Notes for: Machine Learning in Computational Biology, Fall 2024

Taught by Manolis Kellis and Eric Alm

These are automated transcripts, created by ChatGPT 4o in Sept 2024, based on automated transcription of only the audio from the lectures, as automatically generated by youtube.

Link to this page: https://tinyurl.com/MLCB24notes

The youtube playlist is here: https://tinyurl.com/MLCBlectures

Outline

Lecture 1: Course Introduction

Video: MLCB24 Lecture01 Introduction

Slides: Lecture01_Introduction_FA24.pdf

00:00 Introduction

Welcome to the 2024 edition of our **Computational Biology** class! This course will provide a comprehensive overview of key topics in both biology and computational methods, combining foundational knowledge with cutting-edge developments in the field. Throughout the semester, we'll delve into **genomics, epigenomics, disease circuitry, protein structure, immunology, imaging, single-cell analysis, gene regulation, and network modeling**, among other topics. Our aim is to equip you with the theoretical background and practical skills needed to tackle complex biological problems through computational approaches.

Class Structure:

Lectures: Held every Tuesday and Thursday from 1:00 to 2:30 PM.

Recitations: Scheduled on Fridays at 3:00 PM in the same room, where we'll dive deeper into lecture content and address any questions. If you cannot attend, recordings will be made available.

Course Goals:

- Understand key principles of **machine learning** and **algorithm design**.
- Explore influential problems and techniques in **computational biology**.
- Analyze large-scale biological datasets to uncover insights relevant to **human disease** and **drug discovery**.
- Master genome analysis, including **sequence alignment**, **regulatory motifs**, and **epigenomic modifications**.

- Gain expertise in **gene expression**, **network modeling**, and **RNA analysis**.
- Learn about evolutionary principles through **comparative genomics** and **phylogenetics**.
- Investigate the connection between **genetic variation** and **disease**.
- Transition from linear sequences to complex structures, including **protein folding** and **ligand binding**.

We will cover both traditional machine learning techniques like **Gibbs sampling**, **expectation maximization**, and **hidden Markov models**, as well as modern advancements in **deep learning** and **generative AI**, including **graph neural networks** and **deep generative models**.

Faculty and Teaching Staff:

Manolis Kellis: Professor in AI and Computer Science at MIT, and a member of the Broad Institute of MIT and Harvard, specializing in genomics, disease circuitry, epigenomics, and protein structure.

Eric Alm: Co-instructor, known for his work on the microbiome and generative AI, bringing a deep understanding of both computational and biological sciences.

Teaching Assistants: Jared, Sarah, and Dan, who bring a wealth of experience in language models, evolutionary prediction, and single-cell genomics.

What to Expect:

This class will be fast-paced and highly interactive. We expect you to engage actively with the material, participate in discussions, and collaborate on projects.

We've structured the course to maximize your learning and provide ample opportunities for you to apply what you learn through **innovative final projects** that often lead to new research directions, papers, and future career paths in computational biology.

Resources and Logistics:

Course Website: All materials, schedules, and announcements will be available at `compbio.mit.edu/mlcb`

Survey: Please fill out the initial survey to help us tailor the course content to your interests and needs. It will also aid in forming project teams that align with your goals.

This course is designed not just to teach you, but to inspire you to push the boundaries of what is possible in computational biology. We look forward to exploring these exciting topics together!

10:21 Goals of the Course

The primary goals of this course are to provide a comprehensive introduction to **computational biology**, focusing on both the foundational problems and the algorithmic and machine learning techniques that are crucial for data analysis. The skills and methods you will learn are not only applicable to computational biology but also extend to any field of applied data science where real-world challenges, such as noisy data, complex models, and the need for approximation and summarization, are prevalent. Our aim is to equip you with the ability to build meaningful representations of the world and to think critically about how these representations are constructed.

One of the most exciting aspects of this course is that the field is constantly evolving. We're not just exploring established concepts from the past; we're diving into cutting-edge discoveries and innovations that are unfolding right now. By the end of the course, you will be familiar with methods and ideas that are still being refined and discovered. This dynamic environment is what makes teaching and learning computational biology so thrilling—it's a living field, continuously shaped by ongoing research and new findings.

Our goal is not merely to teach you how to use existing tools or software packages. Instead, we focus on explaining **how and why** these methods work, enabling you to think algorithmically and from a machine learning perspective. This deeper understanding will empower you to design the next generation of computational methods, rather than just using those that already exist.

A second key objective is to prepare you to tackle independent research. The course is structured around milestones that will guide you through a final project, allowing you to engage with the latest research literature—often involving papers that are only weeks old. You will have the opportunity to work directly with real data and code from recent publications, exploring questions at the frontiers of the field. Programming assignments and hands-on exercises will challenge you to design parts of algorithms, implement them, and observe their performance in practical scenarios.

The final project is an essential component of the course. You will propose, execute, and present your own research, both in writing and through oral presentations. This experience will not only consolidate your learning

but also position you to contribute novel insights to the field.

We like to think of this course as existing along four axes: **computation and biology** on one side, and **foundations and frontiers** on the other. On one hand, we are deeply invested in solving real biological and medical problems, not just developing biologically inspired algorithms. On the other hand, we emphasize computational fundamentals, equipping you with tools from AI, machine learning, and statistical analysis that can be broadly applied.

The second duality of the course—foundations and frontiers—means that we aim to balance well-established knowledge with exposure to the latest developments. Typically, each module will start with foundational concepts and gradually transition to more advanced, cutting-edge topics. This approach ensures that you not only gain a solid grounding in the essential theories but also stay informed about the evolving landscape of computational biology.

Everyone should have a handout that outlines the course content at a glance. If you don't have one, please raise your hand, and we'll get one to you. This guide will help you navigate the structure of the course and keep track of the topics we'll be covering throughout the semester.

13:52 Course at a Glance

The **Course at a Glance** provides an overview of the entire semester, outlining the key modules, assignments, and important milestones. At the center of this structure, we have the main schedule, broken down into weeks marked by alternating light and dark blue rows. The course is divided into four modules, each spanning several weeks: Module 1 (Weeks 2-4), Module 2 (Weeks 5-7), Module 3 (Weeks 8-10), and Module 4 (Weeks 11-13). Each module will come with a problem set (P-set), which will be distributed as the module begins and due at the end of the module, right before the next one starts. This design ensures that you are always working on current material, without needing to recall content from weeks past while handling new topics.

There are only three P-sets corresponding to the first three modules. Module 4 will not have a P-set but will lead into the quiz, which covers only Modules 1, 2, and 3. Module 4 content, although not included in the quiz, remains important as it could be relevant for your final projects. The P-sets are structured to directly align with the content of each module, ensuring a synchronized learning process. In addition to these assignments, there is also a "Homework 0" designed to get you comfortable with the computational tools we'll be using, such as Jupyter Notebooks, PyTorch, BioPython, and other essential packages. This initial assignment aims to level the playing field, providing everyone the foundational skills required for the rest of the course.

Fridays will often include mentoring sessions or recitations, especially useful for helping you navigate the milestones associated with your final project. These milestones are spaced at the end of every second week, guiding you through the various stages of your project from forming teams to refining your research proposal. The first milestone involves creating a self-profile and recording an introductory video—both designed to ease you into the project phase and help with team formation. Subsequent milestones include selecting key papers, assessing the feasibility of your project, and making sure you can access necessary data and tools before diving deeper into the work.

As the semester progresses, additional milestones will serve as checkpoints where you'll present updates and receive feedback, allowing for adjustments and refinements. One crucial milestone is the mid-course report, scheduled after the quiz. This report functions as a draft of your final project write-up, helping you outline your progress and identify any remaining gaps well before the final submission. Your completed project will be due on the Friday before our last class, with presentations scheduled for the following Tuesday. This timeline is designed to prevent last-minute rushes and ensure that you have ample time to prepare a polished final presentation.

The four modules cover a wide range of topics:

- **Module 1:** Focuses on **genomics, epigenomics, single-cell analysis, and regulatory networks**, providing a deep dive into data that could easily constitute an entire semester on its own.
- **Module 2:** Shifts to **protein structure, protein language models, and geometric deep learning**, exploring the structural and functional complexities of proteins.
- **Module 3:** Covers **chemistry and therapeutics**, including graph neural networks, drug discovery, and molecular interactions, expanding the computational toolkit to tackle challenges in drug development.
- **Module 4:** Wraps up with **electronic health records, imaging, comparative genomics, and metabolomics**—areas that each represent their own specialized fields of study.

Throughout the course, the emphasis will be on active learning through project-based experiences, where you'll explore real-world data and state-of-the-art methods. You'll be encouraged to engage with the material in a hands-on way, learning not just from the lectures but also through collaborations with your peers, Tas, and mentors. The goal is not only to master current computational tools but also to gain the skills necessary to innovate and push the boundaries of what is possible in computational biology.

The upcoming sections of the lecture will dive deeper into the specific content of each module and highlight the exciting topics we'll explore together this semester.

29:40 Why Computational Biology

Computational biology leverages the power of computation to address the complexity of biological systems that cannot be easily captured by traditional mathematical equations alone. Biology is inherently **hacky**—it evolves through a series of tweaks, mutations, and optimizations rather than following a simple, elegant formula like those seen in physics. Biological systems are built from decentralized processes, emerging behaviors, and intricate feedback loops that are not well suited to classical mathematical modeling but can be better understood through computational simulations and machine learning.

Data and Computation play a crucial role in biology due to the sheer **volume of data** generated, particularly with advancements in genomics, proteomics, and other high-throughput techniques. The **iterative process** of hypothesis generation, testing, and refinement becomes far more efficient when supported by computational models, which can run thousands of simulated experiments in the time it takes to perform a single biological experiment in the lab. This approach not only saves time but also reduces the need for ethical dilemmas associated with animal testing and human trials.

Furthermore, the **language of biology**, particularly DNA and RNA, is a **programming language** of its own, vastly different from human languages. Computation allows us to decipher this ancient code, translating sequences into functional molecules, protein interactions, and ultimately, phenotypes. By simulating these processes computationally, we can better understand how genetic information is translated into biological functions and how errors in these processes can lead to diseases.

Computational approaches also offer a way to explore **new hypotheses** that might not be immediately intuitive to human researchers. These models can reveal hidden patterns, suggest novel connections, and even propose mechanisms that researchers might overlook. This ability to look beyond human biases and preconceptions opens up new avenues for discovery, pushing the boundaries of our understanding.

Another critical aspect is the **error-prone nature of biological and technical measurements**. Biology is filled with both intrinsic variability and measurement noise. Computational models help to **distill** the signal from the noise, allowing researchers to extract meaningful insights from complex, high-dimensional data. This capability is crucial in fields like genomics, where patterns are often obscured by the sheer volume of data.

In summary, computational biology serves as a bridge between the raw, unstructured nature of biological data and the structured, analytical world of computation. By combining big data, sophisticated algorithms, and iterative modeling, computational biology enables us to tackle some of the most challenging questions in science today—from decoding the human genome to designing next-generation therapeutics. It's a field where the frontier of knowledge is constantly expanding, driven by the interplay of biology, computation, and innovation.

43:20 Why GenAI is different

This year feels fundamentally different because of the advent of **generative AI** and specifically **representation learning**, which goes beyond mere pattern recognition to grasp deeper meaning within biological data. Unlike earlier AI models that were restricted to recognizing superficial patterns, generative AI and deep learning are enabling machines to **understand complex biological languages**—from the folding of proteins and the intricate chemistry of drug molecules to the genomic sequences that dictate cellular function and the regulatory codes that control gene expression.

Traditional AI has been around for decades, often dismissed as simply benefiting from more data and better computational power. However, **representation learning** is not just about scaling up old techniques; it's about creating models that capture **emergent properties**—capabilities that were not explicitly programmed. These models don't just process data; they learn nuanced, hierarchical abstractions that reflect real-world complexities.

For example, deep learning models can interpret the **language of protein folding**, recognizing how amino acid sequences translate into three-dimensional shapes and ultimately into functional molecules. They can

parse the **language of genomes**, understanding how specific sequences and motifs govern gene regulation, and how these genetic elements interact to orchestrate cellular behavior. Similarly, they are beginning to comprehend the **language of disease and health**, identifying pathways and mechanisms that drive pathological states, all through vast amounts of biological data that no human could parse unaided.

One of the key breakthroughs of generative AI is its ability to build **foundation models**. These are large, self-supervised models trained on massive datasets that allow them to learn representations that are useful across a wide range of tasks. Foundation models possess **emergent capabilities**—they can make connections and generate insights that were not directly encoded during training, which is why this era of AI feels profoundly different.

Self-supervised learning is at the heart of this transformation. Unlike traditional models that rely on human-annotated data, foundation models are trained by hiding parts of their input and learning to predict the missing pieces. This forces the model to develop an understanding of the underlying data structure, learning everything from basic patterns to complex, abstract concepts. For instance, in language models, initial understanding might come from word frequencies, but deeper learning would capture grammar, context, emotions, and subtleties of human interaction—reaching insights that even humans might overlook.

These models are also becoming **multimodal**, capable of interpreting and integrating information from multiple data types simultaneously. Just as models like ChatGPT can blend text and images, new foundation models are being developed to connect **protein structure** and **protein function**, merging 3D molecular geometry with biological descriptions to facilitate novel insights. This capacity to cross domains is key: integrating chemical structures with knowledge graphs, or connecting patient trajectories from clinical notes to structured health data, opens unprecedented opportunities for discovery.

Generative AI and deep learning are thus poised to **transform our understanding of biology** by linking the vast, disconnected pieces of biological knowledge into coherent, interpretable models. Through **hierarchical concepts** and **deep learning**, these models can build layer upon layer of understanding, revealing new insights about disease mechanisms, drug interactions, and much more.

This is why generative AI feels revolutionary: it is not just a new tool but a new way of thinking, capable of expanding human knowledge by interpreting the fundamental codes of life in ways that were previously unimaginable.

57:30 Representation Learning: Images + Genomes

Representation learning, a cornerstone of modern machine learning, transforms how both images and genomes are interpreted by AI. Imagine an AI system analyzing an image of a person carrying a backpack. The process begins with **pixels**, the basic units of a digital image, just as the human retina captures light. These pixels are processed by neurons that recognize fundamental visual features like **edges**, colors, and simple patterns—the first layer of representation learning. From there, the AI builds progressively more complex features: from lines and shapes to identifiable objects like eyes, wheels, or noses. Ultimately, these layers of abstraction allow the AI to comprehend the entire scene as a person with a backpack.

What makes this revolutionary is that these **representations are not pre-programmed** by human engineers; they are learned directly by the machine. The AI is set loose on vast amounts of data and learns by itself which features are most useful for distinguishing objects in the world, from cars and animals to complex scenes involving human activity. This form of self-directed learning represents a significant departure from earlier AI systems, which relied heavily on predefined rules and hardcoded patterns. Instead, the AI learns its own rules for representation at each level of abstraction, building upon the patterns recognized in previous layers.

The **core innovation** here isn't just the increase in data, computation, or model size; it's the shift towards learning representations directly from data, a concept that didn't exist in early AI systems. The transformation from raw inputs (X) to outputs (Y) has always been central to machine learning, but what sets modern deep learning apart is the **latent space (Z)**—the hidden layer of **embedding representations** that captures the underlying structure of the data. This latent space serves as a bridge between raw data and the final output, allowing the AI to make sense of complex inputs through layers of learned features.

In image recognition, for example, convolutional neural networks (CNNs) scan images using filters that identify matching patterns at different levels of complexity. At the lowest level, these filters find basic patterns like lines; at higher levels, they recognize increasingly complex shapes and objects. This **hierarchical learning of representations** is what enables the AI to effectively interpret visual information.

The same principles can be applied to genomic data. In genomics, the AI treats DNA sequences much like an

image, encoding them in a way that makes **patterns recognizable**. DNA can be represented as a series of four bases (A, C, G, T), each acting like a pixel in a 2D image with only four possible colors. By analyzing these sequences, the AI looks for repeated motifs or structures that resemble patterns in images, such as transcription factor binding sites or regulatory elements. These patterns are critical for understanding how genes are regulated, expressed, and how they contribute to cellular functions.

Representation learning thus bridges the gap between diverse data types—whether visual images or genetic sequences—by enabling the AI to identify the key features that drive meaningful distinctions. This approach underpins the advances in generative AI, making it possible for machines not only to classify data but to **understand and generate new data**, mimicking the underlying rules and complexities of natural systems.

1:03:19 Graph Representation Learning in GNNs

Graph Representation Learning is a powerful extension of the principles of representation learning applied to images and genomes, but adapted for complex data structures like graphs. In the context of chemistry, molecules are essentially graphs, where atoms are nodes and chemical bonds are edges connecting these nodes. Unlike the structured grid of pixels in a 2D image or the linear sequence of a genome, graphs are inherently irregular and flexible, making them ideal for representing the intricate connections in chemical compounds.

To learn representations on graphs, we start with **Graph Neural Networks (GNNs)**, which apply the same hierarchical learning principles used in image and sequence analysis. The goal is to iteratively build higher-level abstractions from the basic building blocks of the graph—in this case, the atoms and their connections.

At the **first layer**, each atom is represented with its intrinsic properties: **hydrophobicity**, **charge**, **polarity**, etc. These properties form the **initial representation** of each node in the graph. The next step involves building **layer one representations** by considering how each atom is positioned relative to its neighbors. For instance, a charged atom surrounded by two polar atoms would form a distinctive pattern, contributing to the atom's new representation in the context of the molecule.

As we progress through layers, each node (atom) aggregates information from its neighbors. **Layer one aggregates** data from direct neighbors, while **Layer two** incorporates information from neighbors of neighbors, and so on. This process allows the network to capture increasingly complex interactions within the graph, ultimately creating a **latent representation** of the entire molecule. This abstract, multi-layered representation can then be used to predict properties of the molecule, such as its chemical reactivity, toxicity, or its effectiveness as a therapeutic agent.

This **bottom-up learning approach** allows GNNs to dynamically understand the spatial and relational properties of molecules, much like how convolutional networks learn features in images. By progressively integrating local neighborhood information, GNNs can discern patterns in how atoms and their interactions influence overall molecular function.

The power of this method extends beyond chemistry; it can be used in various fields where relationships and structures are best captured by graphs. For instance, in biology, GNNs can be applied to **protein-protein interaction networks**, **gene regulatory networks**, or **metabolic pathways**. In a drug discovery context, GNNs can correlate the learned representations of chemical compounds with the diseases they affect, enabling predictive models that guide therapeutic design. This ability to model complex, interconnected data makes GNNs a versatile and transformative tool in modern computational biology.

1:06:07 Language Representation Learning in LLMs

Language models, particularly Large Language Models (LLMs), have revolutionized how we process and understand language by employing sophisticated representation learning techniques. At the foundation of this approach is the concept of representing each word with a **ground-level embedding** that captures its contextual meaning based on its usage across vast corpora of text.

The basic idea begins with **predicting missing words** in sentences, which allows models to learn the contextual representation of words. For example, if the missing word is "President," the surrounding context might be similar to that of words like "king," "leader," or "God." This similarity in contexts means these words will be close together in the **embedding space**, which is the foundational principle behind early language models like **Word2Vec**. This approach clusters words with similar meanings, capturing semantic relationships just through exposure to text.

However, the real power emerges when we move beyond representing single words to understanding the

meaning of entire sentences or documents. This is where **attention mechanisms** and **Transformer architectures** come into play. By dynamically adjusting the representation of each word based on the surrounding context, Transformers can capture nuanced meanings that evolve with each word's neighborhood in the text. For instance, the word "Apple" in a sentence can shift its meaning from a fruit to a tech company depending on whether it appears with words like "basket" or "Windows." This context-dependent representation is the core innovation of **Transformer models**.

Attention mechanisms allow models to focus selectively on different parts of a sentence when constructing the meaning of each word, dynamically reshaping its **latent embedding** based on the surrounding text. This enables models to grasp complex sentence structures and subtle semantic shifts, thus moving from basic word embeddings to deeper, contextualized representations that reflect the full meaning of sentences, paragraphs, or even entire documents.

A striking feature of modern LLMs is their **multimodal capabilities**, where they seamlessly integrate different types of data—such as images and text—into a shared representation space. For instance, you can input an image into an LLM, and it can describe what it sees in natural language. Conversely, it can generate images based on textual descriptions, bridging the gap between visual and linguistic information. This joint learning of text and images allows for versatile applications, such as interpreting complex scientific figures or generating novel visuals from descriptions.

This **multimodal embedding space** extends beyond simple image-text pairs; it can encompass representations of proteins, chemicals, genomic sequences, and more, linking them to their functional or descriptive language counterparts. Each point in this space has both a language and a corresponding structural or visual projection, allowing models to navigate seamlessly between different forms of data.

By leveraging these deep, hierarchical, and context-sensitive representations, LLMs enable us to extract, generate, and understand information across diverse fields—from protein folding to chemical interactions and beyond. This makes them invaluable tools not only in computational biology but in any domain that requires the interpretation of complex, multidimensional data. For your final projects, consider exploiting these capabilities to merge language with other forms of data, creating innovative solutions that push the boundaries of what these models can achieve.

1:09:29 Visualizing Z vector Embedding Landscapes

Understanding the hidden, complex relationships in data often comes down to visualizing the **latent space**—the Z vectors—that generative AI and deep learning models create. These vectors, formed through layers of abstraction, encapsulate the hidden features that models learn. Rather than treating AI models as inscrutable black boxes, it's crucial that we actively explore these embeddings, visualizing them to uncover new insights and patterns.

In our research group, we've developed an interactive tool that projects these latent embeddings into two dimensions, allowing us to visually explore complex datasets. For instance, we took the 155,000 papers that have cited our work and projected each into a latent embedding space using a large language model. By applying dimensionality reduction techniques such as **t-SNE** or **UMAP**, we can map these high-dimensional representations into two-dimensional clusters, revealing thematic relationships among the papers that might otherwise remain hidden.

The same approach can be applied broadly across various fields: visualize latent spaces of chemicals, proteins, gene expression patterns, or even entire knowledge domains. The key is to look not at the raw data (X space) or the output (Y space) but to immerse ourselves in the Z space, the world of learned representations.

By overlaying these representations with multimodal data, we can start to see new structures emerge. For example, plotting patient data with Alzheimer's disease against gene expression profiles can reveal distinct axes of variation—like cognition and pathology. Each axis can be associated with specific cell types or genetic expressions, providing new insights into how diseases manifest at the cellular level.

A similar approach has been used to co-embed protein structure and function. In a collaborative project, we combined a **knowledge graph** that connected genes, diseases, drugs, and biological functions with graph convolutional neural networks. This approach allowed us to map drugs, proteins, and phenotypes into a shared latent space, providing a unified view where structural data could directly inform functional insights. This emerging capability to **translate between structure and function** is a step towards bridging gaps like those left by AlphaFold, which solves sequence-to-structure but not structure-to-function.

These techniques are not just limited to biology. We've used them to explore vast collections of textual data, such as the New York Times archive, creating cognitive maps that reveal hidden connections among millions of articles. The same methods can be applied to map scientific literature, grants, and patents, linking them together in ways that reflect their underlying semantic and functional relationships.

Cognitive maps—whether for exploring physical terrain or navigating complex datasets—provide essential simplification and abstraction. They help anchor our understanding, guiding decision-making and sparking new hypotheses. As we progress, the visualization of these latent spaces will be key to unlocking the deeper insights hidden within our data, making this exploration a vital frontier for the field of computational biology and beyond.

1:16:50 The Road Ahead

The landscape of computational biology is undergoing a profound transformation driven by the convergence of multimodal, multi-layer, and hierarchical deep representations of complex biological data. This shift is allowing us to unify the study of diverse data types—from genomes and regulatory circuits to proteins, images, and chemicals. We are no longer limited to studying these elements in isolation; we can now interconnect them in ways that were once unimaginable. This class used to focus solely on genomes and circuits, but now it encompasses proteins, 3D structures, chemical compounds, and more. Why? Because it's not only feasible—it's transformative for our understanding of biology.

This semester, we will dive deep into the frontiers of **representation learning**, dissecting how these powerful models can capture the essence of complex biological phenomena. Our journey will be structured into four main modules:

- **Module 1:** We'll explore gene expression, regulatory networks, epigenomes, and single-cell analysis. This module will set the foundation for understanding how genes are controlled and how cellular decisions are made at the molecular level.
- **Module 2:** We'll connect the regulatory information from Module 1 to protein structure, delving into protein language models, geometric deep learning, and how structure informs function.
- **Module 3:** We will shift to chemistry and therapeutics, focusing on graph neural networks and how they can predict the behavior of small molecules, aid in drug discovery, and model complex biochemical interactions.
- **Module 4:** Finally, we'll tackle imaging, electronic health records, comparative genomics, evolution, and metabolomics, integrating these diverse data types into cohesive models that can transform our understanding of health and disease.

As we cover these themes, I encourage you to **think boldly and creatively** about your final projects. Pull inspiration from any module or combination of modules and design the most ambitious, forward-thinking project you can imagine. The tools we are learning can finally make these ideas a reality. For many of the challenges I've wanted to tackle since I was in your shoes, the technology has now caught up, and we can finally push the boundaries together.

So, take this opportunity to explore, experiment, and potentially change the world. No pressure—but the possibilities are endless, and we are here to guide you every step of the way.

Who's excited? Let's do this!

See you all tomorrow for the recitation on problem set zero—get started on the survey and start brainstorming your project ideas. This is just the beginning.

Lecture 2: Expression Analysis and Clustering

Video: Lecture02 - Expression Analysis Clustering Classification - MLCB24

Slides: Lecture02_ExpressionClusteringClassification.pdf

0:00 Intro

Welcome to Lecture 2! I hope everyone is feeling great today. Just a quick reminder—please fill out the first day survey by Friday. We're organizing teams for the formal mentoring session, and this survey will help us understand you better and form balanced groups. The more information we have, the better we can structure these teams for the mentoring session on Friday, where you will also be completing a one-page profile. This will help your classmates get to know you and facilitate more effective collaboration.

Module 1: Genomics, Epigenomics, Single Cell Analysis, and Networks

Today marks the beginning of our first module, which will cover a wide range of topics in genomics, epigenomics, single-cell technologies, and network analysis. We will delve into the fundamental techniques used to analyze gene expression, explore how these approaches are applied to understand complex biological systems, and introduce key machine learning concepts.

Today's Focus: Expression Analysis

We start with the basics of **expression analysis**, which is foundational in understanding the activity of genes across different conditions or cell types.

We'll introduce clustering and classification, exploring how they fit into broader contexts of **supervised** and **unsupervised learning**.

We will touch upon concepts such as **semi-supervised learning**, which bridges the gap between these two main paradigms.

Key Techniques: Clustering and Classification

K-means Algorithm: A simple yet powerful clustering method that partitions data into groups based on similarity.

Gaussian Mixture Models (GMMs): We'll build on K-means to understand GMMs, which provide a probabilistic approach to clustering that accounts for the inherent variability within data.

These methods will set the stage for deeper exploration of machine learning principles, including Bayesian inference and the contrasts between generative and discriminative models. This will be our first foray into the application of machine learning to computational biology, grounding us in the core techniques that will reappear throughout the course.

Looking Ahead: The Structure of Module 1

Thursday: We will shift focus to sequential data, examining how to align and decode complex patterns using techniques like **dynamic programming**, **Hidden Markov Models (HMMs)**, and **posterior decoding**.

Next Week: We dive into **Gene Regulation**—exploring the regulatory motifs, signals of histone modifications, and the broader regulatory landscape that drives gene expression. This will also involve methods to model these signals computationally.

Single-Cell Genomics: While today's lecture will touch on the basics of clustering and classification, we will dedicate a future lecture to the nuances of single-cell genomics and spatial transcriptomics. These rapidly evolving technologies are reshaping our understanding of cellular heterogeneity and gene expression in situ.

Advanced Visualization: We will also explore nonlinear embeddings for visualizing high-dimensional single-cell data, a topic that we will revisit in detail later in the module.

Graphs and Circuitry: Our journey will culminate in a deep dive into **regulatory circuitry**, using techniques like **principal component analysis** and other dimensionality reduction methods to unravel complex biological networks.

Overview of Future Modules

- **Module 2**: Focus on **protein structure**, diving into the intricate world of protein folding and function.
- **Module 3**: **Chemistry**, examining molecular interactions at a computational level.
- **Module 4**: Exploring **Electronic Health Records (EHRs)** and other large-scale biomedical data sources.

Are you excited to begin Module 1? Today marks our first formal introduction to machine learning within the context of computational biology. We will begin to see how these powerful tools can be leveraged to make sense of the vast and complex datasets that define modern genomics.

4:37 What is Machine Learning

What is Machine Learning?

Machine learning is fundamentally about the ability to **improve performance on a task with more training data**. This concept distinguishes machine learning from traditional artificial intelligence (AI). While AI has been a topic of interest since the 1960s, many AI systems were designed to simulate intelligence by hard-coding decisions and rules. These systems appear intelligent but do not inherently improve over time. If an AI system does not get better with additional data, it is not genuinely engaging in machine learning.

Core Principles of Machine Learning:

1. **Task**: The objective or problem you want the model to solve.
2. **Training Data**: The experience that feeds the model, allowing it to learn patterns and relationships.
3. **Improvement**: The model's performance must enhance as it encounters more data.

Types of Learning Tasks

Machine learning encompasses a broad spectrum of tasks, each with distinct goals:

- **Classification**: A form of supervised learning where the goal is to assign labels to data. For example, distinguishing Alzheimer's patients from healthy controls. Here, we know the correct answer, and the learning is supervised.
- **Clustering**: An unsupervised learning task where we group data without predefined labels, discovering natural groupings or patterns.
- **Regression**: Predicting a continuous value, such as forecasting the age at which an individual might develop Alzheimer's. Regression is about numbers, not categories.
- **Transcription**: Converting data from one form to another, like transforming spoken words into text, as seen with speech-to-text systems.
- **Translation**: Converting information between languages, for example, from English to Greek or French.
- **Structured Output**: Generating organized, meaningful information from raw, unstructured data. For example, translating a doctor's spoken diagnosis into structured fields like "tumor size: 7" or "diabetes stage: severe."

Structured data is organized and quantifiable, while **unstructured data**—like free text or images—lacks such format. Machine learning has traditionally focused on structured data, but this distinction is blurring as techniques for processing unstructured data (e.g., clustering on text or images) have advanced.

Additional Machine Learning Tasks

Anomaly Detection: Identifying unusual or unexpected patterns within data. For example, monitoring patient activities through Wi-Fi signals and detecting falls, triggering an alert to emergency services.

Synthesis: Combining multiple datasets to create a more comprehensive understanding, summarizing key insights from the data.

Imputation: Filling in missing values within a dataset by predicting what those values should be based on the existing data.

Denoising: Identifying and removing noise—unwanted variability in data that can obscure the true signal, such as technical artifacts rather than biological differences.

Across all these tasks, the key is to improve performance with more data. This is the essence of machine learning. To measure improvement, we need a performance metric that quantifies the model's accuracy or effectiveness.

Machine Learning Process

The **training data** serves as the **experience** for the model, akin to how a robot learns from interacting with its environment or how an AI system learns from observing doctor-patient interactions. This ongoing experience allows the model to refine its predictions and responses.

Summary:

- Machine learning is about continuous improvement with more data.
- Performance metrics are essential to validate this improvement.
- It encompasses a wide range of tasks, from classification and clustering to more complex processes like synthesis and anomaly detection.

Understanding these fundamental principles will guide our exploration of machine learning applications in computational biology, as we move from basic techniques to more sophisticated models.

9:32 Making Inferences about the World

In computational biology, and indeed in all scientific analysis, there is a crucial distinction between what we **observe** and what remains **hidden**. Observations might include direct measurements like gene expression values, while hidden states might represent underlying conditions, such as whether a patient has Alzheimer's.

We can think of the world as divided into two realms:

- **The World of Measurements**: This is the domain of observable data, such as expression values, weather conditions, or clinical symptoms.
- **The World of Models and Inferences**: This realm involves the hidden states, hypotheses, and the underlying factors driving those observations.

Inference is the process of deducing hidden states based on observed data. For example, looking outside and seeing bright sunlight might lead you to infer it's summer. If you see snow, you infer it's winter. The observations are direct—light, rain, snow—but the hidden states are inferred: the season, the weather system, or any other underlying condition.

Why Inferences Matter

Understanding hidden states allows us to make **predictions** about future conditions or decisions. For example, if you infer that a storm is building based on current weather patterns, you might decide against sailing. Similarly, in a medical context, observing high expression of a specific gene might allow us to infer a higher probability of a disease like Alzheimer's.

Key Concept: You observe measurable parts of the world to infer hidden, often more critical aspects of that world.

Forward Probabilities and Inferences

Very often, we can directly measure and express **forward probabilities**—the likelihood of observable events given a known state of the world. For example: **Winter** might have a 60% chance of snow. **Summer** might have a 60% chance of sunshine. These are relatively easy to measure because they start from a known condition and predict observable outcomes.

However, **inferences** work in the reverse direction. They aim to determine the hidden state given the observations: **Given it's raining, what's the probability of a storm? Given high expression of a particular gene, what's the probability of Alzheimer's?**

Forward probabilities are often described as **generative probabilities** because they model how known states of the world generate observable data. For instance, if you know it's summer, you can predict temperatures and weather patterns typical of that season.

Generative Models

Generative models describe how data is produced by sampling from a hidden state. For example:

1. If you are in the state of "June," you might sample temperatures typical of June, resulting in a range of observed values.
2. If you know the distribution of these values (mean, standard deviation), you can compare any given observation to these expectations.

When you have an observation, like a temperature reading, you can assess how likely it is to have come from a "June" or "July" distribution. This helps determine whether it is more likely that you are in June or July based on the observed data.

Example: If you record a temperature of 72 degrees, you can compare how probable that reading is under different generative models (e.g., June vs. July) and make an inference about the most likely state.

While weather in places like Boston may not follow neat patterns due to its variability, the broader concept holds: **you use known generative models of the world to interpret and infer hidden states from new observations.**

14:25 Reversing the Arrows: Bayesian Inference

In machine learning and computational biology, we often work with a **generative model** that allows us to calculate the probability of specific observations given a particular state of the world. For example, we can determine the probability of observing snow given that it is winter, or more broadly, the probability of any measurement given a specific condition.

However, our real goal is to **reverse these arrows**. We want to infer the hidden state (e.g., "the patient has Alzheimer's") given the observed data (e.g., gene expression measurements). This requires reversing the direction of the relationship between data and the underlying hypothesis or state of the world.

Using Bayes' Rule for Inference

This reversal is precisely where **Bayes' Rule** comes in. Bayesian inference uses Bayes' Rule to flip the direction of the relationship:

- **Forward Probability (Generative)**: Probability of observing data given a hypothesis, P(Data∣Hypothesis)P(\text{Data} \mid \text{Hypothesis})P(Data∣Hypothesis).
- **Reverse Inference**: Probability of the hypothesis given the data, P(Hypothesis∣Data)P(\text{Hypothesis} \mid \text{Data})P(Hypothesis∣Data).

Bayes' Rule helps us go from knowing the probability of data given a hypothesis to inferring the probability of the hypothesis given the data:

$$P(\text{Hypothesis} \mid \text{Data}) = \frac{P(\text{Data} \mid \text{Hypothesis}) \times P(\text{Hypothesis})}{P(\text{Data})}$$

Understanding Bayes' Rule in Simple Terms

To simplify, consider this analogy: The purple area in a diagram could represent the part of the world that is both red and blue. You can calculate it in two ways:

1. Multiply the fraction of the world that is blue (P(B)P(B)P(B)) by the fraction of the blue that is also red (P(A∣B)P(A \mid B)P(A∣B)).
2. Multiply the fraction of the world that is red (P(A)P(A)P(A)) by the fraction of the red that is also blue (P(B∣A)P(B \mid A)P(B∣A)).

In both cases, the purple area remains the same, demonstrating how these probabilities are interconnected.

Applying Bayes' Rule to Inference

In the context of Bayesian inference, the formula helps us derive what we call the **posterior probability**:

- **Posterior Probability** (P(Hypothesis∣Data)P(\text{Hypothesis} \mid \text{Data})P(Hypothesis∣Data)): The probability of the hypothesis after observing the data.
- **Likelihood** (P(Data∣Hypothesis)P(\text{Data} \mid \text{Hypothesis})P(Data∣Hypothesis)): The probability of observing the data given the hypothesis.
- **Prior Probability** (P(Hypothesis)P(\text{Hypothesis})P(Hypothesis)): The initial probability of the hypothesis before observing any data.
- **Marginal Probability** (P(Data)P(\text{Data})P(Data)): The probability of observing the data under all possible hypotheses.

This marginal probability is calculated as a weighted sum of the likelihoods across all hypotheses. It helps normalize the results but doesn't change the relative ranking of different hypotheses.

Importance of the Prior Probability

One critical insight in Bayesian inference is the role of the **prior probability**. It adjusts the inference based on what we already know before observing the data. For instance, if a disease is extremely rare (like one in a million), even a positive test result might not be enough to strongly suggest you have the disease because the prior probability of having it was so low to begin with.

This principle explains why medical testing often follows symptoms rather than random screening: tests are not perfect, and without a high enough prior likelihood (such as symptoms suggesting a specific condition), the probability of a false positive can outweigh the true detection.

Example in Medical Contexts

Consider taking a test for a rare disease. If the test is positive, is it more likely that you have the disease or that it was a false positive? Often, due to the low prior probability of the disease, the latter is true—you might not have the disease even with a positive result. This highlights the crucial role of priors in Bayesian inference, especially for rare conditions.

Summary

- **Likelihood** tells us the probability of observing the data given the hypothesis.
- **Prior** tells us the probability of the hypothesis before considering the data.
- **Posterior** combines these to provide the probability of the hypothesis after seeing the data.

Bayesian inference allows us to formally incorporate prior knowledge and data to make reasoned judgments

about hidden states, a powerful tool in both scientific research and decision-making.

20:20 Clustering and Classification

In the context of Bayesian inference and machine learning, we are first introducing a simple algorithm for clustering, which is a key unsupervised learning task. We will also touch upon how this relates to more general approaches like Gaussian Mixture Models (GMMs).

Gaussian Mixture Models (GMMs)

- **GMMs** represent scenarios where data is generated from a mixture of multiple Gaussian distributions.
- Imagine sampling data points from several distributions: one green, one red, one blue.
- In a real-world problem, we don't know the underlying distributions, centroids, or spreads.
- **Clustering** attempts to infer the hidden distributions without any labeled guidance.

Supervised vs. Unsupervised Learning

- **Unsupervised Learning**: Finds patterns or structures in data without labels. For example, inferring clusters when you don't know the green, red, or blue labels.
- **Supervised Learning**: Uses labeled data to classify new points. For instance, if we have clusters of labeled points (green, red, blue) and we encounter a new gray point, we use the learned structure to classify it based on proximity.

These two main tasks—**clustering (unsupervised)** and **classification (supervised)**—are central to the lecture. For example, in genomics, you may measure gene expression in different tissues (like brain and liver) and attempt to classify or cluster the data based on these measurements.

Visualizing High-Dimensional Data

We often visualize data in two dimensions due to human limitations, even though biological data can have thousands of dimensions (e.g., 20,000 genes measured simultaneously).

Dimensionality Reduction Techniques: **Principal Component Analysis (PCA)**: Identifies major axes of variation in data. **Nonlinear Embeddings**: Techniques like t-SNE or UMAP that help project high-dimensional data into lower dimensions while preserving the structure.

Clustering vs. Classification

Classification: Uses known labels to develop rules for assigning new data points into existing classes. This involves feature selection and creating metrics like accuracy, false positives, and false negatives.

Clustering: Groups data points based on proximity without predefined labels, identifying structure and patterns within the data. It's challenging to validate because it lacks explicit performance metrics and often requires external validation.

Feature Selection and Representation Learning

Feature Selection: Involves choosing relevant variables from a dataset, especially when you have many more features than samples (e.g., 20,000 genes vs. 70 students).

Feature Construction: Creating new variables from existing ones (e.g., combinations of gene expressions).

Representation Learning: The process of discovering underlying features or patterns that best represent the data's true nature, such as latent variables like disease states or demographic factors.

Semi-Supervised and Self-Supervised Learning

Semi-Supervised Learning: Combines labeled and unlabeled data, allowing the model to learn the structure from unlabeled data and refine it using the labeled points.

Self-Supervised Learning: Uses a pretext task, such as predicting hidden parts of data, to help the model learn underlying structures. This approach blurs the line between supervised and unsupervised learning, especially useful in scenarios with massive datasets but limited labels.

Metrics and Validation

Classification Metrics: When labels are available, we can validate models using performance measures like accuracy, sensitivity, specificity, etc.

Clustering Validation: Often relies on indirect methods, like assessing how well the clusters correlate with additional, unseen measurements.

33:45 AI vs. ML vs. Representation Learning vs. Generative AI

In this lecture, it's crucial to differentiate between various concepts like AI, Machine Learning (ML),

Representation Learning, and Generative AI. Let's explore these concepts in detail.

Artificial Intelligence (AI)

AI encompasses many systems, including those that are not inherently learning from data but are designed to simulate intelligence. **Examples include** rule-based or knowledge-based systems, where rules are hardcoded to make decisions based on input without improvement over time. **ML is a subset of AI** that involves learning from data to improve performance on a task. **Classical ML** often involves hand-designed features. For example, in ML models, we manually choose measurements or features and use data to infer the parameters that weigh these features, improving with more data. **Representation Learning goes beyond ML** by automatically discovering the features from raw data rather than relying on hand-designed features. Instead of specifying the relevant gene expression features manually, representation learning allows the model to infer these features, potentially uncovering complex combinations and patterns.

Deep Learning is a special type of representation learning where features are learned hierarchically: Lower layers learn simple features. Higher layers build on these, learning more complex patterns. This multi-layered learning approach leads to increasingly abstract representations.

Generative AI falls within the broader field of deep learning and focuses on models that can generate new data instances similar to the training data. Examples include image generation, text generation, and more.

Distinguishing AI, ML, and Deep Learning

AI (Broadest Scope): Encompasses systems that mimic intelligent behavior, including non-learning systems.

ML (Within AI): Systems that learn from data to improve at tasks.

Representation Learning (Within ML): Automatically learns features from data rather than relying on human-designed features.

Deep Learning (Within Representation Learning): Uses multiple layers of feature learning, each layer building on the previous one, allowing for more complex abstractions.

Applications and Evolution

Classical AI in Chess: Earlier AI approaches in games like chess involved hardcoded rules, handcrafted scoring functions, and algorithms like Minimax to decide moves. This approach highlighted the limitations of hardcoded intelligence.

Shift from Hard Tasks for Humans to Hard Tasks for Machines:

- **Old Paradigm**: Tasks like chess, logic, and planning were seen as the pinnacle of intelligence—hard for humans, but structured and rule-based, making them easier for machines.
- **New Paradigm**: Recognizing images, driving cars, and understanding natural language are innately easier for humans but were extremely challenging for machines until recent advancements in deep learning.

Deep Learning and Human Cognition

Layers of Abstraction: Similar to how the human brain processes information, deep learning uses layers of neurons (real or artificial) to recognize increasingly complex patterns—from edges to shapes to complete objects.

Neocortex Expansion: Evolutionarily, humans have greatly expanded the neocortex, enabling us to perform complex reasoning, sensory processing, and abstract thinking—all of which are mirrored in the architecture of deep learning models.

General AI Implications: Current models excel at pattern recognition, a domain where humans also excel innately. There is ongoing debate about whether artificial general intelligence (AGI) will emerge from scaling these models or if new, structured approaches are needed.

Summary

The concepts of AI, ML, representation learning, and deep learning represent a progression towards models that can better understand and interact with the world.

These techniques are not just technical; they mirror how we think and learn, offering profound insights into both artificial and natural intelligence.

46:06 ML for Gene Expression Analysis

Machine Learning for Gene Expression Analysis

Gene expression analysis involves studying how genes are expressed in different cells or tissues, which can

reveal important biological insights, such as identifying disease mechanisms or cell types. This process starts with isolating tissue samples and often involves extracting nuclei from complex tissues like the brain. The extracted nuclei are encapsulated in tiny droplets, each tagged with a unique DNA barcode. This barcode ensures that all RNA molecules from a particular cell are identifiable, linking them back to their original cell of origin.

The barcoding process allows researchers to sequence the RNA from individual cells and generate a dataset known as an expression matrix. This matrix contains counts of RNA molecules corresponding to specific genes in each cell. The matrix's rows typically represent individual genes, and the columns represent individual cells, with each entry indicating the expression level of a gene in a particular cell. Modern sequencing technologies have dramatically reduced the cost of sequencing, enabling the quantification of gene expression at a single-cell level with unprecedented resolution.

Once the expression matrix is generated, it is used for downstream analyses such as principal component analysis (PCA) and clustering. PCA helps reduce the dimensionality of the data, making it easier to visualize and identify patterns. Clustering, on the other hand, groups similar cells or genes, revealing hidden structures within the data. For example, clustering might group genes that have similar expression profiles across various conditions or group cells that exhibit similar gene expression patterns, potentially indicating a common cell type or state.

Gene expression data can be analyzed in two main ways: comparing expression profiles across genes or across conditions (e.g., different patients or treatments). In the context of disease, such as Alzheimer's, clustering can reveal how gene expression patterns differ between affected and unaffected individuals, helping to identify disease-associated genes.

Clustering vs. Classification

Clustering and classification are core tasks in gene expression analysis but serve different purposes. Clustering is an unsupervised learning technique used to discover groups of similar data points, such as cells with similar expression patterns, without prior knowledge of group labels. The goal of clustering is to find inherent structure in the data, which can reveal biologically meaningful patterns like distinct cell types or subtypes.

Classification, on the other hand, is a supervised learning approach where the objective is to assign predefined labels to new data points based on features extracted from the data. For example, classification can be used to predict cell types based on gene expression profiles or to distinguish between diseased and healthy cells. This process involves training a model using labeled data, where the classes are known, and then using this model to predict labels for new, unlabeled data points.

In practice, real-world gene expression data is highly complex, often involving millions of cells and thousands of genes, as seen in large-scale datasets with millions of single cells from hundreds of donors. To make sense of this high-dimensional data, dimensionality reduction techniques are often employed, projecting the data into lower dimensions that capture the most relevant biological variation while preserving as much information as possible.

Practical Challenges in Gene Expression Analysis

While the concept of clustering and classifying cells or genes seems straightforward, the actual implementation involves overcoming several practical challenges. These include technical noise, batch effects from different experimental runs, sparsity in the data due to dropout events where certain genes are not detected, and variability between cells that might not be biologically meaningful. Researchers must also contend with the integration of data from different platforms and experimental designs, alignment issues, and the possibility of capturing doublets or multiplets, where more than one cell is encapsulated in the same droplet.

These complexities require careful preprocessing and normalization to ensure that the clustering reflects true biological differences rather than technical artifacts. Understanding these nuances is crucial for interpreting results correctly and for making meaningful biological inferences from gene expression data.

In the subsequent parts of this lecture, we will explore algorithms like Gaussian Mixture Models and K-means clustering, which are fundamental tools in this field. These models help to infer the underlying structure of gene expression data, allowing researchers to make predictions and identify key patterns that can advance our understanding of biology and disease.

54:29 K-means Clustering

K-means clustering is a popular **unsupervised learning** algorithm used to partition data points into a

specified number of clusters, KKK. The primary goal is to group data in a way that points within each cluster are as similar as possible, while being distinct from points in other clusters. The process involves an iterative approach, where clusters are progressively refined until a stable configuration is reached.

The algorithm works through the following steps:

1. **Initialization**: K-means begins by randomly placing KKK cluster centroids within the data space. These centroids serve as initial guesses for the cluster centers.
2. **Assignment Step**: Each data point is assigned to the nearest centroid based on a distance metric, typically **Euclidean distance**. This step groups points into KKK clusters based on proximity.
3. **Update Step**: Once points are assigned, the centroids are updated by calculating the **mean position** of all points assigned to each cluster. This effectively moves each centroid to the center of its associated points.
4. **Iteration**: These steps are repeated—reassigning points and recalculating centroids—until convergence, which occurs when there are no further changes in the assignments of points to clusters or when the centroids stop moving significantly.

How K-means Works: Step-by-Step Illustration

Imagine a set of scattered data points with no prior knowledge of how they were generated. K-means starts by placing KKK centroids randomly. In each iteration, the algorithm assigns each point to the nearest centroid, effectively grouping them into clusters. After assigning the points, the centroids are recalculated based on the average position of their assigned points, effectively refining the clusters.

For instance, if the centroids are initially placed close to certain clusters, the assignment step will immediately group nearby points. As centroids adjust and shift closer to the true centers of their respective clusters, the algorithm gradually refines the cluster boundaries, improving the representation of the data structure with each iteration.

Objective of K-means

The primary objective of K-means is to **minimize the within-cluster sum of squares (WCSS)**, which is a measure of how spread out the points in each cluster are around their centroid. The goal is to make clusters compact and well-defined. Mathematically, this is expressed as:

$$\text{Cost} = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

where CkC_kCk denotes the set of points assigned to cluster kkk, and μk\mu_kμk is the centroid of that cluster. The algorithm's iterative process of assigning points and updating centroids is specifically designed to minimize this cost, thereby improving the quality of the clustering.

Challenges and Modifications of K-means

While K-means is straightforward and effective, it has notable limitations, including sensitivity to the initial placement of centroids and the assumption that clusters are **spherical and equally sized**. Moreover, K-means assigns each point rigidly to a single cluster, which can be problematic when points lie near cluster boundaries.

Fuzzy K-means is one variation that addresses these limitations by allowing partial assignment of points to multiple clusters. Instead of assigning a point entirely to one cluster, Fuzzy K-means assigns probabilities that represent the likelihood of the point belonging to each cluster. For example, a point near the boundary of two clusters might be 60% associated with one cluster and 40% with another, reflecting the inherent uncertainty.

K-means as a Special Case of Gaussian Mixture Models (GMMs)

K-means can be seen as a special case of **Gaussian Mixture Models (GMMs)**, where each cluster is modeled as a Gaussian distribution. In GMMs, points are assigned probabilistically based on the likelihood that a given cluster generated each point. This probabilistic framework allows GMMs to capture more complex data structures, especially when clusters overlap or have different shapes.

In summary, K-means clustering provides a simple, yet powerful way to discover hidden structure in data. It iteratively refines clusters to minimize internal variance, offering a clear, interpretable approach to understanding complex datasets. Its extensions, such as fuzzy K-means and GMMs, further enhance its ability to model the true underlying distributions of data, making it a versatile tool in gene expression analysis and

beyond.

1:01:27 Gaussian Mixture Model Sampling

Gaussian Mixture Models (GMMs) provide a framework for understanding data generated from a mixture of multiple Gaussian distributions. In GMMs, we assume that data points are generated by a set of Gaussian distributions, each with its own mean and variance. This model allows for a more nuanced understanding of data compared to K-means, which only considers cluster centers.

Imagine a scenario where you have different clusters of points, each generated from a different Gaussian distribution. For instance, some points come from a red cluster, others from a blue cluster, and so on. In GMMs, you start by flipping a coin (or rolling a die) to decide which Gaussian distribution to sample from, and then you draw a point from that distribution based on its spread and mean. This process leads to clusters where the density of points decreases exponentially as you move away from the center, following the familiar bell-shaped curve of the Gaussian.

The Ubiquity of the Gaussian Distribution

The Gaussian distribution isn't just a mathematical construct; it frequently appears in real-world scenarios. A classic physical example is the Galton board, where balls randomly fall left or right as they bounce through a grid of pegs, creating a distribution of balls that forms the shape of a Gaussian curve at the bottom. This highlights how randomness in many natural processes often results in Gaussian distributions, making them a powerful model for data analysis.

Understanding Points in the Context of Multiple Gaussians

When dealing with GMMs, each point in your dataset has a likelihood of being generated by each of the different Gaussian distributions. For example, a point near the center of the red distribution is highly likely to have been generated by the red cluster, while a point between the red and blue clusters might have probabilities of belonging to both, with varying likelihoods. This probabilistic assignment differentiates GMMs from K-means, where each point is rigidly assigned to the closest cluster.

However, GMMs go beyond simple cluster centers—they incorporate different spreads and variances, allowing for clusters of different sizes and shapes. This is particularly useful when clusters overlap or have elongated shapes, which are not well captured by K-means.

Expectation-Maximization (EM): Iterative Refinement

To find the optimal set of clusters in a GMM, we use an iterative approach called **Expectation-Maximization (EM)**. EM allows us to iteratively refine our estimates of the cluster parameters (means, variances, and priors) and the assignments of points to clusters. This process involves two main steps:

1. **Expectation Step (E-step)**: Assume that the cluster parameters (means, variances) are known and use them to estimate the probability that each point belongs to each cluster. This step assigns points probabilistically rather than deterministically, accommodating the inherent uncertainty near cluster boundaries.
2. **Maximization Step (M-step)**: Given these probabilistic assignments, update the cluster parameters to better fit the data. Specifically, you compute new means, variances, and mixing proportions that maximize the likelihood of the observed data under the current probabilistic assignments.

These steps are repeated iteratively, gradually refining both the cluster assignments and parameters until convergence, i.e., until further updates no longer significantly improve the fit of the model to the data.

Connection to K-means and Beyond

The EM algorithm can be seen as a generalization of K-means, where instead of hard assignments, we make probabilistic assignments of points to clusters. This probabilistic nature allows GMMs to capture more complex data structures, accounting for varying cluster sizes, shapes, and overlapping areas.

In K-means, each point is assigned to the closest cluster center based purely on Euclidean distance, without considering variance or priors. In contrast, GMMs consider both the probability of a point being generated by a cluster and the shape of the distribution. This makes GMMs a much more powerful tool for modeling real-world data, especially when the assumptions of equal variance and distinct boundaries do not hold.

Overall, Gaussian Mixture Models, paired with the EM algorithm, offer a robust framework for discovering the underlying probabilistic structure of data, accommodating overlapping clusters, variable spreads, and complex shapes, which are beyond the reach of simpler clustering methods like K-means.

1:09:58 Hierarchical Clustering

Hierarchical clustering provides a versatile approach to grouping data without requiring you to pre-select the number of clusters, as is necessary in methods like K-means. It works by building a nested set of clusters, arranged as a hierarchy, which can be visualized in a tree-like diagram called a dendrogram. This approach is particularly useful when you want to explore data relationships at multiple levels of granularity.

How Hierarchical Clustering Works

The basic process of hierarchical clustering involves iteratively merging the closest pairs of points or clusters. Here's how it unfolds:

1. **Initialization**: Each data point starts as its own cluster.
2. **Merging Closest Pairs**: At each step, the two closest clusters are merged into a single cluster. The definition of "closest" can vary, affecting how the clusters form:
    - **Single Linkage**: Merges clusters based on the closest pair of points from each cluster. This approach can create elongated, chain-like clusters.
    - **Complete Linkage**: Merges clusters based on the furthest pair of points between clusters, promoting compact cluster formations.
    - **Average Linkage**: Uses the average of all pairwise distances between points in the two clusters, balancing between the single and complete linkage.
    - **Centroid Linkage**: Merges clusters based on the distance between their centroids (centers of mass), recalculating centroids after each merge.
3. **Forming the Hierarchy**: As clusters are merged, a hierarchy forms. This hierarchical representation allows you to choose the number of clusters dynamically by cutting the dendrogram at different levels.

Advantages of Hierarchical Clustering

No Need to Predefine K: Unlike K-means, hierarchical clustering doesn't require setting the number of clusters beforehand. You can decide how many clusters make sense after examining the dendrogram.

Flexible Cluster Shapes: It can capture complex cluster structures, including nested clusters, which are challenging for methods like K-means that assume spherical clusters.

Visual Insights: The dendrogram offers a clear visual representation of how clusters merge, providing insights into the data's structure at multiple levels.

Challenges and Considerations

Distance Metrics: The choice of distance metric (e.g., Euclidean, Manhattan, Pearson correlation) can significantly influence clustering results. For example, Pearson correlation is useful when you want to cluster based on similarity in pattern rather than magnitude.

Time Complexity: The approach can be computationally intensive, particularly when calculating all pairwise distances ($O(N^2)$). Using centroid linkage can reduce some of this computational burden but requires recalculating centroids each time clusters merge.

Applications and Evaluation

Hierarchical clustering is widely used in various fields, such as biology (e.g., gene expression analysis) and market segmentation, where it's helpful to see data grouped naturally. To evaluate the results, you can compare clusters with external labels or additional measurements to assess how well the clustering captures meaningful patterns.

In summary, hierarchical clustering provides a flexible, intuitive way to group data, especially when exploring hierarchical relationships without needing to commit to a specific number of clusters in advance.

1:13:10 Clustering of Documents and Free-Form Text

In 2024, clustering is no longer confined to numerical data points; it has expanded into the realm of text and free-form documents, enabling the analysis of vast and varied datasets in new, insightful ways. This capability allows us to group not just structured data but also unstructured data like survey responses, patient descriptions, or any textual information that conveys complex, nuanced meaning.

Clustering Text: Beyond Numbers

Traditionally, clustering involves grouping points based on numerical features. For example, we might cluster millions of cells by their gene expression patterns or patients by their clinical profiles. However, text clustering takes this further by allowing us to work directly with language, extracting hidden themes, and patterns that would be impossible to see with numbers alone.

One example involves clustering survey responses about personal experiences. Responses like "I have a relationship that did not go as expected" or "I struggled with personal setbacks" can be clustered based on semantic similarity. This approach reveals common psychological or emotional themes that aren't captured by standard psychiatric categories defined decades ago. It provides a more personalized understanding of human experiences, showing that individual struggles often share common threads, even if they don't fit neatly into traditional categories.

Applications of Text Clustering

Psychiatric and Emotional Analysis: Clustering text responses can highlight shared struggles, like parental health concerns or relationship breakups, allowing for a more dynamic understanding of mental health that adapts to the diversity of human experience.

Gene Function Descriptions: Instead of clustering genes solely by their expression data, one can cluster them based on the descriptions of their functions. For example, by analyzing the free-text descriptions of proteins, you can organize 20,000 genes by their roles, such as lipid metabolism or biosynthesis, creating an intuitive navigation through biological functions.

Exploring Knowledge Spaces: Tools like gene function explorers or curated knowledge maps, built by students and researchers, demonstrate how clusters of text can reveal connections between seemingly disparate topics. For instance, a cluster might highlight how lipid-related genes intersect with various diseases, providing an interactive way to explore complex biological relationships.

Media Content Analysis: Clustering text goes beyond scientific data. For instance, clustering topics from thousands of podcast transcripts, like Lex Fridman's discussions, allows one to visualize thematic landscapes. You might find that philosophical discussions, including "the meaning of life," naturally cluster near other profound topics, revealing the underlying structure of content in a way that's immediately accessible and engaging.

Implications for Broader Applications

The power of text clustering lies in its ability to connect diverse data types, merging quantitative measurements with qualitative descriptions. For example, co-embedding gene expression data with the textual descriptions of those genes can create a unified space that combines biological function with experimental observations, opening new avenues for understanding complex datasets.

These techniques encourage a broader view of what clustering can achieve, moving beyond the limitations of traditional numerical analysis. By embracing text clustering, we can uncover insights in everything from personal experiences to genetic functions, providing a richer, more nuanced understanding of data across multiple domains.

1:17:06 Naive Bayes Classification

Naive Bayes classification is a **simplified version of Bayesian inference** where you already know the classes that data points belong to, such as Class 1 and Class 2. This pre-knowledge allows you to directly assign **probability distributions** to data points within each class. Unlike clustering methods where class membership and distributions must be inferred from scratch, Naive Bayes leverages known class memberships, making it a more straightforward approach.

Estimating Distributions

With Naive Bayes, estimating the **center of mass** (mean) and the **spread** (standard deviation) of each class becomes trivial since you already know the points belonging to each class. From these estimations, you can calculate the **probability of a given class** given the observed data, essentially flipping the probabilities using Bayes' rule. The goal is to compare the probability of Class 1 given the data versus Class 2 given the data.

Forward Probability: This represents the likelihood of observing certain data points given that they belong to a specific class.

Backward Probability: Using Bayes' rule, we reverse the direction of inference, estimating how likely it is that a data point belongs to a class given the observed data.

Bayesian Classification

In Bayesian classification, the task is to determine which class a new data point most likely belongs to by comparing probabilities. We compute:

Probability of Class 1 given the data: Uses the forward probability of data generation multiplied by the prior probability of Class 1.

Probability of Class 2 given the data: Similar calculation with respect to Class 2.

Using these probabilities, you can define a **discriminant function**—a threshold to classify a point based on which class probability is higher. This decision boundary can be tuned by adjusting the log probabilities or raw probabilities until it accurately separates the classes.

Parametric vs. Non-Parametric Models

To estimate these distributions, you have two main approaches:

1. **Parametric Models**: Assume a predefined form (e.g., Gaussian distribution) with parameters like mean and standard deviation. This works well when the distribution fits a known shape.
2. **Non-Parametric Models**: Do not assume any specific distribution shape. Instead, they segment the data space and measure the frequency of observations in each segment, making them more flexible when the data doesn't conform to a simple bell-shaped curve.

Combining Multiple Features with Naive Bayes

The classic Naive Bayes approach makes the **naive assumption** of **feature independence**, meaning it treats all features as if they don't affect each other. This assumption simplifies calculations, as the joint probability of all features given a class becomes the product of individual feature probabilities.

However, this independence assumption is rarely true in real-world data. To address this, you can:

- Use **Principal Component Analysis (PCA)** to reduce the data to independent components.
- Apply Bayesian classification on these principal components, which are designed to be less correlated.

Key Takeaways

- Naive Bayes classification works well when class memberships are known, allowing straightforward estimation of distributions.
- It simplifies multi-feature probability calculations by assuming independence, though this assumption often needs adjustments in practice.
- Despite its simplicity, Naive Bayes remains powerful for many classification tasks, especially when combined with dimensionality reduction techniques like PCA.

This approach offers an accessible yet robust method for tackling classification challenges, especially in domains where quick, interpretable models are valuable.

1:20:50 Summary

In this lecture, we covered several foundational topics in **gene expression analysis** and **machine learning**:

- **Gene Expression Analysis**: We explored how gene expression data is collected, processed, and represented as a matrix of gene counts across various conditions or cells. This matrix forms the basis for further analysis using machine learning techniques.
- **K-Means Clustering**: We discussed this classic algorithm for partitioning data into a fixed number of clusters. By iteratively assigning points to the nearest cluster center and updating the cluster centers based on the mean of the points assigned to them, K-means helps identify underlying structure in the data.
- **Hierarchical Clustering**: Unlike K-means, hierarchical clustering doesn't require the number of clusters to be specified in advance. Instead, it builds a nested set of clusters organized into a hierarchy, which can be visualized as a dendrogram. This method is flexible and can be used to explore data at various levels of granularity.
- **Naive Bayes Classification**: We covered this simple yet powerful probabilistic classifier that assumes independence among features. By applying Bayes' theorem, Naive Bayes calculates the likelihood that a data point belongs to a specific class based on its features, making it an effective tool for classification tasks.
- **Clustering and Classification Beyond Numbers**: We extended the discussion to clustering and classification of text, showing how these techniques can be applied not only to numerical data but also to free-form text and other data types. This highlights the versatility of these methods, especially in the context of modern tools like large language models.

This lecture provided a broad overview of key machine learning techniques used in computational biology, setting the stage for more advanced topics in the next sessions. On Thursday, we'll delve into **Hidden Markov**

Models, **dynamic programming**, **parsing**, and the analysis of **sequential data**—continuing to build on these foundational concepts.

Lecture 3 - Single-cell Analysis

Video: Lecture 03 - Single Cell Analysis - MLCB24

Slides: Lecture03_SingleCellGenomics.pdf

0:00 Intro

Intro: Alrighty, welcome everyone! Today, we're going to be talking about **single-cell genomics**. On Tuesday, we talked about **gene expression analysis**, and I mentioned that there are a lot of complications with single-cell genomics, which we would cover in lecture 13. Well, guess what? We're going to talk about them today because your problem set includes the entire pipeline of single-cell analysis, including **differential analysis, pathway enrichment, and some single-cell epigenomics**—all blended into an awesome **Jupyter notebook**.

Why are we doing this? As you start thinking about your final projects, we want you to have all the tools at your disposal. A lot of people have talked about including single-cell genomics in their projects, so we want to enable and empower all of you to carry out these single-cell analyses, which are at the forefront of where biology is right now. A lot of mysteries about how genes work and how disease impacts the body are very difficult to uncover when you're looking at **bulk analysis**. However, the moment you're able to separate the different cell types in the body, things become much more amenable to these types of analyses.

So, let's dive right in. We're going to start with **why single cells** are important. We will review traditional approaches and the emergence of single-cell **RNA sequencing** across different types of technologies, because understanding the technological underpinnings can help you understand issues with **noise** and when the technology doesn't work exactly right. Then, we're going to dive into the **biology**: what are the questions you can ask about single cells? We'll focus on a lot of the work our lab is doing on understanding the brain in the context of **Alzheimer's, schizophrenia**, and so on.

Then, we'll step back and look at the **basic computational tasks** associated with single-cell analysis. We'll explore some frontiers of the field, including **deep learning and representation learning methods** for understanding single cells. Finally, we'll touch briefly on areas beyond **transcriptomics**, including some of the **epigenomics, multiomics**, and so forth. Excited? Yes? Awesome! Great, let's do it!

Why Single Cell: Again, there are many ways to think about gene expression analysis. The easiest analogy often used in talks, lectures, and newspaper articles is the **smoothie versus individual fruits** or the **fruit salad**. When you're doing bulk transcriptomics, you're essentially blending everything together like a smoothie.

4:48 Bulk vs. Single-Cell

Bulk vs. Single-Cell: Remember in the first lecture, or Tuesday's lecture, I was basically saying, "Oh, you look at brain expression versus liver expression." It sounds reasonable, right? Like if I tell my parents, "Oh yeah, we looked at brain expression relative to liver expression," it makes complete sense to them. But to a scientist, especially someone working on the liver or brain, they're like, "Liver? Okay, which of the 47 cell types in the liver?" And someone working on the brain would ask, "Which of the 143 subtypes of excitatory neurons in the brain?"

So, when you think about liver expression, what you're actually doing is taking this extraordinary diversity of cell types and blending them together into a **smoothie**. You're looking at the liver smoothie, or the brain smoothie, or the lung smoothie. But this analogy is not great; in reality, what you're after are the **individual fruits**—the distinct cell types—and how they behave separately.

Today, we'll especially look at this in the context of **self-projected phenotypes**, a measure and method we've been developing in my group with Gerald Bond and others. Even within a single cell type, there are **continuums of variation** that are very often biologically driven. So, even at the level of sorting the beautiful kiwis, there isn't just one type of kiwi—there's a gradient, and that gradient is associated with different biological properties, sometimes even different disease properties.

When our group started looking at single-cell biology, it became very difficult to go back to bulk analysis. We've gone all out, profiling more single cells than we could have imagined. For example, we've profiled over **20 million cells** from our group alone, which sounds unfathomable. If someone had told you, "I'm going to give you 20 million cells," it would have seemed impossible not long ago. Remember when I showed you the gene expression by condition? It was thousands of genes across hundreds of conditions. Now, we're not just talking about hundreds of conditions or thousands—we're talking about millions and tens of millions of conditions, which are the cells.

This is important because in every tissue in your body—whether it's the brain, lung, liver, or any other—there's an extraordinary diversity of interplaying cell types. That's concept number one. Concept number two is that often, when we talk about single-cell analysis, we're dealing with neurons in the brain that can extend all the way down to our toes. So, how do you do single-cell profiling? You would have to pull out the entire neuron, take all the RNA from all the different axons and dendrites, unweave it, and then profile it.

This is very challenging, especially in the brain, where cell types—like **oligodendrocytes, astrocytes,** and **excitatory or inhibitory neurons**—are intricately interwoven. So, sequencing the entire RNA content of a single cell is hard. As a result, an approximation we often use is **single-nucleus sequencing** because the RNA in the cytoplasm is produced in the nucleus before it gets transported to the rest of the cell. You can think of it as an approximation of what's being produced right now rather than the actual RNA molecules that are being translated into proteins in the cytoplasm.

Moreover, as we push for more resolution, we might not just want to know the total content of RNA inside a neuron; we might want to know how much RNA is sitting in one part of the axon versus another, or in specific dendrites. The **transport of RNA** along the cell body to different parts is extraordinarily important, and that's a technological frontier on its own.

There are some purists who argue, "Oh, single-nucleus sequencing makes no sense," while others say, "RNA expression makes no sense; everything happens at the protein level." Others still argue that, "Proteins don't make sense; everything happens at the post-translational modifications of the protein level," and so on. My answer to all of these purists is that we have to start somewhere. We start where the data is, with the technologies that are accessible today. By using current methods, we drive innovation and push the frontiers of technology further.

To give a historical perspective: in 1995, we could have waited another decade for sequencing costs to drop before starting the **Human Genome Project**, but without doing it with the available technology, we wouldn't have driven the innovation needed to reduce costs, improve technology, and create the market demand that propelled the field forward. So, even though single-nucleus RNA sequencing is an approximation, it's incredibly useful, and we should be aware that better techniques will come as technology evolves. Until then, we will keep pushing the boundaries of what's possible with the current tools.

In summary, even though we know it's an approximation, single-nucleus RNA sequencing provides extraordinary insights. The analogy here is the **smoothie, the individual fruits**, and then the third analogy—the **tart**, which represents **spatial transcriptomics**. Spatial transcriptomics doesn't just look at cells isolated from their context but examines where each cell is positioned within the tissue. This spatial context is crucial because cells located in different parts of the tissue, such as near blood vessels, may behave differently from those elsewhere.

Single-cell technologies are incredibly powerful but also limited because they don't always tell you exactly where each cell is positioned. Spatial transcriptomics, which we'll touch on later, addresses this by linking the cells' transcriptomes to their physical location, revealing spatially distinct patterns that contribute to our understanding of tissue function and disease.

12:08 Why Single Cells

Why Single Cells: Let's talk a little more about why studying individual cells is so important. As I've been emphasizing, it's crucial to study single cell types separately, like analyzing a kiwi distinct from a banana. But it's also essential to study multiple kiwis separately and multiple bananas separately because even within a single cell type, there is a remarkable diversity.

When you look at cells in the body or in a dish, they can look very different from each other. Under the microscope, they display different **morphologies** and have different expression levels of individual proteins. Even cells of the same type can exhibit dramatic differences from one another. For example, in **blood differentiation**, you have an extraordinary diversity and nearly a **continuum of cellular identity** as you go from **hematopoietic stem cells**—the progenitors of all blood lineages—through the various stages of differentiation. These hematopoietic lineages give rise to a wide array of cell types, each with distinct functions, and even within those cell types, there is further diversity.

One of the main reasons for this variability is **stochasticity**. Each cell has an expression program, but which subset of genes gets activated at any given moment can differ. Even though all the necessary genes are there, the order and timing of their activation can vary. Some of this variability is due to **stochastic processes**, like how specific regulators move through the nucleus, interact with various genomic regions, and influence which

genes are expressed.

There's also stochasticity in how cells respond to **intracellular signals**. For instance, receptors on a cell might receive different amounts of a given signal, leading to variations in cellular state transitions even among otherwise similar cells. These transitions can create significant diversity between individual cells.

Developmentally, each cell follows a program that starts with a single genome and evolves through a series of divisions influenced by factors like **maternal deposition of gradients** and the cell's position within the embryo. For example, cells positioned ventrally versus dorsally, or anteriorly versus posteriorly, are subject to different signals and decisions at each cell division. These choices are "remembered" by the **epigenome**—a layer of regulation we'll discuss in more detail next week—and this epigenetic memory influences how cells respond to subsequent signals.

Early methods for studying individual cells involved dissociating tissue, isolating cells, placing each one into a separate well, amplifying the RNA from each well, and sequencing it. This initial approach to **single-cell profiling** allowed us to sequence about 600 cells at a time, which was extraordinary back then. It enabled us to see dramatic differences between cells, distinguish cell types, and observe their varied responses and activation stages, such as how cells react to stimulation after one hour, two hours, four hours, or six hours.

This traditional approach has evolved into more automated methods. Now, we can use technologies like **cell sorting** apparatuses that shoot individual cells through sorting devices and place them into wells. Even more advanced are **microfluidic technologies** that encapsulate each cell in a droplet—a tiny reaction chamber about the size of one cell. Inside each droplet, a bead loaded with millions of **barcodes** tags the RNA molecules from that cell, allowing us to identify which RNA originated from which droplet and therefore from which cell.

However, challenges arise when isolating cells or nuclei. Ambient contamination can affect the quality of the RNA, especially in compromised samples like those from cancer patients or individuals with neurodegeneration. For example, neurons in neurodegenerative conditions might have damaged nuclei, making them more susceptible to contamination during sequencing.

Regarding **spliced versus unspliced RNA**, since we're often working with nuclear RNA, we capture both forms. This can provide insights into the **dynamics of gene expression**. By analyzing the proportion of spliced versus unspliced transcripts, we can infer cellular trajectories, such as differentiation paths from one state to another. If you see that a gene is still unspliced in one cell but fully spliced in another, you can map these observations onto a differentiation lineage, identifying cells transitioning between developmental states.

Single-cell analysis also highlights the pitfalls of bulk analysis. In bulk, you average gene expression across many cells, which can obscure significant variability. For example, if some rare cells express high levels of a particular RNA, bulk analysis might suggest moderate expression across all cells, misleadingly implying that all cells express that RNA when, in reality, most cells have none, and a few have a lot.

This variability—whether due to stochastic processes, environmental influences, or intrinsic differences between cells—is mostly biological, not just technical noise. Single-cell sequencing reveals these subtleties, helping us appreciate the rich complexity of cellular behavior that is otherwise hidden in bulk measurements. Understanding this variability is crucial for exploring the diverse responses of cells and the underlying mechanisms driving health and disease.

24:37 scRNA-seq Technologies

scRNA-seq Technologies: Early single-cell RNA sequencing (scRNA-seq) studies started with just a few cells—sometimes 10 or 100 cells. In less than a decade, we've progressed to capturing millions of cells, which is mind-blowing. So, what are the key technologies that enabled this dramatic innovation?

Let's start with the basics of isolating single cells. In the earliest methods, you could simply pipette individual cells by eye and place them into 96-well plates, treating each well as if it were a sample from a different tissue or individual. Another method involved using **capillary pipettes under a microscope** to capture cells more precisely.

A more advanced technique is **fluorescence-activated cell sorting (FACS)**, where a laser detects the presence of specific cell surface markers or other proteins on the cells. As each cell travels through a tube, you can decide whether to keep it or remove it by applying a current or magnetic field. This approach allows you to isolate specific cell types of interest, such as activated T cells, which can be identified by their markers of activation.

You can also use **laser capture microdissection**, where you look at a tissue and precisely cut out a tiny part

with a laser. However, one of the most widely used methods today is **microfluidics**, which involves cells in suspension flowing through a microfluidic tube while beads loaded with barcodes capture each cell's RNA. These barcodes are sequences of nucleotides, and every bead has millions of barcodes that are identical. For example, one cell's RNA might be tagged with the barcode sequence "AGGTTCGA," while another cell's RNA might get "TGACCAGG."

These microfluidic systems leverage the physical properties of materials, where the lysis buffer and oil do not mix, forming tiny reaction chambers—essentially bubbles—that encapsulate each cell. The bead inside the bubble tags every RNA molecule from the cell with the same barcode, allowing researchers to determine which RNAs came from the same droplet and, therefore, the same cell.

The general methodology across these technologies involves first isolating the cells, then amplifying the RNA, and ultimately counting or measuring the RNA molecules. There are different approaches, such as **full-length sequencing**, which provides the splicing patterns for each RNA, or **digital gene expression analysis**, which focuses on sequencing only the tail end of each RNA molecule.

Why sequence the tail end? The sequencing process starts with a primer that initiates the polymerase reaction. For many eukaryotic genes, this primer can simply be "TTTT," because eukaryotic mRNA transcripts often end with a **poly-A tail** composed of "AAAA." The primer binds to this tail, allowing sequencing to start from the end of the molecule. This method captures the number of times a gene is expressed without providing detailed splicing information.

There are many different technologies for scRNA-seq, each varying in the number of cells captured, sensitivity, cost, and cell capture methods. For example, traditional methods of isolating each cell in a well can cost about $3 to $6 per cell, which sounds cheap until you start working with millions of cells. On the other hand, microfluidic-based methods can bring the cost down to as little as five cents per cell.

Microfluidic systems work by encapsulating each cell in a droplet, where a bead with a specific barcode tags every RNA molecule. After lysis, all the mRNA from a cell gets the same barcode, allowing you to pool all the droplets together and sequence the entire mixture in one bulk reaction. This tagging process makes it possible to trace back each RNA to its original cell, even though the final sequencing is performed on a combined sample.

There's also another technology that uses multiple rounds of barcoding. Imagine you start with a million cells and split them into 100 wells, each well getting a unique barcode. You then blend the cells and split them again, adding a second unique barcode, and repeat the process a third time. By the end, each RNA molecule from each cell has three distinct barcodes, representing the different wells it passed through.

This combination of barcodes allows you to trace RNA molecules back to their original cells with a high level of precision. The probability that two cells would end up with the same three-barcode combination is extremely low—effectively one in a million—making this a robust method for distinguishing RNA from different cells, even when sequenced together.

These technological innovations have transformed single-cell analysis, allowing us to capture the extraordinary diversity of cellular behaviors and states at unprecedented resolution. The microfluidic and multi-round barcoding methods are particularly impactful because they enable high-throughput and cost-effective scRNA-seq, pushing the boundaries of what's possible in biological research.

36:16 scRNA-seq Biological Questions

scRNA-seq Biological Questions: Is everybody with me on technologies? Great. Let's now talk about **biology**. There are three parts to this lecture: the **technological part**, the **biological part**, and the **computational part**. We've covered the technological part; let's dive into the biological aspect. What do we actually do when we have tons of single-cell data? This is a lot of the research happening in my lab, and it's exciting because I get to teach about things that didn't exist just a year or two ago. So, what do we do?

The goal of our lab is to understand the **circuitry underlying disease**. We aim to explore how **genetic differences** or **environmental factors** lead to **molecular changes** that then manifest as disease states. These molecular differences can occur at the **RNA level** or at the **epigenome level**—the latter of which we will discuss next week.

We profile massive numbers of both **healthy and diseased samples** using single-cell profiling. We then perform the analyses that I will show you to identify **driver genes, regulatory regions, regulators, cell types, pathways, and processes**. The key aim is to infer **causality** from this analysis because **genetics** inherently gives us a causal link. Genetic variants can lead to observable differences in a direct, causal manner, unlike

environmental exposures, which occur after genetic differences are already present.

You inherit genetic differences from your parents, so it's far more plausible that the genetic differences are driving the phenotypic changes rather than the other way around. However, genetic differences could also indirectly lead to behaviors, like eating unhealthily, which then contributes to disease, demonstrating the complex interplay of genetics and environment.

We try to infer these **causal paths** by understanding how genetic variants act at the molecular level, allowing us to pinpoint where to **intervene** in the disease process. Now, I'll share some of the lessons we've learned from applying these methods to a variety of disease samples. We've established numerous collaborations, applied for dozens of grants, and we're studying a wide range of traits at single-cell resolution.

This work requires incredible coordination across multiple teams: **doctors** who provide the samples, **patients and their families** who generously contribute samples (often postmortem), **lab technicians** who handle the sorting and profiling, and the **experimental scientists** who push forward the technologies necessary to make all of this possible. Then, there are the **computational analysts** who analyze the data, and the entire team comes together to interpret the results. It truly takes a village to conduct these studies.

We apply this approach across a wide spectrum of conditions, including **Alzheimer's disease, frontal temporal dementia, Lewy body dementia, ALS, Huntington's disease, schizophrenia, bipolar disorder, Down syndrome, autism, depression, PTSD, suicide, and aging**. We profile dozens of cell types across tens of millions of cells, often from multiple regions of the brain and sometimes from other tissues outside the brain. Most of the time, we perform **RNA profiling through single-cell transcriptomics**, which is our main focus today, though we also analyze **DNA accessibility** in some cases—a topic we'll cover more next week.

40:49 84k cells from 48 individuals

84,000 Cells from 48 Individuals: In our study on Alzheimer's disease, we embarked on a groundbreaking exploration of the postmortem human brain, conducting the first analysis of Alzheimer's at single-cell resolution. We collected data from **84,000 cells** across **48 individuals**, a pivotal moment in a long and extraordinary collaboration with Lee Wei, the director of the Picower Institute for Learning and Memory. This work was spearheaded by Hyejung Maddis, who has led the project both experimentally and computationally, and has since established his own lab at the University of Pittsburgh.

This initial study, led by Matheus Jose Davilla—who now has his own lab in Italy—set the stage for a series of papers that have continued to push the boundaries of our understanding of Alzheimer's disease at the cellular level. The core question we aimed to address was: **What do these cells reveal about the disease?** Understanding the roles and states of these cells in the context of Alzheimer's required a series of complex computational analyses.

41:45 Cleaning up Data

The first critical step in our analysis is to **clean up the data**—a process that is crucial for ensuring the accuracy and reliability of our findings. But what does it mean to clean up the data? This involves identifying **quality metrics** that allow us to filter out problematic data points. We may need to remove individual cells that do not meet quality standards, discard entire experiments if they were compromised, or exclude specific genes that exhibit abnormal behavior.

Once this data cleaning is completed, what remains is an **expression matrix**, a foundational tool in our analysis. This matrix represents a comprehensive view of gene expression across thousands of cells, where rows correspond to genes and columns correspond to individual cells. This cleaned and structured dataset provides the basis for all subsequent analyses, enabling us to uncover the hidden dynamics of gene activity within the complex landscape of Alzheimer's disease.

42:22 Clustering and Cell Annotation

Clustering and Cell Annotation: After data cleaning, the next crucial step is **clustering the cells**. This involves grouping cells together based on their gene expression profiles to identify distinct cell types. However, defining these clusters can be challenging. For instance, you might label a group as progenitor cells, but within that group, there could be subtle subtypes. Similarly, in the brain, **excitatory neurons** may not form discrete types but instead exist on a gradient across cortical layers 1 to 5, each with distinct but overlapping expression patterns.

The task of clustering is essentially about identifying the **major drivers of variation** in the data, and it's a form of **unsupervised learning**. Once you've clustered the cells, the next step is to **annotate these clusters**, giving each one a name based on its most characteristic features. For example, cells within a cluster that share

a particular expression pattern might be identified as a specific cell type.

Annotation relies on understanding which **genes are most highly expressed** within each cluster. This process is not done in isolation; it builds on decades of research where scientists meticulously studied gene expression in individual cells using microscopes and small-scale methods. The key difference with modern scRNA-seq is the scale—while traditional studies might have analyzed 20 or 100 cells, today's technologies allow us to examine millions of cells simultaneously, revealing new insights into cellular diversity that were previously hidden.

44:18 DEGs Gene Expression Changes with Phenotypes

With our dataset of 84,000 cells from 48 individuals, one of the first questions we asked was: **How do the expression patterns of these cells correlate with the phenotypes of the individuals?** Each column in our data represents a different person, each with a unique phenotype. Some individuals are categorized as controls or non-Alzheimer's, meaning that, based on pathological analysis, they do not display the classic **signatures of Alzheimer's**—such as neurofibrillary tangles, neuritic plaques, or amyloid-beta deposition. In these individuals, these pathological markers are very low, while they are significantly elevated in others.

Additionally, we analyzed other indicators such as **signatures of parkinsonism** and **cognitive decline**. Among the control group, most individuals maintained high global cognition, except for one who showed signs of parkinsonism. In contrast, those diagnosed with severe Alzheimer's exhibited marked cognitive decline and high levels of pathology. Another subset of individuals showed intermediate levels of pathology, categorized as mild cognitive impairment (MCI) or mild Alzheimer's.

The next step was to determine what differentiates these groups. We focused on identifying **differentially expressed genes (DEGs)** between the Alzheimer's cases and controls. This allowed us to explore whether the cells from Alzheimer's patients exhibited higher expression of certain genes compared to cells from non-Alzheimer's individuals.

Conducting these analyses raises important statistical considerations. Imagine you have 20,000 cells from advanced Alzheimer's patients and another 20,000 from controls. It seems straightforward to compare the gene expression distributions between these groups. However, this approach mistakenly assumes that each cell is an independent sample. In reality, these cells are not sampled in isolation; they come from 24 individuals in each group. Thus, the sampling process is hierarchical: you first sample individuals, and then you sample cells from within those individuals.

This hierarchical nature affects how we assess DEGs. In bulk data, differential expression is simpler because you have one measurement per group. In single-cell data, however, you end up with thousands of measurements for each individual rather than 20,000 independent measurements.

There are three main approaches to DEG analysis in single-cell data:

1. Cell-Centric Approach: This method treats every cell as an independent sample, which can lead to overconfidence in the results because it underestimates the true variance by not accounting for the hierarchical sampling. You mistakenly think you have more samples than you actually do.

2. Pseudobulk Approach: This method averages gene expression across all cells from each individual, effectively creating one measurement per person. While this approach is statistically sound, it oversimplifies the data, losing critical insights into cellular variability that could be biologically meaningful.

3. Hierarchical Mixture Models: These models recognize the hierarchical sampling process, accounting for the fact that cells are nested within individuals. This approach is statistically powerful but more complex, requiring more time and assumptions about the sampling process. If these assumptions are incorrect, the results can be misleading.

Using these methods, we can identify DEGs and explore biological differences between Alzheimer's and control groups. One of our findings was that late-stage Alzheimer's shows extensive gene expression changes that are often shared across multiple cell types. In contrast, early-stage Alzheimer's reveals more **cell-type-specific alterations**, suggesting distinct molecular mechanisms at different disease stages.

We also examined how **biological sex** influences these expression changes. For example, in non-Alzheimer's individuals, there are significant differences in gene expression between male and female excitatory neurons. In Alzheimer's cases, these sex differences in expression patterns are even more pronounced, indicating that sex may play a critical role in disease biology.

Similarly, we assessed the impact of **age** on gene expression. Differences in expression could be driven by

age rather than disease status alone, especially in a cohort where individuals vary widely in age. Recognizing these factors helps refine our understanding of the underlying biology and emphasizes the importance of considering variables such as sex and age in disease studies.

Ultimately, these analyses are not just about finding the answers but about learning **how to think critically** about the data and the complex biological questions it raises.

52:33 Multi-Region Analysis

Multi-Region Analysis: scRNA-seq allows us to go beyond single regions of the brain to explore how **Alzheimer's disease** and other conditions progress through different areas. Alzheimer's progression often follows a stereotypical pattern, starting near the **locus coeruleus** at the base of the brain, then spreading to the **entorhinal cortex**, hippocampus, and eventually reaching broader cortical areas.

We extended our study to multiple regions of the brain to understand these disease dynamics better. We profiled cells from the **entorhinal cortex, hippocampus, thalamus, mammillary body**, and various neocortical regions, such as the **angular gyrus, medial cortex, and prefrontal cortex**. This multi-region approach reveals how different cell types are distributed across the brain and how their expression patterns vary by location.

For example, in some regions, we observe specific **subtypes of excitatory neurons** that are unique to that area, while in others, a broader diversity of neuron types exists. Similarly, different types of **astrocytes** and **oligodendrocytes** show distinct spatial distributions. This spatial variability in cell types and their gene expression is critical for understanding how diseases affect different parts of the brain.

The next step is to **annotate** these additional cell types and determine their distinguishing features or **marker genes**. As we expand the analysis from 84,000 cells to 1.6 million cells, the increased data volume enables us to identify finer distinctions between cell types. Technological advances since our initial study in 2019 have further improved our ability to capture and profile large numbers of cells.

With these detailed annotations, we can begin to explore **cellular trajectories** within each region, painting a picture of how disease-related changes propagate over time and space. For example, some cell types might display a progression of gene expression that corresponds to advancing stages of Alzheimer's, providing clues about how the disease spreads from one brain area to another.

55:45 Module Analysis

In traditional analysis, we often look at individual genes to assess how their expression changes in response to conditions like Alzheimer's disease. This approach involves running **thousands of independent tests**, one for each gene, to determine what changes are occurring. However, the statistical confidence in any one gene's change can be limited. But if multiple genes within the same **pathway** show consistent changes, our confidence increases, revealing a coordinated biological response.

Instead of analyzing each gene independently and then retrospectively examining whether these genes belong to the same pathway, a more powerful approach is to directly group genes based on their co-expression patterns. Imagine taking the vast expression matrix—where rows represent genes and columns represent cells—and **decomposing** it into smaller, meaningful components using techniques like **singular value decomposition (SVD)**. This mathematical method allows us to simplify the data, revealing underlying patterns where genes are not acting in isolation but are instead organized into **modules**—groups of genes that function together.

These gene modules provide a way to think about the complex interplay within cells, allowing us to interpret high-dimensional data as being driven by lower-dimensional spaces. Each module captures a set of genes that are co-regulated, reflecting shared biological processes. By clustering genes into these modules, we can begin to ask deeper questions: **Which genes are grouped together? What biological roles do these modules represent?**

The development of this modular analysis approach was led by researchers including Benjamin James, who has been instrumental in simplifying complex problem sets for further exploration, and Carlos Bacheschi, now a postdoc at Harvard Medical School. They focused on understanding what these gene modules actually represent. By examining correlations between pairs of genes within the matrix, they could identify **networks** of interconnected genes, offering a visual and functional understanding of how genes interact.

These modules aren't just abstract groupings; they can be visualized as **networks** where each gene's position reflects its connections to other genes. This network-based view allows us to go beyond the question of whether a gene is simply part of a cluster. It reveals the **strength and nature of each connection**, showing

how tightly genes are co-regulated and helping us understand the broader functional landscape of the transcriptome.

With this framework, we can start exploring how these **gene modules change in a coordinated manner** across different cell types and in association with various pathological signatures of Alzheimer's disease. This approach enables us to link specific modules to key phenotypes—such as **cognitive impairment, diffuse plaques, neuritic plaques**, or **neurofibrillary tangles**—and assess their roles in the progression of disease. By focusing on gene modules rather than isolated genes, we can capture a more integrated view of cellular behavior, bringing us closer to understanding the molecular underpinnings of Alzheimer's and other complex conditions.

58:50 Q1: Why Modules instead of single-genes

Question 1: Why Analyze Modules Across Clusters Instead of Individual Genes?
A fundamental question arises: Why analyze gene modules across clusters of cells rather than focusing on individual genes in single cells? The primary advantage of this approach is **robustness**. When measuring individual genes in individual cells, the sample size is often too small, making the data inherently noisy. For example, if a gene is expressed at only a few molecules per cell, these measurements can vary widely, leading to high levels of uncertainty.

By shifting the focus to **modules of co-expressed genes** rather than single genes, and by examining these modules across clusters of cells, we gain more stable and reliable measurements. When you're not just looking at one RNA molecule but rather at a set of molecules that are correlated and act together, the overall signal becomes stronger and more consistent. This collective approach captures the underlying biological processes more accurately, reducing noise and increasing the statistical power of the analysis.

59:30 Q2: Difference from Bulk

Question 2: Difference from Bulk Analysis
One of the key limitations of bulk analysis is the loss of **cell-type specificity**. In bulk experiments, all cell types are averaged together, making it impossible to discern how gene expression changes within specific subpopulations, such as particular subclasses of microglia or neurons. For instance, when comparing gene expression between Alzheimer's patients and controls in bulk, you might observe an overall decrease in the expression of certain genes. However, this decrease could be misleading, as it often reflects the loss of neurons—a hallmark of Alzheimer's—rather than a true downregulation of those genes within the remaining cells.

In Alzheimer's, neuron loss skews the data significantly. For example, you might find 4,000 neurons in a control individual but only 2,000 in an Alzheimer's patient. This disparity creates an illusion that thousands of neuron-specific genes are decreasing in expression, simply because there are fewer neurons to contribute their gene expression to the average.

Single-cell analysis overcomes this by allowing you to examine gene expression changes **within each specific cell type**. This enables you to accurately measure how gene expression is altered within neurons, microglia, astrocytes, and other cell types separately, without being confounded by changes in cell composition. Even with pseudobulk analysis, where cells are dissociated and grouped into broad categories like excitatory neurons, inhibitory neurons, or astrocytes, you still maintain the ability to observe expression changes within defined cell populations.

This targeted approach reveals the true dynamics of gene expression changes across different cell types, providing a much clearer picture of the underlying biology than bulk analysis ever could. By focusing on each distinct cell type, single-cell analysis not only enhances our understanding of disease mechanisms but also highlights the critical roles of individual cell populations in conditions like Alzheimer's.

1:01:30 Q3: Robustness and Reproducibility

A common concern when analyzing complex data, especially when moving from individual genes to broader modules, is **robustness and reproducibility**. You may wonder: If I run the analysis again, will the results be exactly the same? Will "Module 13" appear again, and if so, will it look the same as before? This is a valid question, and one that delves into the heart of scientific reproducibility.

However, it's important to recognize that **exact reproducibility** is not always the ultimate goal. If the experiment were repeated with a slightly different set of individuals, what matters more is whether the **biological insights** remain consistent across varying conditions, datasets, or cohorts—not just whether the analysis can be replicated on a different computer with the same data. True robustness lies in the ability to

reproduce findings across different samples, studies, and even populations.

To test robustness, one approach is to perform **subsampling experiments**—randomly selecting cells from the same individuals or splitting the data into two artificial replicates to see if the results hold. Although these methods provide some insight, they are not true biological replicates because each sample or section of tissue might inherently differ due to subtle regional variations.

In a particularly instructive study, two independent teams conducted parallel analyses on overlapping cohorts. Fiorella Jagger's group, initially based at the Broad Institute and now at Columbia, profiled approximately 200 individuals, creating two biological replicates for each person. Our group profiled over 430 individuals, including around 200 that overlapped with Jagger's cohort. This setup allowed us to compare three biological replicates—two from their study and one from ours—for the same individuals.

Such direct comparisons offer a rare opportunity to test how well results align across different research teams and methodologies. An ideal project, for example, would be to analyze how these biological replicates relate to each other. Despite minor variations in experimental execution, many of the same **pathways and gene modules** emerged consistently across studies, reinforcing the validity of key findings.

It is also essential to acknowledge that results can change with different software versions or analytical settings. These changes may or may not affect the overall interpretation, but they reflect the evolving nature of computational biology. In this rapidly advancing field, some hypotheses will withstand the test of time, while others may need revision as new data and techniques become available. The iterative nature of science means constantly learning, adjusting, and moving forward.

Waiting for perfect reproducibility or fully mature technology can paralyze progress. If we had delayed sequencing the human genome until every tool was refined, we would still be waiting. Instead, by engaging with emerging technologies and data, we advance the field, continually building on what we learn. As more studies replicate findings independently, confidence grows in the robustness of specific biological insights, while other hypotheses may be revised or discarded.

The dynamic nature of this field, filled with ongoing discoveries and opportunities for further exploration, underscores the importance of engaging with the data at hand, even as methods and knowledge continue to evolve.

1:07:05 Linked Regions Correlation

One particularly intriguing analysis we conducted was examining the **correlation of changes** in different cell types across brain regions between individuals. Specifically, we asked: How do these changes in cellular behavior co-vary between connected brain areas? For example, we found that specific neurons in the **subiculum** of the hippocampus were highly correlated with corresponding cell types in the **entorhinal cortex**. This observation extended beyond a single brain region, highlighting coordinated patterns of cellular alteration between regions that are anatomically and functionally linked.

This kind of analysis is exciting because it allows us to identify **co-variances** in cellular changes across different brain regions and across individuals. For instance, we could observe that a specific subclass of neurons in one region was consistently covariant with another subclass in a different region across our cohort of individuals. From decades of neuroscience research, we know that these regions often have direct **projections and connections**, suggesting that they communicate and influence each other's function.

These findings align with the well-documented progression pattern of Alzheimer's disease, which often follows a stereotypical pathway from the **locus coeruleus** to the **entorhinal cortex** and then into the **hippocampus** and other cortical areas. One hypothesis is that the pathology progresses due to either **physical damage** or disruptions in neuronal signaling between connected regions. Neurons rely on proper signals from connected regions to maintain their health and function; if these signals are disrupted, it may contribute to correlated patterns of damage.

Understanding these linked patterns provides valuable insights into how Alzheimer's and similar neurodegenerative diseases spread through the brain's complex networks, highlighting the importance of inter-regional connections in maintaining brain health.

1:09:01 Discrepancies between Phenotype and Transcriptome

Another compelling avenue of analysis is exploring discrepancies between an individual's **phenotype** and their **transcriptome**, which can reveal unexpected patterns and potential misdiagnoses. In a recent study on schizophrenia, published in *Science*, we examined whether we could predict whether a person had schizophrenia based solely on their expression patterns. Remarkably, the model worked well for many

individuals, correctly distinguishing between controls and schizophrenia cases. However, the most intriguing findings were the exceptions.

One control individual—let's call them Control 7—was consistently misclassified by the model as having schizophrenia based on their gene expression. On further investigation, it was discovered that this person was the father of a son diagnosed with schizophrenia. This raised the possibility that the expression patterns captured something **inherited**, suggesting a latent genetic or environmental predisposition to the disorder, despite the absence of clinical symptoms.

Conversely, some diagnosed schizophrenia cases exhibited expression patterns more typical of controls. This suggests the possibility of **distinct pathways** leading to schizophrenia-like symptoms that may not align with the conventional understanding of the disorder. Such findings highlight the potential for gene expression analysis to uncover hidden **subtypes** or alternative mechanisms of disease that might not be captured by traditional diagnostic methods.

Building on this approach, we now have a dataset of 430 individuals with postmortem single-cell data from Alzheimer's patients. A fascinating project would be to investigate those diagnosed as having Alzheimer's but whose gene expression profiles appear more typical of healthy individuals. This could help identify **misdiagnoses, overdiagnoses,** or previously unrecognized subclasses of the disease, offering insights that could refine clinical practices.

Another valuable analysis involves exploring the **upstream regulators** of differentially expressed genes, identifying common regulatory elements that might drive pathological changes. Studying the genetic impacts on gene expression is another direction, linking somatic mutations or genetic variants to altered transcriptional patterns. This approach has been applied to various conditions, including psychosis, ALS, frontotemporal dementia, Huntington's disease, and others, each revealing unique regulatory shifts that could inform therapeutic strategies.

For instance, in a recent study published in *Cell*, we identified significant gene expression changes in different subclasses of microglia and vascular cells. Although vascular cells represent only about 0.3% of our data, they exhibit distinct expression patterns depending on whether they are located in venous or arterial regions of the vasculature, with these patterns further altered in Alzheimer's patients.

In another study, published in *Nature Neuroscience*, we explored **single-cell accessibility and epigenomic differences** in Alzheimer's, identifying widespread epigenomic erosion in affected individuals. We also analyzed the **mutational burden** by comparing reference DNA with sequenced RNA. These mutations, which could result from somatic changes at the DNA level or from damage occurring after RNA transcription, were significantly higher in dementia patients compared to controls. This mutational burden was particularly pronounced in specific cell subtypes, hinting at cellular vulnerabilities contributing to disease progression.

Interestingly, individuals with higher mutational burdens also displayed greater **epigenome erosion**, where the typically distinct boundaries between active and repressed genomic regions began to blur. This pattern of **global epigenomic flattening** underscores a critical aspect of neurodegenerative disease: the gradual loss of cellular identity and function at the molecular level.

These insights illustrate the power of transcriptomic and epigenomic analyses to uncover layers of complexity in disease that go far beyond what can be observed clinically, opening new pathways for understanding and eventually intervening in these challenging conditions.

1:14:20 scRNA-seq Analysis Questions

The diversity of analyses that can be performed at the single-cell level is vast, offering a window into the complexities of cellular behavior that bulk analyses cannot match. The workflow typically begins with **raw data**—count matrices that detail the number of reads per gene for each cell. From there, a series of preprocessing steps are essential, including **quality control, data correction, normalization,** and **feature selection**. These steps help refine the data by removing low-quality cells, correcting for technical artifacts, and focusing on the most informative genes, often the highly variable ones.

Once the data is preprocessed, the cells can be **visualized** in a lower-dimensional space using methods like **t-SNE** or **UMAP**, which allow for the identification of distinct cell populations based on gene expression patterns. This visualization helps reveal the structure of the data, highlighting clusters that often correspond to specific cell types or states. These clusters can then be **annotated** using known marker genes to assign biological identities to each group.

Clustering cells based on their expression profiles is a fundamental step, as it organizes the data into

meaningful units that can be further analyzed. This organization allows researchers to explore specific cell types within complex tissues and understand how they interact with each other.

Another powerful analysis is **trajectory inference**, which helps map the developmental or disease progression paths of cells. By examining the proportion of **spliced versus unspliced reads**, we can infer where cells lie along a trajectory, such as the differentiation of stem cells into mature cell types or the degeneration of neurons in neurodegenerative diseases.

Differential expression analysis is central to understanding how gene activity differs between conditions, such as healthy versus diseased states. This approach can reveal the genes driving pathological changes and can be linked to phenotypic shifts within specific cell types.

Beyond differential expression, scRNA-seq can be used to assess **cellular composition differences**, such as the loss of neurons in Alzheimer's disease. This helps identify not just which genes are altered but also how disease affects the overall structure of the tissue at a cellular level.

The breadth of questions that scRNA-seq can address—from basic cell-type identification to complex trajectory analyses—underscores its power in unraveling the intricate dynamics of cellular function and disease progression. As the field continues to evolve, these methods will only grow in their ability to provide deeper insights into the cellular architecture of health and disease.

1:15:30 Cell-Projected Phenotypes

Traditional single-cell analyses often categorize cells into discrete types—such as microglia, astrocytes, or specific neuronal subtypes—based on their expression patterns. However, an even more granular approach involves studying the **transcriptional neighborhoods** of individual cells, moving beyond rigid cell-type definitions. Instead of labeling an entire cluster as, for example, excitatory neurons of cortical layer 5, this method examines each cell's transcriptional proximity to others, creating a continuum of expression states within a broader cell type.

This approach is not about spatial proximity but about **transcriptional similarity**, reflecting how closely related the gene expression profiles of different cells are. By projecting cells into a lower-dimensional space that captures these relationships, we can reveal fine-scale variations that traditional methods might overlook. Given our dataset of 430 individuals, we can then overlay these **phenotypes** onto the transcriptional landscape, effectively projecting Alzheimer's versus control states directly onto individual cells.

What emerges is a fascinating pattern: some cells and their surrounding neighborhoods appear distinctly Alzheimer's-like, while others are more control-like. This discovery suggests that within cell types and subtypes, there are further subdivisions with nuanced roles in disease—subpopulations of cells that diverge from the norm in ways that correspond with disease progression.

This methodology allows us to annotate these neighborhoods according to their disease state, identifying areas within a cell population that are more **Alzheimer's-like** or **control-like**. Remarkably, this approach can be applied at the level of individual patients. By examining an individual's cells without prior knowledge of their clinical phenotype, we can assess whether their microglia, for example, exhibit more Alzheimer's-like or control-like transcriptional states. In some cases, the same individual may have microglia that are Alzheimer's-like but astrocytes that are control-like, highlighting the variability within a single brain.

This high-resolution analysis redefines the concept of phenotype from a static diagnosis to a dynamic attribute measurable at the level of individual cells. This allows for a more personalized and precise characterization of disease states within a patient, providing a powerful tool to correlate cellular phenotypes with other clinical and molecular measurements.

The implications are vast: by using single-cell analyses to dissect these transcriptional landscapes, we can refine our understanding of disease pathology, move beyond simple diagnostic labels, and potentially identify new therapeutic targets. This approach transforms how we think about phenotype—expanding it from a broad label to a detailed map that captures the unique cellular states contributing to disease.

Needless to say, the potential of this field is immense, and the direction it is heading excites me greatly. As we continue to push the boundaries of what single-cell analysis can achieve, I'm eager to hear your project ideas and explore how we can collectively contribute to this rapidly evolving area of study.

Lecture 4 - Alignment

Video: Lecture04 - Sequence Alignment - MLCB24

Slides: Lecture04_DynamicProgramming-BLAST.pdf

0:00 Intro: Aligning Sequential Datasets/Models

Welcome to today's lecture on **sequential pattern matching**, where we'll explore how to align sequences of data across various domains. Whether it's aligning a string of actions, sequences of text, or matching gene sequences between genomes, the core problem is the same: how do we compare and align **sequential data**?

We encounter this challenge in many contexts:

- **Language translation:** Aligning text in one language with its counterpart in another.
- **Comparative genomics:** Matching genes across different genomes to understand evolutionary relationships.
- **Model states over time:** Aligning states of a model that represents the world across different time points.

The techniques we'll discuss are foundational in computation and broadly applicable across biology and beyond. We'll begin with **sequence alignment** as the basis for these techniques, focusing on **dynamic programming**, a key computational strategy that breaks down larger problems into manageable subproblems and efficiently combines them.

We will explore:

1. **Comparative Genomics and Evolution:** How these foundational techniques apply to evolutionary biology, using sequence alignment to study genetic conservation and mutation across species.
2. **Dynamic Programming Concepts:** We'll start with a simple example—computing Fibonacci numbers—to introduce how dynamic programming reuses computations, reducing redundant calculations. This principle extends directly to sequence alignment.
3. **Sequence Alignment with Dynamic Programming:** By building an alignment matrix, we can compare sequences by aligning prefixes of two sequences step-by-step. We'll use this matrix to efficiently construct the optimal alignment path, drastically reducing the runtime from exponential to quadratic.
4. **Advanced Alignment Techniques:** We will explore local alignment, linear time alignment, and linear space alignment, building on our understanding of sequence alignment and dynamic programming.
5. **Hashing for Rapid Lookup:** Beyond dynamic programming, we'll introduce hashing—encoding sequential patterns into unique codes for rapid search. This concept extends to content-based search, where the goal is to find similar documents not by title or keywords but by their actual content, using techniques like latent embeddings and context-sensitive hashing.
6. **Probabilistic Algorithms and Expected Runtime:** We'll delve into the probabilistic foundations of these algorithms, culminating in the BLAST (Basic Local Alignment and Search Tool) algorithm, a highly efficient method widely used in genomics with tens of thousands of citations.
7. **Sequential Data Models:** We'll explore the broader family of models used to handle sequential data, including Hidden Markov Models, Gaussian Mixture Models, and neural network architectures like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks.
8. **Transformers:** We'll discuss how Transformers have revolutionized sequence modeling with explicit positional encoding, allowing the model to maintain context over time without the traditional limitations of RNNs.

Understanding these models requires a grasp of how they handle **contextual information** and **time series data**, managing memory across sequences to preserve or forget information as needed. This sets the stage for exploring Transformers, which leverage sophisticated mechanisms to understand relationships across data points in a sequence.

We'll begin with **comparative genomics** and use the lens of evolution to explore genomes, setting the groundwork for understanding how sequential data can be aligned and interpreted in various applications.

9:31 Comparative Genomics & Evolution

Comparative genomics is central to understanding the diversity of life on Earth. It allows us to compare genomes across different species, tracing back to their common ancestors. This field opens a window into the evolutionary processes that shaped the vast array of life forms we see today.

Consider the interconnectedness of all life: the wood of the desk you sit at comes from trees, our distant cousins; the viruses and bacteria we fight every day are also part of our extended evolutionary family; even the fungi growing in damp places, like the gym locker room, share a common lineage with us. Every life form on

this planet is part of a continuous, unbroken chain of existence stretching back over **three billion years**.

Comparative genomics enables us to compare these life forms by aligning their genomes, shedding light on how species have diverged from one another over time. This comparison is not just about observing genetic differences but understanding the evolutionary forces that led to the emergence of new species. It also allows us to appreciate the vast tapestry of life on Earth, which, while beautiful, is also marked by the **brutality of natural selection**. Species often go extinct, either due to competition that eliminates their niche or catastrophic events like asteroid impacts that reset the evolutionary stage.

Genome alignment is the process of mapping chunks of DNA from one species to corresponding sequences in another. This approach is similar to aligning a series of frames in a movie with a corresponding textual description or translating text between languages. In evolutionary alignment, we take sequences from different species that descended from a common ancestor and infer the sequence of events that led to their divergence.

Through genome-wide alignments, comparative genomics helps us identify **functional elements** in the genome—regions that are preserved across species due to their biological importance. For example, when comparing the genomes of humans, dogs, mice, rats, chickens, and fish, we observe that some regions are highly conserved, indicating their crucial role in cellular functions. These regions often correspond to **protein-coding exons**, the parts of genes that encode proteins.

Interestingly, conservation is not limited to protein-coding regions. Some conserved regions fall between exons and are not annotated as coding. These non-coding regions, often overlooked, may have critical regulatory functions. For example, between exons 5 and 6 of the DBH gene, there is a conserved block that is not part of the protein-coding sequence. This raises the question: what role does this conserved, non-coding sequence play?

The conservation pattern suggests that these regions are under strong **purifying selection**—evolutionary pressure to maintain their sequence integrity. Some conserved elements may have evolved relatively recently, becoming important after diverging from common ancestors, such as chickens or fish. Evolution is not a static perfection from 3.5 billion years ago; it is a dynamic process full of **mutations and adaptations** that sometimes lead to beneficial changes.

One intriguing possibility is that conserved non-coding regions function as **regulatory elements**, controlling when and where specific genes are expressed. Only about **1.5% of the human genome** encodes proteins, yet estimates suggest that **7% to 20%** of the genome is under some form of selection, indicating potential functionality. Moreover, a vast majority of disease-causing mutations—around **93%**—are found in non-coding regions, further emphasizing the importance of these sequences.

Non-coding regions may play roles that are critical yet not fully understood. They could be mistakenly annotated as non-coding, or they might function in ways that are more subtle and complex than simply encoding proteins. Misannotations can happen, as scientists work tirelessly to refine the genome's reference annotations, uncovering new exons and even new genes that were previously missed.

As we map genomes meticulously, aligning each nucleotide across species, we uncover **islands of perfect conservation**—small stretches of DNA that have remained unchanged for millions of years. These sequences often correspond to **regulatory motifs**, such as binding sites for transcription factors that control gene expression. For instance, in yeast, conserved sequences correspond to sites where regulators of glucose and galactose metabolism bind, controlling how the organism adapts to its environment.

By reading the book of evolution, we can infer the **functional elements** of genomes. Each conserved sequence tells a story of selection, adaptation, and survival, offering insights into the fundamental workings of life. As we continue to align sequences and refine our understanding, comparative genomics will remain a cornerstone of modern biology, guiding us through the complex relationships that define the living world.

29:01 Computation Re-use, Dynamic Programming

To align genomic sequences effectively, we need to model the evolutionary process that has shaped them. This involves defining a set of basic operations—mutations, insertions, and deletions—that describe how one sequence can be transformed into another. By simulating these evolutionary operations, we can work backward to infer the most likely sequence of events that led to the observed genetic differences.

The first evolutionary operation is **mutation**, where one letter is changed into another. This happens because the DNA polymerase, while highly accurate, is not infallible. As it replicates DNA during cell division, it occasionally makes mistakes, like a medieval monk painstakingly copying ancient manuscripts and sometimes miswriting a letter. Similarly, **deletions** occur when the polymerase skips over a nucleotide, failing to copy it.

Insertions happen when the polymerase adds an extra nucleotide, perhaps losing track due to repetitive sequences, like a long string of identical bases (e.g., “AAAAAA”) that can cause slippage.

To understand the evolution of sequences, we simulate this forward process of mutations, insertions, and deletions to see how one sequence can evolve into another. The inverse process—inferring how two sequences have diverged—is the challenge we tackle with **sequence alignment**. We formalize the problem computationally by defining these evolutionary operations and establishing an **optimality criterion**—a way to evaluate the best alignment.

For instance, when aligning the genomes of human and mouse, which diverged around 60 million years ago, we want to find the minimum number of operations needed to transform one genome into the other. This requires a reversible model that can infer the likely sequence of events from a common ancestor, accounting for both the forward and backward evolutionary paths.

To refine our model, we introduce the concept of **costs** associated with each operation. Not all evolutionary changes are equally likely; thus, each operation has a different cost based on its frequency in nature. For example, mutations might be more common and therefore cheaper than insertions or deletions. We assign these costs using a **probabilistic interpretation**: rarer events have higher costs, computed as the negative logarithm of their probability, making rare events like large deletions more costly than common point mutations.

With our cost model defined, the next step is to develop an algorithm that finds the optimal alignment between two sequences. We seek a balance between biological relevance and computational feasibility. On one hand, we want our model to capture the nuances of real evolutionary processes; on the other, we need algorithms that are efficient and manageable.

One way to simplify the alignment problem is to look for the **longest common substring** between two sequences, which involves finding the longest matching sequence without allowing insertions or deletions. For example, given two sequences, we might find "TCA" as a matching substring. To find such substrings, we can fix one sequence and slide the other across it, checking for matches. The computational cost of this approach is relatively high, but manageable compared to the exponential possibilities of general alignments.

However, real biological sequences often have insertions and deletions, so we expand our approach to find the **longest common subsequence**, allowing gaps in the alignment. For instance, if the sequences are “AAGT” and “TCA,” the longest common subsequence might include alignments like “AAG-TC-A,” reflecting possible evolutionary operations such as insertions, deletions, and mutations.

Finally, the challenge extends to assigning appropriate penalties for specific substitutions, which reflect the structural similarity between nucleotides or amino acids. For instance, purines (A and G) are often substituted for each other more frequently than purines are substituted for pyrimidines (C and T), reflecting their structural similarities. These substitution patterns are encoded in **scoring matrices** that help determine the most plausible evolutionary paths between sequences, integrating both mutation frequency and the structural context.

Dynamic programming is the key to efficiently searching through the vast space of possible alignments. Instead of evaluating each potential alignment individually, which would take exponential time, dynamic programming breaks the problem into manageable subproblems, reusing previously computed solutions to build up the overall alignment in polynomial time. This approach transforms what would be a computationally intractable problem into one that can be solved systematically, revealing the evolutionary history encoded within the genomes.

46:53 Dynamic Programming Principles and Fibonacci

Dynamic programming is a powerful technique that transforms **exponential runtimes into polynomial runtimes** by avoiding repeated work. It achieves this by storing and reusing previously computed results. To illustrate this concept, we start with a classic example: calculating Fibonacci numbers.

Fibonacci numbers are a sequence found abundantly in nature, such as in the growth patterns of shells, the arrangement of leaves on a stem, and the spirals of sunflowers. The sequence begins with 1, 1, and each subsequent number is the sum of the previous two: 1,1,2,3,5,8,13,21,1, 1, 2, 3, 5, 8, 13, 21,1,1,2,3,5,8,13,21, and so forth. This recursive pattern reflects a natural process of growth where each stage builds upon the sum of the two preceding stages.

To compute Fibonacci numbers, we can define them recursively in a simple Python function:

def fibonacci(n):

```
if n == 1 or n == 2:
    return 1
else:
    return fibonacci(n - 1) + fibonacci(n - 2)
```

This recursive approach, while straightforward, is highly inefficient. When you call `fibonacci(6)`, it triggers calls to `fibonacci(5)` and `fibonacci(4)`, and each of those calls trigger further calls. This creates a tree of function calls, where the same Fibonacci calculations are repeated multiple times, particularly for lower values like `fibonacci(2)` and `fibonacci(3)`. The number of operations doubles with each incremental step, leading to an **exponential growth in computation**.

The recursive approach illustrates the inefficiency inherent in top-down computation: each subproblem is recalculated many times, leading to an enormous waste of computational resources.

Dynamic Programming: Bottom-Up Approach

To avoid this inefficiency, dynamic programming employs a **bottom-up approach**. Instead of starting at the top and working down, you start at the bottom and build up. When asked to compute `fibonacci(100)`, you don't just compute that value—you compute all Fibonacci numbers from 0 to 100, storing each in a table along the way. This approach ensures that every time you need a Fibonacci value, it's already available in the table, eliminating the need to recompute it.

Here's how the bottom-up method works: you start by initializing the first two values, 1,11, 11,1. From there, you compute each subsequent value by summing the two preceding values and storing the result in the table. When you need `fibonacci(100)`, you simply look it up in the table—it's already been computed.

This approach drastically reduces the time complexity. Instead of recalculating `fibonacci(2)` multiple times, you calculate it once, and then **reuse it** every time it's needed. The time complexity drops from exponential to linear, making it much faster.

Key Principles of Dynamic Programming

1. **Identifying Subproblems**: Dynamic programming reveals identical subproblems within the larger problem. By identifying these subproblems, you can solve each only once.
2. **Ordering Computation**: Computation is ordered systematically so that when a subproblem's result is needed, it has already been computed and stored.
3. **Table Filling**: Results are stored in a table, allowing for constant-time lookup instead of recalculating values.
4. **Bottom-Up Approach**: Larger problems are expressed in terms of their smaller subproblems, ensuring that each subproblem is solved once and only once.

The **top-down recursive approach** is slow because the results of small subproblems aren't saved—they're recomputed each time they're needed. In contrast, the **bottom-up dynamic programming approach** systematically fills in a table of subproblems, ensuring that each value is available when required. This shift from recomputation to reuse is the essence of dynamic programming and is what makes it so powerful in transforming intractable problems into manageable ones.

This principle doesn't just apply to Fibonacci numbers; it extends to other computational problems, such as **sequence alignment** in bioinformatics, where dynamic programming techniques help efficiently align sequences by reusing results of smaller alignment tasks. The power of dynamic programming lies in its ability to simplify complex problems by breaking them down into reusable components, making it an essential tool in algorithm design.

51:36 Alignment Matrix, Paths, Traceback, 2^N-vs-N^2

When aligning two sequences, each containing n and m nucleotides respectively, the goal is to efficiently calculate the best alignment scores without recalculating each possible alignment from scratch. This is achieved through **dynamic programming**, which involves systematically storing and reusing scores in an **alignment matrix**.

The alignment matrix is an n×mn \times mn×m grid where each cell (i,j) represents the **score of aligning the first iii nucleotides** of sequence 1 with the first j nucleotides of sequence 2. The idea is to compute these scores iteratively and store them so that each sub-alignment can be reused when needed, rather than starting

from scratch each time.

To fill this matrix, we compute the score at each cell based on three potential previous alignments: moving **diagonally** (representing a match or mismatch between the current nucleotides), moving **vertically** (representing a gap in sequence 1), or moving **horizontally** (representing a gap in sequence 2). The score at each position (i,j) is the **maximum** of these three values, plus the specific cost associated with the operation—whether it's a match, mismatch, or gap penalty.

A crucial aspect of this approach is **traceback**. To reconstruct the optimal alignment, we don't just calculate scores; we also store **pointers** or **arrows** in each matrix cell that indicate which of the three possible moves (diagonal, up, or left) led to the current score. These pointers allow us to trace the optimal path from the bottom-right corner of the matrix (where the final score is located) back to the top-left corner, effectively reconstructing the sequence alignment step-by-step.

The matrix can be filled in several ways—row by row, column by column, or diagonally—as long as each cell is computed after the cells it depends on. By progressively building the matrix from simpler sub-problems (aligning smaller sections of the sequences), we efficiently compute the overall alignment score.

The efficiency of this approach is striking. Although there are 2^N potential paths through the matrix (since each position can involve a decision to align, insert, or delete), dynamic programming reduces the problem to filling a matrix of size N×M. This means the time complexity is only O(N×M), making it feasible to handle relatively large sequences. Each cell computation takes constant time, leveraging precomputed values from adjacent cells, thus transforming an exponentially complex problem into a quadratic one.

Once the matrix is filled, the real magic happens in the **traceback** process. Starting from the bottom-right, we follow the stored arrows backward, constructing the optimal alignment. Each step corresponds to a specific alignment action: matching two characters, introducing a gap in one sequence, or aligning mismatched characters. This allows us to directly visualize how the sequences align, where gaps occur, and which regions match perfectly.

A fun way to visualize and interact with this process is by implementing it in tools like Excel. By defining local scores and dependencies, and allowing dynamic updates based on gap or mismatch penalties, one can see in real time how the optimal alignment path changes. This hands-on approach makes the abstract concepts of dynamic programming and sequence alignment more tangible.

This matrix-based alignment approach is not just limited to nucleotide sequences. It applies broadly to any scenario where two sequences need to be aligned, such as comparing protein sequences, time-series data, or even aligning text. The structured, iterative methodology of dynamic programming makes it a powerful tool for tackling complex alignment problems efficiently, uncovering the best path through a seemingly vast space of possibilities.

1:04:45 Local Alignment, Linear-Time, Linear Space

Dynamic programming's strength lies in its ability to transform the complex problem of sequence alignment into a manageable process. By conceptualizing alignments as paths through a matrix, we can optimize the search for the best alignment, but there's more to explore when it comes to efficiency and precision, especially in biological contexts.

Understanding the Alignment Space

Consider an alignment path that moves all the way down one axis and then across the other, essentially deleting and adding an entire sequence. This kind of alignment is clearly suboptimal as it incurs a massive penalty for insertions and deletions—far from reflecting any meaningful biological relationship between sequences. The question then becomes: **How far off the main diagonal should we allow an alignment to go before it becomes unrealistic?**

If an alignment drifts too far from the diagonal, the penalties from excessive gaps will outweigh any potential gains from matches. To handle this, we can apply a **heuristic approach**: restricting the search to a bounded region around the diagonal. This concept, known as **bounded alignment**, allows us to constrain the search space, focusing only on reasonable alignments that remain close to the diagonal.

This approach is not guaranteed to be globally optimal, as some rare cases might exist where the optimal path lies outside the bounded region. However, for practical purposes, these cases are negligible, and the computational efficiency gained is significant.

Optimizing Space Requirements

Storing the entire alignment matrix requires quadratic space, as each cell stores a score and pointers that guide the traceback process. But what if we only care about finding the optimal score at the matrix's endpoint? In that case, storing the whole matrix is unnecessary. We only need to keep track of the scores for the current and previous rows or columns, reducing space complexity from O(n2)O(n^2)O(n2) to O(n)O(n)O(n).

This optimization allows us to compute the alignment score with just linear space. However, if we also need to reconstruct the alignment path itself, the problem becomes more complex.

Finding the Traceback Path with Limited Space

Even with the space optimization, we can still find the optimal alignment path using a clever recursive strategy. Here's how:

1. **Compute Forward and Reverse Scores**: First, calculate alignment scores from the left to the right, storing the necessary values as you go. Next, compute scores in the reverse direction, from right to left, using the same approach.
2. **Identify the Midpoint**: By summing the scores from the forward and reverse calculations, you can identify the midpoint that provides the best alignment. This midpoint essentially divides the alignment into two sub-alignments, each of which can be processed independently.
3. **Recursive Alignment**: Use the identified midpoint to recursively align the two halves. This recursive divide-and-conquer strategy continues until the complete alignment path is reconstructed.

This method maintains the linear space requirement for each recursive step, allowing you to handle large-scale alignments efficiently.

Local Alignment: Finding High-Scoring Sub-Regions

Sometimes, instead of aligning entire sequences, we are more interested in finding **local alignments**—specific regions within the sequences that align particularly well. Local alignment is crucial when evolutionary rearrangements, inversions, or duplications have occurred, which may disrupt the continuity of a global alignment.

To perform local alignment, we adjust the dynamic programming algorithm to start and end at any position within the matrix rather than being constrained to the matrix's corners. This flexibility captures high-scoring sub-alignments that reflect meaningful biological relationships without requiring the sequences to align perfectly from start to end.

The algorithm's adaptation involves modifying the scoring rules:

- Allow the alignment to start anywhere, adding an additional zero to the options for initial alignment.
- Permit it to end at any position where the score is highest, without requiring it to reach the bottom-right corner.
- Optionally, modify the algorithm to avoid penalties for gaps at the ends of sequences, accommodating incomplete data.

Fast Local Alignment through Hashing

To further accelerate local alignment, particularly when searching for highly conserved motifs or repeated patterns, we can use **hashing**. This technique involves:

- Identifying "seeds" or short segments of exact matches between the sequences.
- Using these seeds to anchor the alignment and then extending outward, rapidly identifying regions of high similarity.

Hashing dramatically speeds up the process by focusing computational resources on promising regions, bypassing the exhaustive pairwise comparison of every possible alignment. It's akin to scanning a video for repeated actions and using those as key points for alignment rather than analyzing every frame in detail.

Conclusion

Local alignment and optimized alignment strategies represent crucial advancements in computational biology, enabling the discovery of meaningful relationships between sequences even in the presence of evolutionary rearrangements and other complex changes. By balancing heuristic approaches, space optimization, and efficient recursive methods, dynamic programming extends far beyond global sequence alignment, offering tools that are indispensable for understanding the genetic code and the evolutionary history encoded within.

1:14:35 Hashing, BLAST, Inexact Matching, and PSI-BLAST

One of the most powerful tools in computational biology is BLAST (Basic Local Alignment Search Tool). Despite its humble name, BLAST has revolutionized the way we compare sequences by employing hashing and other sophisticated techniques to perform local alignments at unprecedented speeds. Let's explore the principles behind this transformative approach.

Hashing: The Foundation of Fast Sequence Matching

The key to rapid sequence alignment lies in a concept borrowed from computer science: hashing. Hashing transforms sequences into numerical representations, allowing for incredibly fast lookups. Imagine you want to find a specific sequence—31415, part of the well-known sequence of pi—within a much longer sequence. Instead of comparing each segment of the long sequence character-by-character, you can treat 31415 as a single number, making the comparison as simple as checking if two numbers are equal.

However, computing such large numbers can be computationally expensive, especially if each calculation involves multiple digits. To make this more efficient, the algorithm uses a rolling hash: it updates the number incrementally by subtracting the contribution of the first digit, shifting the remaining digits, and adding the new digit. This approach reduces the time complexity of each update to constant time.

To avoid the pitfalls of handling extremely large numbers, we apply a modulus operation, which keeps the numbers manageable. This effectively compresses the sequence into a smaller range, allowing fast comparisons. The trade-off is a small chance of collisions, where different sequences yield the same hash. However, these collisions are statistically rare, especially when the hashing space is sufficiently large.

BLAST: Extending Hashing to Biological Sequences

BLAST builds upon these principles, using hashing to rapidly identify matching sequences in a large database. The algorithm breaks the input sequence into small chunks, or "seeds," and searches for these seeds within the database. Once a seed match is found, BLAST extends the alignment outward from the seed, creating a local alignment between the sequences.

The real power of BLAST comes from its ability to handle not just exact matches but also **inexact matches**. BLAST doesn't just look for identical sequences; it also considers close matches, where amino acids or nucleotides are similar but not identical. For instance, it searches for sequences that match a given score threshold, allowing for substitutions that are biologically plausible based on known evolutionary patterns.

In this way, BLAST can find alignments even when there is no perfect match, providing insights into evolutionary relationships and functional similarities between sequences.

PSI-BLAST: Iterative Search for Deeper Connections

PSI-BLAST (Position-Specific Iterated BLAST) takes this approach further by allowing iterative searches. It builds a position-specific scoring matrix (PSSM) based on the sequences found in the initial BLAST search. This matrix captures the likelihood of each amino acid occurring at each position, accounting for sequence variability seen in evolution.

Subsequent searches use this PSSM to refine the search, identifying sequences that are even more distantly related. PSI-BLAST is particularly powerful for detecting remote homologs—sequences that share common ancestry but have diverged significantly over time.

Putting It All Together: Probabilistic Foundations and Efficiency

The probabilistic foundations of BLAST and its extensions are grounded in comparing the likelihood of matching sequences under two models: a related model (where sequences share a common ancestor) and an unrelated model (random sequences). The ratio of these likelihoods informs the scoring of matches, weighting substitutions by their evolutionary plausibility.

Key takeaways include:

- **Hashing** transforms sequences into manageable numbers, allowing rapid matching with minimal computation.
- **BLAST** uses seeds and extensions to perform local alignments, accommodating both exact and inexact matches.
- **PSI-BLAST** builds on BLAST by iteratively refining searches with position-specific scoring, uncovering deeper evolutionary links.
- **Probabilistic scoring** ensures that alignments reflect biologically relevant relationships, not just raw similarity.

These algorithms have become the backbone of comparative genomics, enabling scientists to explore the vast

landscape of genetic data efficiently. They highlight the power of computational techniques in deciphering the complex web of life, from individual genes to entire genomes.

Lecture 5 - Epigenomics - HMMs

Video: Lecture 5 - Epigenomics, HMMs - MLCB24

Slides: Lecture05_Epigenomics_HMMs.pdf

0:00 Logistics

Introduction to Epigenomics and Hidden Markov Models. Welcome, everyone. Today, we're going to focus on **epigenomics** and use it as a gateway to introduce **Hidden Markov Models (HMMs)**, a type of **sequential learning model** that we touched on briefly last time. Here's the **progression of topics** we've covered so far: **Gene expression analysis:** We explored how to understand **gene expression patterns**, which is crucial in many areas of study. We used this to introduce **clustering**, **classification**, and **Gaussian mixture models**. Then, we delved into **single-cell analysis**, understanding single-cell data in a foundational way.

The problem set (P-set) you're working on tackles **epigenomics**, **single-cell analysis**, and **gene expression analysis**. This P-set is a hands-on way to apply what we've learned so far. The goal of the problem set is to **teach and reinforce** your learning, not to make it unnecessarily hard. Some steps may seem complex, but we're here to walk through them together. There were questions about scheduling office hours around the holiday. To accommodate, we'll hold **office hours** and likely record them for those who cannot attend live. Let's poll for availability: Who's available today? (Raising hands for different times of day). Alternatively, we could schedule the office hours on **Monday**. (Show of hands for availability on Monday). It seems like Monday is the preferred day. We'll finalize this with a **doodle poll** and confirm whether the office hours will be at **3 PM, 4 PM**, or **5 PM** on Monday. As you work on the problem set over the weekend, don't hesitate to ask for **help** from **ChatGPT**—not for the answers, but to better understand the **code** and **concepts**. The goal is for you to gain clarity and confidence in the material. Next week, we'll transition into **Regulatory Genomics** and **Regulatory Networks**, but today, let's dive deep into **epigenomics**.

4:00 Lecture Overview

Today, we will be exploring the fascinating world of **epigenomics**—the study of how **diverse cell types** can emerge from the same underlying genome. Here's an outline of what we'll cover:

1. **Epigenomics Overview**: We will begin with a broad look at what **epigenomics** is and its role in cellular diversity.
2. **Chromatin Modifications**: We'll discuss the specific **chromatin modifications** that play a critical role in regulating gene expression.
3. **Technologies in Epigenomics**: A look at the **experimental technologies** used to study epigenomics, setting the foundation for how we obtain and process data.
4. **Primary Data Processing**: After collecting experimental data, how do we process it? We'll talk about **signal processing** and introduce the **Burrows-Wheeler Transform (BWT)**, an efficient approach for **read mapping** in epigenomic data. I'll also share an interesting connection: I recently met **Langmead**, the original author of the BWT algorithm, who visited MIT.
5. **Peak Calling**: Once we map the data, how do we find significant regions or peaks in epigenomic marks? This is crucial for interpreting the data.
6. **Combining Multiple Epigenomic Marks**: It's not enough to study just one mark in isolation. We'll discuss how to combine multiple **epigenomic marks** to characterize **chromatin states**, revealing patterns of regulation in the genome.
7. **Hidden Markov Models (HMMs)**: We will introduce **Hidden Markov Models**, starting with the basics, and then extend this to a **multivariate HMM** for epigenomic data.
8. **Model Complexity**: A key challenge in using HMMs is deciding the **model complexity**—how many parameters and states are sufficient to describe the system. We'll address how to make these decisions.
9. **Learning Chromatin States Jointly**: We'll explore how to **learn chromatin states** not only across multiple epigenomic marks but also across **multiple cell types**, giving us a holistic view of chromatin dynamics.

Ready to dive in? Let's begin with **epigenomics** and explore how we go from a single genome to the

extraordinary cellular diversity found in different cell types.

5:34 Introduction to Epigenomics

The **diversity of cell types** in our body is remarkable. In the brain alone, we find a vast range of cell types within the **neocortex**, and even more diversity in the **subcortical regions**—with both **excitatory** and **inhibitory neuron subtypes**. Similarly, the **immune system** displays a vast variety of **blood cells**, not just the red blood cells but an extensive range of **white blood cells** with specialized roles in immune function. A small section of **skin** contains **hair follicles**, **innervations**, **sensory organs**, **epithelial cells**, and **veins**, all working together, yet originating from the same genetic material.

Despite the **same underlying genome** in all these cells, their **phenotypes** exhibit extraordinary diversity. This diversity is made possible by the fact that cells use **different subsets** of the genome, regulated by **epigenomics**.

The genome in each of our cells is packed extremely **compactly**. If you were to take the DNA from a single cell, it would measure **2 meters long** when stretched end-to-end. Given that our body consists of **trillions of cells**, the total length of DNA would stretch from Earth to **Jupiter**, and back, multiple times.

This incredible amount of DNA is compacted within each cell by **structural mechanisms**, primarily involving **nucleosomes**. Each nucleosome contains around **200 nucleotides** of DNA wrapped around **histone proteins**. These histones are decorated with **chemical modifications** that signal to the cell whether a region of DNA is **active**, **repressed**, **poised**, or **transcribed**.

Every nucleosome consists of **eight histone proteins**, which, in addition to compacting DNA, play a **functional role** by encoding **epigenomic memory**. This memory allows cells to maintain their identity throughout cell divisions, ensuring that neurons don't suddenly start functioning like **immune cells**.

Thus, the **DNA packaging** is not just structural but also **functional**—enabling the cell to access the DNA it needs for its specific role. During development and through cell divisions, **epigenomic factors** help maintain the specific **chromatin state** needed for each cell type. Additionally, DNA **accessibility** and **chromatin state** are critical, as **DNA interactions** across different segments also contribute to gene regulation.

In summary, **epigenomics** is the mechanism that enables the vast cellular diversity from a single genome, controlling which parts of the DNA are accessible or functional in each cell type.

9:57 Three types of Epigenomic Modifications

Three Types of Epigenomic Modifications

The structural **compaction** of DNA has functional implications. The more **compact** a region of DNA is, the harder it is to access. For example, highly compacted regions in neurons might not be useful for neuron function but could be important for heart, liver, or lung cells. This **compaction** plays a critical role in controlling which genes are available for a cell type to use.

The second aspect is **DNA accessibility**, which goes beyond general compaction. It involves the **local positioning** of nucleosomes. Even in less compact regions, nucleosomes can shift or roll back and forth to open specific segments of DNA, allowing **regulatory factors** to bind to the exposed sequences. This local accessibility determines whether a regulatory protein can engage with that particular part of the DNA.

The third key modification is **DNA methylation**. This occurs directly on the DNA sequence, specifically at **cytosine bases** (C in the ACGT code). A **methyl group** can be added to cytosine, creating a **methyl-C**. This alteration can influence whether a **transcription factor** (TF) can bind to the DNA. Transcription factors don't read the DNA by pulling it apart; instead, they interact with it from the side, feeling the atoms along the major groove of the DNA. If a **methyl group** is present, it can block the transcription factor from recognizing the cytosine, preventing it from binding. Conversely, methylation might **enable binding** for some regulators that specifically recognize **methylated cytosines**. Thus, **DNA methylation** acts as a mechanism for changing how transcription factors interact with the DNA, modifying gene expression.

In addition to **DNA methylation** and **accessibility**, there are also **histone modifications**. Each **nucleosome** consists of eight **histone proteins**, with the most common version being made up of two copies each of **H2A, H2B, H3, and H4**. These histones can undergo various modifications, including the replacement of regular histones with **histone variants** (e.g., **H2A Z**). Beyond this, **post-translational modifications** can occur on the **tails** of the histone proteins. These tails serve as a **landing pad** for regulatory proteins that read the signals embedded in these modifications.

These **histone tail modifications** help determine whether a region of DNA will be interpreted as an **enhancer**,

promoter, **transcribed region**, or **repressed region**. For example, **methylation** on histones can indicate a repressed region, while **acetylation** loosens the chromatin, increasing accessibility and transcriptional activity. Some histone modifications have **informational effects**, while others have **biophysical impacts** that alter the physical **compactness** of the chromatin.

In summary, there are three main types of epigenomic modifications:

1. **Compaction and accessibility** of the chromatin, which affects whether certain regions are available for transcription.
2. **DNA methylation**, which alters how transcription factors bind to DNA.
3. **Histone modifications**, which involve chemical changes to the histone tails that regulate chromatin structure and gene expression.

These modifications provide a multilayered system for controlling the accessibility and interpretation of the genome, ensuring that the right genes are turned on or off in each specific cell type.

15:03 Q1: Non-standard modifications

Q1: Non-Standard Modifications

There was a question about lab-generated epigenomic modifications, specifically **oxidation** and **carbonation**. While I had mainly discussed **DNA methylation** and **acetylation** as common modifications, there are indeed many others, including **ubiquitination**, **sumoylation**, and some lab-generated or synthetic ones like **oxidation** and **carbonation**. These lab-generated modifications can be used experimentally to **mark specific regions** of the genome. However, once these modifications are added, they must be maintained, especially after **DNA replication**, because many of these modifications will be lost during the replication process.

This maintenance requires regulatory mechanisms that establish, read, and reapply these marks after each replication cycle. The nature of the **double helix structure** enables **semi-conservative replication**, where one strand of the DNA can serve as a template for the other. While the **DNA sequence** can easily be copied in this manner, the challenge lies in **maintaining the epigenomic modifications**. These marks need to be reestablished after replication by specialized regulatory proteins.

For instance, **histone marks** are part of the chromatin that wraps around DNA. During replication, these histone proteins might fall off or be partially retained, and the appropriate **histone modifications** need to be **reestablished** after the replication process. This process is similar during **RNA transcription**, where the **chromatin** must be opened up to allow the transcription machinery to access the DNA. Some epigenomic marks are **transcription-associated** or **co-transcriptional**, meaning they travel with the transcription machinery and are modified along the way.

Therefore, we need to consider both the **informational aspect** of the genome and the **biophysical aspects** when thinking about how epigenetic regulation is maintained during these processes.

17:20 Q2: Epigenetic inheritance

Q2: Epigenetic Inheritance

Many people often think of **epigenetics** as something that carries through generations, similar to genetic mutations, and thus associate it with **inheritance**. This has led to the term **"epimutation"**—referring to changes in gene expression or function that occur without altering the underlying DNA sequence. For instance, when an organism is exposed to extreme cold, radiation, or other environmental factors, there might not be changes to the DNA itself, but there could be changes in the **epigenome**. The natural question that arises from this is: are these epigenomic modifications **inherited** by future generations?

The answer, nearly all of the time, is **no**. Humans and other organisms actively **wipe out epigenomic states** between generations. When **sperm** and **egg cells** are formed, their epigenomic landscape is reset, making them very sperm-like and egg-like. For example, if I experience epigenomic modifications in my **brain cells**, it's not clear how these would be reflected in the sperm cells I produce.

When a **zygote** (the fertilized egg) forms, there's an additional **epigenomic wiping out** to ensure that the developing organism starts with a clean slate. This reset allows the zygote to rely on **primary sequence signals** encoded in the DNA itself, rather than epigenetic modifications. The egg, in particular, plays a crucial role here, as it is packed with **proteins and signals** that help establish the foundational epigenetic state of the zygote.

This resetting process happens **multiple times** during development. So, even if I acquire some epigenomic

change due to an experience—say, watching a violent movie—that modification would have to make its way into my sperm, survive the fertilization process, persist through the developmental stages of the zygote, and then manifest in the relevant cells (e.g., brain cells) of my child. As you can imagine, this is an incredibly complicated and unlikely process.

However, while this seems nearly impossible based on what we know, there are theoretical ways in which **epigenetic inheritance** might still occur. One possibility involves **small RNAs** associated with **epigenomic regulation**. These small RNAs could theoretically travel throughout the body, enter the germ line (the cells that give rise to sperm or eggs), and pass on to the next generation. They might even make their way through the **mature egg** and play a role in the developing embryo, continuing to influence the epigenome. While this is **possible**, it remains **extremely difficult** and rare.

The overall takeaway is that, while epigenetic modifications do occur, their inheritance across generations is generally wiped out, though rare mechanisms like **small RNA-based inheritance** may provide a potential pathway in certain cases.

20:29 Q3: Developmental memory establishment

Epigenetic memory during development is established by **regulatory elements** that were present in the previous cell type. The process begins with the **zygote**, which initially has only one cell type—because it is a single cell. As the first **cell division** occurs, one of these daughter cells will become **anterior**, while the other will become **posterior**. This distinction between cells can happen in different ways depending on the species.

In some species, this **marking** happens **randomly**, while in others it is influenced by the **position** of the cells inside the mother's body, inside the egg, or within the replication environment. Additionally, some species establish **gradients** of gene expression for various regulatory signals. These gradients help define the body plan by marking different regions of the developing organism.

For instance, an **anteroposterior gradient** is established, followed by two additional gradients that run in different directions. Specific regulatory regions, or **response elements**, react to different combinations of these gradients. For example, some regulatory regions might require **two-thirds** of one gradient and **one-third** of another, while others might need different proportions, such as **three-fifths** and **two-fifths**. These varying combinations result in distinct **bands of gene expression** and help to further **differentiate cell types**.

In **Drosophila** (fruit flies), these bands of gene expression are often referred to as **stripes**, and in **C. elegans**, each replication is highly programmed. The identity of each cell is precisely tracked, for example, a cell labeled **13A** will divide into **27B** and **27D**, and the fate of each subsequent cell is determined down to specific replication patterns. This highly detailed **cellular differentiation** process has evolved over time in numerous organisms to ensure the correct formation of various cell types.

This process of **breaking symmetry**—where cells that were once identical start to adopt distinct fates—is one of the most **elegant** and **miraculous** aspects of biology. It's incredibly complex, and as I mentioned, it's a feat that would challenge any engineer to replicate. Imagine creating a **self-replicating robot** that, using just three billion letters of code, not only builds itself but also makes real-time decisions about how to differentiate its parts. This intricacy is likely a reason it took so long to evolve from **unicellular** to **multicellular** organisms, as the rules governing **self-reprogramming** and differentiation needed to be perfected over billions of years of **evolution**.

Who else finds these processes amazing? The beauty and complexity of how life develops and diversifies truly demonstrates the extraordinary ingenuity of evolution.

23:25 Diversity of Histone modifications

Histone modifications play a dual role, contributing both **biophysically** and through **read/write mechanisms**. Histone tails can undergo a wide range of modifications, which makes histone modifications one of the **most complex and versatile** mechanisms of epigenetic regulation.

When we talk about histone modifications, we refer to specific **amino acids** in the histone tails that get modified. For example, you might encounter terms like **H3K4 methylation**. Here's how to break it down:

- **H3** refers to the **H3 histone protein**. Histones are classified into **H2A**, **H2B**, **H3**, and **H4**.
- **K4** refers to **lysine** at position 4 on the H3 histone tail.
- **Me3** refers to the addition of **three methyl groups** (trimethylation) to the lysine.

So, **H3K4me3** means that the **lysine at position 4** on the **H3 histone** has been **trimethylated**. These shorthand notations will recur frequently, and it's important to understand how they denote specific

modifications.

Histone modifications do not act in isolation. For example, **H3K4me3** might mean one thing when it's present alone, but its functional impact can change significantly when it's found in combination with other modifications. Histones can be modified through processes like **methylation**, **phosphorylation**, **acetylation**, and others, creating a vast array of regulatory possibilities.

In addition to histone modifications, there are other factors like **DNA methylation**, **nucleosome positioning**, and **DNA accessibility** that also regulate gene expression. This interplay between different epigenetic mechanisms can be thought of as a **territorial struggle** on the DNA. **Regulatory factors** fight for access to DNA by pushing aside nucleosomes. Some regulators, known as **pioneer transcription factors**, are strong enough to **displace nucleosomes** and gain access, while others will only bind if the nucleosomes are already displaced.

This balance of access involves both **biophysical forces** and the **read/write system**, where proteins can recognize specific histone modifications and alter the chromatin landscape. These modifications can result in **loosening** or **tightening** of the chromatin structure, preparing it for processes like **RNA polymerase binding** to initiate transcription.

Histone modifications also correlate with specific regions of the genome:

- **Enhancers** are associated with marks like **H3K27ac** and **H3K4me1**.
- **Promoters** are linked to **H3K4me3** and **H3K9ac**.
- **Transcribed regions** have multiple modifications, such as **H3K36me3**, **H3K79me2**, and **H4K20me1**.

Repression marks are classified into three types:

1. **H3K27me3**: Indicates facultative repression (repression that can be reversed in different cell types).
2. **H3K9me3**: Associated with **stable heterochromatin** repression, forming regions that are highly compact and difficult to reactivate.
3. **DNA methylation**: Often marks **repressive regions** in regulatory sequences. Interestingly, in transcribed regions, DNA methylation can correlate with **more expression**, potentially acting as a marker for preventing reinitiation of transcription.

Untranslated regions (UTRs) are also important to consider. A **5' UTR** exists between the **start of transcription** and the **start of translation** (the **ATG** codon), while a **3' UTR** lies between the **stop codon** and the **end of transcription**. These regions are critical for stabilizing the mRNA transcript, with processes like **capping** and **polyadenylation** helping to ensure transcript stability.

29:00 Methylation (Bisulfite) and DNase Profiling

In the realm of **epigenomic modifications**, numerous techniques are still **emerging**, and to achieve a systematic mapping of the epigenome, researchers employ methods like **chromatin immunoprecipitation (ChIP)**, **bisulfite sequencing**, and **DNase accessibility profiling**.

Let's break down **bisulfite sequencing** first. The process involves treating DNA with **bisulfite**, a chemical that selectively converts **unmethylated cytosines** into **uracil** (which will be read as thymine during sequencing), while leaving **methylated cytosines** unchanged. After sequencing the treated DNA, researchers compare it with untreated DNA sequences. If the **cytosine** remains unaltered, it indicates **methylation** at that position. This method is widely used for **detecting DNA methylation** patterns across the genome.

Next, **DNase sequencing** involves the use of an enzyme called **DNase I**, which digests DNA at **open chromatin regions**, where DNA is more accessible because nucleosomes or other binding proteins are absent. By selectively cleaving in these accessible regions, researchers can identify **DNA segments** that are not densely packed with **histones** or other regulatory proteins. Interestingly, within these accessible regions, there can be **protected sites** where **regulatory factors** bind to the DNA, preventing the enzyme from cutting those specific regions. This creates a clear pattern where the **accessible chromatin regions** flank the **binding sites** of these regulatory factors, revealing the **regulatory landscape** of the genome.

Lastly, **chromatin immunoprecipitation (ChIP)** is a technique used to identify **protein-DNA interactions**. It involves using an **antibody** that specifically binds to a target **histone modification** or **protein**. After the antibody pulls down the chromatin (the DNA-protein complex), the associated DNA fragments are sequenced. These **sequences** reveal the locations in the genome where the particular modification or protein of interest is present, allowing us to map various epigenomic marks across the genome.

Numerous large-scale projects, like the **Epigenomics Roadmap**, **ENCODE**, and **Blueprint**, are working to map these epigenomic features across various tissues and cell types. These efforts involve identifying a wide range of **histone modifications**, **open chromatin regions**, **DNA methylation sites**, and **gene expression profiles** in multiple tissues. The key takeaway here is that while we may only have **one human genome**, there are many distinct **epigenomes** that differ across various **cell types**, **tissues**, **developmental stages**, and even between **individuals**. For example, the **epigenome** in your **brain** will differ from that in your **lungs**, and even within your brain, different types of neurons will have distinct epigenomic landscapes. Understanding these differences is essential for **decoding the complexity** of human biology.

This broad and systematic mapping of **epigenetic landscapes** helps researchers understand the **dynamic regulation** of genes, providing insights into how **different environments**, **conditions**, and **stages of development** influence our genome.

32:31 Antibodies, ChIP-Seq, data generation projects, raw data

One of the most widely used methods for studying epigenomics is **chromatin immunoprecipitation (ChIP-Seq)**. This technique relies on **antibodies** that are designed to specifically recognize **histone modifications** or **DNA-bound proteins**. Here's how it works: scientists first create an antibody by injecting an animal, such as a rabbit, with a specific **histone modification** like **H3K4me3**. The animal's immune system responds by generating antibodies against the introduced modification. These antibodies are then collected and used in the ChIP-Seq process.

The process begins by **chopping up the DNA** and isolating the fragments bound by the antibodies. These fragments are then sequenced to determine which regions of the genome are associated with specific histone modifications or bound by transcription factors. For example, if an antibody is specific to **H3K4me3**, it will pull down all DNA regions where this modification is present. After sequencing, these regions are mapped to the genome to identify the locations enriched with the histone mark.

Through this approach, researchers can identify regions of the genome that are marked by different histone modifications, transcription factor binding, or chromatin states. For example, **H3K4me1** is a mark of **enhancers**, while **H3K4me3** is associated with **promoters**. This mapping helps researchers construct a detailed view of gene regulation, chromatin structure, and transcriptional activity across the genome.

These technologies are employed in large-scale projects like the **Epigenomics Roadmap** and **ENCODE**, which aim to map histone modifications, open chromatin, DNA methylation, and gene expression across various tissues and cell types. Unlike the human genome, which is relatively stable across individuals, the **epigenome** varies significantly between tissues and stages of development. For example, the **brain epigenome** is distinct from the **lung epigenome**, and even within a single organ, such as the brain, different types of neurons have unique epigenomic profiles.

In addition to mapping histone modifications, researchers can also study **DNA accessibility** and **chromatin states**. Some marks indicate active regulatory regions, while others signify repressed chromatin. Understanding how these marks interact provides insights into gene regulation, cellular function, and differentiation.

The ultimate goal of these efforts is to uncover the **language of the epigenome** by identifying **enhancers**, **promoters**, **transcribed regions**, and **repressed regions**. While promoters are often **active and ready** for transcription, enhancers tend to be more **tissue- and cell-type-specific**, activating genes only under certain conditions. A gene might have multiple **enhancers** but typically only one or a few **promoters**.

Understanding these complex layers of regulation requires sophisticated computational tools and algorithms to analyze the large volumes of **sequencing data** generated from these experiments. These tools help researchers decode the **combinations of marks** that define different **chromatin states** and provide insights into the dynamic regulation of the genome across different **cell types**, **conditions**, and **developmental stages**.

38:02 Read mapping: Hashing, Suffix Trees, Burrows-Wheeler Transform

The challenge of mapping millions of short sequencing reads back to the genome is immense. Imagine sequencing **100 million reads** across multiple experiments and needing to map them efficiently to a **3 billion base pair** human genome. Traditional dynamic programming techniques, which we discussed earlier in the course, can handle sequence alignment but are too slow when applied at this scale. So, how do we map these vast numbers of reads efficiently?

Dynamic Programming is great, but its quadratic time complexity can be too slow for large-scale genomic

data. In this context, we need something faster. One solution is **hashing**, where instead of doing full alignments, we create a **hash table** for small segments of the genome, known as **k-mers** (short subsequences), and rapidly match reads to these pre-computed hashes. This allows us to avoid recalculating the alignment for every read, but it can still be memory-intensive, as it requires building and maintaining large hash tables.

An even more advanced solution, which is both **memory efficient** and **fast**, is the **Burrows-Wheeler Transform (BWT)**. The **BWT** was implemented in a highly efficient mapping tool called **Bowtie**, which significantly accelerated the read mapping process. In fact, Bowtie, developed by **Ben Langmead**, became a cornerstone in genomics for mapping sequencing reads to the genome in a fraction of the time compared to older methods.

How the Burrows-Wheeler Transform Works:

1. **Traditional Approaches:** Traditional methods like hashing involve chopping the genome into **k-mers**, creating hash tables, and matching reads to these k-mers. While this method is fast, it consumes a lot of memory due to the large number of possible subsequences in a genome.
2. **Enter Burrows-Wheeler Transform (BWT):** The **Burrows-Wheeler Transform** uses a clever sorting strategy. To understand how BWT works, consider taking all **rotations** of a string (like "banana"), and sorting them alphabetically. The BWT is then the last column of this sorted list. The interesting property of this transformation is that it clusters repeated characters together, allowing for efficient compression and searching.
3. **Efficient String Searching with BWT:** BWT is especially useful for **string matching**. It allows us to progressively narrow down the search space by matching one character at a time. For example, if we are searching for the string **"CAG"**, we can first identify all locations where **"C"** occurs, then refine this to locations where **"CA"** occurs, and finally pinpoint where **"CAG"** occurs. This method of searching significantly reduces the need to store massive lookup tables.
4. **Handling Large Genomic Data:** The **BWT** is able to compress and sort the genome in such a way that allows for very fast lookup of subsequences, such as sequencing reads. Instead of maintaining a massive hash table, BWT keeps only **sorted fragments of the genome** and a few pointers, dramatically reducing memory usage while retaining the ability to quickly map reads.
5. **Mapping Millions of Reads:** Once the genome is pre-processed using **BWT**, you can map millions of reads very efficiently. This pre-processing step is performed once, and afterward, the **BWT** allows for repeated fast mapping of reads to the genome. The time complexity is **linear** with respect to the number of reads, making it vastly more efficient than quadratic or exponential approaches.
6. **Accommodating Mismatches:** One of the challenges in genome mapping is accounting for sequencing errors or **mismatches**. The **BWT** can handle this by introducing a certain number of allowable mismatches in the search, either by artificially inserting them during the search or by other optimizations that efficiently handle small variations in the sequence.

In summary, the **Burrows-Wheeler Transform** and tools like **Bowtie** revolutionized the way we map reads to genomes. This approach is faster, more memory-efficient, and scales effectively with the massive amounts of data generated by modern sequencing technologies.

The genius of BWT lies in its ability to **compress and sort** genomic data in a way that allows for fast retrieval without the need for massive memory consumption. This has enabled the rapid analysis of millions of reads across many experiments, dramatically advancing fields like genomics and personalized medicine.

54:45 Quality Control, Cross-correlation, Peak calling, IDR (similar to FDR)

Once we have mapped our reads to the genome, the next challenge is to interpret the results. We've generated **intensity signals** from mapping 100,000 reads, but how do we now infer biologically relevant information from them? Specifically, we need to determine which regions of the genome are bound by regulators or marked by histone modifications. To do this effectively, we perform a series of **quality control steps** and ultimately identify regions of interest, known as **peaks**, where significant binding or marking events occur.

1. **Quality Control:** The first step is to ensure that the experimental data is reliable. To assess this, we need to check a number of factors, such as:
    - Are the reads distributed in a way that suggests the experiment was successful?

    - Do the reads match across different marks in a consistent manner?
    - Were there sequencing errors or regions that were not properly mapped?
2. We also need to run a **control experiment** where we sequence the **DNA by itself** to differentiate between regions that are truly bound by the regulator and regions that are merely **accessible**.
3. **Handling Redundancy and Fragmentation:** When pulling down DNA fragments bound by a regulator, we often generate redundant reads if the same DNA molecule is sequenced multiple times. To address this, we calculate the **non-redundant fraction** of binding in our library and check if the reads are piling up artificially. Additionally, DNA fragmentation can occur in the experiment, so we need to ensure that we're capturing valid fragments that reflect true binding events rather than experimental artifacts.
4. **Cross-Correlation Analysis:** One key step is to look at the **distribution of reads** across forward and reverse strands of the DNA. Since the sequencer captures DNA fragments, reads from the **Waton strand** will align to the left of a binding site, and reads from the **Crick strand** will align to the right. By scanning the forward and reverse reads past each other, we can identify a **peak** where they converge, indicating the **fragment length** of the captured DNA.
   This analysis allows us to:
    - Confirm the expected **fragment length**, ensuring it's longer than the read length.
    - Identify artifacts like **re-sequencing** of the same reads, which would show a high degree of redundancy at the read length.
5. **Peak Calling:** Once we have high-quality data, we move on to **peak calling**, where we identify regions of the genome that exhibit strong binding signals. There are several tools for peak calling, including **MACS** and **PICS**, which use different algorithms to identify significant peaks based on the distribution of reads. These tools rely on models of read distribution and calculate probabilities for identifying real binding events versus noise.
6. **Combining Replicates:** When we perform the same experiment multiple times, the question arises: how do we combine the results from **replicate experiments**? Several strategies exist:
    - **Intersection:** Only call peaks that appear in both experiments. This is conservative but may miss strong signals if one experiment was noisier.
    - **Union:** Call all peaks detected in either experiment. This may lead to an overabundance of false positives if one replicate was noisy.
    - **Signal Summation:** Sum the signals from both experiments. This can dilute strong signals from one experiment if the other is weaker.
7. A more sophisticated approach is to sort peaks by **signal strength** in each replicate and determine how well they replicate across experiments. This allows us to determine a **cutoff** where replication falls off, ensuring that we capture the strongest signals from both experiments without introducing noise.
8. **Irreproducible Discovery Rate (IDR):** IDR is a measure similar to **False Discovery Rate (FDR)**, but it focuses on reproducibility across replicates. By comparing the peaks from multiple experiments, we can determine where the signal starts to diverge between replicates. At this point of divergence, we set the **cutoff** for peak calling. IDR allows us to keep peaks that are **highly reproducible** between replicates, even if one experiment has lower overall signal strength.
   The key idea behind IDR is that as you go down the list of peaks sorted by intensity, the replication rate will start to drop at a certain point. By identifying this drop-off, we can determine which peaks are reliable.

In conclusion, **quality control** ensures that our sequencing data is reliable, **cross-correlation** helps us identify valid fragment lengths, and **peak calling** combined with **IDR** ensures that the peaks we identify are both statistically significant and biologically relevant. This approach allows us to confidently infer regions of the genome that are bound by regulators or marked by histone modifications, leading to deeper insights into gene regulation and chromatin structure.

1:05:38 Discovery and characterization of chromatin states

At this stage, we've **carried out our experiments**, mapped our reads, called our peaks, and validated them through replication. The next challenge is to combine the information from **multiple histone modification marks** and uncover biologically relevant patterns across the genome. This leads to the goal of discovering **chromatin states** that represent different functional regions of the genome.

1. **The Challenge of Multiple Marks:** We have **dozens of histone modification marks** that can occur in various combinations. These combinations are **complex**, **dynamic**, and potentially **diverse in function**. For example, different combinations of marks may correspond to distinct chromatin states, such as enhancers, promoters, or repressed regions. We aim to determine which combinations of marks are biologically meaningful.
2. **The Goal:** Our ultimate goal is to **learn the language of the epigenome from scratch**. This requires an **unsupervised approach**, where we provide the algorithm with a large dataset, and it identifies patterns in the data without prior assumptions. The algorithm should be able to find **distinct chromatin states**—such as promoters, enhancers, and transcribed regions—that represent different functional aspects of the genome.
3. **Probabilistic Modeling:** To accomplish this, we need a **probabilistic model** that can identify the combinations of marks in a way that reflects biological reality. The model should not only recognize individual marks but also take into account the **relationship between neighboring genomic positions**. This is important because chromatin states are not random; for example, promoter states are more likely to be found near other promoter states, and transcribed states are more likely to be found near other transcribed regions.
4. **Hidden Chromatin States:** The task is to infer a **hidden chromatin state** at every genomic position. At any given point in the genome, we may be in a promoter state, enhancer state, or transcribed state, each of which will **emit** a specific combination of histone modification marks. The **hidden state** refers to the actual chromatin state that we are trying to infer based on the **observed histone marks**.
5. **Transition and Emission Matrices:** The approach relies on two key concepts:
    - The **transition matrix** represents the likelihood of moving from one chromatin state to another as we walk along the genome. For instance, promoter states are likely to be near other promoter states, and enhancer states near other enhancer states.
    - The **emission matrix** defines the probability of observing certain histone marks in a given chromatin state. For example, promoter states might emit certain marks like H3K4me3, while enhancer states might emit H3K27ac.
6. **Hidden Markov Model (HMM):** To model these relationships, we use a **Hidden Markov Model (HMM)**, which allows us to represent the sequence of chromatin states as a probabilistic process. The HMM enables us to infer the **hidden states** (such as enhancers or promoters) from the **observed emissions** (the histone marks). By training the model on the data, we can learn the **transition probabilities** between chromatin states and the **emission probabilities** of histone marks from those states.
    - The **transition probabilities** capture the likelihood that a given chromatin state will transition to another chromatin state.
    - The **emission probabilities** reflect the likelihood that a specific combination of histone marks will be observed in each chromatin state.

Through this approach, we can discover and characterize **chromatin states** across the genome, helping us understand how the epigenome regulates gene expression and cellular function. This method allows us to move beyond individual histone marks and focus on the **higher-order structure** of chromatin, which ultimately governs how the genome functions in different contexts.

1:08:02 HMM Foundations, Generating, Parsing, Decoding, Learning

Hidden Markov Models (HMMs) are a foundational tool for understanding how **sequential data** is generated, interpreted, and learned. HMMs allow us to model sequences where the underlying states are **hidden**, but they generate observable data.

1. **Generating Sequences:** HMMs are designed to model processes where the sequence of observed data is **influenced by hidden states**. For instance, in **speech recognition**, an HMM can transcribe words by mapping **utterances** (the observable data) to **hidden linguistic states** (words or phonemes). These utterances come in a sequence, and the interpretation of each sound is influenced by the one that precedes it. Similarly, in genomics, the position along the genome influences the current chromatin state. For example, if the model is in an **intergenic region**, it's more likely to stay in that region, while if it enters a gene, it's more likely to stay in a transcribed region.
2. **Temporality in HMMs:** HMMs incorporate the notion of **temporality**, where the sequence of hidden

states follows a specific order. In genomics, this temporal structure corresponds to the **position along the genome** from left to right. For example, if the model detects a promoter region, it's likely to stay in a nearby promoter state for several positions. This use of **neighborhood information** constrains the number of possible states at each position, reducing the complexity of the sequence modeling.

3. **Emitting and Recognizing Sequences:** The main goal of an HMM is to emit observable data based on the hidden states and to recognize these sequences by learning the distinguishing characteristics of each state. For instance, in the genome, HMMs emit **DNA sequences** based on the hidden chromatin state—such as promoters, enhancers, or transcribed regions—and allow us to recognize these functional regions from the observed marks.
4. **Observations and Hidden States:** At the beginning of the course, we discussed how there's a distinction between **observations** (the data we see) and **hidden states** (the unobserved, underlying process that generates the data). The observations might seem independent, but they are sampled from a **continuous process** that contains temporal relationships. In HMMs, we combine this framework with **probabilistic models** to make inferences about the hidden states based on the observed data.
5. **Markov Chain vs. Hidden Markov Model:** In a **Markov Chain**, the sequence of observations directly influences the next state, with no hidden variables. However, in an HMM, the Markov Chain operates on the **hidden states**. These hidden states generate the observable data in a **generative framework**. For instance, in genomics, we transition between hidden states such as **promoters, enhancers, and transcribed regions**. These hidden states then emit the observable **histone marks** or other biological signals.

In summary, **HMMs** are powerful tools for **modeling sequences** with hidden states, allowing us to infer complex patterns in sequential data like speech, text, or genomic sequences. By learning the **transition probabilities** between hidden states and the **emission probabilities** of observable data from these states, we can decode hidden structures within the data.

1:11:15 Two Sets of HMM parameters: Emissions, Transitions

In **Hidden Markov Models (HMMs)**, we work with two key sets of parameters: **transition probabilities** and **emission probabilities**. These parameters govern how hidden states evolve over time and how they generate observable data.

1. **Hidden States and Observations:** In an HMM, we have:
    - A set of **observations** (denoted as **X**), which represent the data we can observe.
    - A sequence of **hidden states** (denoted as **π**), which are not directly observable but determine the behavior of the observations.
2. **Transition Probabilities:** The **transition probabilities** define how the model moves from one hidden state to another. Specifically, the probability of transitioning from **state K** to **state L** is determined only by the current state (state K). This means that the HMM is **memoryless**, which is a key property of Markov models. In this context, **memorylessness** means that the transition to the next state depends solely on the current state and not on any prior states.
3. **Emission Probabilities:** The **emission probabilities** define how the hidden states generate the observable data. For each state, there's a probability distribution over the possible observations. For example, the probability of emitting an observation $\mathbf{X_i}$ while being in **state K** is part of the emission probabilities.
4. **HMM Structure:**
    - We have a **transition matrix**, which encodes the probabilities of moving between hidden states.
    - We have an **emission vector**, which encodes the probabilities of the observed data being generated from each hidden state.
5. **Bayesian Framework:** HMMs extend the basic **Bayesian framework** by including both transition and emission probabilities. Using these, we can compute the likelihood of a sequence of observations by considering both the likelihood of staying in certain states and the likelihood of generating specific observations from those states.

In summary, an HMM models sequences through two types of probabilities: **transition probabilities** for moving between hidden states and **emission probabilities** for generating observable data from these states. Together, they provide a powerful way to model sequential data with underlying hidden structures.

1:12:40 Example 2-state HMM, observations, scoring, inference

In a **2-state Hidden Markov Model (HMM)**, we aim to model sequences of data, such as genomic sequences, by transitioning between different hidden states (e.g., **background** and **promoter**) and generating observable sequences from these states.

1. **Background vs. Promoter States:**
    - The **background state** is characterized by equal probabilities for nucleotides (**A, C, G, T**), while the **promoter state** might favor **G** or **C** with higher probabilities, such as 40%, and have a lower probability for **A** and **T** (e.g., 10%).
    - The model includes **self-transition probabilities** for both the background and promoter states. For example, the background state might have a 99% chance of remaining in the background, and the promoter state might have a 95% chance of staying in the promoter state.
2. **Transitioning Between States:**
    - The transition probability from the background to the promoter state could be 1%, while transitioning back from the promoter to the background could be 5%.
    - **Self-transition probabilities** determine how long the model stays in a given state. Since the background has a higher self-transition probability (99%), it is more likely to remain in the background for a longer duration compared to the promoter state.
3. **Observation Sequence:** Suppose we observe a sequence like **ATAGTC**. To determine if it came from the background state or the promoter state, we calculate the joint probability of the sequence being generated by each state.
    - For the **background state**, we calculate the product of emission probabilities (e.g., 0.25 for each nucleotide in the background) and the self-transition probabilities (e.g., 0.99 for staying in the background at each step).
    - For the **promoter state**, we calculate the product of emission probabilities, such as 0.1 for **A, T** and 0.4 for **G, C**, and the self-transition probabilities (e.g., 0.95).
4. **Likelihood Comparison:** After calculating the joint probabilities, we can compare the likelihood of the sequence being generated by the background or the promoter state. In this example, the **background state** might be **2.5 times more likely** to generate the sequence because it is **A/T-rich**, which is typical for the background state.
5. **Transitioning Between States:** We can also consider models where the sequence transitions between the **background** and **promoter** states. For instance, transitioning from background to promoter and back to background might better capture the **G/C-rich** segments of the sequence. However, **transitioning between states** incurs a cost, and excessive transitions make the model less likely.
6. **Inference:** The challenge with this approach is that there are an **exponential number of possible paths** through the states, making it computationally expensive to evaluate every possible path. Therefore, more efficient algorithms (such as the **Viterbi algorithm**) are needed to find the most likely path without calculating every possible transition.

In summary, this 2-state HMM example illustrates how we model sequences by assigning probabilities to both staying in a state and transitioning between states, while considering the costs of transitions and the likelihood of observing specific nucleotide sequences in each state.

1:16:38 Viterbi algorithm: Find best parse $\pi^* = \text{argmax}_\pi P(x,\pi)$

The **Viterbi algorithm** is used to find the **most likely sequence of hidden states** (or "parse") that generates the given sequence of observations, using **dynamic programming** to efficiently navigate through the exponentially large space of possible paths.

1. **Dynamic Programming to Avoid Exponential Complexity:**
    - Instead of evaluating every possible path through the hidden states (which would be exponential in number), we use dynamic programming to align **states** to **positions** in the sequence.
    - The key insight is that we can compute the best score up to a certain position, then reuse this information to compute the best score for the next step. This reduces the computational complexity from exponential to **polynomial**.
2. **Aligning States to Positions:**

- For each position in the sequence, we align it to one of the possible hidden states.
- The alignment of states doesn't happen linearly but rather transitions between states according to the **transition probability matrix**, which governs the likelihood of moving from one state to another.

3. **Transition and Emission Costs:**
   - As we transition between states, we incur a **transition cost** specific to that transition, rather than a fixed gap penalty as in sequence alignment.
   - Each state also has an **emission probability** for generating the observed character at the given position. The **total score** for a state depends on both the **transition** from the previous state and the **emission** of the observed character.
4. **Recursive Calculation of Scores:**
   - The algorithm recursively calculates the **best score** for each state at each position in the sequence.
   - For the current state, the score is the product of:
     - The **score** of the best previous state,
     - The **transition probability** to the current state,
     - The **emission probability** for the observed character.
   - This ensures that we find a state with both a **high previous score** and a **low transition penalty**.
5. **Storing the Best Path:**
   - As we compute the maximum score at each position, we store both the **score** and an **arrow** pointing to the previous state that gave the best score. This allows us to trace back the optimal path once we've processed the entire sequence.
6. **Backtracking to Recover the Best Path:**
   - Once we reach the end of the sequence, we follow the **stored arrows** backwards to recover the optimal sequence of states (the **best parse**).
   - This backtracking ensures that the entire sequence is optimally aligned to the most likely sequence of states.
7. **Time and Space Complexity:**
   - The Viterbi algorithm runs in **O(k^2n)** time, where kkk is the number of states and nnn is the length of the sequence. The space complexity is **O(kn)**, since we only need to store the scores for each state at each position and the arrows for backtracking.

In summary, the Viterbi algorithm efficiently finds the most likely path through a Hidden Markov Model by dynamically computing scores, storing intermediate results, and backtracking to reconstruct the optimal state sequence. This method avoids the exponential complexity of brute-force approaches while ensuring a globally optimal solution.

1:20:15 Posterior Decoding: Most likely state πi(over all paths)

1. **Total Probability Calculation:**
   - Posterior decoding aims to determine the **most likely state** at each position, considering **all possible paths**. Instead of finding a single most likely sequence of states (like in Viterbi decoding), it computes the probability of being in a particular state at each position, summing over all paths that pass through that state.
   - To do this, we need to calculate the **total probability of generating a sequence**, which involves summing the probabilities of **all possible paths** through the sequence.
2. **Forward-Backward Algorithm:**
   - **Forward pass**: Starting from the left, we calculate the probability of generating the sequence **up to** a particular state. This gives us the likelihood of being in a specific state based on all prior observations.

- **Backward pass**: Starting from the right, we calculate the probability of generating the sequence **from** a particular state **to the end**. This captures the likelihood of being in a specific state given all future observations.
- By combining these two passes (forward and backward), we can calculate the probability of being in a particular state at each position.

3. **Combining Forward and Backward Probabilities:**
   - The **forward probability** represents the likelihood of observing all the states and emissions **up to** the current state.
   - The **backward probability** represents the likelihood of observing all the states and emissions **from** the current state **to the end**.
   - Adding the forward and backward probabilities gives us the **total probability** of being in a specific state at a given position, accounting for both the preceding and following observations.
4. **Posterior Probability for Each State:**
   - By combining the **forward and backward** probabilities, we calculate the **posterior probability** of being in each state (e.g., enhancer, promoter, transcribed, repressed) at each position in the sequence.
   - This allows us to determine the **most likely state** for each position **independently**, unlike the Viterbi algorithm, which finds the most likely sequence of states as a whole.
5. **Application in Epigenomics:**
   - In the context of epigenomics, this approach helps infer the probability of a genomic position being in a particular **chromatin state** (e.g., enhancer, promoter) based on the **emitted histone modification marks**.
   - By considering all observations before and after a given position, posterior decoding provides a more **granular view** of the likelihood of each state, rather than committing to a single sequence of states.

In summary, **posterior decoding** calculates the probability of being in each hidden state at every position, considering both past and future observations. This method is especially useful when we want to independently infer the most likely state at each position, rather than finding the single best sequence of states.

1:21:48 Summary

In today's lecture, we covered key concepts in **epigenomics** and explored various techniques for data processing, sequence mapping, and quality control:

1. **Epigenomics Foundations**:
   - We began with an overview of **epigenomics** and its role in regulating gene expression and chromatin structure.
2. **Primary Data Processing**:
   - We discussed methods for processing raw epigenomic data, including the use of the **Burrows-Wheeler Transform (BWT)** for fast read mapping, which helps in aligning sequencing reads to the genome efficiently.
3. **Quality Control and Peak Calling**:
   - We examined methods for **quality control** of the data and determining where **peaks** (representing binding sites or marks) occur, which indicate areas of interest in the genome.
4. **Hidden Markov Models (HMMs)**:
   - We introduced the basics of **Hidden Markov Models** for modeling genomic data. HMMs allow us to model sequential data where the **observations** (histone marks, for instance) are generated by **hidden states** (e.g., chromatin states like enhancers or promoters).
   - We covered **sequence generation**, **parsing**, and **decoding** using HMMs, with a focus on the **Viterbi algorithm** for finding the best state path and **posterior decoding** for calculating the probability of each state at each position.
5. **Next Steps**:

- While we discussed HMMs in detail, we still need to cover how to **learn the parameters** of an HMM, such as transition and emission probabilities, and how to model more complex **state transitions**. This will be picked up in the next lecture.

In summary, we've laid the groundwork for understanding **epigenomic data processing**, **HMMs**, and key algorithms for sequence decoding. The next lecture will continue from here, expanding on the learning mechanisms and tying this into **regulatory motifs**.

Looking forward to seeing you all for recitation on Monday and for continuing this lecture on Tuesday. Have a great long weekend!

Lecture 6 - Regulatory Circuitry

Video: Lecture 6 - Regulatory Circuitry - MLCB24

Slides: Lecture06_RegulatoryGenomics.pdf

0:00 Introduction

Welcome, everyone! Today, we're diving into **regulatory circuitry**. We've previously discussed different parts of the genome, but today's focus will be on understanding how **non-coding regions**—particularly **regulatory elements** like **enhancers** and **promoters**—play a key role in disease-associated genetic variants. Remarkably, 93% of these disease-linked variants are found in non-coding regions, not in **protein-coding regions**. This raises critical questions: Where are these variants located? Are they targeting promoter regions, enhancer regions, or other control regions? If they are targeting **enhancers**, how do they affect target genes, and what are the **upstream regulators** that control these regions? Additionally, how do these regulatory regions behave dynamically across different **cell types**?

We'll tackle these key questions regarding regulatory circuitry today. Our focus will be on laying the **foundations at the genomic level**, and later in the week, we'll transition to the networks that arise from these circuits, helping us to understand **biological networks** more comprehensively.

There are several core topics on the agenda:

1. **Epigenomic Dynamics**: Building on what we discussed previously, we'll dive deeper into understanding **chromatin states** across multiple cell types and how this contributes to gene regulation dynamics.
2. **Enhancer-Gene Linking**: We will examine how enhancers are linked to their **target genes**. This includes exploring different methods such as **3D genome conformation**, **correlation-based methods**, and a brief discussion of **expression quantitative trait loci (eQTLs)**, which are genetic variants that affect nearby gene expression, rather than influencing **phenotypic traits** directly.
3. **Regulatory Motif Discovery**: We will explore various methods for discovering **regulatory motifs**. These include traditional methods such as **enrichment analysis**, **expectation maximization**, and **Gibbs sampling**, as well as more advanced approaches like **deep learning** and **convolutional neural networks (CNNs)**.
4. **Comparative Genomics**: We'll also delve into **global motif discovery** through comparative genomics, focusing on conserved sequence identification across species.

Finally, time permitting (though it's unlikely), we'll explore **massively parallel reporter assays** for discovering new regulatory elements.

In this lecture, we'll touch on concepts from previous weeks, such as **expression analysis**, **single-cell analysis**, **sequential data in hidden Markov models (HMMs)**, and **epigenomics**, as we integrate all of these into today's topic of **regulatory genomics**. We'll also get ready to move on to **biological networks** next time.

Let's begin!

3:45 Epigenomics review: signals, mapping, HMMs

Enhancers and promoters play a crucial role in regulating gene expression, as they contain **motifs** that bind to **regulatory proteins**, forming key parts of the **regulatory circuitry**. This circuitry is physically instantiated through **DNA interactions**, where regulatory proteins bind to enhancers, which in turn loop around to physically interact with promoters. This allows **RNA polymerase** to bind and initiate **transcription** of RNAs.

Last time, we discussed the challenge of mapping **hundreds of millions of short reads** to the genome efficiently. We explored the **Burrows-Wheeler transform**, a computational breakthrough initially implemented

in **BWA** (Burrows-Wheeler Aligner), which allows for rapid and memory-efficient genome mapping. This technique builds on the idea of **growing suffixes** of a search string and progressively **narrowing the search space** by using pointers to guide the process.

We then introduced the idea of using **multiple independent epigenomic marks** to analyze genomic signals. By combining these signals, we can leverage **Hidden Markov Models (HMMs)**, which is the focus of today's lecture. HMMs offer a powerful framework for distinguishing between **observed** and **hidden** states, enabling us to model dependencies between **adjacent genomic positions**.

This approach is critical because **genomic features**, such as enhancers and promoters, tend to occur near each other. For instance, it's far more likely to find an **enhancer** near a **transcribed region** rather than in the middle of **repressed regions**. These relationships are encoded in the **transition probability matrix** of the HMM, which tells us the probability that the next state at position iii is LLL given that the state at position i−1i-1i−1 was KKK. This captures transitions such as **enhancer to promoter**, **promoter to transcribed region**, and so on.

Additionally, the **emission probabilities** describe the likelihood of observing a particular signal (or "character") given that we are in a specific **hidden state**. This allows us to model the generative process behind the **observed genomic data** using a **probabilistic framework**.

6:30 Parsing, scoring, Viterbi decoding, optimal path

In our last lecture, we explored the foundation of **Hidden Markov Models (HMMs)**, focusing on how we transition between different states and self-transition within the same state. These transitions help determine the **length** of certain regions like **promoter sequences** or **background sequences**. This is crucial because the **probability of staying in the same state** may be very high (e.g., 99%), but it decreases **geometrically** with each time step, resulting in a **geometric length distribution** for these states.

We also discussed **emission probabilities**, which tell us how likely certain states are to emit particular **characters**. For example, in a **background state**, **ACGT** might have equal probabilities, while in a **promoter state**, **C** and **G** might have a higher probability than **A** and **T**.

Next, we introduced how to **parse** sequences by analyzing transitions between background and promoter states. The **transition penalty** for switching between states (e.g., from **background** to **promoter**) was high, meaning that switching states frequently incurs a substantial **cost**. Therefore, while the emission probabilities might offer slight gains in likelihood, the high cost of transition discourages frequent changes, which is one of the central features of HMMs: avoiding **jittery transitions** between states. Instead, the model ensures **smooth transitions** to allow **nearby positions** to influence the probability of a given state.

To handle the exponential complexity of possible **parses** (since each position could have two possible states, leading to 2n2^n2n combinations for nnn positions), we introduced **dynamic programming** for optimal parsing. Using the **Viterbi algorithm**, we computed the **optimal path** by:

1. **Setting up a variable V** for each position, representing the maximum score at that position based on the best possible score from the previous position.
2. The current **maximum score** is computed by choosing the best score from the previous state, factoring in the **transition cost** and **emission probability**.

This approach ensures that we efficiently compute the best path without scoring every possible path. The key is balancing a **high score** in the previous state with a **low transition cost**.

Once we compute the best score for the final state, we can **trace back** the arrows (similar to **dynamic programming alignment**) to reconstruct the entire **optimal parse**, which is the **best interpretation** of the sequence of **hidden states** given the observations.

This process led to the **Viterbi decoding equation**, where the **current maximum** is the **product** of the previous state's maximum score and the transition cost to the current state, ensuring that we find the **best path**.

11:47 Posterior decoding, Forward Algorithm

We introduced a new way to approach parsing sequences by focusing on individual **positions** rather than the entire path. In the previous Viterbi method, we found the **best single path** through the sequence. However, this approach might miss important information, since there are an **exponential number of paths** that can explain the sequence, and many of these paths could pass through different **hidden states** at a given position.

For instance, while the best path might pass through an **enhancer** state at position 523, many other

high-probability paths could pass through the **promoter** state at that same position. In this case, it's not enough to focus only on the best path. Instead, we should sum over **all possible paths** that pass through the enhancer and compare them to the sum of all paths that pass through the promoter at position 523. This gives us a better understanding of the likelihood of being in one state versus another at that specific position.

To calculate this, we introduced the concept of **total probability mass** for each state at a given position. This required us to move away from the Viterbi approach of maximizing probabilities and instead to sum all probabilities up to a given point. This process is done using **recursion**: if we know the total probability up to a certain point, we can sum the probabilities at the next position by factoring in **transition** and **emission** probabilities.

This leads us to the **Forward Algorithm**, which calculates the probability mass for each state moving **forward** through the sequence. Once we reach the end of the sequence, we need to do the same process but moving **backwards**, which is accomplished with the **Backward Algorithm**. The Forward Algorithm tells us the total probability mass going up to a point, and the Backward Algorithm gives us the probability mass after a point, working in tandem.

A small adjustment is needed: we don't want to count the **emission probabilities** twice, so we include them in the Forward Algorithm but exclude them from the Backward Algorithm.

Once we've completed these two steps, we can calculate the **posterior probability** of being in a given state at any position. This is known as **Posterior Decoding**, which looks at the **posterior probability** of being in a state after observing the entire sequence. This method doesn't necessarily give us a valid transition path, but it allows us to determine the most likely state at each individual position.

The key difference between this method and Viterbi is that Posterior Decoding focuses on the **best state at each position**, while Viterbi gives the **best overall path**. This means that Posterior Decoding might produce a sequence of states where transitions are not valid according to the model, but it still reflects the **most likely states** at each position when considering all possible paths.

Does this distinction between Viterbi and Posterior Decoding make sense?

18:39 Learning HMM parameters: Supervised

In a **supervised learning** context, we already have labeled data, meaning we know both the **observations** (the sequence of emitted symbols) and the **hidden states**. With this information, it's straightforward to estimate both the **emission** and **transition** probabilities for the Hidden Markov Model (HMM).

Estimating Emission Probabilities

The emission probability describes the likelihood of a state emitting a particular symbol. Given a labeled sequence, we simply count how often a particular symbol is emitted from a particular state and divide it by the total number of emissions from that state.

For example, imagine you have a sequence of symbols like **G, C, A, A, G, C** labeled with states **B, B, B, P, P, B**. If we want to estimate the probability of emitting the symbol "A" from state **P**:

- **Numerator:** Count the number of times **A** is emitted from state **P** (which might be 2 times).
- **Denominator:** Count the total number of emissions from state **P** (which might be 3 times).

Thus, the **emission probability** of **A** from **P** is 23\frac{2}{3}32. You repeat this process for all symbols and states to build the full **emission probability matrix**.

Estimating Transition Probabilities

The **transition probability** is the likelihood of moving from one state to another. For example, if we are in state **B**, how likely is it that we will transition to state **P**?

You can estimate this by counting how many times you observe a transition from **B** to **P** and dividing by the total number of times you are in state **B**. For instance:

- **Numerator:** Count the number of transitions from **B** to **P**.
- **Denominator:** Count the total number of transitions starting from **B**.

For example, if you observed 5 transitions from state **B**, and one of those transitions led to **P**, the **transition probability** from **B** to **P** would be 15\frac{1}{5}51.

Similarly, if you were in **P** three times and transitioned to **B** once, the **transition probability** from **P** to **B** would be 13\frac{1}{3}31. If you stayed in **P** twice, the **transition probability** from **P** to **P** would be 23\frac{2}{3}32.

This is a **maximum likelihood estimation** approach, which works well when there's plenty of data.

Addressing Zero Counts

One challenge in this supervised approach arises when you don't observe certain transitions or emissions. If you only have a small dataset, some transitions might never occur, leading to **zero probabilities**, which can skew the model. To address this, we introduce **pseudo counts**, which add a small number (like 1) to each count to avoid zeros in the probabilities. This technique smooths the estimates and ensures the model remains robust, even with limited data.

Summary:

1. **Emission Probabilities:** Count how often a symbol is emitted from a state and divide by the total number of emissions from that state.
2. **Transition Probabilities:** Count how often you transition from one state to another and divide by the total number of transitions from that state.
3. **Pseudo Counts:** Add small values to counts to avoid zero probabilities in small datasets.

21:22 Learning HMM parameters: Unsupervised (Best Path, Viterbi)

When learning HMM parameters without labeled data, the problem becomes significantly more interesting and challenging. In an **unsupervised** setting, we're given sequences but no information about the underlying states. Our task is to infer both the **hidden states** (like promoters, enhancers, repressed regions, etc.) and the **parameters** (the emission and transition probabilities).

The Basic Approach

1. **Initial Parameters:** We start with some **initial guess** for the model parameters, including the **transition** and **emission probabilities**. These initial values might be based on general biological assumptions or randomized.
2. **Infer States:** Using these initial parameters, we apply the **Viterbi algorithm** to find the **best path** (sequence of hidden states) through the given sequence. This allows us to predict which regions are likely promoters, enhancers, etc., based on the sequence and the current model.
3. **Parameter Updates:** Once we have the predicted sequence of hidden states (the **Viterbi path**), we can now update the **transition** and **emission probabilities** by counting how often different symbols (e.g., nucleotides) are emitted by different hidden states, and how often transitions between hidden states occur.
4. **Iterate:** With the updated parameters, we run the Viterbi algorithm again to find a **better path** through the sequence. This process is repeated iteratively, improving both the parameters and the inferred path over time, gradually **converging** to a solution where both the parameters and the hidden states are well-estimated.

Dynamic Interplay Between Parsing and Parameters

Parsing with Viterbi: At each step, the Viterbi algorithm identifies the **best path** through the hidden states, based on the current parameters. This path represents the most likely sequence of hidden states that generated the observed sequence.

Parameter Updates: After obtaining this best path, we treat it as if it were the **true labeling** and use it to update the model's parameters. For example, if a certain region is inferred as a promoter, we update the emission probabilities for promoters based on the symbols (nucleotides) observed in that region.

Convergence to Better Annotations. Initially, the model might not be very accurate, but as it repeatedly alternates between inferring the best hidden state path and refining the parameters, it **converges** to a better solution. Over time, the model learns to annotate the genome with no prior labels by: Identifying **distinct patterns**, like regions with high or low GC content. Learning that certain **sequential dependencies** (e.g., enhancers tend to be near transcribed regions) are more likely than others.

Comparison to K-means Clustering. This unsupervised HMM learning is conceptually similar to **K-means clustering**: In K-means, we start with arbitrary cluster centers and iteratively assign points to the nearest center and update the centers. In HMM learning, we start with arbitrary parameters, use them to assign hidden states, and then update the parameters based on these assignments. Both methods involve **iterating between assignment and updating** until convergence.

Example: Partitioning Genome by GC Content

As an example, imagine the task of clustering the genome into regions of **high GC** and **low GC** content. By

starting with an initial guess, the model can progressively refine these regions by:

- Assigning hidden states based on the current parameters (e.g., regions with more GC nucleotides being classified as "high GC").
- Refining the parameters to better reflect the data, such as adjusting the emission probabilities for GC nucleotides in the high GC state.

Over time, the model will find regions that are consistently high or low GC and will update its parameters and state assignments accordingly.

The Sequential Homogeneity Constraint

One key difference between unsupervised HMM learning and other clustering methods (like K-means) is that **sequential structure matters** in HMMs. The model assumes that certain hidden states are more likely to follow others (e.g., a promoter might be near a transcribed region), which constrains the sequence of state transitions. This constraint is encoded in the **transition matrix**, ensuring that the genome is **parsed** in a way that respects the biological context.

Iterative Process in Summary

1. Start with some initial parameters.
2. Use **Viterbi** to find the best hidden state path.
3. Use the inferred hidden states to update the parameters.
4. Repeat until convergence, alternating between finding the best path and refining the parameters.

This iterative back-and-forth between estimating hidden states and updating parameters is key to unsupervised learning in HMMs.

27:02 Learning HMM parameters: Unsupervised (All Paths, Baum-Welch)

The **Baum-Welch algorithm** represents a more sophisticated method for unsupervised learning in Hidden Markov Models (HMMs), extending beyond the limitations of the Viterbi algorithm. While Viterbi focuses on finding a single optimal path through the hidden states, Baum-Welch leverages the **entire distribution of possible paths** to derive more accurate estimates of the HMM parameters, such as transition and emission probabilities.

Baum-Welch operates through an **Expectation-Maximization (EM)** framework. In the **Expectation step (E-step)**, it calculates the total probability across all possible paths through the hidden states, rather than only considering the most likely path. In the **Maximization step (M-step)**, the algorithm updates the HMM parameters based on these probabilities, refining the estimates iteratively.

A key element of this approach is using the **forward-backward algorithm**, which computes the probability of the sequence up to a certain point (forward probability) and from that point onward (backward probability). These two probabilities together give the **posterior probability** of being in a particular state at any given position in the sequence. Baum-Welch thus captures the **probability mass of all possible paths**, rather than just the single best path, making it more comprehensive and accurate.

In contrast to Viterbi training, which generates an annotation based on the best path, Baum-Welch performs **posterior decoding**, where the algorithm estimates the most likely state for each position based on the probability mass summed over all paths. This allows for a more nuanced understanding of the sequence, as it accounts for the contribution of many paths that might pass through different states at any given position.

The **core process** of Baum-Welch involves:

1. **Initial parameters**: Starting with some guesses for the transition and emission probabilities.
2. **Forward-backward calculation**: Using the forward algorithm to compute the total probability mass up to each point and the backward algorithm to compute the total mass from that point forward.
3. **Parameter updates**: Summing over all possible paths to determine the expected number of transitions and emissions, updating the HMM parameters accordingly.

For example, instead of simply counting how many times a particular state emits a symbol in a single best path, Baum-Welch considers the **weighted contributions** of all paths that could emit that symbol from that state, providing a more accurate estimate of the emission probabilities. Similarly, it accounts for **all transitions** between states, weighted by the probabilities of the paths that make those transitions.

Ultimately, **Baum-Welch training** converges on a set of transition and emission probabilities that best explain

the observed data, given all the possible paths the HMM might follow. This approach contrasts with the **supervised learning scenario**, where labeled data simplifies the task of counting transitions and emissions directly. In Baum-Welch, however, the lack of labeled data necessitates this iterative approach of refining both the path estimates and the model parameters over time.

In summary, Baum-Welch allows us to handle **unsupervised learning** of HMMs in a more robust and statistically grounded way, accounting for the full range of possibilities inherent in the data, rather than relying on a single optimal interpretation. This method significantly enhances the ability of HMMs to capture the complexity of real-world biological sequences, making it a powerful tool in fields like **epigenomics** and **gene regulation** analysis.

31:45 Chromatin State Characterization

Once we've developed the capabilities to analyze chromatin states, we can essentially unleash the model on the human genome to explore its architecture. The model identifies patterns, such as regions with **higher GC content**, and we then ask: **What are these regions?** Through enrichment analysis, we can check if these regions overlap with known genomic elements, such as **repeat elements**, **enhancers**, **transcription start sites**, and **exons**. For example, if we find that these regions are enriched around transcription start sites, we may hypothesize that the identified hidden state represents a **promoter**.

This process involves **annotating the genome** by dividing it into a set of distinct states based on their epigenomic signatures. Each state represents a combination of chromatin marks, and by examining the enrichment of these states in different genomic regions, we can give meaningful labels to them, such as **promoter**, **transcribed**, **enhancer**, or **repressed** states.

Model Output and Genome Parsing

For example, after partitioning the genome into 50 chromatin states, we may find that certain states are consistently associated with **gene start sites**, others with **gene body regions**, and others with **repressed regions**. This allows us to start interpreting the roles of these chromatin states in gene regulation. A key discovery is that some states appear in regions near active genes, while others occur in **intergenic** or **repressed** regions.

The model also helps us understand the genome's overall structure. A large portion of the genome may fall into what we might call "boring" states, representing regions where chromatin activity is low or uniform. In contrast, smaller portions of the genome, where **active chromatin marks** are concentrated, may be more **biologically interesting**. For example, the model might show intense activity in **gene-dense regions** and little activity in **centromeric** or **heterochromatic regions**. The model may even reveal differences between **chromosomal territories**, such as the **A compartment** (associated with active chromatin) and the **B compartment** (associated with repressed chromatin, often located at the **nuclear periphery**).

Assigning Functional Labels to Chromatin States

To assign meaningful labels to the chromatin states discovered by the model, we examine their enrichment for known genomic features. For instance:

- **Promoter states** are strongly enriched around transcription start sites.
- **Repressed states** may show strong enrichment for **nuclear lamina** association, reflecting their inactive nature.
- **Repeat states** may be enriched in **repetitive elements** such as **LINEs** or **SINEs**.
- **Transcribed states** show enrichment in gene bodies, corresponding to active gene transcription.

This analysis allows us to systematically annotate the genome based on epigenomic features, moving from a purely **data-driven model** to one that is **biologically interpretable**. For example, we may find that certain states are enriched near **developmental enhancers** or regions marked by **long non-coding RNAs**.

Discovering New Functional Elements

Through this process, we can also uncover previously unknown elements in the genome. For instance, we may identify **new protein-coding genes**, **developmental enhancers**, or regions corresponding to **long intergenic non-coding RNAs (lincRNAs)**, which may have critical regulatory roles despite being non-coding. These discoveries are made possible by combining epigenomic signatures with **functional enrichment analyses**.

Relevance to Disease: Genome-Wide Association Studies (GWAS)

Perhaps one of the most significant applications of this chromatin state analysis is in understanding

disease-associated non-coding regions. Initial genome-wide association studies (GWAS) revealed that only a small percentage (around 7%) of disease-associated variants are found within protein-coding regions. The majority are located in **non-coding regions**, previously termed **"dark matter"** of the genome. By characterizing chromatin states, we now have a powerful tool to shine a light on these non-coding regions. We can identify that many of these disease-associated variants are located in **enhancer regions**, providing a new understanding of their regulatory roles in disease.

Thus, chromatin state annotation allows us to map the non-coding genome in a way that directly links genetic variation to gene regulation and disease mechanisms, transforming our understanding of the genome's regulatory architecture.

43:26 Model complexity: selecting the number of states/marks

When addressing **model complexity** in Hidden Markov Models (HMMs) and deciding on the number of states or marks, it's crucial to determine an appropriate balance between model simplicity and detail. The question arises: how many states are sufficient to capture the biological complexity without overfitting or introducing unnecessary redundancy?

One approach is to assess whether increasing the number of states improves the model's ability to capture **mark dependencies**. For example, as you add more states (e.g., moving from 10 to 50 or 100), you can analyze the **occurrence frequency of pairs of marks** within a state. As the number of states increases, the **deviation from expectation decreases**. This indicates that, with a sufficient number of states, the model can capture more complex dependencies between marks. When using fewer states, the model tends to only capture **mark-independent effects**.

Another method to determine the optimal number of states is through **nested initialization**. This technique avoids fixing the number of states from the start. Instead, you start with a larger number of states (say 80) and then **prune redundant states** after training. This approach allows the model to initially explore more states, increasing the chance of discovering important but rare states that may not appear in a smaller model. Once the model is trained, you can remove states that are redundant or less informative, refining the model down to the most meaningful states.

This approach offers several advantages. For instance, if you randomly initialize the model with a specific number of states (e.g., 50), it might capture the same state multiple times due to random fluctuations, missing important ones. By starting with a higher number of states and then pruning, the model can discover rare states and avoid redundancies. This is analogous to running a clustering algorithm multiple times and selecting the most consistent clusters across runs.

Nested initialization further ensures that **once a state is discovered**, it remains part of the model, even as you reduce the number of states. This robustness makes the model less sensitive to random fluctuations in initialization. For example, you might only start consistently recovering a particular state after reaching seven states, but once recovered, it remains stable even as you reduce the total number of states.

In biological interpretation, the model first operates in an **unbiased manner**. The interpretation of each state comes after the model is fully pruned and refined. At this stage, biological meaning is assigned to the states based on their observed characteristics. However, there's a risk of **overfitting** to known biological stories. For instance, if a particular state isn't recovered until much later in the process (e.g., after discovering 75 states), it might be disregarded. This could mean missing an important biological feature simply because it wasn't obvious at the start.

In summary, selecting the number of states in HMMs involves balancing model complexity with interpretability. Techniques like **nested initialization** allow for a more exploratory and refined approach, discovering rare and informative states while avoiding redundancy. However, care must be taken not to overfit the model based on current biological knowledge, as previously overlooked states may hold significant value with further data or insights.

49:50 Learning chromatin states jointly across multiple cell types

When you have data from **multiple cell types**, it's possible to learn chromatin states not just for individual types but jointly across them, allowing for more powerful insights into the dynamic behavior of gene regulation. For example, consider a scenario where you have various cell types like **human vein endothelial cells, keratinocytes, fibroblasts, and embryonic stem cells**, and you're able to profile chromatin marks across each of them. The challenge becomes how to model these multiple cell types together.

There are several strategies you can adopt. One approach is to treat all the data from different cell types as

one large dataset. This is the **stacking approach**, where you stack all the chromatin marks from different cell types together, learning the chromatin states from this collective set. This allows you to capture chromatin states that may only appear in specific cell types, helping to identify **cell-type-specific enhancers** or **promoters**.

Alternatively, you can use a **concatenation approach**, where instead of stacking, you concatenate the data across all chromosomes for each cell type. This method treats each cell type's chromosomal data as a continuous genome, learning the same chromatin states for each cell type but with different dynamic locations. The advantage of this method is that it forces the model to use a common set of chromatin state definitions across cell types, making it easier to compare and track chromatin state changes between cell types.

A third strategy is to learn **independent models** for each cell type and then **cluster the emission matrices** to find common patterns across different types. This allows for a more granular view of how certain chromatin states are shared or unique between cell types. For example, you might discover that certain chromatin states (like promoter or enhancer states) are found consistently across many cell types, while others may only appear in specific tissues.

Once the chromatin states are learned, you can observe **dynamic transitions** in chromatin states across cell types. For example, a gene might be in a **poised state** in embryonic stem cells, showing both repressive and activating marks, but then shift to an **active state** in differentiated cell types. These dynamic shifts help in understanding how genes are regulated during development and differentiation.

Handling Missing Data and Imputation

A common issue when profiling multiple cell types is that certain marks might be missing from some cell types due to technical limitations or failed experiments. In these cases, one approach is to **impute missing marks** by using the information from other marks that are present in the same cell type. This imputation can be incredibly useful, especially when you are dealing with rare cell types or samples that cannot be easily replicated. Instead of discarding incomplete datasets, imputation allows for the recovery of potentially valuable data and enables a more complete model of chromatin state dynamics.

58:20 Three ways of linking enhancers to target genes

Once you have learned chromatin states across multiple cell types, the next step is to start linking **non-coding regions** (like enhancers) to their **target genes**. One method of linking is based on **correlation**. By observing when enhancers become active in certain cell types and correlating that with the activity of nearby genes, you can make functional inferences about which enhancers regulate which genes.

Another method is **genetics-based linking**, using **expression quantitative trait loci (eQTLs)**. eQTLs are genetic variants that affect gene expression. By studying how different genetic variants (e.g., single-nucleotide polymorphisms) influence gene expression, you can link genetic differences in non-coding regions to changes in the expression of specific genes. This is a powerful approach, as it uses both genetic and expression data to make connections between enhancers and target genes, often revealing previously unknown regulatory relationships.

Both of these methods—**correlation-based** and **genetics-based linking**—allow for a more comprehensive understanding of how **non-coding genetic variation** impacts gene regulation and ultimately contributes to different phenotypes, including disease states.

1:00:50 Three-dimensional Genome, Chromatin Conformation Capture, Hi-C

The third method for linking enhancers to their target genes is based on **physical interactions** within the **three-dimensional structure of the genome**. This is where techniques like **Chromatin Conformation Capture** (Hi-C) come into play.

To visualize this, imagine looking at regions of the genome, such as the region around the **FTO gene**, which is linked to obesity. Through Hi-C, we can observe **long-range physical interactions** between the FTO gene and other distant genes, such as **IRX3** and **IRX5**. These interactions indicate that certain genomic regions, though distant in terms of linear sequence, may fold and loop in three-dimensional space to come into close proximity.

The Concept of Chromatin Conformation Capture

Imagine taking a bowl of spaghetti (representing the genome), and you want to know which strands of spaghetti (or regions of DNA) are physically near each other. In the same way, chromatin is compacted in the cell nucleus, and different regions of the genome may come into contact as the DNA folds. To capture these interactions, you start by **fixing** the DNA in place using a chemical fixation agent, which essentially freezes the

spatial organization of the genome.

Next, you **chop** the genome into smaller segments and allow these fragments to randomly rejoin. Most often, these segments will rejoin with nearby regions, forming loops and re-ligations. However, some of these fragments will reconnect with more **distant regions** in the genome, because those distant regions were physically close due to the three-dimensional folding of the DNA. These distant re-ligations create **off-diagonal dots** in Hi-C interaction maps, representing long-range interactions between different genomic regions.

Hi-C and Chromatin Territories

Hi-C is a powerful technique that allows us to investigate the **spatial organization of the genome** within the nucleus. Inside the nucleus, chromosomes do not just float randomly. Instead, they occupy distinct **chromosomal territories**. Using Hi-C, we can identify these territories and observe how chromosomes are arranged relative to each other. By staining chromosomes with different colors (through combinatorial labeling), it becomes clear that **chromosomes occupy distinct, non-overlapping domains** within the nucleus.

Hi-C begins with **3C** (Chromosome Conformation Capture), evolves through **5C** (Carbon Copy Chromosome Conformation Capture), and culminates in **Hi-C**, which adds a biotin label to every cut and religation event, making it easier to pull down and sequence these regions. This technique allows for a **comprehensive view of the three-dimensional genome architecture** by mapping interactions between all regions of the genome.

Understanding Chromosome Interaction Maps

When the Hi-C data is plotted, you can see a **checkerboard pattern**. This pattern shows that different regions of the genome tend to interact with one another in a structured manner. The diagonal of the matrix represents interactions within close proximity (cis-interactions), while off-diagonal elements indicate long-range interactions (trans-interactions). Through this analysis, two main classes of chromatin emerge:

1. **Gene-rich regions**: These regions tend to cluster together and are often found in the **middle of the nucleus**.
2. **Gene-poor regions**: These regions are more likely to be located near the **nuclear periphery**, often associated with the **nuclear lamina**.

This insight reveals a spatial segregation between **active** and **repressed chromatin domains**.

Loop Extrusion and Topologically Associated Domains (TADs)

A key discovery from Hi-C is the identification of **Topologically Associated Domains (TADs)**, regions of the genome that interact more frequently within themselves than with other regions. These domains are formed by mechanisms such as **loop extrusion**, where **CTCF binding proteins** and **cohesin** create loops in the chromatin, bringing distant regulatory elements and genes into close proximity. This loop extrusion process helps to define boundaries between active and repressive chromatin regions, allowing for **spatial organization** of gene regulation.

Overall, Hi-C and chromatin conformation capture techniques provide a crucial tool for studying the **three-dimensional architecture of the genome** and for understanding how spatial interactions influence gene regulation. These methods allow us to map how the genome folds and loops to bring **regulatory elements**, such as enhancers, into physical proximity with their **target genes**, providing deeper insight into **gene expression** and **disease mechanisms**.

1:08:50 Linking Enhancers to Function, Motifs, Upstream Regulators

We can now **link non-coding regions** like enhancers to their **target genes** using various methods, including correlated activity, 3D genome structure, and genetic associations. For instance, by **unbiased clustering** of **2.3 million genomic regions** based on their enhancer activity across **127 epigenomes**, we can organize these regions into **modules**. These modules consist of enhancers that show similar activity patterns across different cell types.

Each genomic region that lights up as an enhancer in a given cell type can be represented as a **vector** of activity. This vector is 127 cell types long, indicating how **enhancer-like** that element is across all 127 types. By clustering these vectors, we can identify **enhancer modules** that are specific to certain cell types or show more generalized activity. For instance, some modules are highly **cell type-specific**, while others are active across a range of cell types, such as in **T-cells** or **embryonic stem cells**. These modules can also be linked to **specific functions**, such as neuronal development, immune response, or heart function, based on the nearby genes.

Once these modules are identified, we can investigate the **genes nearby**. By clustering the enhancers and

identifying the genes close to them, we can see whether these genes share **common functions**. For example, a group of enhancers active in **embryonic stem cells** and the **brain** may be enriched for **neuronal development genes**, while enhancers active in **T-cells** and **B-cells** might be enriched for **immune-related genes**.

Motif Enrichment and Upstream Regulators

A key question is whether these enhancers turn on **randomly** or if they share common **regulatory motifs** that drive their activity. By analyzing the **motif enrichments** of these enhancer modules, we can identify the **regulators** that bind to these motifs. These regulators might be transcription factors that are **expressed** in the same cell types where the enhancers are active. This allows us to link **enhancers** with **upstream regulators** and identify how gene regulation occurs across the genome.

For example, we can study the **motif enrichments** within enhancer modules to see which transcription factors are likely to be binding to these regions. Then, by linking this information with gene expression data, we can determine whether the corresponding **regulators** are expressed in the relevant cell types. This provides a powerful tool to connect **enhancer activity**, **regulatory motifs**, and **gene expression**.

Correlating Enhancers and Gene Expression

We can also study the **correlation** between **enhancer activity** and **gene expression** directly. By analyzing enhancer marks such as **H3K27 acetylation** (a mark of active enhancers) and comparing it to gene expression data, we can predict which genes are regulated by which enhancers. For instance, if an enhancer's activity is strongly correlated with the expression of a nearby gene, we can hypothesize that the enhancer is regulating that gene.

By systematically analyzing the **enhancer-gene links** across the genome, we can start building a **regulatory map** that connects enhancers to their **target genes** and identifies the **regulatory motifs** and **transcription factors** that control these processes. This integration of chromatin state, motif analysis, and gene expression data allows us to uncover the **regulatory circuits** governing gene expression in different cell types and contexts.

1:14:30 Distinguishing Upstream Activator vs. Repressor TFs

We can uncover a **three-way correlation** between the **activation of an enhancer region**, the **enrichment of corresponding motifs**, and the **expression of the associated regulator**. This enables us to distinguish **activators** from **repressors** among transcription factors (TFs).

For each **module of enhancers**, which might be active in different cell types, such as stem cells, immune cells, or hepatic cells, we can ask: What is the **enrichment** (red) or **depletion** (blue) of a specific **motif** in those regions? A **depletion** of a motif might indicate that a **repressor** TF binds to that motif and prevents the region from becoming active. Thus, when the motif is absent, the enhancer is **active**, implying that the absence of the repressor allows the enhancer to turn on.

We can use this information to identify **activators** and **repressors**. For example:

- **Activators** are identified when a motif is **enriched**, and the associated factor is **expressed**, leading to enhancer activity.
- **Repressors** are identified when the motif is **depleted**, and the factor is expressed, or conversely, when the motif is **enriched**, but the factor is **not expressed**.

For instance:

- **OCT4** and **RFX** appear as **activators** in **embryonic stem cells**.
- **GATA1**, **SPI1**, and **STAT** factors act as **activators** in **immune cells**.
- **GFI1** functions as a **repressor** in immune cells.

Building Regulatory Networks

This insight allows us to **build regulatory networks**, linking enhancer modules to their transcription factors. By identifying motifs associated with specific enhancers, and correlating them with the activity of corresponding TFs, we can infer how **activators** and **repressors** control gene expression across different cell types. For example, in immune cells, **ETS1** and **SPI1** (also known as PU.1) activate immune-specific enhancers, while **GFI1** acts as a repressor.

High-Resolution Motif Localization

We can visualize how transcription factor binding motifs are distributed around specific genomic features, like

the **transcription start site** (TSS), and how chromatin states differ around active motifs. For example, the **HNF4** motif (hepatocyte nuclear factor 4) in **hepatocytes** (liver cells) is surrounded by a strong **enhancer state** in hepatic cells, and the **repressed chromatin state** is depleted around this motif. This provides a **high-resolution map** of regulatory motifs and their surrounding chromatin context.

Functional Impact of Motif Changes

By conducting experiments like **luciferase assays**, we can validate the functional significance of specific motifs. For example, altering a single base pair in the **HNF4 motif** results in a significant reduction in gene expression, demonstrating the critical role of specific motifs in enhancer function. While motifs don't act in isolation, changing one can dramatically affect gene regulation in the context of other nearby motifs.

This systematic approach allows us to integrate **motif enrichment**, **chromatin state data**, and **gene expression** to understand the complex regulatory networks controlling gene expression across different cell types and biological contexts.

1:20:10 Summary

In today's lecture, we explored the **dynamics of the epigenome** and discussed approaches for **joint chromatin state learning** across multiple cell types. We delved into **hidden Markov models (HMMs)**, examining both **supervised** and **unsupervised learning** methods for modeling chromatin states. In the supervised approach, we simply **count** transitions and emissions, while in the unsupervised method, we first **infer** hidden states and then update our model parameters iteratively. We explored the two unsupervised techniques: **Viterbi decoding**, which relies on the **best path**, and **Baum-Welch**, which sums over **all possible paths**.

We also covered the process of **annotating chromatin states**, where we assign meaningful biological labels to states based on **enrichments** for known genomic features, such as promoters, enhancers, and repetitive elements. We discussed **model complexity** and how increasing the number of states captures **pairwise dependencies** between chromatin marks. This led to the concept of **nested initialization**, where models are built with a larger number of states and later **pruned** to remove redundant or less informative states, resulting in a more **robust model**.

Next, we covered three different ways to learn chromatin states **jointly across multiple cell types**:

1. **Stacking** the chromatin marks for each cell type.
2. **Concatenating** the genomes from different cell types.
3. Learning **independently** in each cell type and later aligning the states across cell types.

We then examined how we can **link regulatory elements** like enhancers to their target genes using three strategies:

1. **Correlated activity** between enhancers and nearby genes.
2. **Genetic information** (e.g., expression quantitative trait loci or eQTLs).
3. **Physical proximity** revealed through **chromatin conformation capture** techniques, such as **Hi-C**, which allows us to understand **three-dimensional genome architecture** by analyzing regions of the genome that physically interact.

Lastly, we discussed how to identify **activators** and **repressors** by examining the correlation between enhancer activity, the **motifs** present or absent within enhancer regions, and the **expression** of the transcription factors that bind those motifs. This allowed us to **map regulatory networks** and infer how transcription factors control gene expression across different cell types.

We'll continue exploring the **theme of networks** in the next lecture, which will further expand on how these regulatory circuits form complex biological networks.

Lecture 7 - Regulatory Networks

Video: Lecture 7 - Regulatory Circuitry and Networks - MLCB24

Slides: Lecture07_RegulatoryNetworks.pdf

0:00 Intro to regulatory motifs / gene regulation

Welcome to today's discussion on **regulatory circuitry and networks**, where we'll explore how regulatory motifs function as the foundational elements of gene regulation. This is the final part of our first module, and we will begin by delving into **regulatory motifs**—the key molecular mechanisms underlying the regulatory edges

we see in genetic networks.

When we talk about gene regulation, the edges connecting regulator A to gene B aren't just abstract lines in a network diagram. They represent **physical interactions** on the genome, typically at specific sequences called **regulatory motifs**. These motifs are short DNA sequences that bind transcription factors (TFs) or other regulators, allowing control over gene expression.

Motifs are generally located in larger **regulatory regions** like promoters and enhancers, which **compete with nucleosomes** (protein complexes that package DNA) to keep the DNA accessible. This competition is **biophysical**, not orchestrated by any higher-level signaling, but driven by the **probability of regulator binding** and detachment. In these regions, **multiple motifs** often work together, reinforcing each other's binding through their interactions with nucleosomes and protein-protein contacts. This makes regulatory regions much more likely to be functional than motifs isolated in the middle of genomic deserts.

How Regulators Bind to DNA

Regulators don't bind by pulling the DNA strands apart and reading the sequence in a straightforward way. Instead, they "**feel**" the DNA from the side. The bases A, T, G, and C have unique **molecular signatures** that can be recognized by the transcription factors from the edges of the DNA helix, without needing to disrupt the double strand.

If a regulator binds to a **single side of the DNA**, it will recognize a **continuous motif**. However, if the protein binds two sides, the motif may have a **gap** or spacer region in between. In cases where the regulator is a **homodimer**—where two identical proteins bind together—it creates a **palindrome** (a sequence that reads the same in both directions), but in reverse on the opposite DNA strand due to the complementary base pairing.

Motifs and Disease-Associated Variants

These motifs are not just abstract sequences; they represent critical points of control in the genome. When **genetic variation** disrupts a motif, it can have significant consequences, such as causing **disease**. Mutations within regulatory motifs may alter transcription factor binding, leading to misregulation of gene expression. This regulatory disruption is a common cause of **non-coding genetic variants** associated with disease.

A striking example comes from work on **obesity** genetics, where a single-nucleotide variant disrupted a conserved motif involved in energy balance. This **regulatory variation** results in individuals being less able to burn calories, leading to an average gain of 10 pounds in homozygous carriers. This demonstrates the powerful effects that small changes in regulatory motifs can have on biological processes like **thermogenesis**.

In summary, today's lecture will explore **how regulatory motifs govern gene expression**, how to discover them, and how to analyze the networks that emerge from these interactions.

6:18 Motifs (generative model) vs. instances (single occurrence)

Motifs vs. Instances: Understanding the Difference

When we talk about **motifs**, it's essential to distinguish between the **concept of a motif** and an **instance** of that motif. A **motif** is a **generative model**—a probabilistic representation that describes a recurring pattern, such as a sequence of DNA, RNA, or protein that can bind to regulators. This model captures the likelihood of certain nucleotides or amino acids appearing at each position. In contrast, an **instance** is a **specific occurrence** or realization of the motif in a sequence.

A **generative model** allows us to predict or generate potential sequences by sampling from a distribution. For instance, a motif may specify that a position has an equal likelihood of being an A or G, but C is more likely to appear than T at another position. These motifs are typically represented using **position weight matrices (PWM)**, which describe the frequency of nucleotides across multiple observed instances. By lining up all instances where a regulator binds, you can create a PWM that captures the statistical preference for each base at every position.

For example, in the **abf1 regulator**, there are certain highly conserved positions such as TCA in the motif, and other positions that are more flexible, like the first base that can be either A or G. The **motif's structure** also informs how the regulator binds to the DNA. Some motifs may have variable **spacing** between parts of the sequence, or depend on a specific **arrangement of nucleotides** due to how the protein interacts with the DNA.

Motif Independence and Dependencies

A key assumption made by position weight matrices is that **nucleotide positions are independent**—the probability of having a particular base at one position tells you nothing about the probability at another.

However, in many cases, this isn't true. Some motifs exhibit **dependencies between positions**, where having a particular base influences the likelihood of seeing another base nearby. In those cases, you might split the motif into multiple PWMs to capture these dependencies.

Similarly, motifs may vary in **spacing**. If the binding of a regulator allows flexibility in the distance between two parts of a motif, you might create several versions of the PWM to account for this variable spacing.

Motifs Beyond DNA

While motifs are most often discussed in the context of **DNA-binding** for gene regulation, they can also exist in **RNA** (e.g., determining whether a transcript will be spliced) and **proteins** (e.g., motifs for post-translational modifications like phosphorylation). In each case, the concept of recurrent patterns holds, allowing motifs to function across different biological levels.

Discovering Motifs

Motifs can be discovered using several methods. One approach is **expectation maximization** or **Gibbs sampling**, where you iteratively refine a PWM by aligning multiple sequences. Another method involves **enumerating** potential motifs by scanning the genome for specific sequences. Alternatively, **evolutionary conservation** can help identify functional motifs by focusing on sequences preserved across species. You can also leverage **protein domain knowledge** to infer what sequences a protein is likely to bind based on its structure (e.g., **zinc finger domains** or **helix-loop-helix domains**).

Experimental Methods for Motif Discovery

Experimentally, motifs can be discovered through techniques such as **selection assays**. In these assays, DNA fragments that bind to a protein of interest are progressively enriched through multiple rounds of selection, allowing you to identify the bound fragments. Another approach uses **DNA microarrays**, where short DNA sequences are synthesized on a chip, and proteins are washed over the surface. By detecting which sequences bind to the protein, researchers can identify motifs experimentally.

In summary, **motifs** are powerful representations of recurring sequences in gene regulation, and they can be discovered through computational and experimental means. By understanding both the **generative model** (motifs) and their **instances**, we can gain insights into how regulators control gene expression at the molecular level.

12:40 Regulatory Genomics Challenges: Motifs - Regulators - Targets

When approaching regulatory genomics, there are several key elements and relationships to consider. First, we have **the motif**, which is the **generative model** representing the recurring sequence pattern that binds specific regulators. Next, we have **individual instances** of that motif, which are the specific occurrences of the motif in the genome. Finally, there is the **regulator** that physically binds to a given instance of the motif and has an **affinity** for the broader generative model.

Understanding regulatory genomics involves answering several core questions:

1. **How do we discover regulators?**
    - One approach is to identify proteins with **homology** to known regulators, suggesting they may also have regulatory functions. Once identified, they can be tested experimentally.
2. **How do we discover motifs?**
    - **Comparative genomics** can help by highlighting **conserved regions** across species. Within these conserved regions, common sequence patterns (motifs) can be identified, allowing for de novo motif discovery.
3. **How do we find individual binding sites (instances) for a motif?**
    - Given a motif, matching sequences in the genome can be found, but not all matches will be **functional**. The **chromatin context**—whether the region is accessible or actively bound by regulators—plays a critical role in determining functionality.
4. **How do we identify the targets of a regulator?**
    - From a known regulator, you can find its targets using **binding experiments** such as **ChIP-seq**, which identifies regions in the genome where the regulator binds. These regions are potential target sites for regulation.
5. **How do we find the regulator for a given motif?**

- If a motif is known, but not its corresponding regulator, you can experiment by generating multiple variants of the motif, allowing them to interact with various proteins. After isolating the proteins that bind to the motif, **mass spectrometry** can be used to identify them.

6. **How do we find a motif from a collection of target sequences?**
    - If you have a set of target regions bound by a common regulator, you can align these sequences and search for recurring patterns. This process helps in identifying the motif that binds the regulator.

To summarize, the main challenge in regulatory genomics is navigating between these entities—motifs, instances, regulators, and targets—and determining their relationships. Whether you start with a **regulator**, a **motif**, or a set of **targets**, different experimental and computational approaches allow you to uncover the missing links between these elements, helping to map the regulatory networks governing gene expression.

17:40 Enrichment-based motif discovery

After conducting a **ChIP-seq experiment** and identifying a set of genomic regions bound by the same regulator, the next challenge is to discover the **motif** that is common to these regions. Similarly, if you've performed an **expression analysis** and identified genes that co-express under the same conditions, it's likely that they share common regulatory elements in their **promoter regions**. In either case, the goal is to search for **common motifs** that indicate shared upstream regulation.

Once you've identified regions of interest—whether through ChIP-seq, expression analysis, or protein binding microarray experiments—you generally don't have the exact sequence of nucleotides bound by the regulator. As a result, **motif discovery** becomes necessary. Here are some approaches for motif discovery:

1. **Local Alignment**: One approach is to look for local sequence alignments across the regions to identify common patterns. These regions may be otherwise unrelated but share similar **motif sequences** that are responsible for regulatory binding.
2. **Expert Knowledge**: Sometimes, knowledge of known regulatory motifs or binding sites can guide the search. Using a **database of known motifs**, you can compare the discovered regions with well-characterized motifs to find matches.
3. **Enumeration**: You can systematically search for motifs by enumerating **k-mers** (e.g., 5-mers, 6-mers, or 7-mers) and looking for sequences that occur frequently across the regions. The most common sequences are likely candidates for motifs that the regulator binds to.
4. **Evolutionary Conservation**: Another powerful approach is to restrict the search to **conserved regions** across species. This is based on the idea that functional regulatory motifs are under **evolutionary pressure** and will be conserved in orthologous sequences, even if the surrounding regions vary. However, this approach has its limitations because motifs can exhibit **degeneracy**—allowing for sequence variation such as **A or G** or **C or T** in some positions—making evolutionary conservation imperfect for motif discovery.

If a motif is **known**, the process is simpler: you can scan the genome for **instances** of that motif by matching the sequence at every position. If the motif is not fully characterized, an **iterative approach** can be used to refine the motif. By aligning the discovered instances and re-evaluating the frequencies of nucleotides at each position, the motif model can be continuously improved.

This iterative refinement allows for increasingly accurate **position weight matrices** (PWMs), which can then be used to scan for further instances of the motif across the genome.

20:53 Expectation Maximization for Motif Discovery - starting positions - motif definitions

In the process of **motif discovery**, **expectation maximization (EM)** is a key iterative approach used to improve both motif definitions and motif instance detection. The basic idea behind EM is to use an initial **motif definition** to search for possible instances of that motif across a genomic region, then refine the motif based on those detected instances, and continue cycling through this process to get progressively better results.

To start, once we have a **motif definition**, we can use it to search for possible instances across a genome. At each position where a potential motif instance appears, we can calculate the **likelihood** that the given DNA sequence was generated by the motif model versus being generated by the background. This is expressed as a **likelihood ratio**—comparing the likelihood that a sequence belongs to the motif versus a random background sequence.

Once potential motif matches are found, the next step is determining how to handle **uncertainty**. Do we

always take the most likely match, even if the likelihood is only slightly higher than the alternatives? Or should we consider all possibilities?

Here's where various methods come into play:

- **Greedy method**: Always selects the maximum likelihood match, simplifying the process by focusing only on the most likely candidate.
- **Expectation maximization**: Considers all possible motif instances, but weighs them according to their likelihoods. This method uses a **weighted average** to refine the motif, considering all matching positions, which helps avoid bias but may slow down convergence.
- **Gibbs sampling**: Involves a probabilistic selection where a candidate is chosen based on its likelihood. For example, a motif with 70% probability will be selected 70% of the time, offering a more exploratory approach.

The key challenge with EM is navigating the **landscape of possible motifs**. This involves adjusting the motif model iteratively, trying to discover the best motif sequence. **Expectation maximization** typically converges at a **local maximum** based on the starting point, while **Gibbs sampling** is more flexible, sometimes jumping between different solutions, potentially helping avoid local maxima.

A critical strategy in EM is to use **multiple initializations**, starting from different positions in the sequence space. This helps increase the chances of finding the **global maximum** motif rather than getting stuck in a local optimum.

Ultimately, all of these methods aim to balance **accuracy** with **speed of convergence**, refining the motif model step-by-step until it captures the true underlying regulatory sequence pattern.

This EM approach is analogous to the iterative processes we've seen in **K-means clustering**, **hidden Markov models**, and **Baum-Welch training**, where machine learning is used to refine models iteratively based on data patterns.

26:58 Deep Learning and Convolutional Neural Networks for Motif Discovery

Deep learning has recently emerged as a powerful tool in the field of **genomics**, providing new ways to understand genomic sequences through **convolutional neural networks (CNNs)**. The same principles that allow CNNs to detect features in images—like lines, edges, and eventually objects—can be applied to genomics, where raw sequence data can be transformed into progressively more abstract representations.

In this approach, CNNs can **learn motifs** from scratch, just as they learn filters or patterns in image data. The process involves feeding the network a series of nucleotide sequences and training it to predict specific outcomes. For example, using experimental data, the model can be trained to predict whether a given sequence is **bound by a specific regulator**, such as **ABF-1**. You provide the CNN with labeled data—sequences known to be bound or not bound by a particular regulator—and the network learns which patterns in the sequence lead to binding.

A significant advantage of CNNs is their ability to scale beyond predicting a single regulatory binding site. You can implement **multitask learning**, where the network is tasked with predicting the binding of **multiple regulators** simultaneously. This involves creating a vector of tasks for each sequence, where each task corresponds to whether a specific regulator binds to the sequence. The **shared representations** learned by the network can capture common motifs and patterns that are useful across multiple regulators, making the model more generalizable and versatile.

By learning in this layered fashion, CNNs can capture increasingly complex representations. At the **base layer**, the network might learn individual motifs. In the layers above, it might start combining those motifs into **motif parts** or more intricate **combinations of motifs** with varying spacing. This is analogous to how the structure of a protein, such as a **zinc finger**, can be decomposed into functional parts, and the CNN can learn the building blocks at the lower layers before assembling them into full motifs at higher layers.

Another powerful feature of deep learning is its ability to not only predict whether a sequence is bound or not, but also to predict the **shape** or **signal intensity** of the binding at very high resolution. The model can be trained to handle multiple types of data simultaneously, such as **regulator binding**, **nucleosome positioning**, and **histone modifications**. This versatility allows the network to detect more complex patterns, incorporating various regulatory elements like **splicing sites** or **initiator factors**.

Traditionally, motif discovery involves fishing for motifs that explain experimental results, such as determining which sequences are bound by a known regulator. Deep learning extends this by allowing for more complex

joint learning tasks, such as predicting multiple types of genomic signals in tandem. This enables a deeper understanding of genomic regulation and opens up new possibilities for exploring **gene regulation** in a more holistic and integrated manner.

32:03 Evolutionary signatures for de novo motif discovery

De novo motif discovery using **evolutionary signatures** leverages the principle that **conserved genomic regions** often harbor **functional elements** like regulatory motifs. The idea is straightforward: within regulatory regions, some DNA sequences are **conserved across species**, indicating functional importance. These conserved regions often correspond to **motif instances**, but finding individual instances is not enough; you need many instances to infer a **motif**.

In this method, we focus on identifying **conserved islands** across the genome and then search for **sequence patterns** that explain those islands. This involves discovering **recurrent patterns** that are conserved in multiple regions of the genome, which would indicate the presence of a regulatory motif.

For instance, consider the well-known **Gal4 motif** involved in regulating genes like **GAL1** and **GAL10**. By identifying conserved instances of the Gal4 motif across the genome, we can compare these conserved motifs to **randomly shuffled sequences** to distinguish true motifs from background noise. Shuffling, however, must be done carefully—maintaining certain properties of the original motif (e.g., **dinucleotide frequencies**) ensures that the shuffled sequences are reasonable controls.

The **shuffling process** plays a critical role in motif discovery, as the way you shuffle affects which **distinguishing features** you detect. For example, if you shuffle nucleotides randomly without considering spacing or other contextual factors, you might miss important regulatory signals.

Once these conserved motifs are identified, we can investigate their **conservation patterns** in different genomic contexts, such as intergenic versus coding regions, or upstream versus downstream of genes. For example, **Gal4 motifs** are often more conserved in **intergenic regions**, where they perform their regulatory functions, than in **coding regions**.

This approach allows for **systematic motif discovery**. By **enumerating sequence patterns** with gaps and looking for those that are highly conserved, we can build a catalog of **motifs**. This strategy has been applied across various species, such as yeast, humans, flies, and mammals, leading to the discovery of both **known motifs** and **novel motifs**, many of which were validated experimentally.

By using **evolutionary signatures**, this method enables **de novo motif discovery** without any prior knowledge of the regulatory protein or experimental data. It uncovers conserved regulatory elements across species, expanding our understanding of **gene regulation** at a fundamental level.

42:11 Evolutionary signatures for motif instance identification

Evolutionary signatures can also be used to identify **individual motif instances** with high confidence. After discovering motifs, the next step is to pinpoint **specific occurrences** of these motifs in the genome that are functionally relevant. This is where **phylogenetic branch length** becomes crucial.

To identify **high-confidence motif instances**, we assess the **phylogenetic branch length** over which the motif is conserved. The key idea is to tolerate minor **mutations** (e.g., an A or G in certain positions) and **gaps** that occur due to insertions or deletions in alignments. However, we focus on **long branches** in phylogenetic trees, which provide more informative signals of conservation compared to **short branches** where mutations are less frequent.

We start by searching sequences independently and calculating a **branch length score** for each motif instance. This score reflects how conserved the motif is across species, measured in terms of **substitutions per site**. Even with multiple mutations, if a motif is highly conserved across many species, the **total branch length** may exceed one substitution per site, indicating that the motif is real.

Next, we generate **shuffled motifs** as controls, ensuring these shuffled versions maintain properties such as length and nucleotide composition similar to the real motif. By comparing the conservation of real motifs against shuffled motifs, we can measure **motif-specific conservation enrichment**. This comparison tells us how much more conserved the motif is than expected by chance, given its length and genomic distribution.

The results are visualized by plotting **confidence scores** based on the ratio of real to shuffled motif conservation. **Higher confidence scores** are associated with greater branch lengths, indicating strong conservation across species, while lower confidence scores suggest random or weak conservation.

As we increase our confidence, we find that **transcription factor motifs** tend to be enriched in **promoters**,

while **microRNA motifs**—which act on mRNA—are enriched in the **5' untranslated regions** (5' UTRs) of genes. This correlation between motif confidence and known biological functions reassures us that the evolutionary conservation we are detecting is meaningful.

The key takeaway is that we now have a **universal confidence scale** for motif identification, allowing us to compare motifs of different lengths and types using the same metric. This confidence score can be applied across all motifs, making it a powerful tool for identifying functionally relevant **motif instances**.

Finally, this method enables us to predict **individual targets** of motifs with high precision. The combination of **motif discovery** and **instance identification** using evolutionary conservation provides a comprehensive framework for mapping **regulatory elements** across the genome.

50:44 Network Analysis Section Overview

We are transitioning from motif and regulatory element discovery to understanding how to **analyze the resulting networks** that emerge from knowledge about regulator bindings. For example, if we know that a certain **regulator binds** to multiple locations in the genome, we can begin to connect these into a **regulatory network**. Similarly, we might understand how other types of networks, such as **metabolic** or **signaling networks**, are structured.

The goal is to understand how to **analyze and interpret** these networks. We will explore several different **types of networks**, such as:

- **Regulatory Networks**: Networks that depict the interactions between transcription factors and the genes they regulate.
- **Metabolic Networks**: Networks of biochemical reactions within a cell.
- **Signaling Networks**: Networks that represent pathways where information flows, such as signaling cascades triggered by external stimuli.
- **Interaction Networks**: Networks of protein-protein or protein-DNA interactions.
- **Functional Networks**: Networks where nodes represent biological entities (genes, proteins, etc.), and edges indicate functional relationships.

Next, we will examine both **Bayesian probabilistic views** and **algebraic views** of networks to understand the **structure** and **functionality** of these networks. This will also involve an introduction to **network centrality**, which helps measure the importance or influence of nodes within a network.

The discussion will include a **linear algebra review**, covering concepts such as **eigenvalues**, **singular value decomposition**, and **low-rank approximation**. These mathematical tools are critical for **dimensionality reduction** techniques like **Principal Component Analysis (PCA)**, which help us uncover patterns in complex network data.

Finally, we will introduce **Graph Neural Networks (GNNs)**, a modern machine learning approach for analyzing and predicting properties of networks. GNNs are especially useful for working with **non-Euclidean data** such as graphs, which don't fit into traditional neural network architectures. Although GNNs will be introduced here, they will be further explored in **subsequent lectures**.

This section will provide the foundation for performing in-depth **network analysis** across various biological and computational domains.

51:55 Network types: regulatory, metabolic, signaling, interaction, functional

The **organization of living systems** is structured in layers. At the base is the **genome**, which is modulated by the **epigenome** (discussed in previous lectures), and the genome also produces the **transcriptome** (mRNAs, microRNAs, etc.). The **transcriptome** then translates into the **proteome** through the process of protein synthesis. We can begin to understand **regulatory networks** at each of these levels:

1. **Gene Regulatory Networks**: These are networks that capture the interactions between transcription factors and their target genes, representing **pre-transcriptional regulation**. This includes understanding which **regulator binds** to a region of DNA and **activates or represses** a particular gene.
2. **Post-transcriptional Regulation Networks**: These focus on how elements like **microRNAs** regulate gene expression **after transcription** by influencing processes such as gene stability, degradation, and localization of transcripts.
3. **Protein-Protein Interaction Networks**: Proteins that **physically interact** with one another can form

complexes that either help bind to other proteins or regulate certain pathways.

4. **Signaling Networks**: These depict **signaling cascades**, where one protein modifies another (e.g., phosphorylation), leading to a chain reaction that eventually affects gene expression. For example, a protein might phosphorylate another, activating a cascade that leads to the expression of genes.
5. **Metabolic Networks**: These focus on how different **metabolites** are transformed into one another via enzymatic reactions. The **nodes** in these networks are metabolites, while the **edges** represent the enzymes that catalyze transformations between them.

Characteristics of Network Types:

- **Gene Regulatory Networks**: These are typically **directed** and **signed** (indicating either activation or repression) networks. They often show **weighted edges**, where the strength of activation or repression between nodes may vary.
- **Metabolic Networks**: These are usually **bidirectional**, where an enzyme can catalyze both forward and reverse reactions. However, some reactions might be **irreversible**. In these networks, **nodes** are metabolites, and **edges** represent enzymes.
- **Signaling Networks**: These are **directed** and **unweighted** networks, as phosphorylation or signaling events are binary (either occurring or not).
- **Protein-Protein Interaction Networks**: These are **undirected** and **unweighted** networks, where a physical interaction between two proteins is symmetrical (if protein A interacts with protein B, then protein B interacts with protein A).
- **Functional Networks**: In contrast to physical interactions, these represent **functional relationships** such as **coexpression** of genes. The edges in functional networks represent **correlations**, and the networks are typically **undirected** but **weighted** based on the strength of these correlations.

Cross-network Interactions:

There is a substantial amount of **cross-talk** between these networks. For instance:

- A **signaling cascade** might activate a regulator that turns on a gene, which then produces a protein that participates in the regulatory or signaling networks.
- A **gene regulatory network** may control the transcription of enzymes that catalyze reactions in a **metabolic network**, and the metabolites produced may serve as **signals** in a **signaling network**, completing the cycle of regulation.

These **interconnections** between networks illustrate the complexity of biological systems, where events at one layer can influence others, contributing to an integrated system of regulation and function.

57:20 Network applications and challenges

We face several challenges when working with networks, such as **finding regulators**, **discovering motifs**, and **identifying target genes**. Beyond these, we also want to understand how **networks themselves predict cellular activity**. This is where **graph neural networks** (GNNs) come in. GNNs allow us to learn **representations from networks** to predict specific functions.

For example, given a chemical compound, we might want to predict its effect on a biological system, such as whether it has a **detrimental effect on E. coli** (helping patients by killing the bacteria). We can also use this approach to predict the **properties of molecules** or **proteins** from their network structures.

Predicting Functions with Graph Neural Networks:

GNNs help predict the **function of a protein** based on its network. For instance, a protein could be part of a **regulatory network**, a **signaling molecule**, or an **enzyme**. By analyzing the **graph structure**, we can infer these functions. GNNs are also used to **infer networks from functional data**, such as gene expression patterns, helping us determine which genes are **correlated** or **interacting** based on the observed data.

Structure Analysis of Networks:

Network analysis also enables us to explore the **vulnerabilities** or key points of networks. For example, in the case of a viral attack, we might want to know the most **vulnerable hubs** where the virus attacks the host or bacterium. Networks have **hubs**, which are highly connected points, and **network motifs**, which are recurring patterns of connections. For instance:

- **Feedforward loops** occur when regulator A controls both B and C, and B also controls C.
- **Feedback loops** involve mutual regulation among multiple entities, such as when A regulates B, B

regulates C, and C regulates A.

We can also **discover functional modules** within networks, which are sub-networks that are **densely connected**, revealing higher levels of cooperation or coordination within certain areas of the graph.

Network Types:

We distinguish between different types of networks:

- **Physical networks** involve direct physical interactions, such as binding between proteins.
- **Relevance or functional networks** describe how **similar** entities are in function (e.g., coexpression networks where genes exhibit similar expression patterns without necessarily interacting physically).
- **Probabilistic networks** capture **dependencies** between nodes. For example, observing one variable can change the probability distribution of other variables based on the **flow of information** through intermediate nodes.

In **probabilistic networks**, we can also explore **conditional independence**. If certain intermediate variables are fully observed, then nodes that previously depended on one another may become independent. This helps us understand how information propagates through the network and how it changes when new observations are introduced.

By understanding these networks and their structures, we can **predict functions, discover new motifs, and analyze vulnerabilities**, offering powerful insights into **biological systems**.

1:02:27 Matrix Representation of Networks and Linear Algebra

We can represent **networks** as **matrices**. For example, a **graph** where node **A** is connected to node **B** with a weight of 1.5 can be represented in an **adjacency matrix**. In the case of an **undirected network**, this adjacency matrix is **symmetric**. The matrix contains entries that reflect the weights of the edges between nodes, allowing us to mathematically capture the network structure.

Degree of Nodes in a Network:

We can also calculate the **degree** of a node, which is the number of edges connected to it. In a **weighted network**, the degree is simply the **sum of the weights** of all edges connected to a node. For instance, node **B** in our network may have a **degree of three**, meaning it is connected to three other nodes. In a **directed network**, we distinguish between the **in-degree** (edges coming into the node) and **out-degree** (edges going out from the node). This distinction helps identify **in-hubs**, which aggregate a lot of information, and **out-hubs**, which propagate a lot of information.

Matrix Operations on Networks:

Once we have a matrix representation of a network, we can perform various **matrix operations** to understand how information flows through the network. Applying the **adjacency matrix** to a **vector of starting weights** (such as concentrations of metabolites or initial states of nodes) shows how these values propagate through the network.

In this context, the **matrix multiplication** transforms the **vector**, which could represent initial conditions at each node. For example, if we have an initial value of **7 at node A** and a **rate of 2** for the connection between nodes, the multiplication propagates the value through the network based on these rates.

Iterative Matrix Applications:

By applying the matrix multiple times (e.g., **A × vector**, **A^2 × vector**, **A^3 × vector**), we can see the **cumulative effect** of repeatedly applying the network transformation. This iterative process shows how the network evolves over multiple steps and allows us to compute **transitive closure**, representing the cumulative effect of applying the transformation repeatedly.

Eigenvectors and Eigenvalues:

Eigenvectors and eigenvalues are important in **network analysis**. They help us understand the long-term behavior of matrix transformations by identifying **principal directions** along which transformations act. This is particularly useful in identifying **central nodes** in the network or understanding **steady-state distributions** when the network stabilizes after multiple iterations of transformation.

This formalism of representing networks using matrices enables a wide range of **computational techniques** to analyze their behavior, simulate signal propagation, and extract key structural insights.

1:06:03 Eigenvalues, Eigenvectors

In network analysis, **eigenvectors** are vectors that remain aligned in the same direction when a matrix

transformation is applied, but their magnitude is scaled by a scalar factor called an **eigenvalue**. This concept is essential in understanding how networks behave and in simplifying complex data representations.

What are Eigenvectors and Eigenvalues?

Every **matrix** transformation alters the space on which it is applied, shifting or stretching vectors. However, certain **eigenvectors** remain in their original direction, and their length is multiplied by a constant factor—this factor is the **eigenvalue**. In other words, applying the matrix to an eigenvector results in a scaled version of the same vector, represented mathematically as: $A \cdot v = \lambda \cdot v$, where A is the matrix, v is the eigenvector, and λ is the eigenvalue. This means that instead of shifting or rotating the vector, the matrix only stretches or shrinks it.

Why are Eigenvectors and Eigenvalues Important?

Eigenvalues and eigenvectors allow us to **decompose complex systems** and identify **underlying patterns**. For example, in gene regulatory networks, a large set of genes may exhibit highly correlated behavior. By finding the **eigenvectors** of this matrix (the gene expression data), we can reduce the system to a smaller set of "eigen-genes" that capture the most important trends in the data.

This approach helps us:

- **Simplify complex data**: Instead of analyzing hundreds of genes individually, we can analyze a few **principal components** that capture the most significant variations in the system.
- **Reveal patterns of variation**: Eigenvectors can represent large **modules** of co-expressed genes, providing insights into how groups of genes behave under different conditions (e.g., how a set of genes responds to changes in brain vs. liver tissue).

Geometric Interpretation:

From a **geometric perspective**, eigenvectors represent **directions** in which the transformation acts by simple scaling, rather than more complex rotations or shifts. We can think of matrix transformations as altering the space in multiple dimensions, and **eigen-decomposition** allows us to break down that transformation into simpler operations:

1. **Rotation** of the matrix into a different space.
2. **Scaling** the eigenvectors within that transformed space.
3. **Rotation** back to the original space.

This **decomposition** is useful for making complex transformations more understandable and manageable.

Eigen-Decomposition and Principal Component Analysis (PCA):

By decomposing a matrix into its **eigenvectors** and **eigenvalues**, we can start to uncover the **principal components** of variation in a system. Principal component analysis (PCA) uses this approach to reduce the dimensionality of a dataset, focusing on the most significant sources of variation. For example, instead of analyzing each gene individually, we can analyze the **principal axes** of variation in gene expression data to understand the main trends that explain differences between conditions or tissues.

Eigen-decomposition is a powerful tool in network analysis, helping us break down complex structures into simpler, more interpretable components.

1:09:19 Principal Components Analysis PCA, SVD

Principal Component Analysis (PCA) is a fundamental technique used in data reduction and pattern discovery, particularly in the context of gene expression and other high-dimensional biological data. Here's a breakdown of the key concepts:

Principal Component Analysis (PCA)

PCA helps to transform a large, complex dataset into a simpler representation by identifying the principal components—directions in the data along which the variation is maximized. These components allow for more efficient exploration and interpretation of the data. In biological systems, this might mean reducing thousands of gene expression measurements to a few meaningful variables.

Steps in PCA:

1. **Variation Across Genes:** You start with a dataset where gene expression levels vary across different conditions or samples. These variations can be described in terms of different variables or gene expression patterns.
2. **Transformation of the Space:** Instead of focusing on the raw expression of individual genes, PCA rotates the data space to find new axes, or "principal components," that capture the most significant

patterns of variation.

3. **Dimension Reduction:** By finding the principal components (often just a few), you reduce the complexity of the dataset. For example, instead of focusing on the expression of 20 individual genes, PCA might reveal that those genes collectively describe a process like heat shock stress or nutritional stress.

These new dimensions are combinations of the original variables and typically explain the **maximum variance** in the data. As a result, PCA allows you to capture the essential features of a dataset in fewer variables.

Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a broader mathematical technique than PCA and can be applied to matrices that are not square. It's essentially a generalization of eigenvalue decomposition. Like PCA, it is useful for **dimension reduction** and **data compression**.

Matrix Factorization: SVD decomposes a matrix into three components: U, Σ, and V. These represent different ways of understanding the relationships between rows and columns of the matrix. U represents the left singular vectors (e.g., gene expression). Σ is a diagonal matrix of singular values (analogous to eigenvalues in PCA). V represents the right singular vectors (e.g., samples or conditions).

Non-Square Matrices: Unlike PCA, which operates on square covariance matrices, SVD can be applied to rectangular matrices, making it more flexible for analyzing datasets where the number of variables does not match the number of samples.

Applications in Biology:

In biological data analysis, both PCA and SVD help to **reduce the dimensionality** of complex datasets like gene expression matrices or metabolic profiles. This makes it easier to identify underlying patterns and relationships among genes or experimental conditions.

For example, in a matrix of gene expression data, SVD allows us to find **latent variables** or "eigen-genes," which can be used to describe major trends in the data. These trends might represent fundamental biological processes or cellular states, reducing the need to analyze individual genes separately.

Singular Value Decomposition in Practice:

- **Generalization of PCA:** While PCA deals with square matrices, SVD provides a more general method applicable to rectangular data matrices, making it ideal for datasets with unequal numbers of variables (e.g., thousands of genes measured across a few experimental conditions).
- **Eigen Genes and Eigen Experiments:** By applying SVD, we can create abstract representations such as "eigen-genes" and "eigen-experiments." These concepts allow us to reduce a complex biological system to a small number of variables, providing insights into high-level processes, such as how metabolic stress or heat shock affects the system as a whole.

Sparse PCA and Nonlinear Dimensionality Reduction:

While traditional PCA uses linear combinations of the original variables, **sparse PCA** imposes additional constraints, ensuring that only a small subset of variables is used to describe the principal components. This makes the interpretation of results more biologically meaningful, as it highlights only the most critical genes or features. Additionally, **nonlinear dimensionality reduction techniques** can be applied to capture more complex relationships that cannot be explained by linear transformations alone.

1:11:58 Non-linear Dimensionality Reduction: t-SNE

Non-linear dimensionality reduction techniques like **t-SNE** have become widely used for understanding and visualizing high-dimensional data, particularly in complex biological datasets such as single-cell RNA sequencing data. Here's a detailed breakdown of the process and its significance:

Why Use Non-linear Dimensionality Reduction?

Traditional dimensionality reduction techniques like **PCA** focus on linear transformations. They are useful when data vary linearly across a set of principal components. However, biological systems and high-dimensional data, such as gene expression profiles, often exhibit **non-linear relationships**. These patterns are complex, and PCA may fail to capture important structures in the data.

To address this, **non-linear dimensionality reduction techniques** like t-SNE are applied. t-SNE can capture **non-linearities** in the data, revealing patterns that are not visible in a linear projection.

t-SNE Overview

t-SNE aims to reduce the dimensionality of high-dimensional data (such as gene expression profiles) while

preserving the **local structure**—the relationships between close neighbors—of the data. It achieves this by focusing on **pairwise similarities** between data points in high-dimensional space and embedding those into a lower-dimensional space (typically 2D or 3D).

High-dimensional Relationships: Imagine having data in a very high-dimensional space (e.g., 20,000 gene expression profiles). In that space, some points (data samples) are very close to each other, representing similar biological profiles (e.g., similar gene expression patterns).

Neighbor Preservation: t-SNE tries to ensure that points that are close together in high-dimensional space remain close in the reduced space, while points that are far apart are spread out accordingly.

t-SNE Mechanics

Preserving Neighborhoods: The method begins by measuring how close each point is to its neighbors in the high-dimensional space. This is achieved using **pairwise distances** or a probability distribution that reflects the likelihood that two points are close in the original space.

Stochastic Mapping: The algorithm then attempts to embed the data in a lower-dimensional space, adjusting the points such that their **pairwise distances** in this lower space reflect their relationships in the original high-dimensional space. The process is iterative, with the algorithm making small adjustments to the positions of points in the lower-dimensional space over multiple steps to improve the match between high-dimensional and lower-dimensional relationships.

Non-linear Transformation: Unlike PCA, which relies on linear transformations, t-SNE uses **non-linear transformations** to better capture the complex relationships in the data. It ensures that clusters of similar data points are tightly packed together in the reduced space while pushing dissimilar points far apart.

High-dimensional to Low-dimensional Mapping: The key goal of t-SNE is to map a **high-dimensional k-nearest neighbor graph** to a **low-dimensional k-nearest neighbor graph** while preserving the relationships between pairs of points as closely as possible.

Visualization and Insights

One of the primary uses of t-SNE is for **visualization**. For example:

In **single-cell RNA sequencing (scRNA-seq)**, t-SNE can be used to project tens of thousands of gene expression profiles into a 2D or 3D space. This reduced representation allows researchers to visualize clusters of cells with similar gene expression profiles, revealing potential subpopulations of cells.

Cell Differentiation: Cells that are similar in function or in the same developmental lineage will cluster together, making it easier to identify and interpret biological patterns.

Limitations

Interpretation of Distances: While t-SNE excels at preserving **local structures**, it is less effective at preserving **global structures**. This means that while it's good for identifying local clusters, distances between clusters may not be meaningful.

Computational Complexity: t-SNE is computationally expensive and can be slow to converge, especially for large datasets. Moreover, the algorithm can sometimes produce different results for the same dataset due to the randomness involved in the initialization process.

Application in Single-cell Data

In the context of **single-cell data analysis**, t-SNE has revolutionized how we visualize and interpret large-scale gene expression datasets. For example, in **scRNA-seq**, the algorithm is used to reduce the data from tens of thousands of dimensions (each dimension representing a gene's expression level) to just two or three dimensions. This reduced space allows researchers to easily spot **cellular subpopulations** and **transcriptional states**, as distinct groups of cells naturally form clusters in the t-SNE plot.

Summary

t-SNE provides a powerful way to visualize complex, high-dimensional data by transforming it into a lower-dimensional space while preserving the local relationships between points. This makes it invaluable for biological applications where understanding clusters and patterns in data, such as in gene expression or cellular differentiation, is critical.

1:16:23 Machine Learning on Networks

Learning on Graphs allows us to directly model complex relationships between entities. These networks can represent **gene-gene interactions**, **physical structures**, **textual relations**, or **molecular structures**. In each case, **edges** between nodes can carry additional constraints, such as **angles in 3D molecular geometry**.

The key challenge is to **learn representations** of these graphs that capture essential features for **predictive modeling**. This can be done at multiple levels:

1. **Node-level representations**: Predict properties of individual nodes based on their relationships with neighbors.
2. **Edge-level representations**: Learn interactions or dependencies between nodes.
3. **Subgraph representations**: Capture local graph structures to infer larger patterns.
4. **Graph-level representations**: Summarize the entire graph for higher-level predictions.

Representation Learning on graphs is analogous to deep learning on images. Just as pixels are abstracted into **edges, corners, shapes, and objects**, **subgraphs** can be used to build **node properties** in layers. For example:

- **Level 0**: Initial node properties.
- **Level 1**: Properties influenced by direct neighbors.
- **Level 2**: Properties influenced by neighbors of neighbors.

This progressive abstraction allows for more complex **relational understanding**. At each level, the representation is updated based on neighborhood interactions, and this is done at both **node** and **edge levels**.

In practice, **graph neural networks (GNNs)** enable this multi-level abstraction. Once the model is trained, we can **benchmark** performance, analyze **explainability**, and refine the learned **representations** for various tasks, from biological networks to molecular property prediction

1:18:49 Summary

Today we focused on the **building blocks of regulatory motifs**—the physical sequences bound by transcription factors or regulators—and how we can discover these motifs through techniques such as **expectation maximization**, **enrichment analysis**, or **evolutionary signatures**. These motifs are fundamental to understanding gene regulation, and from these motifs, we build regulatory networks.

We then explored how to analyze the **resulting networks** to uncover key properties such as **centrality**, **eigenvectors**, **principal components**, and **lower-dimensional projections**. Finally, we discussed how **machine learning** can be applied to these networks, using graph neural networks to infer complex relationships at different levels—nodes, edges, subgraphs, or the entire graph.

Looking ahead, the next two modules will extend these concepts to **protein structure** and **chemical structure**, where we'll use the same methodologies to deepen our understanding of these biological and chemical systems. This lecture provided a foundation for that exploration, and I encourage you to begin preparing for the next steps by reviewing and attending office hours early to stay on track.

Lecture 8 - Intro to Protein Structure

Video: Lecture08 - Intro to protein structure - MLCB24

Slides: Lecture08_IntroToProteinStructure.pdf

00:00 Introduction

The focus of this module on protein structure brings together **computational and biological disciplines**—a merging that has significantly shaped modern bioinformatics and structural biology. This lecture is a personal and academic journey into protein structure, tracing the lecturer's evolution from an initial disinterest in biology, due to rote memorization, to a profound appreciation for evolutionary biology and genetics, spurred by a transformative realization about the **tree of life** and humanity's place within it.

With a strong background in **math and computer science**, the lecturer's shift toward biology began in college. Initial explorations in protein folding during graduate studies under **David Baker** introduced him to the **protein folding problem**—an inquiry into the time required for proteins to fold, which varies greatly depending on structural characteristics. This early work sparked a broader interest in biological functions beyond static genetic codes, driven by emerging technologies like the **Human Genome Project**. The excitement surrounding genome sequencing projects at the time involved discovering genes within vast nucleotide sequences and exploring potential applications, from human health to even unexpected uses in consumer products.

This course marks a **return to protein structure** for the lecturer, aiming to reconnect with and inspire an interest in **structural biology** among students. This lecture introduces **key themes in structural biology**:

- **Defining structural biology** and its importance in revealing function at the molecular level.
- **Case study on transcription factors**, specifically the LacI family, to illustrate practical applications.
- **Techniques for determining protein structures**, covering the main experimental and computational methods.
- **Principles of protein structure and comparison**, laying out foundational concepts.
- **Energy functions and physics**, essential for connecting structural biology with physics principles, and foreshadowing the use of **deep learning** in protein structure prediction.

The lecture highlights that **understanding energy functions** remains relevant even with advanced machine learning approaches. The interplay between **statistical mechanics** and deep learning showcases how models originally rooted in physics can translate into modern **neural networks** used in biological modeling, forming a bridge between **quantitative sciences and structural biology**.

This session serves as a gateway into **protein structure fundamentals** while setting the stage for upcoming discussions on **deep learning's transformative role** in protein structure prediction and the broader implications for biology and computational science.

9:48 What is structural biology

Structural biology is the study of molecular structures to understand their function, particularly focusing on proteins. It involves both analyzing **pre-determined structures** to identify patterns (such as common features across enzymes) and **determining structures experimentally or computationally** to uncover specific details. The central challenge in structural biology, particularly for proteins, has been how to transition from a linear **protein sequence (derived from genomic data)** to a **three-dimensional structure**. Such a structure can then shed light on the molecule's function, which is essential for understanding biological processes.

Proteins form the primary focus of structural biology, especially with advancements in **AI** over the past five years, which have made significant progress in protein structure prediction. Currently, we have around **200,000 experimentally-determined protein structures** and many more that have been **predicted computationally** with increasing accuracy. This vast dataset opens up the possibility of exploring structural patterns at scale.

The importance of structural biology lies in the fact that **we have extensive genomic data**, but the functional interpretation of much of this data remains elusive. If we could **convert all genomic sequences into structures**, and thus infer their function, we could vastly improve our understanding of the genome and its implications. Structural biology also has direct implications for **disease research**, as many disorders result from **protein misfolding**. Diseases like **Alzheimer's, Parkinson's, Huntington's, and cystic fibrosis** are tied to the improper folding or aggregation of specific proteins, leading to cellular dysfunction. Similarly, conditions like **prion diseases** (e.g., Mad Cow Disease) highlight how misfolded proteins can even act as infectious agents, influencing the folding of other proteins.

Furthermore, **protein design** is an emerging field that leverages our understanding of structure and function to **engineer proteins with desired properties**. This field aims to create proteins that can perform specific reactions, bind to targeted molecules, or even inhibit harmful cellular processes. Recent breakthroughs in **deep learning** have enhanced the accuracy and potential of protein design, allowing scientists to conceptualize and build custom proteins for applications in **medicine, biotechnology, and synthetic biology**.

Key resources for visualizing and understanding protein structures include:

- The **Protein Data Bank (PDB)** for accessing structural data.
- **PyMOL** and other visualization tools for inspecting molecular structures, many of which now support **Python scripting** for advanced analysis.
- **AlphaFold** and **ESM Fold**, cutting-edge deep learning tools that have revolutionized structure prediction, offering databases of **pre-computed structures** accessible for research.

This overview sets the stage for understanding **structure-function relationships** in proteins and highlights the pivotal role that structural biology plays in both **basic science** and **applied research**. Through case studies and hands-on resources, students will explore how molecular structures can be visualized, analyzed, and designed, building a foundation for more advanced discussions on **predictive modeling** in the following lectures.

16:40 Using PyMOL for Structural Analysis of Transcription Factors

In structural biology, **understanding the relationship between a protein's structure and its function** is

crucial, and PyMOL is a valuable tool for visualizing these interactions. Here, we focus on a **transcription factor** from the **LacI family**, commonly found in prokaryotes, which features a **helix-turn-helix motif** for DNA binding. The goal is to explore how this protein binds DNA and controls gene expression, using PyMOL to uncover key structural details.

Steps in PyMOL:

1. **Initial Structure and Crystal Duplication**: The visualized structure shows multiple copies of the DNA-protein complex due to the **x-ray crystallography process**, where protein molecules are crystallized in repeating units. To focus on a single protein-DNA complex, extraneous copies are removed by selecting specific chains (e.g., `chain A` and `chain B`).
2. **Electrostatic Surface Analysis**: Generating **electrostatic surfaces** helps reveal regions with **net positive or negative charges**:
    - **Positive regions (blue)** likely interact with the **negatively charged DNA backbone** (phosphodiester bonds), suggesting where DNA binding occurs.
    - **Negative regions (red)** orient away from DNA due to charge repulsion.
3. **DNA Binding and Major/Minor Grooves**: After hiding extraneous features, we examine how the **protein fits into the DNA double helix**. DNA features **two grooves**:
    - **Minor Groove**: Tighter with closer backbones.
    - **Major Groove**: More open, allowing transcription factors to access and bind DNA.
4. **Helix-Turn-Helix Motif and Recognition Helix**:
    - The **recognition helix** is positioned in the **major groove**, enabling specific interaction with DNA bases. This helix, in close proximity to the DNA backbone, allows residues to make both **sequence-specific** contacts (recognizing particular DNA sequences) and **non-specific contacts** (stabilizing the interaction through backbone contacts).
    - **Conservation Across Species**: For residues interacting nonspecifically with the backbone, **high conservation** is expected, whereas residues involved in sequence specificity may vary.
5. **Symmetry and Palindromic DNA Sequences**:
    - Because this transcription factor binds DNA as a **homodimer**, it binds to two separate DNA sites, often separated by about **10 base pairs**. This configuration requires the DNA binding sites to be **palindromic** to accommodate the symmetry of the homodimer.
    - **Recognition Motifs**: These palindromic motifs are conserved across various species and are used to identify **DNA binding motifs** in prokaryotic genomes.

By examining this transcription factor, PyMOL provides insights into the **physical and chemical principles** underlying DNA binding and transcriptional regulation. This example illustrates how structural biology tools can reveal **functionally significant structural features**, such as charge distribution, binding motifs, and protein symmetry, essential for understanding gene regulation.

31:45 Transcription factor-DNA binding and functional specificity

In **structural biology**, understanding how transcription factors (TFs) interact with DNA is essential for elucidating gene regulation mechanisms. This example focuses on the **helix-turn-helix motif** in transcription factors from the **LacI family**, illustrating how the structure of transcription factors influences their DNA-binding specificity and regulatory function.

Helix-Turn-Helix Motif and Recognition Helix

In the LacI family, the **recognition helix** of the helix-turn-helix motif is positioned within the **major groove** of DNA. This positioning allows specific residues in the recognition helix to interact directly with the DNA bases, thereby determining the specificity of DNA binding. For example:

- In **PurR** and similar proteins, three critical residues in the recognition helix protrude into the DNA's major groove, where they make **specific contacts** with the DNA bases, determining the protein's binding specificity.

By comparing different members of this transcription factor family, researchers identified that **three primary residues** are largely responsible for specificity. These residues allow some transcription factors to bind the same DNA sequences, while others target distinct sequences, depending on these key residue differences.

Allosteric Regulation by Small Molecules

The **binding of small molecules** to transcription factors such as **LacI** or **PurR** alters their conformation, thereby influencing DNA binding. For instance:

- **LacI** binds lactose or a lactose derivative, which changes its structure, reducing its affinity for DNA and preventing it from binding to its operator sequence.
- This conformational shift means that **in the presence of lactose**, the LacI repressor is removed from the DNA, allowing transcription to proceed.

Each member of the LacI family often binds different small molecules (e.g., purines, carbohydrates) but has a similar regulatory effect, allowing cells to respond to specific environmental signals by turning on or off gene expression.

Synthetic Applications: Designing Molecular Switches

The discovery that **changing these three specificity-determining residues** could transfer DNA binding specificity between transcription factors across species provides a foundation for designing **synthetic molecular switches**. By engineering these switches to respond to various cellular chemicals, scientists can potentially target them to specific genome regions, enabling precise control over gene expression.

Operon Structure and Regulatory Binding Sites

On a genomic level, the layout of **genes, promoters, and operator sequences** is essential for regulatory control:

- In the **Lac operon**, LacI binds to its operator sequence downstream of the promoter, blocking **RNA polymerase** from initiating transcription of genes necessary for lactose metabolism (LacZ, LacY, and LacA).
- **Repressors**, such as LacI, typically bind downstream or overlapping with the promoter, physically blocking RNA polymerase from binding.
- **Activators**, in contrast, bind upstream of promoters, enhancing transcription by recruiting RNA polymerase to weak promoters, where the polymerase would otherwise have a lower affinity.

This hierarchical organization—from specific amino acid contacts in DNA-binding helices to operon-level gene layout—illustrates how structural elements at multiple scales collectively **determine the function** of transcription factors and their role in cellular regulation.

37:41 Experimental methods for determining structure

In **structural biology**, accurately determining the three-dimensional structure of proteins and protein complexes is crucial for understanding their function. Three primary experimental techniques—**X-ray crystallography**, **nuclear magnetic resonance (NMR)**, and **cryogenic electron microscopy (cryo-EM)**—are widely used to achieve this, each with distinct advantages, limitations, and suitability depending on the protein's size, stability, and the environment in which it functions.

X-ray Crystallography

X-ray crystallography is often considered the gold standard for high-resolution protein structures. In this method, researchers must first grow a **high-quality crystal** of the protein. This process can be challenging because proteins naturally prefer to be in solution, and not all proteins readily form crystals. Often, scientists use complexes of proteins bound to DNA or ligands (such as drugs) to understand not only the protein structure but also how it interacts with these molecules.

Once a crystal is obtained, it is exposed to an **X-ray source**. As X-rays pass through the crystal, they are diffracted, creating a **diffraction pattern** on a detector. Each point in this pattern provides information about the electron density within the crystal, which represents the spatial distribution of atoms. The **electron density map** generated from this pattern can then be used to deduce the protein's 3D structure through a **Fourier transform**.

However, **phase information**—critical for constructing the electron density map—is lost during diffraction. To recover this phase information, researchers use techniques such as:

- **Heavy atom substitution**: Adding atoms with high electron densities (e.g., mercury or gold) to create differences that can help determine phases.
- **Molecular replacement**: Using the structure of a closely related protein as a template to estimate the phase information.

The resulting structure is characterized by its **resolution**, a key metric of quality. Higher resolution (indicated by smaller numbers, typically below 3 Ångstroms) means more detailed and precise atomic positions. Some modern structures are determined at **sub-angstrom resolutions** (below 1 Å), allowing researchers to see nearly every atom's position in exquisite detail.

Nuclear Magnetic Resonance (NMR) Spectroscopy

Nuclear magnetic resonance (NMR) spectroscopy offers a solution for studying proteins that cannot be crystallized. This technique is based on the magnetic properties of atomic nuclei, particularly hydrogen atoms in proteins. NMR is advantageous for examining proteins in **solution**, providing a more natural environment that is close to physiological conditions.

NMR spectroscopy measures **interatomic distances** by analyzing signals from hydrogen nuclei that are close to each other in space. By identifying pairs of hydrogen atoms that are close enough to interact, researchers can infer structural proximity and gradually construct a 3D model of the protein. This makes NMR particularly useful for proteins in which **dynamic flexibility** or **functional conformational changes** are essential aspects of their activity.

However, NMR is limited by:

- **Protein size**: Generally, NMR is most effective for smaller proteins (typically under 30 kDa), as larger proteins produce more complex spectra that are challenging to interpret.
- **Complexity**: Interpreting NMR spectra requires extensive computational processing to assign signals to specific residues and atoms, which can be time-consuming and computationally demanding.

Despite these challenges, NMR provides invaluable insights, especially for studying protein dynamics and interactions under near-physiological conditions.

Cryogenic Electron Microscopy (Cryo-EM)

Cryogenic electron microscopy (cryo-EM) has revolutionized the field of structural biology in recent years, particularly for **large macromolecular complexes** that are difficult to crystallize, such as **viruses, ribosomes, and multi-protein assemblies**. In cryo-EM, samples are flash-frozen to preserve their native structure and then subjected to an electron beam in a transmission electron microscope.

Cryo-EM provides a **direct image** of large protein complexes, capturing them in various conformational states. The technique allows researchers to observe not only the architecture of individual proteins within a complex but also how these proteins fit together and function as a unit. The following features characterize cryo-EM:

- **Native state preservation**: The sample remains in a frozen, hydrated state, which can reveal natural conformations without the need for crystal packing constraints.
- **Large complex suitability**: Cryo-EM is ideal for proteins and complexes that are too large for NMR or difficult to crystallize, such as membrane proteins.

Cryo-EM images often reveal the overall shape and arrangement of large complexes, but at **moderate resolutions** compared to X-ray crystallography. However, the increasing power of modern electron microscopes and advances in image processing have enabled cryo-EM to approach resolutions close to those achieved by X-ray crystallography, with some structures determined below 3 Å.

In cases where researchers have high-resolution structures of individual components (from crystallography or NMR), they can use these structures as **rigid-body fits** within the cryo-EM density map to create a composite model of the entire complex.

Summary of Techniques and Applications

Each of these methods provides unique benefits and insights:

- **X-ray crystallography** is best for high-resolution structures of stable, crystallizable proteins and complexes, often used when fine atomic details are required.
- **NMR spectroscopy** allows proteins to be studied in solution, capturing dynamic interactions and flexibility, ideal for small proteins and proteins with conformational variability.
- **Cryo-EM** enables visualization of large macromolecular assemblies in their native state, invaluable for studying complex molecular machines and large viral structures.

Together, these methods enable researchers to obtain comprehensive structural and functional information across a range of protein sizes, shapes, and interaction complexities. As structural biology continues to integrate deep learning methods, each of these experimental approaches remains indispensable for validating

and refining AI-predicted protein structures, ensuring that models align with the physical reality of molecular interactions.

42:15 Protein structure: conformations and Folding

Understanding **protein conformations** is central to grasping how proteins achieve their functional shapes. Proteins are **polymers** of amino acids, and while the **bond lengths and bond angles** between atoms in the backbone remain relatively fixed, the **rotatable bonds** along the backbone allow for flexibility. This flexibility enables proteins to explore multiple **conformations** until they find their lowest-energy folded state.

Backbone Rotational Angles: Phi, Psi, and Omega

The **protein backbone** is primarily defined by three main angles associated with **rotatable bonds**:

1. **Phi (φ)**: This is the angle between the **nitrogen atom** and the **alpha carbon** (Cα). The φ angle allows rotation around the bond connecting the nitrogen of one amino acid to the alpha carbon of the same amino acid.
2. **Psi (ψ)**: This is the angle between the **alpha carbon** and the **carbonyl carbon** of the same amino acid. Rotation around this bond influences the orientation of the backbone downstream of the alpha carbon.
3. **Omega (ω)**: This bond is between the **carbonyl carbon** of one amino acid and the **nitrogen** of the next amino acid in the chain. The ω angle usually adopts a **trans configuration** (180°) due to steric hindrance, which is generally stable. However, when **proline** is present, the ω angle may adopt a **cis configuration**, introducing a specific type of rigidity.

While each amino acid has unique side chains that influence these angles, the **phi and psi angles** are the most influential in defining the overall **three-dimensional shape** of the protein, dictating how it will fold and pack into its final structure. A complete description of a protein's shape includes specifying these angles for each residue in the chain.

Conformational Preferences: Energy Minimization

Despite the seeming **continuum of rotational possibilities** (from 0° to 360°) for each bond, proteins tend to adopt only a few **low-energy conformations**. This preference arises because the **spatial arrangement of atoms** naturally favors configurations that **minimize steric clashes** and maximize stability. For example, if we examine the simple molecule **ethane**, where two large atoms are attached, the lowest energy conformation positions these bulky groups as far from each other as possible. This concept also applies to larger, more complex protein structures, where **rotational freedom** is restricted to avoid high-energy, sterically unfavorable arrangements.

In practice, for a given **phi and psi angle pair**:

- **Three preferred states** often represent the most stable conformations due to steric and energetic considerations.
- The **three-state approximation** is commonly used in protein modeling, acknowledging that within each preferred state, there can be some **minor fluctuations** around the stable conformation. These fluctuations allow proteins to retain some **conformational flexibility** even when fully folded, important for functions like **ligand binding** and **allosteric regulation**.

Folding Landscape and Conformational Space

To conceptualize the enormity of **conformational space**, consider a simple **thought experiment**:

- Imagine a **100-amino acid protein**, where each phi and psi bond pair can adopt **three stable conformations**. For each amino acid residue, there are thus **three possibilities for phi and three for psi**, totaling nine per residue.
- With 100 residues, the number of potential conformations becomes approximately 32003^{200}3200, or about 1010010^{100}10100. This vast number demonstrates the **astronomical diversity** of possible shapes the protein could theoretically adopt.

Despite this staggering number, only **one conformation** (or a small set of closely related structures) represents the **folded, functional state**. This fold results from a complex **energy landscape** where the protein "searches" for the configuration that minimizes its **free energy**, a process often referred to as **the protein folding problem**.

The **probability of achieving this folded state by chance** is infinitesimally small (roughly 1/101001/10^{100}1/10100), underscoring the idea that protein folding is not a random process. Instead, it follows specific **folding pathways** guided by **intrinsic chemical properties** and **environmental factors** (e.g.,

temperature, pH).

Implications of Conformational Flexibility

Conformational flexibility is vital for many **biological functions**. For instance:

- **Enzyme catalysis** often requires small **conformational adjustments** to bring substrates into the optimal orientation.
- **Signal transduction proteins** may undergo significant conformational changes upon ligand binding, allowing them to relay information across cellular membranes or to other molecules.
- **Allosteric proteins** use conformational shifts to modulate their activity in response to binding events at sites distant from the active site.

Conformational Energy Wells and Fold Stability

The **rotational preference** of bonds, coupled with **interactions among residues**, means that proteins tend to fall into **local energy minima**, or **wells**, in their energy landscape. Within these wells, **microstates** allow for minor rotations and adaptations, contributing to a protein's **stability and functional flexibility**. In completely folded proteins, the **entropy loss** associated with adopting a single conformation from numerous possible ones is offset by **enthalpic stabilization** provided by favorable **hydrogen bonding, van der Waals interactions**, and **hydrophobic effects**.

Understanding these principles of **conformational stability and flexibility** provides essential insights into why proteins fold in specific ways, how they achieve functional forms, and how their **misfolding** can lead to dysfunction or disease. For instance, in misfolding diseases like **Alzheimer's** or **Parkinson's**, proteins fail to reach their low-energy folded state, often aggregating into toxic assemblies that disrupt cellular function.

In summary, **protein conformations** are governed by a combination of **rotational freedom** around specific backbone angles and **energetic preferences** that guide the molecule towards a stable, functional state. This delicate interplay of **structure, flexibility, and stability** is foundational for understanding protein function and dysfunction in a biological context.

50:40 The protein folding problem: Levinthal's Paradox and the Energy Landscape

The **protein folding problem** is one of the most profound and long-standing questions in molecular biology, centered on understanding how proteins achieve their **functional three-dimensional structures** from a linear chain of amino acids. The process is remarkably efficient, yet theoretically complex, as proteins seem to "know" how to fold rapidly despite the astronomical number of possible conformations.

Christian Anfinsen's Hypothesis: Energy Minimization

In the 1950s, **Christian Anfinsen** proposed that a protein's folded conformation represents its **global free energy minimum**—the lowest energy state it can achieve in its given environment. This hypothesis suggested that proteins would naturally adopt their **native conformation** because this structure is thermodynamically the most stable. Essentially, the idea was that, given the right conditions, the sequence alone contains all the information needed for a protein to fold into its functional form.

Levinthal's Paradox: The Search Problem

In 1969, **Cyrus Levinthal** articulated a challenge to Anfinsen's hypothesis, which became known as **Levinthal's Paradox**. He observed that if proteins were to search through all possible conformations to find the one with the lowest energy, it would take **longer than the age of the universe** for a protein to reach its folded state. For a small protein of 100 amino acids, the possible conformations number around 1010010^{100}10100—an astronomically large space to search.

Yet, in practice, many proteins fold **within milliseconds to seconds**. This apparent contradiction—between the need for exhaustive conformational sampling and the rapidity of actual folding—highlighted that proteins must employ an **efficient search strategy** to locate their folded state, circumventing a brute-force approach.

Resolving Levinthal's Paradox: The Folding Funnel and Energy Landscape Theory

The resolution to Levinthal's Paradox lies in the concept of the **energy landscape**, often visualized as a **folding funnel**. Rather than randomly sampling every possible conformation, proteins fold through a series of **thermodynamically favorable intermediate states**:

- **Local energy minima** represent stable intermediate structures along the pathway to the native state.
- **Energy barriers** separating these minima are overcome through **local conformational changes** that progressively guide the protein toward its folded form.

The folding funnel model suggests that as a protein folds, it moves down the funnel, where the number of possible conformations decreases while stability increases. The **depth** of the funnel represents the energy, with the **native state** at the global minimum. This landscape structure implies that while proteins can sample various states, they are energetically biased toward progressively lower-energy conformations, thus streamlining the folding process.

Contact Order and Folding Speed

In 1998, researchers discovered that the **contact order**—the **average sequence distance between amino acids that interact in the folded structure—correlates with folding speed**. Proteins with **low contact order** (i.e., where interacting residues are close in sequence) tend to fold faster than those with high contact order. This finding supports the idea that **local interactions** form quickly, helping to stabilize portions of the structure early in the folding process and guiding the overall conformation.

Levels of Protein Structure

Proteins are organized hierarchically, with each level contributing to the final folded form:

1. **Primary Structure**: The **linear amino acid sequence** of the protein, determined directly by the gene encoding it.
2. **Secondary Structure**: **Local structural elements**, such as **alpha helices** and **beta sheets**, stabilized by hydrogen bonding patterns between backbone atoms. These structures form quickly, reflecting local energy minima and helping reduce conformational space.
3. **Tertiary Structure**: The overall **three-dimensional shape** of the polypeptide chain, encompassing all interactions within a single molecule. This structure includes the folding of secondary elements and interactions between side chains.
4. **Quaternary Structure**: The **assembly of multiple polypeptide chains** (subunits) into a functional complex. For example, the **Lac repressor** protein discussed earlier achieves functionality as a **homodimer** by forming a stable unit with two identical subunits.

Each level builds upon the last, with the tertiary and quaternary structures representing the final, functional form of the protein. This hierarchical folding allows proteins to efficiently narrow down the folding options, leveraging local structure formation to inform and stabilize the overall fold.

Implications of the Folding Problem

The protein folding problem remains an area of active research with implications across **biomedicine and biotechnology**. Misfolding and aggregation are central to many **neurodegenerative diseases**, such as Alzheimer's and Parkinson's, where misfolded proteins aggregate, leading to cell damage. Additionally, understanding protein folding mechanisms aids in **protein design** efforts, where scientists aim to engineer new proteins with specific functions by predicting and controlling their folded structures.

The development of **computational techniques** such as **machine learning** (e.g., AlphaFold) has revolutionized our ability to predict protein structures with high accuracy, bypassing some of the limitations of experimental structure determination. However, the protein folding problem remains a rich field that bridges physics, chemistry, and biology, underscoring the elegance of nature's approach to creating functional biological molecules.

55:00 Protein secondary structure

Protein secondary structure represents a critical level of organization within protein folding, characterized by **localized structural motifs** such as **alpha helices** and **beta sheets**. These motifs are stabilized primarily by **hydrogen bonds** and are foundational elements that contribute to the overall three-dimensional shape and function of proteins.

The Ramachandran Plot: Mapping Accessible Dihedral Angles

The **Ramachandran plot** is a powerful tool in structural biology, visualizing the **dihedral angles (phi and psi)** of amino acid residues in proteins. By plotting the phi (φ) and psi (ψ) angles for residues across many proteins, we observe distinct regions where these angles are more favorable, reflecting the **limited conformational space** due to **steric hindrance** and **backbone rigidity**:

- **Alpha helices** typically appear in a dense region on the plot with characteristic phi and psi angles that allow for **helical hydrogen bonding**.
- **Beta sheets** occupy another prominent region, with angles favoring **extended strands** stabilized by hydrogen bonds between adjacent strands.

- **Glycine**, due to its lack of a side chain, exhibits more flexibility and can adopt unique angles outside these regions, often occurring in **turns or loops**.

The **density of points** in the Ramachandran plot reflects the stability of certain backbone angles, with highly populated areas indicating **preferred conformations** that minimize energy through favorable bonding and spatial arrangement.

Secondary Structure Elements: Alpha Helices and Beta Sheets

Alpha Helices:

- An alpha helix is a **right-handed coil** where each amino acid residue forms a **hydrogen bond** with the residue four positions ahead in the sequence. This results in a tightly packed, rod-like structure with the side chains projecting outward from the helix, minimizing steric clashes.
- The alpha helix is a stable configuration due to **internal hydrogen bonding** along the backbone, which propagates down the helical axis, contributing to its structural resilience.

Beta Sheets:

- Beta sheets consist of **extended strands** that align side-by-side to form a sheet. These strands are connected by **hydrogen bonds** between backbone atoms of adjacent strands, creating a **pleated sheet** appearance.
- Beta sheets can be either **parallel** or **anti-parallel**:
    - In **parallel beta sheets**, the strands run in the same direction and typically form larger, less tightly bonded sheets.
    - In **anti-parallel beta sheets**, strands run in opposite directions, resulting in stronger hydrogen bonds and greater stability.
- These sheets play a crucial role in structural scaffolding and are often found in the core of proteins, where they contribute to the **rigid, planar stability** necessary for maintaining the protein's overall shape.

Methods for Predicting Secondary Structure

Predicting secondary structure has been a focal point in computational biology, as it provides insights into protein function and helps guide more complex three-dimensional modeling.

1. **Chou-Fasman Method**:
    - This early method used **statistical probabilities** based on the frequency of each amino acid in secondary structures derived from known protein crystal structures.
    - For each amino acid, the **probability of occurrence** in an alpha helix, beta sheet, or random coil was calculated. By analyzing stretches of amino acids with a high likelihood of forming specific secondary structures, this method could reasonably predict segments of helices and sheets.
2. **Neural Network Approaches**:
    - In the 1990s, neural networks became a transformative tool for **secondary structure prediction**, with methods like **PHD (Profile network from Heidelberg)** leveraging **three-layer neural nets** to improve accuracy.
    - These neural networks incorporated **multiple sequence alignments** (MSAs), which analyze the conservation of amino acids across homologous proteins, providing clues about the likelihood of certain residues being part of an alpha helix or beta sheet.
    - This approach marked a pivotal moment in structural biology, allowing the prediction of secondary structures with greater precision by combining sequence information with homologous structure data.
3. **Modern Predictive Techniques**:
    - Today, advances in **machine learning** and **deep learning**—exemplified by models like **AlphaFold**—have significantly improved **three-dimensional protein structure prediction**, often rendering isolated secondary structure prediction unnecessary.
    - By incorporating extensive training on sequence-structure relationships and leveraging large protein structure databases, modern algorithms can predict not only the overall tertiary structure

but also local secondary structures with high accuracy.

The Role of Secondary Structure in Protein Folding

Secondary structures form the **initial scaffold** of a protein's folded state, guiding the organization of the tertiary structure. The hydrogen bonding patterns and structural rigidity inherent in alpha helices and beta sheets create a **stable framework** upon which other interactions, such as side-chain packing and hydrophobic interactions, can build. This **hierarchical folding process** is critical in ensuring that proteins adopt a functional, low-energy conformation efficiently.

In summary, secondary structures like alpha helices and beta sheets are integral components of protein architecture, supported by the limited conformational space outlined in the Ramachandran plot. Predictive models have evolved from statistical and neural network methods to sophisticated deep learning systems, enabling us to reliably infer protein structure and function. These advancements underscore the connection between **sequence, structure, and biological function**, facilitating our understanding of how proteins achieve their specific roles within cells.

59:00 Protein tertiary structure: From Sequence to 3D Configuration

Tertiary structure encompasses the unique three-dimensional shape that a protein assumes, dictating its functionality, stability, and interactions. Predicting this structure from a **linear amino acid sequence** has been a longstanding goal, capturing the intricate relationships between residues and leveraging insights from both structural biology and computational methods.

Contact Maps: Visualizing Residue Interactions

A foundational tool in tertiary structure prediction is the **protein contact map**, a two-dimensional grid representing **residue-residue proximities** within the protein:

- On a contact map, residues are plotted on both the **x-axis** and **y-axis**, with **contacts marked at coordinates** where two residues are spatially close.
- **Alpha helices** appear as linear stretches along the diagonal (y = x line) due to consistent bonding patterns within the helical structure.
- **Parallel and anti-parallel beta sheets** manifest as lines perpendicular or parallel to the x = y axis, reflecting the hydrogen bonding patterns between aligned beta strands.

These contact maps reveal not only **secondary structure elements** but also insights into tertiary structure by displaying which residues interact across the protein's 3D conformation.

Evolutionary Insights: Using Co-evolution for Structure Prediction

Proteins evolve under selective pressures that preserve structural integrity, often leading to **co-evolution of residue pairs** that interact. This is critical for understanding tertiary structure:

- By analyzing **multiple sequence alignments (MSAs)** across homologous proteins, scientists observe that certain amino acid pairs change in tandem. When one amino acid in an interaction changes, the paired amino acid may change to maintain the interaction's stability.
- This **coevolutionary data** offers clues about residue proximity, aiding in the construction of contact maps. Proteins predicted to maintain specific residue pairs often reveal **functionally critical sites** or structural features that are conserved across species.

This co-evolutionary approach laid the groundwork for **machine learning methods** that integrate MSAs with contact maps, significantly improving predictive accuracy in protein folding.

Computational Advances: From Rosetta to AlphaFold

The quest to solve protein structure prediction has evolved through notable computational tools and milestones:

1. **Rosetta**:
    - Developed in **David Baker's lab**, Rosetta represented an early, transformative approach that combined **statistical and physics-based insights**. Unlike traditional models, Rosetta used **short sequence fragments** (typically five amino acids) from known structures to assemble potential 3D configurations, inferring the **phi and psi angles** from these fragments.
    - This approach iteratively assessed which configurations minimized **free energy**, leading to compact, energetically favorable structures.
    - Rosetta's **fragment-based methodology** allowed for plausible predictions even for regions

without direct homologs, setting a new standard in structure prediction and revealing the utility of leveraging **empirical data** in structural biology.

2. **The Critical Assessment of Protein Structure Prediction (CASP)**:
    - CASP, a biennial competition, became a proving ground for structure prediction methodologies. In this competition, crystallographers provide protein sequences for which structures are known but unpublished, allowing researchers to test their prediction models against experimentally validated data.
    - CASP exposed the limitations of existing methods, revealing that no approach, until the advent of deep learning models, consistently predicted unknown structures accurately.
3. **AlphaFold 2**:
    - In 2020, **AlphaFold 2** dramatically advanced the field, using **deep learning techniques** to predict protein structures with near-experimental accuracy for many proteins.
    - AlphaFold 2 integrates **evolutionary data from MSAs** with deep learning architectures, which enables it to infer spatial relationships between residues, effectively capturing **long-range interactions** that are difficult to model in traditional approaches.
    - Its unprecedented success at CASP highlighted the power of combining **large-scale evolutionary data** with neural networks, shifting the landscape of structural biology.
4. **Protein Language Models and Attention Maps**:
    - Recent models incorporate **protein language models** that leverage attention mechanisms, similar to natural language processing (NLP) techniques. These models capture context-dependent relationships between residues, allowing them to predict contacts and structural features with high accuracy.
    - **Attention maps** visualize which residues communicate strongly with others, aiding in understanding **functional and structural dependencies** across the protein.

Tertiary Structure Prediction Challenges and Future Directions

Tertiary structure prediction remains an area of active research, especially for proteins with complex folding patterns or those that undergo conformational changes. Despite advances, challenges include:

- **Dynamic and flexible regions**: Many proteins contain intrinsically disordered regions that do not adopt a fixed structure, complicating prediction.
- **Membrane proteins and large complexes**: These proteins pose unique difficulties due to their diverse environments and complex interactions, requiring specialized modeling approaches.
- **Integrating multi-scale data**: Future models aim to integrate various types of biological data, such as cryo-electron microscopy for large complexes, enabling holistic predictions that incorporate different structural and environmental contexts.

In summary, predicting protein tertiary structure has evolved from a theoretical challenge into a computational reality, thanks to insights from **contact maps, co-evolution, and machine learning**. With each breakthrough, the field moves closer to a future where **structure-based functional prediction** becomes integral to understanding and manipulating biological systems, from basic research to therapeutic development.

1:04:20 Comparing protein structures: Quantitative and Structural Alignment Methods

Once we have protein structures, whether from **experimental determination** (e.g., X-ray crystallography) or **predictive models** like AlphaFold, we often need to **compare** these structures. Structural comparison can reveal **conformational similarities**, help assess the **accuracy of predicted models**, and provide insights into **evolutionary relationships** and **functional conservation**.

RMSD: Root Mean Square Deviation for Structural Comparison

A common metric for assessing similarity between two protein structures is the **root mean square deviation (RMSD)**, which quantifies the average distance between corresponding atoms (usually **C-alpha atoms**). RMSD provides a single numerical value that summarizes the **overall alignment quality**:

- **Formula**: RMSD is calculated by taking the squared differences of the coordinates (x, y, z) of corresponding atoms between two structures, averaging these squared differences, and then taking the square root of the result:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((x_i^1 - x_i^2)^2 + (y_i^1 - y_i^2)^2 + (z_i^1 - z_i^2)^2)}$$

where NNN is the number of atoms, and (x_i^1, y_i^1, z_i^1) and (x_i^2, y_i^2, z_i^2) are the coordinates of the i-th atom in the two structures.

- **Interpretation**: Lower RMSD values indicate better alignment (closer structural similarity), with values below **2 Å** often reflecting very similar structures.

Structural Alignment Process

1. **Residue Correspondence**: When comparing two homologous proteins or evaluating a model against an experimentally solved structure, we first need to **map corresponding residues**. For identical sequences, mapping is straightforward, while for homologous proteins, we use **multiple sequence alignments** to match residues across similar regions.
2. **Centering and Superposition**:
    - **Centering**: To minimize differences due to translational offsets, both proteins are first centered by calculating the **center of mass** and adjusting coordinates so that this center aligns with the origin.
    - **Optimal Rotation**: After centering, the structures are rotated for optimal alignment. This rotation ensures that RMSD measurements reflect structural rather than positional differences. Using **singular value decomposition (SVD)** on the **covariance matrix** of the atomic coordinates allows the calculation of a rotation matrix that aligns the structures optimally.

Once the structures are superimposed, we calculate the RMSD to determine how well they align in three-dimensional space.

Beyond RMSD: Evaluating Functional and Structural Fidelity

While RMSD provides a robust baseline, additional comparisons may be necessary to capture **local structural deviations** or **specific functional sites**. Other metrics and approaches include:

- **Local RMSD Calculations**: Focusing on specific structural motifs (e.g., binding sites) provides insights into **functional conservation** even if the global structures vary.
- **Secondary Structure Element Matching**: Comparing helices, beta strands, and loops can offer a more detailed view of structural similarity, as these elements often contribute to **stability** and **function**.
- **Global vs. Local Conformational Similarities**: Certain comparisons emphasize **local similarities** within active sites, useful in functional studies or ligand-binding assessments, while global RMSD captures the entire structure.

Energy Functions in Protein Structure Evaluation and Prediction

In computational modeling and evaluation, **energy functions** play a central role. These functions approximate the **physical interactions** that stabilize protein structures, guiding predictions and offering insight into the effects of **mutations**, **binding affinities**, and **dynamic stability**.

Key components of energy functions include:

1. **Electrostatics**: Models **charge-based interactions** between residues. Attractions and repulsions between charged side chains (e.g., lysine and glutamate) are key to overall structure.
2. **Van der Waals Interactions**: These account for **steric effects** and **non-bonded interactions**, capturing the balance between attractive forces at moderate distances and repulsive forces at very close ranges.
3. **Hydrogen Bonding**: Particularly important in **secondary structures** (alpha helices and beta sheets), hydrogen bonds stabilize the backbone configuration and are vital for folding.
4. **Hydrophobic Effect**: Reflects the tendency of non-polar side chains to cluster away from water, stabilizing protein interiors and driving the folding process.

Applications of Energy Functions

Energy functions form the basis of several critical applications in structural biology:

- **Molecular Dynamics (MD)**: By integrating energy functions over time, MD simulations provide insights into **protein motion**, **folding pathways**, and **conformational changes** under physiological conditions.
- **Drug Binding and Docking**: Predicting how a drug or ligand binds to a protein involves assessing binding energy, optimizing binding affinity, and modeling structural stability.
- **Protein Design and Mutagenesis**: By evaluating energy differences between mutations, researchers can predict the **stability** and **functionality** of modified proteins or design entirely new proteins with desired structural properties.

In sum, structural comparisons, RMSD calculations, and energy functions enable researchers to **quantify protein structure accuracy**, **predict stability**, and **design functional proteins** for biomedical applications. Together, these methods provide the foundation for both **experimental validation** and **computational exploration** in structural biology.

Lecture 9 - Protein Folding Algorithms

Video: Lecture 09 - Protein Folding Algorithms - MLCB24

Slides: Lecture09_10_AlgorithmsForProteinStructure.pdf

00:00 Ab initio protein structure prediction

The prediction of protein structure without relying on homologous templates, known as **ab initio protein structure prediction**, represents one of the most challenging yet exciting areas of computational biology. Ab initio approaches strive to determine a protein's **three-dimensional structure** purely from its **amino acid sequence**, leveraging **fundamental principles of physics** and **empirical energy functions**. These methods have been especially pivotal in exploring **novel protein folds** and **unique structural motifs** not present in existing databases.

The Role of Energy Functions in Ab Initio Methods

Historically, ab initio methods heavily relied on **energy functions** to predict structures. Energy functions represent **theoretical models of the physical forces** governing protein stability, and they encompass a range of interactions:

- **Electrostatics**: Interactions between charged groups (such as positively charged lysines and negatively charged glutamates) are governed by Coulomb's law. These interactions help stabilize the protein's overall structure.
- **Van der Waals Forces**: These forces account for **steric interactions** between atoms, balancing short-range repulsions and attractive forces. Van der Waals forces are crucial in determining **packing density** within the protein core.
- **Hydrogen Bonds**: Often a defining feature in secondary structures, hydrogen bonds stabilize **alpha helices** and **beta sheets** by creating repetitive patterns along the backbone.
- **Hydrophobic Effects**: Proteins naturally fold so that **hydrophobic residues** cluster within the core, away from water, while **hydrophilic residues** are exposed to the solvent. This arrangement is central to the **thermodynamic stability** of proteins in aqueous environments.

By combining these forces, energy functions create a **landscape of possible conformations**, where each point on the landscape represents a potential protein fold. The **global minimum** of this energy landscape typically represents the most **thermodynamically stable structure** of the protein.

Applications of Energy-Based Ab Initio Methods

These energy functions have enabled several key applications in structural biology:

1. **Structure Prediction**: Early ab initio methods used energy minimization to search for low-energy states that approximate the protein's native fold.
2. **Docking and Ligand Binding**: Predicting how proteins interact with each other or with small molecules, such as drugs, relies on calculating the **binding energy** between interacting partners.
3. **Molecular Dynamics (MD)**: By simulating the dynamics of a protein over time, MD provides insights into **conformational changes**, **stability**, and **interaction pathways** within cellular environments.
4. **Predicting Mutation Effects**: Energy functions can also model the impact of specific mutations on protein stability, which is crucial for understanding **genetic disease mechanisms**.
5. **Protein Design**: One of the most ambitious applications, protein design, involves creating **novel**

proteins with specified functions. By understanding the relationships between sequence, structure, and function, researchers can hypothesize new structures and design proteins for functions beyond what exists in nature.

The Emergence of Machine Learning and Deep Learning in Protein Structure Prediction

With the rise of **deep learning** methods, structural biology has seen transformative advancements. Early energy-based approaches have now been complemented—and in many cases, surpassed—by **AI-driven models**. These models bring a new approach to ab initio structure prediction:

- **Training on Large Datasets**: Modern deep learning algorithms like AlphaFold and ESMFold use vast amounts of **structural data** to learn sequence-to-structure relationships, effectively bypassing the need for exhaustive energy calculations.
- **Integration with Evolutionary Data**: Machine learning models can incorporate **multiple sequence alignments (MSAs)** to infer evolutionary relationships, which often correlate with structural and functional conservation.
- **Rapid and Accurate Prediction**: While energy-based ab initio methods can be computationally intense, deep learning models can produce highly accurate predictions in a fraction of the time.

For **single-sequence predictions** where MSAs are unavailable, recent methods have shown substantial improvements, allowing predictions even in **de novo** cases where template structures and evolutionary information are sparse.

The Future of Ab Initio Protein Structure Prediction

While energy functions remain critical for **fine-tuning structural models** and **simulating molecular dynamics**, the integration of **deep learning** has redefined the field of protein structure prediction. Ab initio prediction continues to benefit from combining **physical principles** with **data-driven approaches**, leading to insights that extend beyond static structures to dynamic **protein function and interaction** within cells.

2:56 Energy functions in Protein Structure and Dynamics

Energy functions play a crucial role in modeling and predicting the structure, interactions, and dynamics of proteins. By quantifying the physical forces at play, they allow us to calculate the **stability of specific protein conformations** and simulate interactions with other molecules, including drugs and other proteins. Here, we'll delve into the main components of energy functions, each of which contributes to our understanding of how proteins fold, maintain stability, and interact within the complex environment of a cell.

1. Electrostatics

Electrostatic interactions between charged particles within proteins follow **Coulomb's law**, which states that the energy of two interacting charges depends on the **magnitude of each charge** and their **distance** apart. For proteins, this typically involves charges on individual atoms or groups within amino acids. Coulomb's law also incorporates the **dielectric constant**, which adjusts the interaction based on the environment. For instance, the **dielectric constant in water** (around 80) significantly reduces the energy of interaction compared to a vacuum, due to water's polarity and its tendency to **shield charges** by aligning its molecules around charged sites.

A useful concept for estimating electrostatic effects in proteins is the **Bjerrum length**, which is the distance at which two charges experience an interaction equal to thermal energy, about **2.5 kJ/mol** (equivalent to **kT** at biological temperatures). This length is approximately **7 Å in water**, and it provides a sense of how close charges need to be to have an energetically significant interaction, with larger separations leading to weakened effects due to thermal fluctuations.

2. Van der Waals Forces and the Lennard-Jones Potential

Van der Waals forces are weak attractions that occur between all atoms, contributing to the tight packing seen in protein cores. These forces are described by the **Lennard-Jones potential**, which combines a **short-range repulsive force** (arising when electron clouds overlap, violating the Pauli exclusion principle) with a **long-range attractive force**. The attractive component peaks when atoms are in close proximity, driving the formation of **densely packed regions** within folded proteins.

Van der Waals interactions are fundamental in **stabilizing protein cores** where hydrophobic residues aggregate, but if two atoms approach too closely, repulsive forces sharply increase, preventing overlap and giving proteins a defined shape and rigidity.

3. Hydrogen Bonds

Hydrogen bonds are critical in maintaining protein secondary structures, such as **alpha helices** and **beta sheets**. While sometimes approximated by electrostatics, hydrogen bonds are fundamentally **quantum mechanical in nature**, involving partial sharing of electrons between donor and acceptor atoms. With an energy between **5-10 kJ/mol**, hydrogen bonds are considerably stronger than simple electrostatic interactions, though still weaker than covalent bonds. They are frequently assessed empirically due to their complex nature, and they play a major role in stabilizing regular patterns along the protein backbone.

4. Bonded Interactions

To model the **specific geometry of protein backbones** and side chains, energy functions include terms for **bond lengths**, **bond angles**, and **torsional angles**. These parameters ensure that the protein's backbone and side chains are arranged in a way that respects the **natural bond constraints** found in real proteins, keeping atoms at optimal distances and angles. Bonded interaction terms allow flexibility for minor deviations but impose penalties when bond lengths or angles deviate too far from ideal values.

5. Hydrophobic Effect and Solvent Exposure

The **hydrophobic effect** is a major driving force in protein folding, wherein hydrophobic (water-repellent) residues cluster within the protein core to avoid water, while hydrophilic (water-attracting) residues are generally found on the surface. The **solvent-accessible surface area (SASA)** is often calculated to approximate how much of the protein surface is exposed to water. This can be visualized by "rolling" a water molecule around the protein's surface, tracing an imaginary boundary that highlights regions accessible to the solvent.

The hydrophobic effect is primarily **entropic** rather than enthalpic: burying hydrophobic groups reduces the order imposed on surrounding water molecules, effectively increasing entropy and favoring folded conformations. By treating this effect in energetic terms, SASA can be included in energy functions, with greater solvent exposure penalizing hydrophobic groups, thereby encouraging their burial within the protein's core.

6. Popular Energy Functions in Structural Biology

Several widely used energy functions incorporate these elements to simulate protein behavior and predict structure:

- **AMBER** (Assisted Model Building with Energy Refinement)
- **CHARMM** (Chemistry at HARvard Macromolecular Mechanics)
- **GROMOS** (Groningen Molecular Simulation)
- **OPLS** (Optimized Potentials for Liquid Simulations)

Each of these frameworks has been optimized to perform well in specific applications, including **molecular dynamics (MD) simulations**, **docking** (modeling interactions with small molecules or other proteins), and **structure refinement**. For MD simulations, these functions can be used in **explicit water** environments, where a protein is placed within a virtual box of water molecules, or **implicit water models**, where water's effects are approximated without modeling each water molecule.

7. Hydrophobic Effect as an Entropic Force

The hydrophobic effect, though included in energy functions, is fundamentally **entropic**. This is because it relates to the ordering of water molecules around hydrophobic groups: when hydrophobic residues cluster in the protein interior, they **release water molecules**, allowing these molecules to become more disordered, which **increases entropy**. This effect provides a powerful driving force for folding proteins into compact, stable conformations.

In summary, energy functions capture the complex interplay of forces within proteins, modeling interactions down to the atomistic level. From **electrostatics and van der Waals attractions** to **hydrophobic entropic effects**, these functions allow researchers to simulate and predict protein structures, investigate mutations, and design novel proteins with potential therapeutic applications.

16:00 Relationship between energies and probabilities

The Relationship Between Energies and Probabilities

In statistical mechanics and thermodynamics, understanding the relationship between energy and probability is fundamental. This concept enables us to predict the likelihood of various **states** of a system, such as configurations of molecules in different environments or conformations of proteins. Here, we explore how

energy differences translate into probabilities using **Boltzmann distribution** and **entropy considerations**, which are crucial in biological systems for understanding molecular behavior under thermal fluctuations.

Configurational Space and Probability

Imagine a particle in a **large box** versus a **small box** connected by a passage. If the larger box has 100 times the volume of the smaller one, the particle will spend approximately **99% of its time** in the larger box simply because there are far more possible positions, or **configurations**, in the large box. This setup illustrates how **entropy**—or the number of accessible configurations—impacts probability.

Now consider two boxes of **equal size** but placed at different **altitudes**—one at sea level and the other at the top of Mount Everest. In this case, gravitational potential energy influences the particle's distribution: particles are more likely to occupy the box at sea level due to the **lower potential energy** there. This scenario illustrates how **energy differences** can also dictate probability distribution, favoring states of **lower energy**.

The Boltzmann Distribution

The **Boltzmann distribution** provides a formal way to calculate probabilities based on both **energy differences** and **entropy**. For two states with different energies, the Boltzmann distribution expresses the **relative likelihood** of observing a particle in each state. The formula is:

$$P_i \propto e^{-\frac{E_i}{kT}}$$

where:

- P_i is the probability of the system being in state iii,
- E_i is the energy of state iii,
- k is the **Boltzmann constant**,
- T is the **temperature** in Kelvin.

This distribution tells us that **lower-energy states** are exponentially more likely than higher-energy states, particularly at **biological temperatures**. The factor **kT** represents the average energy associated with **thermal fluctuations**. In practical terms, **kT≈2.5 kJ** at room temperature, meaning any energy difference less than or close to this value can be easily overcome by random thermal movements, allowing the system to explore both low and moderately high-energy states.

Probabilities of Protein States: An Example

Consider a protein that can exist in two conformational states: **open** and **closed**. Suppose the **closed state** is energetically favored, being **5 kJ/mol** lower than the open state. To calculate the fraction of proteins in each state, we use the **Boltzmann factor**:

1. Calculate the relative energy difference in units of kT:
$$\frac{\Delta E}{kT} = \frac{5\,\mathrm{kJ/mol}}{2.5\,\mathrm{kJ/mol}} = 2$$
2. The probability ratio of open to closed is then given by e^{-2}, which is approximately **0.135**.
3. To find the fraction of proteins in the **closed state**, we use:
Fraction in closed state=11+e−2≈0.88\text{Fraction in closed state} = 1/(1 + e^{-2})≈0.88

This means about **88% of proteins** will be in the closed state, while the remaining **12%** will be in the open state, reflecting a strong but not absolute preference for the closed configuration.

General Formula: The Partition Function

The probabilities derived from the Boltzmann distribution rely on calculating the **partition function (Z)**, which normalizes probabilities across all possible states:

$$P_i = \frac{e^{-\frac{E_i}{kT}}}{Z} \quad \text{where} \quad Z = \sum_j e^{-\frac{E_j}{kT}}$$

The **partition function** Z sums the contributions of all possible states, each weighted by its Boltzmann factor. This normalization ensures that the total probability sums to 1, balancing all configurations based on their respective energy levels.

Energy and Entropy in Protein Conformations

In protein folding, the interplay between **energy minimization** and **entropy maximization** determines the protein's final structure. Lower-energy states are generally more favorable, but the **entropic cost** of ordering a protein must also be considered. When a protein folds, it typically minimizes energy by adopting specific contacts and conformations, but it sacrifices configurational freedom, lowering entropy. The **free energy** of a folded protein thus reflects both enthalpic (energy-related) and entropic contributions.

By understanding these relationships, we can better predict how proteins will fold, interact with other molecules, and respond to **environmental changes**. This foundational concept links **thermodynamic stability** with **biological function**, showing why proteins assume certain shapes and how mutations or other modifications might alter their behavior.

26:00 DNA Looping in the Lac Operon: Understanding Tetramerization and Binding Dynamics

The **Lac repressor** (LacI) plays a critical role in regulating the **lac operon** by binding to DNA at specific **operator sites**. Typically, LacI forms **dimers** to bind DNA, but it can also assemble into higher-order **tetramers**. This tetramerization allows the Lac repressor to engage with multiple DNA sites, facilitating **DNA looping** and enhancing repression of the lac operon. The tetrameric structure of LacI enables it to bind simultaneously to two separate **operator sites** (O1 and O2 or O1 and O3), causing a loop in the DNA that effectively blocks transcription.

Tetramer Formation and DNA Binding Sites

LacI binds to three key operator sites in the lac operon:

- **O1**: The primary binding site.
- **O2**: A secondary site.
- **O3**: Another secondary site, located upstream or downstream depending on the operon context.

When LacI binds to **O1 and O2** or **O1 and O3** as a tetramer, it physically brings these two DNA regions closer together, forming a **loop**. This looped configuration stabilizes the repression state, as it makes the bound repressor harder to dislodge. Importantly, **tetramerization** allows LacI to control operon expression with increased efficiency, as it doubles the chances of a repressor binding to the operon through either of the two possible site combinations (O1-O2 or O1-O3).

Enhancing Binding via Energetic Compensation: A Thought Experiment

In this system, the presence of two possible configurations (binding to O1-O2 or O1-O3) essentially **doubles the effective binding probability** of LacI to the operon. To illustrate this, consider a scenario where **O3** is mutated to eliminate LacI binding. With O3 out of play, LacI can only form loops by binding to **O1 and O2**.

To maintain the same overall **binding frequency** to the operon as before the mutation, we need to compensate for the loss of the O1-O3 option. This compensation requires strengthening the **binding affinity** between LacI and the remaining **O2 site**. This adjustment means lowering the **binding energy** at O2 to account for the reduced number of binding configurations.

Calculating the Necessary Energy Adjustment

The relationship between **probability** and **energy** differences follows from the **Boltzmann distribution**. To compensate for the reduction in configurations, we need to reduce the energy of the O2 site by: $\Delta E = -k\,T\ln(2)$

where:

- **k** is the Boltzmann constant,
- **T** is the temperature in Kelvin (often in the context of biological systems, room temperature, so $kT \approx 2.5$ kJ/molkT

Given that $\ln(2) \approx 0.693$\ln(2) \approx 0.693$\ln(2) \approx 0.693$, the energy reduction needed at the O2 site would be

approximately: $\Delta E \approx -2.5\ \text{kJ/mol} \times 0.693 \approx -1.73\ \text{kJ/mol}$

This reduction increases the likelihood of LacI binding effectively at O1-O2, compensating for the loss of the O1-O3 binding pathway.

Strengthening the Lac Repressor-O2 Interaction

To achieve this energy adjustment without altering LacI's binding affinity for other sites (e.g., O1), changes should ideally be made at **O2's DNA sequence** rather than the protein itself. Adjusting the **base pairs** at O2 can create a more favorable binding environment for LacI by enhancing the **hydrogen bonding** or **electrostatic interactions** between O2 and the LacI residues involved in binding. This selective modification preserves LacI's affinity for O1 while increasing its binding strength at O2, thereby maintaining repressive control over the operon.

The Physics of DNA Looping and Lac Repressor Function

This example illustrates how **energetics and structural configurations** determine the functional dynamics of gene regulation. The **DNA looping** induced by LacI tetramerization demonstrates a sophisticated form of regulatory control, where spatial organization and binding flexibility increase the operon's responsiveness. By manipulating binding affinities through mutations at specific sites, researchers can precisely tune regulatory pathways, providing insights into the energetic requirements for **transcriptional repression** and the broader mechanics of **gene regulation** at the molecular level.

32:00 Deep Learning and Protein Structure Prediction: AlphaFold and Beyond

The development of **AlphaFold**, a deep learning-based approach to predicting **protein structure from amino acid sequence**, represents a breakthrough in computational biology. By leveraging **neural network architectures**, AlphaFold achieves remarkably accurate predictions, rivaling even experimentally determined structures.

Background and Goals of AlphaFold

The challenge of **protein folding**—predicting the three-dimensional structure of a protein from its linear sequence of amino acids—has perplexed scientists for decades. In 2020, during the **14th Critical Assessment of protein Structure Prediction (CASP14)**, AlphaFold 2 achieved a remarkable feat, often matching or surpassing traditional methods and even **solving the protein folding problem** for many cases. AlphaFold's predictions align closely with **crystal structures** obtained through **experimental methods** such as X-ray crystallography, particularly for **root mean square deviation (RMSD)**, which measures how closely two structures overlay.

Key Results from AlphaFold's Development

In AlphaFold's performance at CASP14, it dramatically outperformed other protein prediction methods:

- The **RMSD values** of AlphaFold predictions against experimentally derived structures were significantly lower, showcasing precise alignment of **C-alpha atoms** (the backbone atoms of amino acids) between the predictions and experimental results.
- The model effectively captured fine details in structures. For example, in cases with **zinc ions**, AlphaFold correctly placed **histidine residues** around the approximate position of the ion, even though it doesn't explicitly model ions. This ability to **implicitly infer presence** of non-standard protein elements highlights the robustness of AlphaFold's learned representations.

The AlphaFold Approach: Learning Protein Structure through Deep Neural Networks

AlphaFold 1 integrated some **energy functions**, but **AlphaFold 2** entirely relies on deep learning, which **replaces traditional physics-based energy terms** with a complex, multi-layered neural network. AlphaFold's success lies in the sophisticated use of **transformer-based architectures**, a type of neural network particularly suited for interpreting sequence-based data.

Why AlphaFold Works: Information Encoding and Model Architecture

AlphaFold 2 achieves its performance by carefully encoding **sequence information** and **coevolutionary data**:

1. **Multiple Sequence Alignments (MSA)**: AlphaFold identifies **patterns in evolutionarily related sequences**. If two residues consistently vary together across species, the model infers that they are structurally proximate or interact. This coevolution data helps AlphaFold predict **contact maps** (proximity of residues within the folded protein).
2. **Representation of Structural Constraints**: Unlike other models, AlphaFold can incorporate **spatial constraints** such as bond angles and distances, which are crucial in forming realistic protein folds.

3. **Deep Learning Architectures**: AlphaFold uses **attention mechanisms** to learn relationships between amino acids, resembling methods from **statistical mechanics** that relate **energy and configurational probability**. These mechanisms allow AlphaFold to learn dependencies across long sequences and predict secondary, tertiary, and quaternary structures.

Future Directions: Protein Language Models and Transformers

The next stage of research in this field includes **protein language models**—transformer-based models that can treat protein sequences similarly to language, capturing **contextual and evolutionary patterns** in amino acid sequences. These models promise to expand our capacity to **interpret sequence data** for applications beyond static structure prediction, such as understanding **protein dynamics**, **mutational impacts**, and **protein-protein interactions**.

The Implications of AlphaFold's Achievements

AlphaFold's success opens up numerous possibilities:

- **Genome-wide Structural Predictions**: Having access to high-accuracy structures for all proteins in an organism's genome facilitates **comparative genomics**, **drug design**, and **functional annotation**.
- **Structure-Guided Drug Design**: With accurate structural predictions, researchers can design drugs to target proteins even without experimentally obtained structures, potentially accelerating **drug discovery**.
- **Novel Protein Design**: Understanding structure-to-function relationships enables **de novo protein design**, creating proteins with novel functions not found in nature.

Potential Limitations and Challenges Ahead

While AlphaFold provides highly accurate predictions for many proteins, challenges remain:

- **Complex Assemblies**: Predicting structures of **multi-protein complexes** and **proteins bound to non-standard ligands** still requires further refinement.
- **Dynamic Structures**: Proteins are inherently flexible, and AlphaFold typically predicts a **single static structure**, which may not fully capture functionally relevant conformations.
- **Integration with Experimental Techniques**: While AlphaFold's predictions are highly accurate, experimental techniques like **cryo-EM and NMR** are still essential for validating predictions and understanding conformational changes over time.

Conclusion: AlphaFold and the Frontier of Protein Science

AlphaFold exemplifies how **deep learning models can revolutionize** fields traditionally dominated by **physics-based approaches**. Its success illustrates the transformative potential of **neural networks** in capturing complex biological phenomena. As AlphaFold and similar models evolve, they are likely to integrate deeper insights from **language models**, statistical physics, and bioinformatics, empowering researchers to tackle previously insurmountable challenges in **biology** and **medicine**.

38:45 Neural Network Refresher: Fundamentals and Applications to Protein Structure Prediction

To understand how deep learning models, like those in **AlphaFold**, tackle the **protein folding problem**, we need to revisit the **fundamental structure of neural networks**. Neural networks, especially deep neural networks, serve as the backbone of many AI-driven prediction systems by transforming input data through multiple layers of learned transformations, eventually yielding a predictive output. Here's a structured overview of the essential components of neural networks and the mechanisms that make them powerful tools for complex problems, including **biological sequence modeling**.

Structure of a Neural Network

A **neural network** comprises several interconnected layers that progressively transform input data. Each layer contains **neurons** (or nodes), which represent values that propagate through the network.

1. **Input Layer**: This layer receives the initial data, typically represented as a vector of numbers. For a biological sequence, these numbers could encode **amino acid properties** or other features derived from a **protein sequence**.
2. **Hidden Layers**: These intermediate layers perform the transformations that are key to the network's learning. Each **neuron** in a hidden layer receives input from neurons in the previous layer, which it combines using **weights** (scalars specific to each connection) and sometimes **biases** (constant terms added to the output). These transformations are essentially **matrix multiplications**, where the weights and biases are parameters that are fine-tuned during training.

3. **Output Layer**: This layer produces the network's final prediction, which could represent probabilities for different outcomes or specific predictions, such as the coordinates of atoms in a protein.

Activations and Non-Linearities

Each neuron in a layer computes an **activation**, determined by combining inputs through weights and adding biases. The activations pass through a **non-linear activation function**, such as:

- **ReLU (Rectified Linear Unit)**: Common in deep networks, ReLU outputs zero if the input is less than zero and returns the input itself if greater than zero. This simplicity makes ReLU computationally efficient and effective for many tasks.
- **Sigmoid and Tanh**: These functions squash the output to a range (0 to 1 for sigmoid, -1 to 1 for tanh), commonly used in binary classification tasks or recurrent architectures.

These **non-linearities** allow neural networks to approximate complex functions. Without non-linear activations, a network would only be capable of performing **linear transformations**, limiting it to a single layer's equivalent transformation.

Learning Weights and Biases through Optimization

The parameters—**weights** and **biases**—are optimized using an objective, or **loss function**, which measures the error between the network's prediction and the actual output. **Backpropagation** and **gradient descent** are typically employed to minimize this loss function by adjusting the weights and biases across the network. Through repeated updates, the network learns to approximate the function that maps inputs to outputs.

Logits, Probabilities, and Softmax

The network's output often comes in the form of **logits**—values that represent the log-likelihood of each possible outcome. To convert these into **probabilities**, the **softmax function** is applied. The softmax operation exponentiates each logit and then normalizes these values to sum to 1, effectively transforming them into a probability distribution.

Mathematically, softmax is analogous to the **Boltzmann distribution** in statistical mechanics, relating to **energies and probabilities**. This connection is crucial in fields like protein structure prediction, where probabilistic interpretations of molecular states underlie the learning process.

Practical Role of Neural Networks in Protein Structure Prediction

In AlphaFold and similar models, neural networks use sequences of **linear transformations** and **non-linear activations** to learn patterns from **amino acid sequences** and **co-evolutionary data**:

- **Encoding Sequence Information**: By transforming sequences through layers of learned weights, AlphaFold encodes features that capture structural tendencies.
- **Multiple Sequence Alignment (MSA)**: Incorporating MSA data enables the model to identify **evolutionary relationships** between amino acids, which helps predict physical interactions within the protein.
- **Prediction of Contacts and Angles**: Through complex architectures, such as attention mechanisms in transformers, these models can predict which amino acids are likely to interact spatially.

Connection to Thermodynamics and Probabilities in Deep Learning

In complex biological systems, just as in neural networks, **energy states** can dictate configurations. Proteins tend to settle in their **lowest-energy configurations**, and neural networks can approximate these configurations by treating certain patterns as **low-energy states** that are highly probable. The **softmax function** in neural networks, mirroring the **Boltzmann distribution**, makes predictions based on a probabilistic view of "energy" across possible outcomes.

Summary

Neural networks, with their structured layers, non-linear activations, and probabilistic output transformations, offer a flexible, powerful approach to modeling complex biological systems. By stacking multiple layers and introducing **non-linear functions**, they can learn to approximate highly intricate functions that govern protein structure and behavior. This capability is the cornerstone of **AI-driven protein prediction models** like AlphaFold, where advanced architectures bring us closer to accurately predicting and understanding **the architecture of life at the molecular level**.

48:30 Softmax, the Boltzmann Distribution, and Probabilistic Interpretation in Neural Networks

In deep learning, **softmax** serves as a bridge between the **logits** (raw outputs of the neural network) and a **probabilistic distribution** across possible outcomes. This transformation process is not only mathematically

elegant but also carries a profound connection to **statistical mechanics**, specifically to the **Boltzmann distribution**. Understanding this link is key to grasping how neural networks, like those in AlphaFold, convert complex computations into predictions that can be interpreted as **probabilities**.

Softmax: Converting Logits to Probabilities

When a neural network completes its computation, the **final layer** often outputs a set of values known as **logits**. Logits are raw, unbounded scores that don't immediately represent probabilities. To interpret these logits probabilistically, softmax applies an **exponential transformation** and normalizes the outputs, creating a probability distribution over the possible classes or states. This is especially useful when dealing with **multi-class classification tasks** or **protein folding predictions** in models like AlphaFold.

The **softmax function** for a set of logits ($z_1, z_2, \ldots, z_n$) is defined as:

$$P(y = i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

where P(y=i) is the probability of the iii-th class or state. This function ensures that all probabilities are non-negative and sum to one, thereby representing a valid probability distribution.

Connection to the Boltzmann Distribution

In **statistical mechanics**, the **Boltzmann distribution** describes the probability of a system occupying a particular **energy state** at a given temperature TTT. The Boltzmann distribution can be represented as:

$$P(E_i) = \frac{e^{-\frac{E_i}{kT}}}{\sum_{j=1}^{n} e^{-\frac{E_j}{kT}}}$$

where:

- E_i is the energy of state iii,
- k is the Boltzmann constant, and
- T is the absolute temperature.

The Boltzmann distribution implies that lower-energy states (which are more "favorable") are more probable. Similarly, in deep learning, the softmax function interprets lower logit values as less favorable (lower likelihood) and higher logits as more favorable, aligning closely with the interpretation of **energy states** in thermodynamics.

To simplify computations in deep learning, we often assume that **kT=1kT = 1kT=1**. This means we treat logits directly as if they were analogous to energies without needing a scaling factor, making the softmax function directly usable as a probability distribution.

Why This Matters for Neural Networks and Protein Structure Prediction

1. **Probabilistic Predictions**: Using softmax, neural networks can provide **probability distributions** over multiple potential outputs, allowing models to estimate confidence levels in predictions. For example, in protein folding prediction, softmax could help determine which folded structure is most likely based on computed logits for various possible conformations.
2. **Interpretability via Energies**: Thinking of logits as energies provides a **natural interpretation** of neural network outputs in terms of **energy minimization**. This is particularly valuable in models like AlphaFold, where protein folding corresponds to finding a **global energy minimum**—the conformation with the lowest free energy.
3. **Efficiency and Scalability**: Softmax is computationally efficient and scales well with the number of output states, making it highly suitable for models that deal with large sets of possibilities, such as predicting protein structure from numerous possible configurations.
4. **Deep Learning as Statistical Mechanics**: The use of softmax to approximate the Boltzmann distribution showcases the **shared principles** between **deep learning** and **statistical physics**. This

analogy reinforces the idea that deep learning models can function similarly to physical systems, finding stable, low-energy configurations (i.e., optimal predictions) through iterative computations.

Final Thoughts

The softmax function, by mirroring the Boltzmann distribution, enables neural networks to handle complex probabilistic tasks in a way that aligns with **natural physical processes**. For fields like **protein structure prediction**, this provides both a robust probabilistic framework and a bridge to **thermodynamic interpretations**—essential for translating intricate biochemical sequences into practical predictions about structure and function.

49:46 AlphaFold 2: Core Mechanics and Innovations in Protein Structure Prediction

AlphaFold 2 represents a groundbreaking advancement in **protein structure prediction**, achieving accuracy comparable to experimental methods in many cases. Its architecture integrates **deep neural networks** with **evolutionary data**, producing highly reliable structural predictions even without traditional energy functions. Let's delve into how AlphaFold 2 operates, examining its input structure, neural network architecture, and iterative refinement process.

Input Components: Sequence, Multiple Sequence Alignments, and Templates

1. **Primary Sequence**: AlphaFold 2 starts with the **primary amino acid sequence** of the target protein, a linear sequence of residues that needs to be translated into a 3D structure.
2. **Multiple Sequence Alignment (MSA)**: One key to AlphaFold's success is the use of **evolutionary information** derived from MSAs. By aligning the target sequence with homologous sequences from various organisms, AlphaFold extracts **patterns of conservation and co-evolution**. These patterns provide insight into which residues are functionally important or may interact with each other within the folded structure.
    - **Conserved Residues**: Residues that remain unchanged across species suggest functional importance. Charged residues like lysine or arginine that are conserved often appear on the **protein's surface**, potentially interacting with other molecules. Hydrophobic residues like tryptophan, if conserved, are likely part of the **protein's core**, aiding stability.
    - **Co-evolutionary Signals**: Patterns of correlated changes between residues (e.g., a positive charge in one position matched by a negative charge in another) imply **physical proximity** within the protein structure, as these residues may form **stabilizing interactions**.
3. **Structural Templates**: If homologous structures exist in databases like **RCSB**, AlphaFold can incorporate them as templates to guide its predictions. Remarkably, however, AlphaFold performs almost equally well without templates, indicating its capability to independently learn structural rules.

Representations: Encoding Protein Information for Neural Networks

AlphaFold utilizes two main data representations to process the input:

- **Pair Representation**: This is a **tensor** of size L×L×128L \times L \times 128L×L×128 (where LLL is the protein length), encoding information about **inter-residue relationships**. Each entry in this tensor represents a **128-dimensional vector** that captures specific interactions or distances between pairs of residues.
- **MSA Representation**: This tensor, shaped as L×N×32L \times N \times 32L×N×32 (where NNN is the number of aligned sequences in the MSA), captures evolutionary patterns across sequences. Each entry holds a **32-dimensional vector**, encapsulating interaction potentials and conservation scores across the multiple sequence alignment.

Evoformers: Transforming Information Iteratively

The **Evoformer** module is central to AlphaFold 2's architecture. It is inspired by **Transformer neural networks** (hence the name "Evoformer") and iteratively refines both the MSA and pair representations to encode structural and evolutionary insights:

1. **Information Flow**: The Evoformer exchanges information between the **MSA representation** (evolutionary context) and **pair representation** (residue-residue relationships), enabling a complex understanding of **spatial and evolutionary constraints**.
2. **Attention Mechanisms**: Similar to Transformers, Evoformers utilize attention mechanisms to focus on important interactions, learning which residues or residue pairs are most relevant for accurate folding.
3. **Iterative Refinement**: The process is repeated over several cycles, each pass yielding a more refined

representation that integrates spatial and functional information about the protein.

Structure Module: Generating the 3D Structure

The refined information from the Evoformer feeds into the **Structure Module**, which constructs a **3D model** of the protein. This module directly translates the processed information into spatial coordinates for each residue, effectively "folding" the protein sequence into its three-dimensional conformation.

Confidence Measures: Gauging Prediction Reliability: An essential feature of AlphaFold 2 is its ability to provide **confidence scores** on a **residue-by-residue basis**. These scores indicate the model's certainty regarding specific regions of the predicted structure. This feature helps researchers assess which parts of the structure are reliable and which might require further validation.

Iterative Feedback Loop: AlphaFold's structure prediction is not a one-pass process; rather, it **loops through the sequence and MSA multiple times**, refining its understanding with each cycle. This iterative feedback enables the model to correct and optimize the predicted structure progressively, achieving high accuracy.

Evolutionary Data and Neural Networks: A Symbiotic Approach

The combination of **deep learning and evolutionary insights** is crucial to AlphaFold's success. By aligning sequences across species, the model gains a robust understanding of functional constraints imposed by evolution, enabling it to predict structures that are **biologically plausible** and **functionally meaningful**.

1. **Evolutionary Constraints**: Conservation data provides context for understanding which interactions and structural motifs are critical. For example, the hydrophobic core and active sites are often better conserved and thus prioritized in structure prediction.
2. **Neural Network Power**: The Evoformer, inspired by Transformer architectures, learns relationships between residues, capturing the complex dependencies and interactions that underlie protein structure.

Why AlphaFold 2 Is Revolutionary

AlphaFold 2's achievement in solving the protein folding problem is more than just producing accurate structures—it is **redefining structural biology**. The model has shown that it can generate **reliable predictions for nearly all proteins** encoded in a genome, something previously unattainable with purely experimental methods. This capability opens doors to **drug discovery**, **protein engineering**, and **understanding fundamental biological mechanisms**.

Future Potential

While AlphaFold 2 has brought the field closer to a "solution" for the protein folding problem, the broader **applications of these models are just beginning** to unfold. With structures for countless proteins now readily available, the next challenge lies in understanding and leveraging these insights to drive innovations in **medicine, biotechnology, and synthetic biology**.

58:40 Embedding Discrete Data: Transforming Protein Sequences into High-Dimensional Vectors

AlphaFold's approach to predicting protein structures relies on **embedding discrete data** in a way that retains contextual and relational meaning within high-dimensional space. Embeddings transform simple, discrete identifiers—such as amino acid residues or protein sequence positions—into **continuous vector representations**. This transformation is foundational for neural network models to process structured biological data meaningfully.

The Concept of Embedding in Neural Networks

1. **Discrete Data as Scalars**: In their raw form, **amino acids** in a sequence are discrete entities, represented by numbers (e.g., integers from 1 to 20, each corresponding to one of the 20 amino acids). These are essentially categorical values with no inherent numeric relationships between them.
2. **Expansion to High-Dimensional Vectors**: To make these discrete values useful for a neural network, AlphaFold **expands them into vectors** with continuous, real-valued elements. For example, instead of representing an amino acid with a simple integer, it is transformed into a **vector of length 32 or 128**. This transformation captures more nuanced relationships between amino acids, such as their physical or chemical properties.
3. **Embedding Matrices**: An embedding can be thought of as a large matrix where each discrete input (e.g., an amino acid) corresponds to a unique vector in high-dimensional space. For instance, if the model uses a **128-dimensional embedding**, each amino acid would be represented by a unique vector of length 128. This matrix is **optimized during training** to ensure that similar inputs (e.g., hydrophobic residues) have similar vector representations, which can be highly informative for

predicting structure.

Embeddings in Natural Language Processing (NLP): An Analogy

AlphaFold's embedding approach is closely related to embeddings in **natural language processing** (NLP), where a similar method transforms discrete words into vectors. In NLP, embeddings capture semantic similarities between words, making it possible to represent complex relationships through vector math. For example, **Word2Vec**, a popular NLP tool, creates embeddings such that relationships like "king - man + woman ≈ queen" hold true, reflecting an implicit understanding of **gender and hierarchy** within the vector space.

- **Analogous Relationships in Proteins**: In protein embeddings, similar amino acids (e.g., those with similar charges or hydrophobicity) cluster together in the vector space, just as words with similar meanings do in NLP. This clustering aids AlphaFold in inferring the likely locations of residues within a protein structure based on evolutionary and structural patterns.

Creating Embeddings for AlphaFold's Input Data

AlphaFold uses embeddings not only for individual amino acids but also for other structural features in the input data. It builds separate embeddings for **multiple sequence alignments (MSA) and pairwise interactions** between residues. These embeddings are integrated into the model to represent complex relationships within the protein sequence, as well as spatial interactions.

1. **Multiple Sequence Alignment (MSA) Representation**: AlphaFold creates a matrix where one dimension represents **sequence positions** and the other represents the **aligned sequences**. Each position is represented by a vector (e.g., of length 32), capturing both **conserved positions** and **co-evolutionary patterns**. Conserved positions might indicate structural or functional importance, while co-evolving residues hint at physical proximity or interactive roles in the folded structure.
2. **Pair Representation**: This is a matrix of **residue-residue interactions**, where each element is a vector (e.g., of length 128) representing the interaction strength or likelihood between two residues. This matrix becomes a tensor with dimensions L×L×128L \times L \times 128L×L×128 (where LLL is the protein sequence length), and it encodes potential spatial relationships, essential for accurately modeling 3D structure.

Constructing AlphaFold's Input Features with Embeddings

Once embeddings for each feature are created, AlphaFold combines them through two main methods:

- **Concatenation**: When two embeddings provide distinct information (e.g., MSA and pairwise interactions), AlphaFold combines them by concatenating their vectors, effectively creating a more informative, high-dimensional representation.
- **Addition**: When embeddings contain redundant or complementary information, AlphaFold merges them through **vector addition**. This addition in high-dimensional space allows AlphaFold to consolidate related features without increasing dimensionality unnecessarily.

The Power of Embedding in High-Dimensional Space

Embedding discrete data into continuous high-dimensional vectors is central to **AlphaFold's success**. By transforming categorical biological data into vectors that capture relationships, the model can handle complex structural and evolutionary information in ways that traditional discrete representations would miss. This embedding method, borrowed from NLP, is crucial in AlphaFold's ability to infer intricate structural patterns and relationships across residues.

In conclusion, **embedding enables AlphaFold to convert raw biological data into a form that neural networks can interpret**, preserving essential patterns and relationships across the protein structure. This innovation, along with deep learning architectures like Evoformers, allows AlphaFold to make highly accurate predictions of 3D protein structures from sequence data alone.

1:05:35 Adding High-Dimensional Vectors in Protein Language Models

In high-dimensional spaces, the process of adding vectors takes on unique properties that diverge significantly from our intuitions in lower dimensions. This is especially pertinent in **protein language models**, where the dimensionality of vector representations (e.g., 1024 or even 1280 dimensions) allows for the encoding and manipulation of complex information, like entire protein chains, in ways that retain meaningful patterns.

Intuition of High-Dimensional Vector Addition

1. **Loss of Retraceability in Low Dimensions**: In a three-dimensional space, adding two vectors results

in a new vector, but the initial components of that sum are effectively "lost." This means we cannot reverse the operation to determine the original vectors just from their resultant sum. In high-dimensional spaces, however, the outcome of adding vectors behaves differently.

2. **Maintaining Distinguishability in High Dimensions**: In spaces with a very high number of dimensions, adding vectors does not obscure their individual contributions. For instance, even if we sum many vectors in a 1024-dimensional space, the **individual properties** of those vectors can still be detected within the resultant sum by using specific mathematical tools, such as **dot products** or **cosine similarity**. This allows for reconstructing or identifying the contributing components to a much greater extent than in low-dimensional spaces.
3. **Signal Preservation**: High-dimensional spaces allow for an almost **orthogonal arrangement** of vectors, meaning that randomly chosen vectors have minimal overlap in direction. If we take a specific vector as a signal (e.g., a vector with a 1 in a certain dimension followed by 0s), randomly chosen vectors added to it will interfere very little with its direction. This limited interference ensures that the original signal remains largely intact even after multiple vectors are added.

Probability and Cosine Similarity

1. **Quantifying Interference with Cosine Similarity**: In a 1024-dimensional space, if we add vectors together, each additional vector will only slightly distort the original signal. The degree of distortion is measured by **cosine similarity**, which calculates how aligned two vectors are. Due to the large dimensionality, the **cosine similarity between random vectors is typically low**, meaning they don't significantly overlap or interfere with each other.
2. **The Power of Averaging Effects**: The likelihood of a randomly chosen vector substantially affecting the original signal vector (say, by more than a small percentage of its length) is extremely low. This ability to add vectors with minimal interference is what enables the model to sum multiple vectors and still retain meaningful information about each.

Application in Protein Language Models

In the context of protein language models, such as AlphaFold's **ESM2 model**, these high-dimensional properties are harnessed to create a **single representation of an entire protein chain**:

- **Summing Amino Acid Embeddings**: Each amino acid in a protein sequence is represented by a high-dimensional vector. By summing these embeddings, the model constructs a **composite vector** that encapsulates information about the whole sequence.
- **Interrogating the Composite Vector**: After constructing this composite vector, we can query it to retrieve specific information. For instance, if we want to know the prevalence of a particular amino acid (e.g., tryptophan) in the sequence, we can take the **dot product** of the composite vector with the embedding vector of tryptophan. This operation will give us an approximate count of that amino acid in the sequence.

This approach demonstrates how **high-dimensional vector addition enables compact and retrievable storage of complex sequence information**. In the composite vector, the entire sequence's structural and chemical properties are preserved, which can then be accessed as needed through further vector operations. This capability is a cornerstone of how protein language models efficiently encode and retrieve patterns, relationships, and quantities of amino acids in complex biological data.

1:10:30 Detailed Walkthrough of AlphaFold 2's Mechanisms

AlphaFold 2, a pioneering model in protein structure prediction, builds on the intricate embedding and sequence comparison techniques discussed earlier, but adds a new layer of sophistication by integrating **attention mechanisms** and **iterative information sharing** across representations to predict protein structures with high precision.

1. Embeddings and Information Passing Between Residues

The foundation of AlphaFold 2's success lies in its **ability to capture and share information** among amino acid residues effectively. AlphaFold 2 doesn't just treat each residue as an isolated entity; rather, it enables residues to **"communicate" information** about their properties and relationships within the entire protein chain. This is achieved primarily by leveraging **attention mechanisms**, which allow each residue to **evaluate its relationship with other residues** and decide the importance of that connection. Key aspects include:

- **Row-Wise and Column-Wise Attention**: AlphaFold 2 applies attention in two directions on the **multiple sequence alignment (MSA)** — both rows and columns. This helps AlphaFold 2 to:

    - **Capture evolutionary correlations** across different species (row-wise).
    - **Identify patterns within the sequence** (column-wise), such as residues that consistently appear together, which often hints at physical interactions.
- **Integration with the Pair Representation**: The pair representation, which stores data about residue pairs' interactions, informs how attention is applied across MSA rows. By allowing **pairwise comparisons to influence attention**, AlphaFold 2 can prioritize meaningful interactions based on structural clues.

2. Structural and Spatial Encoding of Protein Geometry

AlphaFold 2 goes beyond typical sequence-based information, embedding spatial relationships directly into the model:

- **3D Spatial Constraints Through Pairwise Distances**: One of AlphaFold 2's training objectives is to **predict the distance between residue pairs**, which provides strong spatial constraints that reinforce 3D structure accuracy. By using a **probabilistic distribution of distances**, AlphaFold 2 can express structural uncertainty where needed, which is crucial for complex, flexible protein segments.
- **Triangle Inequality Enforcement with Triplet Interactions**: The model introduces a **triangle inequality** mechanism, ensuring that spatial relationships remain geometrically valid:
    - Triplet interactions allow residue pairs like (i,j)(i, j)(i,j) and (j,k)(j, k)(j,k) to impact the relationship between residues iii and kkk.
    - This setup enforces realistic geometric constraints in 3D space, ensuring that the model maintains valid structural relationships.

3. The Evoformer Block: Iterative Refinement Through Attention Layers

The **Evoformer** is a core component of AlphaFold 2's architecture, facilitating iterative refinement and information integration across sequences and pairs:

- **Passes Through 48 Evoformer Blocks**: AlphaFold 2's model passes through 48 Evoformer blocks, each containing unique parameters. This repetition allows the model to **refine its understanding incrementally**, simulating how residues and segments interact structurally over successive layers.
- **Combining the MSA and Pair Representations**: Within each Evoformer block, both the **MSA and Pair Representations** interact to improve predictions, sharing refined insights about evolutionary patterns and spatial constraints across layers.

4. Final Structure Prediction and Confidence Scoring

After completing all Evoformer passes, AlphaFold 2 generates a final representation of the protein's MSA and pair relationships. From here:

- **Direct Prediction of Protein Structures**: Rather than relying on energy-based models as AlphaFold 1 did, AlphaFold 2 utilizes a neural network to **output a final 3D protein structure** directly. This approach bypasses traditional modeling steps, relying on the neural network's refined internal representation.
- **Confidence Measures for Residue Positions**: AlphaFold 2 assesses its confidence in each residue's placement, providing scores that help researchers evaluate which segments of the predicted structure are most reliable. This is crucial for identifying regions that may need experimental validation.

5. Positional and Sequence Distance Embeddings

Another innovation in AlphaFold 2 is its handling of positional information:

- **Embedding Numerical Position in the Sequence**: Each residue's position within the sequence is embedded as a numerical feature, which the model uses to assess relative positions, independent of the protein's linear sequence order.
- **Incorporating Relative Sequence Distance**: Throughout the model, **relative distance within the sequence** is used multiple times to enhance predictions, particularly for regions that are physically far apart but might still interact closely in the folded structure.

Summary and Next Steps

AlphaFold 2 represents a monumental shift in protein structure prediction by leveraging high-dimensional embeddings, attention-based information sharing, and direct structure prediction via neural networks. This multi-layered approach allows AlphaFold 2 to capture both local and global structural relationships within

protein sequences, yielding results that are comparable to experimentally derived structures.

Next, the exploration will delve into **protein language models** like ESM and Transformer architectures, examining how these models push boundaries in biological data analysis by further enhancing the representation and interpretability of protein sequences.

Lecture 10 - Protein Structure with Transformers

Video: Lecture 10 - Protein Structure with Transformers

Slides: Lecture09_10_AlgorithmsForProteinStructure.pdf

00:00 Algorithms for protein structure using transformers

In this lecture, we delve into advanced **algorithms for protein structure prediction** with a focus on **Transformer architectures**, building on AlphaFold's foundational ideas. Here, we explore how Transformers manage to capture the intricacies of protein sequences and structures, utilizing **attention mechanisms** and **rich intermediate representations**.

1. Recap of AlphaFold 2's Mechanisms

AlphaFold 2 uses two main internal representations:

- **Multiple Sequence Alignment (MSA)**: Carries evolutionary information, helping to inform the structure by highlighting conserved regions that may be crucial for folding and function.
- **Pair Representation**: Encodes information about the **3D spatial relationship** between residue pairs, capturing the geometry and relative positioning.

Beyond simply generating a structure, AlphaFold was optimized to solve intermediate objectives:

- **Distance Prediction**: Predicts a distribution of distances between residue pairs.
- **Orientation and Rotation**: Forecasts relative orientations and angles, embedding geometric constraints.
- **Local Distance Difference Test (LDDT)**: Measures local structural accuracy, allowing the model to assess confidence in different parts of the predicted structure.

LDDT is particularly valuable because it enables **confidence scoring**:

- Scores under **50** indicate unreliable predictions.
- Scores between **70 and 90** suggest a good backbone structure.
- Scores above **90** mean the structure is highly reliable and suitable for molecular modeling.

2. Introduction to Transformers in Protein Structure Prediction

Transformer architectures have become a powerful tool for protein structure prediction, especially due to their ability to **handle sequential data** efficiently and to capture **long-range dependencies** between residues. The core of this capability lies in **self-attention mechanisms**, which allow every residue to assess its importance relative to every other residue in the sequence.

- **Self-Attention Mechanism**: Key to the success of Transformers, this mechanism enables each residue to interact dynamically with all others in the sequence, allowing the model to identify meaningful interactions that contribute to the overall structure.
- **Multi-Headed Attention**: This technique divides the self-attention into multiple "heads," each focusing on different relationships within the sequence. It can capture a range of interactions that might vary in significance, from local short-range interactions to critical long-range structural constraints.

3. Evolutionary Data and Pairwise Interactions in Transformers

Incorporating evolutionary and geometric data, Transformers for protein prediction can leverage:

- **Position Embeddings**: Encoding relative or absolute positions of residues helps the model maintain structural order, critical for interpreting long protein chains.
- **Pairwise Residue Representations**: Similar to AlphaFold's pair representation, Transformers use additional embeddings to represent residue-residue interactions across the protein sequence.

4. Leveraging Intermediate Representations

A distinguishing feature of modern protein-predictive Transformers is their use of **intermediate layers** to refine the information iteratively:

- **Layer-wise Refinement**: Each layer of the Transformer refines the representation of the protein, capturing increasingly abstract structural relationships.
- **Attention Maps**: Visualization of attention weights across layers can reveal which residues are more interdependent, providing insights into potential folding patterns or structural motifs.

5. Objective Functions and Optimization

Similar to AlphaFold, Transformers for protein prediction utilize:

- **Distance and Angle Predictions**: Objective functions in training include distance and angle constraints that guide the model towards more physically realistic structures.
- **Confidence Scoring and Error Prediction**: By predicting structural confidence (e.g., LDDT) as an intermediate task, the model can determine which regions of the prediction are reliable, which can be valuable for experimental follow-ups.

Key Takeaways

1. **Transformer models offer new insights into protein structure prediction** by using attention mechanisms to model both local and global residue interactions effectively.
2. **Intermediate representations and objectives** (such as pairwise distance constraints) help the model capture intricate structural details that improve final predictions.
3. **Confidence scoring mechanisms**, such as LDDT, allow for a realistic assessment of the prediction's reliability, making it possible to focus on the most reliable regions for experimental validation.

Next Steps

In future discussions, we'll delve deeper into **Transformer language models**, examining how these models can leverage linguistic-style embeddings and attention mechanisms specifically tailored to protein language modeling. This will enhance the understanding of **sequence-function relationships** and potentially lead to breakthroughs in designing synthetic proteins or understanding protein misfolding in diseases.

4:30 AlphaFold 3: Key Architectural Changes and Advancements

AlphaFold 3 builds upon the successes of AlphaFold 2 with notable architectural changes and **significant enhancements in functionality**, marking a new step forward in protein structure prediction and modeling.

1. Simplification and Bottlenecking of Representations

One of the **major changes in AlphaFold 3** is the **simplified architecture** compared to AlphaFold 2:

- **Focused Pair Representation**: In AlphaFold 2, information flowed dynamically between the **Multiple Sequence Alignment (MSA)** and **pair representation** matrices, allowing evolutionary and geometric data to interact. AlphaFold 3 further centralizes this by **bottlenecking all critical information into the pair representation matrix**.
- **Geometric Constraint Storage**: The pair representation matrix now holds essential data for predicting geometric constraints directly, reducing the need for complex cross-layer communication while still capturing the essential spatial information of residues and atoms.

2. Integration of Diffusion Models

A transformative addition in AlphaFold 3 is the use of a **diffusion model** in place of the neural network structure seen in AlphaFold 2:

- **Diffusion-Based Structure Prediction**: Diffusion models, emerging as a prominent tool in generative AI, introduce a probabilistic approach to gradually refining protein structures. By starting with a rough approximation and iteratively "diffusing" toward a more accurate structure, these models capture intricate details and variability with increased precision.
- **Enhanced Structure Prediction**: The diffusion model excels in handling uncertainty and providing a more comprehensive view of protein folding landscapes, potentially yielding more accurate predictions, especially in challenging cases where proteins may have multiple stable conformations.

3. Expanded Interactions and Molecular Complex Modeling

AlphaFold 3 expands its scope beyond individual proteins to **model complex biomolecular interactions**, which is pivotal for advancing our understanding of cellular function and drug discovery:

- **Protein-Protein Interactions**: The model now incorporates data on **protein interactions**, which is essential for studying **multimeric complexes** and understanding how proteins assemble and interact functionally in cellular environments.

- **Protein-DNA Interactions**: AlphaFold 3 includes **protein-DNA binding predictions**, providing insights into how proteins regulate genetic information, bind to specific DNA sequences, and influence transcriptional processes.
- **Protein-Ligand Interactions**: By integrating ligand interactions, AlphaFold 3 allows for **drug-target modeling**, an essential feature for designing pharmaceuticals that target specific protein sites with high affinity and specificity.

4. Practical Applications and Potential Impact

These updates make AlphaFold 3 a **powerful tool for a wide range of applications** in structural biology and biomedicine:

- **Drug Discovery and Protein Engineering**: The ability to model protein-ligand interactions directly within AlphaFold 3 opens new possibilities in **rational drug design** and **therapeutic development**.
- **Genomic Regulatory Modeling**: By predicting protein-DNA interactions, AlphaFold 3 supports research into **gene regulation mechanisms**, allowing scientists to explore how proteins bind to and modulate DNA.
- **Complex Protein Assemblies**: The model's capacity for accurate protein-protein interaction predictions facilitates research into **cellular complexes**, advancing our understanding of cellular machinery and the pathways that underpin disease.

Conclusion

With its **simplified and focused architecture**, reliance on **diffusion models**, and expanded ability to model **interactions across biomolecular types**, AlphaFold 3 significantly broadens the scope of computational structural biology. These advancements make it possible not only to predict individual protein structures but also to model complex biological systems, a leap that will likely reshape research in **molecular biology, genomics, and drug discovery**.

The architecture of AlphaFold 3 exemplifies how AI can enhance our understanding of biological processes, marking an exciting development for **in silico modeling** and **predictive biology**. This new version promises to have profound applications across many fields, extending the utility of AI in understanding life at the molecular level.

7:50 Protein Design with Generative AI and Deep Learning

Protein design has emerged as a critical area where generative AI and deep learning are transforming how we develop proteins with **novel functions and unprecedented stability**. Following the success of models like AlphaFold in solving the protein folding problem, researchers are now exploring how to **design proteins from scratch** for applications that go beyond what is found in nature.

Goals of Protein Design

The primary aim of protein design is to create **proteins with specific, often novel functions** that do not naturally occur. These applications range from:

- **Custom Enzymatic Activity**: Designing proteins to catalyze chemical reactions that do not exist in nature.
- **Targeted Protein Interactions**: Creating proteins that can activate or inhibit specific genes or pathways, useful for therapeutic applications.
- **Controlled Molecular Assemblies**: Designing proteins that form molecular cages to encapsulate and release molecules in response to stimuli.
- **Biosensors**: Engineering proteins that bind small molecules and produce a signal, useful in diagnostic applications.

Milestones in Protein Design

One of the earliest achievements in protein design was **Top7**, a protein created in David Baker's lab that exhibited a **novel fold**—a structure not observed in naturally evolved proteins. Top7 demonstrated the potential for deep structural innovation and **extreme stability** in designed proteins, even exceeding the stability of naturally occurring proteins.

A Workflow for Protein Design Using AI

1. **Generating a Protein Backbone**:
    - The process begins by defining the **shape of the protein**, often driven by the desired function.

For example, if an enzyme is needed to stabilize a specific reaction transition state, this requirement informs the backbone structure.
    - This step may involve rational design, where specific structural elements are chosen, or a generative model like **RF Diffusion** to create various backbone possibilities.
2. **Predicting the Amino Acid Sequence**:
    - With a desired backbone structure, the next step is to determine a **suitable amino acid sequence**. Here, software such as **Protein MPNN (Message Passing Neural Network)** can predict an amino acid sequence that will likely fold into the desired shape.
    - This step allows the model to "guess" which residues will best suit the structure's constraints and function.
3. **Verifying with AlphaFold**:
    - After selecting a candidate sequence, the predicted structure can be validated using **AlphaFold**. AlphaFold generates a 3D structural prediction and assigns a **Predicted Local Distance Difference Test (PLDDT)** score, indicating confidence in the model's accuracy.
    - If AlphaFold indicates a reliable structure (often with a PLDDT above 70 for backbone reliability), the sequence may be considered for synthesis and laboratory testing.
4. **Iterating the Design Process**:
    - If the design does not yield high confidence in AlphaFold, adjustments are made. Designers may either:
        - **Modify the backbone** by selecting nearby designs from RF Diffusion.
        - **Optimize the sequence** using Protein MPNN for more favorable residues.
    - This iterative process continues until a design achieves a satisfactory level of predicted stability and accuracy.

The Role of Generative Models and Deep Learning

The protein design workflow benefits significantly from **AI-driven tools**, which make it possible to automate and refine each stage:

- **Diffusion Models**: RF Diffusion, similar to methods used in AI image generation, explores potential protein backbones by starting with a rough outline and iteratively refining it to produce realistic structures.
- **Machine Learning in Sequence Prediction**: Tools like Protein MPNN leverage machine learning to predict sequences that will conform to desired structural parameters.
- **Modeling Protein Interactions**: Advances in models, particularly seen in AlphaFold 3, make it feasible to incorporate **protein-protein, protein-DNA, and protein-ligand interactions** directly, which is essential for creating functionally integrated protein systems.

Future Implications

Protein design holds immense promise for **drug development, synthetic biology, and biotechnology**. With AI's help, it's possible to create proteins tailored to nearly any molecular function, paving the way for innovations that extend the limits of biological engineering.

As AI tools continue to evolve, the prospect of designing proteins that are not only functional but also **hyper-stable, efficient, and customizable** will redefine possibilities across numerous fields, from **medicine to environmental science**.

14:05 Diffusion Models in Protein Structure Prediction

Diffusion models have become a significant force in generative AI, with applications extending far beyond image synthesis. Initially popularized by **image-generating models like Stable Diffusion**, these models are now proving transformative for **protein structure prediction** and **design**, offering a novel approach to sampling and generating realistic molecular structures.

Overview of Diffusion Models

The foundational idea behind diffusion models is **learning to reverse a noising process**. Starting with a clear image (or, in the case of proteins, a structured model), noise is gradually added, creating a degraded or fully randomized version. The model is then trained to **"denoise"** progressively, recovering the original structure

step-by-step. This denoising process allows the model to generalize by starting from pure noise and iteratively reconstructing coherent structures—whether they be images or proteins.

1. **Image Diffusion Example**: For visual intuition, imagine a clear image of a dog. As noise is added in stages, the image degrades until it appears as complete random noise. By training a neural network to reverse this process, it becomes possible to synthesize entirely new images by reversing the noise to clarity.
2. **Protein Diffusion Example**: For proteins, the process is similar. Noise is introduced by **randomly moving atoms and adjusting their rotations**, distorting the original structure. The model learns to reverse these distortions, generating valid protein structures from noise. This capability enables us to **sample new protein conformations** by starting with random configurations and iteratively refining them into coherent structures.

Applications of Diffusion Models in Protein Design

In protein design, diffusion models offer a powerful approach for:

- **Exploring the Protein Conformational Space**: Diffusion-based models enable sampling from the broad and complex space of possible protein structures, which is essential for finding viable new folds or functional designs.
- **Reverse-Sampling for Novel Protein Structures**: By training on a set of known protein structures, a diffusion model can start from random configurations and generate unique, plausible proteins. This process aids in the discovery of **structural motifs** or **binding sites** not present in nature.

Diffusion in RF Diffusion and Protein MPNN

- **RF Diffusion**: Specifically for protein structures, RF Diffusion takes advantage of the denoising framework to sample from structural possibilities, creating new protein backbones with varied configurations and properties. This model iterates through noisy representations to converge on **physically plausible structures**, effectively "sculpting" proteins from random initial configurations.
- **Protein MPNN (Message Passing Neural Network)**: While Protein MPNN uses a slightly different architecture (encoder-decoder) to predict sequences from structures, it shares conceptual similarities in transforming high-dimensional representations (such as protein structure) into actionable formats (such as amino acid sequences). The encoder captures the spatial and structural information of a protein, while the decoder translates it into a viable sequence.

State of the Art in Protein Design with Diffusion Models

Diffusion models, along with other generative techniques, enable groundbreaking advancements in:

- **Designing Protein-Protein and Protein-Small Molecule Interactions**: These tools allow researchers to model proteins that can bind specifically to other molecules or proteins, which is key for therapeutic and diagnostic applications.
- **Creating Novel Protein Folds**: With sufficient structural diversity, diffusion models facilitate the creation of previously unseen protein folds, expanding the repertoire of available protein structures for engineering.
- **Engineering Protein Complexes**: By generating proteins with predefined **quaternary structures** or **multiple conformations**, diffusion models can create proteins with complex, adaptable functions, such as biosensors or molecular cages.
- **Modulating Enzymatic Activities and Protein Dynamics**: Future models may incorporate specific **dynamic behaviors** within proteins, allowing the design of enzymes with new reaction mechanisms or proteins that change shape under certain conditions.

As diffusion models evolve, they open a vast landscape for **customizable protein design**. This marks a shift from understanding existing structures to **actively generating novel biomolecules** tailored for specific functions, whether for drug discovery, synthetic biology, or biotechnological innovation.

19:00 Protein Language Models (PLMs) and the Transition to Transformers

Protein Language Models (PLMs) apply the principles of large language models (LLMs), like GPT, to understand the “language” of proteins. These models aim to capture the sequence patterns, structures, and functional relationships inherent in protein sequences, using a vast amount of sequence data to predict and infer biological information. PLMs are typically **large, computationally intensive models** that require considerable resources to train, making them more accessible to researchers through pre-trained models

available for fine-tuning.

Applications of PLMs:

1. **Embedding Extraction**: By processing protein sequences through PLMs, we can extract embeddings from the final layers of the model. These embeddings provide a high-dimensional representation that encapsulates structural and functional attributes of the protein.
2. **Transfer Learning**: Using embeddings for tasks like **subcellular location prediction** or **functional classification** allows PLMs to act as transfer learning models, where rich, general-purpose embeddings are used to support various specific prediction tasks with fewer training examples.
3. **Increased Informational Density**: Instead of working with simple numerical representations (like sequence integers), PLMs transform sequences into embeddings with hundreds of thousands of dimensions, carrying far more context and structural insight.

In the practical tasks for this unit, you will explore these embeddings with the **ESM protein language model**, observing how protein sequences look in the neural network's embeddings and using this data to perform downstream predictions.

Foundations of PLMs in Transformer Architecture

The Transformer architecture, which forms the backbone of PLMs and LLMs, was first introduced in **2017** and revolutionized sequence modeling by offering a highly effective alternative to recurrent neural networks (RNNs). To understand why Transformers became the standard, let's explore the previous sequence modeling approaches and their limitations.

1. **Recurrent Neural Networks (RNNs)**:
    - **Goal**: Originally used for tasks like **language translation**, RNNs were effective in handling input and output sequences of different lengths.
    - **Process**: In translation, for instance, each word in the input (e.g., "The cat ate the mouse") was processed recursively, with each word's context being passed along in the sequence to accumulate meaning progressively.
    - **Limitations**: RNNs struggled with capturing long-range dependencies in text due to difficulties with gradient flow, often losing context for distant elements in a sequence. This limited their effectiveness for complex sequence patterns, such as those in biological data.
2. **Transformers' Solution**:
    - Transformers introduced an **attention mechanism**, allowing the model to focus on relevant parts of the input sequence without needing to rely on recursion. This attention mechanism could assess the importance of each element in the input sequence in relation to others, capturing dependencies regardless of their position.
    - The architecture proved superior in managing long sequences, as each input token could directly "attend" to every other token, enabling richer and more holistic context retention.

Structure of Transformers in PLMs and LLMs

In the PLM context, the Transformer architecture allows for the encoding of protein sequences by processing **sequence embeddings** layer-by-layer through self-attention mechanisms. This allows the model to capture the relationships between amino acids across the sequence, forming representations that incorporate **spatial and functional insights**.

- **Embedding Layers**: Protein sequences are embedded as high-dimensional vectors at the start, representing each amino acid not just by its identity but by a position-aware, learned vector.
- **Self-Attention Mechanism**: Every amino acid "attends" to every other amino acid, helping the model discern which residues have significant functional relationships.
- **Layer Stacking**: The output of the attention layers is passed through feedforward networks, and these are stacked, allowing the model to capture complex relationships within and across amino acid groups.

In PLMs, as in natural language models, the **attention mechanism** enhances the model's ability to understand patterns and dependencies. This approach aligns well with how protein structures function, as residues interact in complex spatial arrangements, not merely sequentially.

Upcoming Topics: Designing Custom Transformer Architectures

The lecture will cover Transformer architecture in depth, offering the foundational intuition for designing custom

Transformer-based models. With a solid grasp of Transformers, you'll be equipped to design or adapt models for unique sequence-based challenges in protein science, advancing beyond existing sequence modeling techniques to tackle highly specialized biological questions.

By the end of this discussion on PLMs and Transformers, you'll understand how models like AlphaFold employ the Transformer structure to learn protein sequence representations that contribute to structure prediction, protein design, and functional understanding across diverse applications.

24:30 Encoder-Decoder architectures and the Introduction of Attention

Overview of Encoder-Decoder Models

The **encoder-decoder architecture** is a foundational structure in many AI models, especially in tasks like translation and image captioning. This model structure is designed to process complex inputs, convert them into a compact, latent representation, and then decode that representation back into a structured output. Here's a breakdown of each component:

1. **Encoder**: The encoder processes input data, like a sequence of text or an image, and converts it into a **latent (hidden) representation**. This hidden representation encapsulates essential features in a reduced or expanded form, making it easier to perform transformations on the data.
    - **Dimensionality Reduction**: Encoders can take complex data (e.g., an image with millions of pixels) and distill it into a simpler, lower-dimensional form. This condensed form is useful for tasks that require understanding key features without the full data complexity.
    - **Dimensionality Expansion**: In some cases, like protein sequence modeling, the encoder starts with simple data (like integers representing amino acids) and adds depth by enriching the sequence with learned representations, which enables complex predictions.
2. **Decoder**: The decoder is the **generative part** of the model, transforming latent representations back into structured outputs, like a translated sentence or a generated image. In generative AI, this approach allows models to synthesize new instances by navigating the latent space, where small variations can produce meaningful outputs.
    - **Generative Potential**: Decoders in generative models can sample the latent space to generate new data (e.g., images of cats based on a label "cat"). This capability is critical in applications like **sequence generation** (e.g., creating protein sequences with desired properties) and **conditional generation** (e.g., generating images with specific attributes).

Limitations in Traditional Encoder-Decoder Models

Before the introduction of Transformers, traditional encoder-decoder models encountered challenges, especially with handling long sequences. With each step in the encoder, data is compressed into a **fixed-length vector** that acts as a bottleneck, limiting the model's ability to capture complex dependencies in the input sequence.

As sequences get longer and more complex, this bottleneck creates issues:

- **Loss of Long-Term Dependencies**: Information from earlier parts of the sequence may not be fully retained, leading to degraded performance on long texts or complex sequences.
- **Information Degradation**: In the "telephone game" analogy, where information is sequentially passed down, some details get lost or altered. This is especially problematic in translation or any task that requires nuanced contextual understanding.

The Innovation of Attention

The key insight leading to **Transformers** was the concept of **attention**. Attention allows models to bypass the limitations of sequential data processing by enabling access to specific parts of the input at any point in the process. Here's the fundamental change that attention introduced:

1. **Direct Access to Input**: In traditional models, information must sequentially flow from input to output. However, with attention, the model can "peek" back at the input directly. This significantly improves the model's ability to retain and utilize long-term dependencies.
2. **Self-Attention in Transformers**: The concept of self-attention, introduced in Transformers, enables every part of the sequence to relate to every other part. For example, when generating a translation, the model can examine all previously generated words and align them with the input sequence dynamically, rather than relying on a rigid, fixed representation.

Transitioning to Transformers

With the attention mechanism, **Transformers** allowed models to overcome the bottleneck of traditional encoder-decoder architectures. This advancement led to significant improvements in natural language processing tasks, and later, it proved equally transformative for biological sequence modeling.

Key Takeaways for PLMs

In protein language models:

- **Attention enables context-sensitive predictions**: Just as in language translation, attention allows PLMs to model interactions between distant residues in a protein sequence.
- **Enhanced accuracy**: By capturing the nuanced "grammar" of protein sequences, PLMs with attention mechanisms are better suited for tasks like protein structure prediction and functional annotation.

The next segment delves deeper into **attention mechanisms**, focusing on how they enable Transformers to dynamically capture dependencies across a sequence, paving the way for more sophisticated applications in both language and protein modeling.

34:35 Limitations of Encoder-Decoder Models and the Birth of the Transformer Architecture

Challenges with Encoder-Decoder Models

The encoder-decoder architecture, which was groundbreaking in earlier neural network applications like translation, encountered key limitations when dealing with long or complex sequences. Here are the primary issues:

1. **Recursive Structure Bottleneck**: Traditional encoder-decoder models rely on a recursive neural network (RNN) that processes tokens (words or sequence elements) sequentially. This process imposes a bottleneck, as each token passes information forward in a sequential chain. By the end, much of the information from the beginning may have degraded or been lost.
2. **Fixed-Length Vector Constraint**: The information gathered in each sequence is ultimately condensed into a single, fixed-length vector. This limited vector struggles to retain detailed information in longer sequences, especially as the complexity of inputs and required outputs grows.
3. **Dependency on Sequential Processing**: Because RNNs rely on sequential data flow, each token must pass through the same network in order, making it challenging to capture long-term dependencies or complex contextual relationships effectively.

These limitations became pronounced in tasks like translation of lengthy sentences, image captioning, and, later, in protein structure prediction.

Enter the Transformer: "Attention Is All You Need"

To address these issues, the **Transformer** model introduced a fundamental shift, focusing on a new concept: **attention**. This approach enabled models to process tokens in parallel rather than sequentially, dramatically improving efficiency and performance. The paper "Attention Is All You Need" laid the foundation by proposing a model where attention would replace the recursive structure entirely. Here's how the Transformer model was constructed to overcome encoder-decoder limitations:

1. **Parallel Processing with Attention**: Instead of processing tokens sequentially, Transformers allow each token in the input sequence to be processed independently and in parallel. This enables the model to capture complex dependencies and relationships across the entire sequence simultaneously.
2. **Self-Attention Mechanism**: Self-attention enables each token to "attend to" or reference every other token in the sequence. This allows the model to dynamically weigh the importance of each part of the sequence relative to other parts, capturing context and dependencies without sequential processing.
3. **Multi-Head Attention**: To enrich the model's capacity to interpret different aspects of a sequence, the Transformer uses **multi-head attention**. This mechanism allows the model to examine multiple relationships between tokens simultaneously by learning multiple attention "heads." Each head focuses on different parts of the sequence, capturing varied types of information, such as word relationships or positional context in language translation, or sequence dependencies in protein modeling.
4. **Positional Encoding**: Since self-attention operates on tokens without considering order, **positional encodings** are introduced to give the model information about the sequence order. These encodings are vectors that represent the position of each token and are added to the input embeddings, helping the model understand the structure of the sequence.

The Transformer Architecture in Detail

The Transformer model is structured as an encoder-decoder model, with several important innovations:

1. **Encoder and Decoder Blocks**: The architecture consists of a stack of encoder and decoder blocks, each containing:
    - **Multi-Head Attention**: Each token attends to other tokens to understand relationships and dependencies.
    - **Feed-Forward Neural Network**: Each token's representation is then passed through a feed-forward network, adding nonlinear computation.
2. **Residual Connections and Layer Normalization**: To stabilize training and preserve information, **residual connections** are added, where the input to each block is summed with its output before moving to the next block. This approach helps retain information across multiple layers. Additionally, **layer normalization** is applied to stabilize learning and improve convergence.
3. **Autoregressive Decoding with Masking**: In tasks like language modeling, the Transformer uses an autoregressive process, where previously generated tokens are fed back as inputs to predict the next token. The model masks certain parts of the input to ensure it only attends to relevant context during generation.

Transforming Sequence Processing in Protein Language Models (PLMs)

The Transformer architecture revolutionized sequence modeling across fields, including natural language processing and biological sequence analysis:

1. **Parallelized Processing of Protein Sequences**: PLMs, like ESM and AlphaFold's later iterations, benefit from the Transformer's ability to process sequences in parallel. This capability allows the model to efficiently capture relationships among residues across the entire protein sequence, which is essential for accurate structure prediction and functional annotation.
2. **Enhanced Contextual Understanding**: Self-attention enables PLMs to capture long-range dependencies, crucial for understanding secondary and tertiary structure in proteins. By attending to each residue's relation to others, PLMs can capture complex spatial and functional relationships within protein sequences.
3. **Positional Encoding for Biological Sequences**: In PLMs, positional encoding informs the model of the residue order within the sequence. This is essential since the order of amino acids dictates a protein's structural and functional properties.

Moving Forward: The Transformer as a Foundation for Protein Modeling

The Transformer's architecture provides a powerful framework for capturing dependencies and contextual relationships in sequences of all types, making it particularly valuable for **protein structure prediction** and **functional annotation**. Its ability to handle complex dependencies and enable contextual learning marks a significant advancement for protein language models, which use attention to uncover the intricate "grammar" of protein sequences. This sets the stage for breakthroughs in protein design, drug discovery, and more specialized applications in biological sciences.

The next section will delve into the **mechanics of multi-head attention** and **how the model dynamically weighs different parts of the sequence**, furthering our understanding of how Transformers model interactions within sequences like protein chains.

49:20 Tokenization, Semantic Embedding, and Positional Embedding in Transformer Models

The Transformer model's functionality relies on three foundational steps for processing input sequences: **tokenization**, **semantic embedding**, and **positional embedding**. Here's how each step plays a role in transforming raw data (like language or protein sequences) into representations the model can use for deeper analysis.

1. Tokenization

Tokenization is the process of breaking down a sequence into discrete, manageable parts, often words or subwords in language models, or individual **amino acids in protein models**.

- **For language translation**: The model tokenizes input by segmenting sentences into words or subwords. This is essential to provide structure and units the model can process independently.
- **For protein sequences**: Tokenization is simpler, as the 20 amino acids act as natural tokens. Special

tokens, such as start-of-sequence and end-of-sequence markers, may also be included to define the boundaries of the sequence.

Each token is initially just an integer representing an element in the sequence. The next step is to transform these into more meaningful representations.

2. Semantic Embedding

Once tokens are identified, the model converts them into **semantic embeddings**—high-dimensional vectors that encode the properties and relationships between tokens.

- **Embedding vectors**: For language models, embeddings capture syntactic and semantic relationships between words. In protein language models, each amino acid is represented by a high-dimensional vector (for instance, a 1280-dimensional vector in ESM2). These embeddings aim to capture biochemical properties, such as hydrophobicity or charge, which influence protein behavior.
- **Learning from vast datasets**: Semantic embeddings are learned from extensive datasets. For ESM2, embeddings are trained on hundreds of millions of protein sequences, allowing the model to encode subtle and complex relationships that reflect the vast diversity of protein structures and functions.

The result of embedding is a large vector representation that encapsulates both the identity and, indirectly, the behavior or characteristics of each token. While this vectorized representation is powerful, it still lacks information on the order of tokens, which brings us to positional embeddings.

3. Positional Embedding

In Transformer models, **positional encoding** addresses the lack of inherent sequence information in embeddings, enabling the model to differentiate where each token lies within a sequence.

- **Why positional embedding is needed**: Unlike RNNs that process tokens sequentially, Transformers process tokens in parallel. This parallel processing makes Transformers faster but loses the sequential context that is essential, especially for sentences and protein chains where order impacts meaning and structure.
- **Sinusoidal encoding**: The original Transformer paper introduced a sinusoidal function-based positional encoding scheme. These encodings consist of sine and cosine functions with varying frequencies, creating a unique, continuous-valued vector for each position. This allows the model to calculate relative positions easily through mathematical operations.
    - **Visual interpretation**: Each position within a sequence has a distinctive encoding, but because of the sinusoidal patterns, similar positions (such as consecutive residues) yield similar encodings, supporting the model's recognition of local patterns.
    - **Information spread**: The model learns to differentiate the embedding vector space to store properties like hydrophobicity or charge on one side, while positional information, which guides sequence order, occupies another part of the vector. This ensures that both types of information coexist without interference.

Benefits of Combining Semantic and Positional Embedding

With both semantic and positional embeddings, Transformer models can operate on a rich, high-dimensional space where:

- **Semantically related tokens** (e.g., similar amino acids) have closer representations, and
- **Positional context** helps the model distinguish between different positions, facilitating the understanding of sequence order and structure.

In protein language models, these embeddings allow the Transformer to interpret amino acid sequences not just as a chain of residues but as context-aware units within a biological framework, encoding both **biophysical characteristics** and **sequence-specific order**. This dual awareness is essential for tasks like protein folding predictions, where both individual amino acid properties and their specific sequence order determine the final structure.

Overall Impact in Protein Models

Tokenization, semantic embedding, and positional embedding collectively equip Transformers with the nuanced understanding necessary for complex biological sequence analysis. This framework has revolutionized fields beyond language processing, enabling breakthrough applications in **protein structure prediction**, **drug design**, and **functional annotation** of proteins.

Next, we'll explore how **attention mechanisms** leverage these embeddings to allow tokens to "communicate," further enhancing the model's interpretative capabilities within complex sequences like proteins.

55:00 Information Flow in a Transformer and Normalization Techniques

In Transformer models, efficient processing and information sharing across sequence tokens (like words or amino acids) rely on **parallelized attention mechanisms** and **normalization** methods. Let's dive into these concepts to understand how they help optimize model performance, especially in handling complex sequences.

Parallel Processing and Information Sharing in Transformers

Unlike recurrent neural networks (RNNs), Transformers handle sequence data (e.g., a sentence or protein sequence) **in parallel**. Each token, whether a word or an amino acid, passes through the Transformer layers simultaneously, allowing the model to process data more efficiently.

- **Parallel Processing**: Each token in a sequence is fed through identical Transformer blocks at the same time. This enables the model to avoid the sequential processing limitations of RNNs, where tokens are processed one by one. This parallelism makes Transformers highly scalable and faster for training on large datasets.
- **Attention Mechanism**: Within each Transformer block, an **attention mechanism** allows tokens to share information with one another. Each token (or word/position in the sequence) can focus on others based on their relevance to the current token's context. This is visualized as each token connecting to every other token in the sequence.
    - **Weighted Connections**: Though every token can connect with every other token, the strength of these connections (or weights) varies, influenced by how significant one token is to another in context. This dynamic, context-sensitive communication enables the model to capture long-range dependencies in sequences.

Residual Connections and Normalization

Following the attention mechanism, the Transformer model incorporates **residual connections** and **normalization layers** to stabilize learning and improve convergence.

- **Residual Connections**: After each attention operation, the Transformer adds the original input (embedding vector) back to the output of the attention layer. This "skip connection" helps maintain the original input information even after complex transformations and aids in training deep networks by preventing the vanishing gradient problem. Residual connections ensure that even after multiple transformations, some of the initial information remains accessible.
- **Normalization Techniques**:
    - **Batch Normalization**: In traditional deep learning, batch normalization is used to scale and shift the outputs within a layer, normalizing them across an entire batch of inputs. This technique sets the mean of activations to zero and the variance to one for each feature, stabilizing training.
    - **Layer Normalization**: In Transformer models, **layer normalization** replaces batch normalization due to variations in sequence length, which can range widely (e.g., some protein sequences might be 100 amino acids long, while others might have 1,000). Layer normalization operates on each token individually, normalizing within the token's sequence instead of across a batch. For each layer of the neural network, it sets the mean of activations to zero and the standard deviation to one, but only within the current sequence.
        - **Why Layer Normalization?**: This approach is better suited for models handling sequences of varying lengths, ensuring stable learning across the entire sequence. It also keeps each sequence's structure intact without introducing inconsistencies due to batch-level normalization, which can vary with input size.

Importance of Normalization in Transformers

Normalization is essential in Transformer models because it:

- **Improves model convergence** by stabilizing activations at each layer, making learning more efficient.
- **Prevents gradient instability**, especially in very deep networks, allowing the model to maintain informative gradients through layers.
- **Enhances generalization** by reducing the model's sensitivity to input variations, enabling it to perform well across different types of sequences.

Together, **attention mechanisms, residual connections, and normalization** form a robust framework for processing complex, high-dimensional data like language and protein sequences. This allows Transformers to not only process vast data in parallel but also retain key features and relationships within sequences, essential for tasks like protein structure prediction, language translation, and beyond.

59:30 Attention mechanism

The **attention mechanism** in Transformer architectures serves as the foundation for modeling complex relationships between sequence elements, such as amino acids in a protein or words in a sentence. The attention mechanism dynamically determines which parts of a sequence are contextually significant for each other, a capability especially vital in language models and protein modeling.

Key Concepts of the Attention Mechanism

1. **Representation as Queries, Keys, and Values (Q, K, V):**
    - **Query**: Represents the current position or token (e.g., an amino acid or word) for which we seek relevant context.
    - **Key**: Represents potential context elements that might relate to the query. Keys are derived from other tokens within the sequence.
    - **Value**: The actual information content to be retrieved and incorporated if a connection between a query and a key is established.
2. Each input token (e.g., residue in a protein sequence) is transformed into queries, keys, and values by multiplying the original embedding by learned matrices specific to each type. This approach allows the model to focus on different aspects of the token (e.g., hydrophobicity or charge in proteins).
3. **Dot-Product Attention:**
    - Each query vector is compared with each key vector across the sequence using a **dot product**, measuring their alignment.
    - Tokens with high alignment values indicate a strong contextual relationship, prompting the model to pay more attention to these tokens.
    - This produces an **attention matrix** that records the relevance of every token pair in the sequence.
4. **Softmax Normalization:**
    - To ensure that the sum of attention weights for each query token equals one, the model applies **softmax normalization** to the dot products.
    - This process converts raw dot product scores into probabilities, indicating how much attention should be given to each token in the sequence relative to the query.
5. **Generating the Attention Output:**
    - For each query token, the attention output is computed as a weighted sum of values from all other tokens in the sequence, using the normalized attention weights.
    - These outputs allow tokens to dynamically incorporate context from relevant parts of the sequence, significantly enhancing the representation's expressiveness.
6. **Multi-Head Attention for Richer Contextual Understanding:**
    - Transformers use multiple attention "heads," each with unique Q, K, V matrices, to capture diverse relationships (e.g., hydrophobic-hydrophobic or positive-negative charge interactions in proteins).
    - **Concatenation**: Outputs from multiple attention heads are concatenated and transformed back to the original input size, preserving the enhanced, multi-dimensional context from each head.
7. **Residual Connection and Normalization:**
    - The output of the attention layer is added back to the initial embedding via a **residual connection**, which helps maintain original input features while also incorporating learned contextual information.
    - **Normalization**: To stabilize training, the residual-connected outputs are layer-normalized, ensuring the distribution of outputs remains consistent across layers.

8. **Scaled Dot-Product Attention**:
    - In practice, dot-product scores are divided by the square root of the key dimensionality to prevent overly large values, ensuring smoother gradients and more stable learning.

Attention Mechanism in Action for Transformers

Each attention head captures a unique type of relationship, and the combined output from all heads gives a rich, multi-faceted view of the token's context. For tasks like protein structure prediction, this allows the model to identify biologically relevant interactions within a sequence, supporting highly accurate predictions of protein folding and interactions.

By enabling each token to dynamically draw context from any other token, the attention mechanism is crucial for the remarkable performance of Transformers in tasks requiring nuanced contextual understanding, such as language translation and protein language modeling.

1:13:50 Residual connections

Residual connections (also known as skip connections) are a fundamental mechanism in deep learning, especially in deep architectures like Transformers and ResNets. They serve to stabilize training, prevent issues with vanishing gradients, and allow information from earlier layers to persist through the network.

Purpose of Residual Connections

1. **Information Preservation:**
    - By directly adding the input of a layer to its output, residual connections help retain some of the original information throughout the layers, preventing the network from deviating too far from the initial signal.
2. **Mitigating Vanishing Gradients:**
    - In very deep networks, gradients can diminish (or "vanish") as they propagate backward through many layers, making training difficult. Residual connections allow gradients to flow directly through the network, improving the stability of training and the network's ability to learn.
3. **Easier Training in Deep Networks:**
    - Residual connections enable networks to scale deeper, as they effectively allow each layer to learn adjustments to the previous layers rather than starting from scratch. This capacity for depth was demonstrated in residual networks (ResNets), where the error rate continued to improve even with over 100 layers, whereas traditional architectures would worsen as depth increased.

How Residual Connections Work

In practice, a residual connection simply adds the input vector of a layer to its transformed output vector. This process can be represented as:

Output=Layer(Input)+Input\text{Output} = \text{Layer}(\text{Input}) + \text{Input}Output=Layer(Input)+Input

In Transformers, residual connections are applied after key operations, like the **self-attention mechanism** and **feed-forward networks**, to retain the initial input information at each step.

Example in Transformers

1. **Self-Attention Layer:**
    - After computing the attention scores and weighting the values, the result is added back to the original input vector before passing through layer normalization. This maintains both the initial embedding information and any context-based modifications made by the attention layer.
2. **Feed-Forward Network Layer:**
    - Similarly, after passing through the feed-forward network, the transformed vector is added back to the residual input, allowing the network to build more complex representations over many layers without losing the underlying data.

Impact on Performance

The addition of residual connections in deep networks has been a game-changer:

- **Scalability**: Residual connections allow architectures to scale with hundreds or even thousands of layers without degradation in performance.
- **Improved Accuracy**: As shown in the ResNet paper, deep networks with residual connections

outperform shallower ones, allowing them to achieve lower error rates on tasks such as image recognition.

Overall, residual connections have become a ubiquitous feature in deep learning architectures, enhancing both the depth and performance of models by preserving and propagating information efficiently.

1:17:18 Comparison with other models

The structure of Transformers, with their encoder-decoder configuration, offers advantages over other deep learning models, like recurrent neural networks (RNNs) and convolutional neural networks (CNNs), especially in tasks involving sequential or contextual understanding, such as language processing or protein sequence analysis.

Here's how Transformers and their variants (like encoder-only, decoder-only, and encoder-decoder models) compare to other architectures:

1. **Recurrent Neural Networks (RNNs)**:
    - **Pros of RNNs**: They handle sequence data by processing each element in order, making them effective for tasks where sequence information is crucial, like time-series prediction and language translation.
    - **Cons**: RNNs struggle with long sequences due to issues like vanishing gradients, limiting their effectiveness in capturing long-range dependencies. Training is sequential and computationally intensive, making them slower for large-scale models.
    - **Transformers' Advantage**: Transformers use attention mechanisms to focus on important parts of the sequence, making it possible to process all tokens in parallel rather than sequentially. This allows for more efficient training, captures long-range dependencies better, and provides superior performance in tasks with lengthy or complex sequences.
2. **Convolutional Neural Networks (CNNs)**:
    - **Pros of CNNs**: CNNs excel in tasks involving spatially structured data, like image recognition, due to their ability to capture local patterns through convolutions.
    - **Cons**: Although they can be adapted for sequential data, CNNs aren't inherently designed to handle sequence dependencies and often require complex configurations (such as dilated convolutions) to manage long-range dependencies.
    - **Transformers' Advantage**: Transformers are inherently suited for capturing relationships across any two points in the sequence, regardless of distance. For tasks like protein structure prediction, which require understanding long-range dependencies in amino acid sequences, Transformers provide more direct and comprehensive relational modeling than CNNs.
3. **Encoder-Only Models**:
    - **Example**: The ESM protein language model you'll work with.
    - **Function**: Encoder-only models focus solely on transforming the input data into meaningful embeddings without generating output sequences. They are primarily used for tasks requiring feature extraction, like classification or understanding.
    - **Use Case**: Ideal for protein structure and function prediction, where understanding the protein sequence's embedded features is crucial.
4. **Decoder-Only Models**:
    - **Example**: ChatGPT.
    - **Function**: Decoder-only models are designed to generate sequences and are typically used in language generation tasks.
    - **Use Case**: These models are used when the task involves generating coherent sequences, such as text generation, where the model relies on its prior context without an explicit encoded source sequence.
5. **Encoder-Decoder Models**:
    - **Example**: Google Translate.
    - **Function**: Encoder-decoder models are structured to take an input sequence (such as a

sentence in one language), process it with an encoder, and then use a decoder to generate a corresponding output sequence (such as the translated sentence).
    - **Use Case**: Suitable for machine translation and other tasks that require an output sequence generated based on an input sequence.

Workflow in Transformers

The encoder-decoder model in Transformers leverages self-attention for intra-sequence relationships and cross-attention to incorporate information from the encoder into the decoder.

- **Encoder Process**: The input sequence is tokenized, encoded with positional embeddings, and processed through layers of multi-head attention, feed-forward networks, and residual connections to generate a rich sequence representation.
- **Decoder Process**: The decoder uses self-attention to process its own sequence and cross-attention to refer back to the encoder's output, allowing it to incorporate contextual information from the input sequence.

The final output layer, with a softmax function, generates probabilities for the output tokens, enabling applications like translation, language generation, or protein sequence modeling.

Summary

- **ESM (Encoder-Only)**: Focuses on understanding and encoding sequence data, useful for tasks like protein function prediction.
- **ChatGPT (Decoder-Only)**: Generates language sequences without requiring explicit input context, suitable for conversational AI.
- **Google Translate (Encoder-Decoder)**: Processes input and output sequences for translation, allowing for bidirectional sequence modeling.

This overview demonstrates how Transformers, in various configurations, provide flexibility and power for a wide array of AI applications, from NLP to structural biology.

Lecture 11 - Protein Language Models

Video: Lecture11 Protein Language Models

Slides: Lecture11_ProteinLanguageModels_PLMs.pdf

0:00 Parallels of NLP and Protein language models

As the field of **AI and machine learning advances**, it's becoming clear that natural language processing (NLP) models—such as those used in **chatbots, sentiment analysis, and language translation**—share significant parallels with **protein language models**. These models, leveraging deep learning, open new avenues in **protein structure prediction, interaction analysis, mutation effect assessment, and de novo protein design**. Let's dive into the core parallels between natural languages and proteins, explore available data sources, and examine key applications.

1. Parallels Between Natural Languages and Protein Sequences

Natural language models process human languages by analyzing patterns in texts, translating, predicting sentiment, and generating coherent sequences. Similarly, **protein language models** process protein sequences to predict biological functions and structures.

- **Alphabet and Building Blocks**:
    - **Natural Language**: English, for instance, has a 26-letter alphabet, which combines to form **words** and **sentences**.
    - **Protein Sequences**: Proteins consist of sequences of **20 amino acids** (e.g., alanine, glycine, etc.), which form chains with specific biological functions. DNA and RNA sequences, on the other hand, use **four nucleotides** (A, T, C, G for DNA and A, U, C, G for RNA).
- **Tokenization**:
    - In NLP, tokenization often involves grouping letters into **words**.
    - In protein modeling, tokenization generally occurs at the level of **individual amino acids** rather than groups. However, we could explore "k-mers"—sets of three or more consecutive amino acids—as a token unit to capture context similarly to words in natural languages.

- **Training Data Sources**:
    - **Language models** draw on massive text corpora like Wikipedia, Reddit, and even **code repositories** like GitHub.
    - **Protein models** use data from specialized repositories:
        - **NCBI Sequence Archives**: Comprehensive repositories where experimental and in silico sequences are deposited.
        - **GISAID**: A resource focused on viral evolution and surveillance, crucial for tracking emerging variants, especially for viruses like **SARS-CoV-2**.
        - **Protein Data Bank (PDB)**: Contains **3D protein structures** derived from crystallography and cryo-EM, offering spatial information alongside sequences.
        - **Specific Databases**: For example, **SabDab**, an antibody structure database, supports models specialized in **antibody prediction and design**.

2. Key Applications of Protein Language Models

a) Protein Structure Determination. With the recent breakthroughs in **AlphaFold** and similar models, AI has begun to predict protein structures from sequence alone. Protein language models analyze evolutionary patterns within sequences—especially by **examining co-evolution of amino acids** within proteins. If mutations occur in pairs of amino acids that are close in 3D space, compensatory changes can indicate **spatial contacts in protein folding**. By training on these **coevolutionary constraints**, models can predict 3D structures accurately.

b) Protein-Protein Interaction Prediction. Moving beyond single proteins, protein language models can also **predict interactions between proteins**. By analyzing pairs of related protein sequences (e.g., protein A and B across species), models can identify **coevolutionary sites** where changes in one protein are often paired with changes in the other. These coevolved sites likely reflect direct physical interactions, which are critical for predicting **contact sites** and understanding **molecular interactions** in pathways and complexes.

c) Mutation Effect Prediction. Models such as **EVE** (Evolutionary Model of Variant Effects) predict how single-point mutations affect protein function. These predictions leverage **natural evolutionary sequences** and are particularly valuable in **disease variant prediction**:

- By examining evolutionary conservation, models predict the effects of rare mutations (e.g., potential to cause disease) based on whether a mutation deviates from natural variation.
- Mutation impact analysis has potential clinical applications, enabling **predictive diagnostics** for genetic disorders.

d) De Novo Protein Design. De novo design aims to engineer entirely new proteins for specific purposes—a critical field in **synthetic biology** and **therapeutics**. David Baker's lab, recently honored with a Nobel Prize, is known for pioneering this work:

- Language models assist in designing novel proteins that can act as **receptors, inhibitors, or binders**.
- For example, a model can design a binder protein for **influenza hemagglutinin** to prevent infection by blocking viral entry into cells.

3. Understanding and Using Protein Language Models

Protein language models learn from sequence data to perform tasks analogous to those in NLP, such as **sequence generation**, **classification**, and **translation** (sequence to structure, for instance).

- **Embedding and Representation Learning**:
    - Just as NLP models map words to embeddings, protein models map amino acids and sequences to a high-dimensional space that reflects their biological properties and relationships.
    - These embeddings enable **downstream applications**, from structure prediction to interaction analysis, by capturing the **latent biochemical and structural properties** of proteins.
- **Transfer Learning and Fine-Tuning**:
    - Protein language models, like NLP models, benefit from **pre-training** on massive datasets, then fine-tuning on specific tasks.
    - For example, a model could be pre-trained on general protein sequences and later adapted to focus specifically on **antibody design** or **enzyme functionality**.

4. Toward the Future: Expanding Protein Language Models

The current advances in protein language modeling hint at an exciting future, where these models may perform increasingly complex biological tasks, extending beyond structure prediction into realms like:

- **Drug discovery** by predicting not only binding sites but also the **specific effects of small molecules on protein conformations**.
- **Synthetic biology** applications where models predict the design and function of **entire biochemical pathways** involving multiple interacting proteins.
- **Therapeutic development** with models that can predict and design **peptide therapeutics or synthetic antibodies** to treat infectious diseases, autoimmune conditions, and more.

By understanding the principles of **language models in proteins**, researchers and practitioners in bioinformatics and molecular biology are well-equipped to leverage these tools, opening new possibilities for **precision medicine, drug design, and synthetic biology**.

9:02 The Protein Alphabet: Structure, Similarity, and Functional Constraints

Protein language models are built on the **20 standard amino acids**, each of which brings a unique set of **biochemical properties** that influence protein folding, stability, and function. Unlike natural language, where words might have nuanced but easily categorized similarities, amino acids exhibit complex biochemical similarities that affect how they can substitute for each other in proteins. Let's explore the protein alphabet, how amino acids interact, and how **evolutionary constraints** inform functional sequences.

1. Amino Acid Representation in Models

Historically, amino acids in sequence data have been represented using **one-hot encoding**, where each amino acid is represented by a vector of length 20. Each position in this vector corresponds to one of the 20 amino acids, and a "1" in a specific position denotes that amino acid, with the other positions set to "0." However, this approach does not account for **similarities and dissimilarities** between amino acids.

To improve on this, more sophisticated representations in protein language models consider the **biochemical properties** of amino acids:

- **Charge**: Amino acids can be positively charged, negatively charged, or neutral.
- **Polarity**: Some amino acids are **polar** (water-soluble) while others are nonpolar, affecting their placement within protein structures.
- **Hydrophobicity**: Hydrophobic (water-repellent) amino acids tend to be buried inside protein cores, while hydrophilic (water-attracting) ones are found on surfaces.
- **Size and Shape**: The physical bulk of amino acids, such as **aromatic** or **greasy** groups, affects how closely they can pack and interact.

Protein language models can either incorporate these features explicitly or **learn them implicitly** through large-scale training on sequence data. This approach allows the model to recognize, for instance, that **lysine and arginine** (both positively charged) are more similar to each other than to **serine** (a polar but uncharged amino acid).

2. Evolutionary Similarity and the BLOSUM Matrix

Evolutionary data offers insight into how amino acids can substitute for one another in a functional protein. In biological terms, amino acids that frequently replace one another without disrupting protein function are likely to share critical **structural or biochemical characteristics**. This concept is captured in **substitution matrices** like the BLOSUM (BLOcks SUbstitution Matrix), which summarizes evolutionary substitutability.

- **BLOSUM Matrices**: These matrices indicate how often one amino acid is replaced by another across evolutionary lineages. If an amino acid is frequently substituted by another in nature, it suggests they can fulfill similar roles within a protein's structure, even if they differ in finer details.
- **Evolutionary Substitutability**: Amino acids that can substitute without loss of function indicate **similarity in function or structure**. For instance, **isoleucine** and **leucine** can often substitute for each other due to similar hydrophobic properties, suggesting they play interchangeable roles in many contexts.

In practice, models that use BLOSUM-derived features can gain insight into **likely and unlikely substitutions** in proteins, allowing predictions of which mutations are functionally neutral or harmful.

3. Protein Backbone and Side Chain Interactions

A protein's three-dimensional structure is largely determined by its **backbone** and **side chains**:

- **Backbone**: This is the continuous chain of repeating units (N-C-C) that forms the primary scaffold of the protein.
- **Side Chains**: Each amino acid has a unique side chain that projects from the backbone, determining its chemical properties.

The **spatial interactions** between side chains are essential for maintaining the structural stability and functional configuration of a protein:

- **Bonding and Clashes**: Amino acids must fit spatially without causing clashes. If two bulky amino acids are next to each other, they may interfere with one another, disrupting the protein's stability.
- **Size Compatibility**: Amino acids of similar sizes and properties can often substitute without disrupting bonds or introducing instability. However, if smaller amino acids are replaced by larger ones (or vice versa), this can create gaps or overcrowding that destabilizes the protein.

For example, in a protein that relies on hydrogen bonds or ionic interactions, **substituting an amino acid with a different charge** may disrupt these stabilizing interactions. Thus, models trained on sequence data can learn that amino acids within certain environments cannot be freely substituted without compromising function.

4. Learning Functional Constraints through Evolution

Evolution has naturally selected for **functional sequences** that fulfill specific roles, filtering out many sequences that would be non-functional. Thus, evolutionary data acts as a guide for predicting which mutations can maintain protein function:

- **Sampling of Functional Space**: Evolutionary sequences serve as samples of the functional space that proteins can occupy. While not all functional sequences are observed, the ones that are can be assumed to be viable and stable under natural conditions.
- **Inference of Constraints**: By examining a dataset of **evolutionarily conserved sequences**, a protein language model can infer constraints that limit which amino acids can appear in specific contexts. This inference helps the model predict if a novel mutation will likely retain function or if it might destabilize the protein.

By leveraging evolutionary data, models can predict which **amino acid substitutions** would disrupt structural or functional integrity, as these changes have rarely or never been observed in natural evolution. For example, if a particular position in a protein sequence has consistently been a **hydrophobic residue** across species, changing it to a hydrophilic one may result in loss of function.

5. Integrating Biochemical and Evolutionary Insights

When designing protein language models, we can integrate both **biochemical properties** and **evolutionary substitutability** to create robust and nuanced embeddings:

- **Biochemical Property-Based Embeddings**: Models can include explicit biochemical properties, such as charge, size, and hydrophobicity, to improve representation quality.
- **Evolutionary-Based Embeddings**: Evolutionarily derived matrices like BLOSUM add another layer, guiding models in identifying amino acids that can function interchangeably in specific protein contexts.

Combining these approaches allows the model to:

1. Learn general patterns about **amino acid similarities** from biochemical principles.
2. Refine predictions based on **functional constraints** observed in nature, filtering out substitutions that would destabilize or alter protein function.

Conclusion

The **protein alphabet** is more than just a sequence of amino acids; it is a complex system shaped by biochemical properties and evolutionary pressures. By incorporating these aspects, protein language models are capable of not only understanding protein sequences but also making sophisticated predictions about their structure, interactions, and potential responses to mutations. This foundational understanding empowers further advances in **protein engineering, drug design, and therapeutic discovery**, as AI learns to decode the rich language of proteins.

13:49 Multiple Sequence Alignments: Capturing Functional and Evolutionary Constraints

Multiple sequence alignments (MSAs) are a powerful tool for understanding functional protein sequences. They allow us to study evolutionary relationships between sequences, infer structural and functional

constraints, and predict the potential impact of mutations. An MSA takes one protein and aligns it with evolutionarily related proteins, aligning similar or conserved positions across species or strains. This alignment provides insights into which positions are conserved due to **functional or structural importance** and which are more variable, indicating **less evolutionary constraint**.

1. Structure of an MSA

In a typical MSA, each column represents a specific position in the protein sequence, and each row is a sequence from a related protein:

- **Conserved Sites**: Columns with similar or identical amino acids across sequences indicate **high conservation**, often due to essential roles in protein structure or function. For instance, a column dominated by **A (alanine)** or **T (threonine)** suggests these amino acids are critical for maintaining the protein's integrity or function.
- **Variable Sites**: Columns with diverse amino acids, such as Q (glutamine), H (histidine), E (glutamate), and N (asparagine), imply lower functional or structural constraint, allowing for **greater sequence flexibility**. Variability at these sites suggests that mutations here might not significantly impact the protein's functionality.

The evolutionary **patterns** observed in MSAs reveal the sequence features necessary for function and help in **modeling evolutionary pressures** across sequences.

2. Modeling Approaches for MSAs

There are several ways to model MSAs to learn about protein function and stability, with each approach offering different insights into amino acid interactions:

Site-Independent Models. A **site-independent model** evaluates each position in the alignment independently, ignoring context from other sites. This approach calculates the **position-specific amino acid frequencies** by tallying occurrences of each amino acid at each site in the alignment.

Steps in Site-Independent Modeling:

- For each position i in the alignment, calculate the **frequency** of each amino acid at that position. For example, if position iii has a high frequency of R (arginine), it may indicate a preference for positively charged residues at that site.
- To **score a new sequence**, calculate the product of probabilities for each amino acid at its respective position. This product is often normalized by a factor Z to ensure it represents a probability distribution.

In equation form:

$$P(\text{sequence}) = \frac{1}{Z} \prod_{i=1}^{N} P(a_i)$$

where $P(a_i)$ is the probability of observing amino acid aaa at position i.

By taking the **log of the probability**, one can derive a **log-likelihood score** for the sequence, which is useful for comparing sequences.

Applications:

- **Phylogenetic Inference**: Site-independent models are often sufficient for reconstructing evolutionary trees, where each position is treated independently.
- **Substitution Rate Estimation**: These models can estimate the **rate of substitution** for each position in a protein, helping to identify conserved and variable sites.
- **Mutation Effect Prediction**: Site-independent models can predict whether a mutation at a particular position is likely to be harmful or neutral, based solely on the observed frequency of amino acids at that position. However, these predictions may lack precision, as they ignore **contextual dependencies** between sites.

3. Limitations of Site-Independent Models

Site-independent models, while useful, fall short in capturing **interdependencies between sites**:

- **Context Ignorance**: These models fail to account for co-evolving positions, where a change in one amino acid might necessitate a compensatory change in another to maintain function.
- **Structural Oversimplification**: Protein structure is complex, with three-dimensional relationships

between residues. Ignoring interactions between sites overlooks important structural and functional constraints.

For instance, two adjacent amino acids might form a critical bond, and changes at one site might require compensatory changes at the other. Site-independent models do not capture such interdependencies, leading to **less accurate predictions** for mutation effects.

4. Beyond Site-Independent Models: Pairwise and Higher-Order Interactions

To improve upon the limitations of site-independent models, **pairwise interactions** and **higher-order interactions** can be incorporated:

Pairwise Interaction Models:

- These models consider **correlations** between pairs of sites, capturing co-evolutionary relationships. For example, if position iii often has a positively charged residue and position jjj often has a negatively charged residue, this might indicate an electrostatic interaction critical for the protein's function.
- Such correlations can be quantified using **covariance or mutual information** metrics, providing a **richer representation** of functional constraints.

Higher-Order Interaction Models:

- These models extend beyond pairs, capturing **multi-residue dependencies** within the protein sequence. Although more computationally intensive, they provide a more nuanced understanding of complex **inter-residue networks** that contribute to protein stability and function.

By capturing dependencies between residues, these models enable a **more comprehensive view** of the constraints shaping protein evolution and function.

5. MSA in Protein Language Models

Protein language models leverage MSAs to learn which mutations are permissible and which might disrupt function. They analyze **patterns of conservation and co-evolution** to infer constraints that have been shaped by millions of years of evolutionary selection. This is especially valuable for tasks like:

- **Predicting Disease Variants**: By identifying mutations that are rare or absent in natural sequences, models can flag variants likely to disrupt function.
- **Understanding Functional Domains**: MSAs reveal regions where specific amino acids are essential, indicating functionally important domains.
- **Guiding Protein Design**: For designing novel proteins, these models help ensure that engineered sequences respect evolutionary constraints, improving the likelihood of achieving desired functions.

Conclusion

Multiple Sequence Alignments (MSAs) provide a fundamental dataset for understanding the evolution and function of proteins. By examining conservation and variability across sequences, MSAs highlight the **functional constraints** governing proteins. Site-independent models, while useful for certain tasks, are limited by their inability to capture dependencies between residues. More sophisticated models, incorporating **pairwise or higher-order interactions**, enhance our understanding of **structural and functional constraints** in proteins. Through MSAs and advanced modeling, protein language models can predict the effects of mutations, guide protein engineering, and deepen our understanding of protein evolution and function. This lays the groundwork for applications ranging from **disease variant prediction** to **de novo protein design**, with transformative implications for biotechnology and medicine.

17:14 Mutation effect prediction: Understanding Functional Impact of Genetic Variants

Mutation effect prediction aims to assess how genetic changes (mutations) impact the function and fitness of proteins. Changes in the protein's **genotype** (amino acid sequence) can affect various **molecular phenotypes** such as stability, optimal working temperature, binding capabilities, and overall function. These phenotypic changes are critical for understanding how mutations influence the organism's **fitness** and can help predict whether a mutation might be **disease-causing**. This is especially useful in medical genetics, where understanding the functional implications of mutations can provide insights into potential health risks.

1. Challenges in Mutation Effect Prediction

Mutation effect prediction is inherently difficult due to several factors:

- **Epistasis**: This concept describes how the effect of a mutation at one site depends on the presence of other mutations. For example, a single mutation might have a different effect if another mutation is also

present. This **context-dependent behavior** means that mutations cannot be evaluated in isolation, as their impact may vary significantly depending on interactions with other residues.

- **Complexity of Experimental Validation**: Experimentally testing the effect of every possible mutation is time-consuming and expensive. The vast sequence space makes it impractical to assess each mutation experimentally, so computational methods play a vital role in prioritizing mutations for further study.

Analogy for Epistasis: A useful analogy is in naming conventions. Suppose "Muhammad" is the most common first name globally, "James" is the most common middle name, and "Wong" is the most common last name. However, combining these into "Muhammad James Wong" does not result in the most common full name. This illustrates that **context matters**, as certain combinations are more likely to occur than others. Similarly, in proteins, simply combining the most common amino acids at individual sites does not guarantee a functional protein; the sequence context is crucial for function.

2. Moving Beyond Site-Independent Models with Pairwise Interactions

To capture the context in which mutations occur, we can incorporate **pairwise interactions** between residues, going beyond the limitations of site-independent models.

Pairwise Interaction Models. In a **pairwise interaction model**, we consider not only the frequency of each amino acid at individual sites but also **joint frequencies of pairs of amino acids** at specific positions. This allows the model to account for **co-evolutionary dependencies**, where the presence of one amino acid at a given site may necessitate a specific amino acid at another site to maintain protein function.

For instance:

- **Two amino acids that frequently co-occur** at certain positions may indicate a structural or functional relationship. If one of these residues changes, the other may need to adapt to maintain interactions, such as hydrogen bonds or hydrophobic packing, crucial for the protein's stability and function.

Example: In a protein sequence, a larger amino acid at one position might require a smaller neighboring amino acid to prevent steric clashes, while two similarly sized residues at adjacent positions could destabilize the structure.

Mathematical Representation: A pairwise model can score a sequence by combining:

1. **Position-specific frequencies** of amino acids at individual sites.
2. **Pairwise frequencies** for pairs of positions, representing how likely two specific amino acids are to appear together in specific positions.

In formula terms:

$$P(\text{sequence}) = \frac{1}{Z} \prod_{i=1}^{N} P(a_i) \prod_{(i,j)} P(a_i, a_j)$$

where $P(a_i)$) is the probability of amino acid aia_iai at position i, and $P(a_i,a_j)$ is the joint probability of observing amino acids a_i and a_j at positions i and j, respectively. This approach accounts for the interaction between residues, providing a more realistic picture of the protein's functional constraints.

Applications:

- **Structure Prediction**: Highly correlated sites are often in close proximity in the 3D structure, allowing pairwise models to infer **structural contacts** critical for folding and stability.
- **Protein-Protein Interactions**: Pairwise interactions can also predict which residues are involved in binding interfaces between two proteins, as co-evolving residues between two proteins are likely to interact.
- **Enhanced Mutation Effect Prediction**: By considering interdependencies, pairwise models improve the accuracy of mutation effect predictions, especially in cases where two or more residues contribute collectively to a specific function.

3. Limitations of Pairwise Models and the Need for Higher-Order Interactions

While pairwise models capture some dependencies, they are often insufficient for fully understanding the

complexity of protein behavior:

- **Higher-Order Interactions**: Some functional dependencies in proteins involve interactions among three or more residues, which pairwise models cannot capture. For example, a mutation in one residue might only be tolerable if compensatory changes occur at multiple other positions.
- **Complex Epistasis**: Proteins exhibit complex epistatic interactions beyond pairwise relationships. Certain structural or functional constraints may involve intricate networks of residues working in concert, making pairwise approximations overly simplistic.

4. Protein Language Models for Complex Mutation Prediction

To address the limitations of pairwise models, **protein language models** can capture higher-order interactions without requiring explicit multiple sequence alignments (MSAs). These models, which include advanced architectures such as **transformers** and **autoregressive models**, learn complex patterns within protein sequences:

- **Contextual Embeddings**: Protein language models create **contextualized representations** of each residue by considering the entire sequence, capturing dependencies across distant residues.
- **Implicit Learning of Constraints**: By training on large protein datasets, these models implicitly learn the functional and evolutionary constraints without explicitly modeling each pairwise or higher-order interaction.

Benefits:

- **Higher Accuracy in Mutation Prediction**: Protein language models can provide **more accurate predictions of mutation effects** by capturing nuanced interdependencies across multiple sites.
- **Flexibility**: These models do not require predefined alignments, making them suitable for analyzing diverse protein families or engineered proteins with no natural homologs.

Through complex representations, protein language models are advancing our understanding of protein function and mutation effects, contributing to **precision medicine** and **rational protein engineering**. By identifying mutations that may cause disease or enhance protein stability, these models offer powerful tools for biomedical research and therapeutic development.

Summary

Mutation effect prediction is central to understanding protein functionality and the impact of genetic variation. While site-independent models provide a baseline, **pairwise interaction models** capture co-evolving residues, revealing critical interdependencies in protein structure and function. However, to fully capture the complexity of protein behavior, **higher-order models and protein language models** offer advanced solutions by learning from the entire sequence context, addressing the challenges posed by epistasis and context-specific mutations. These sophisticated approaches are crucial for tasks ranging from **predicting disease-causing variants** to designing **novel proteins with specific properties**. Through these models, we continue to unlock the potential of **computational biology** and **genomic medicine** in addressing real-world biological and clinical challenges.

23:22 Protein Language Models Beyond Multiple Sequence Alignments

Protein language models are revolutionizing the field of protein modeling by moving beyond the limitations of **multiple sequence alignments (MSAs)**. Unlike MSAs, which require aligning evolutionary related sequences into a fixed structure, protein language models process proteins as sequences without needing alignment. This is akin to how natural language models, such as ChatGPT, are trained—by analyzing a vast body of text rather than aligning sentences or words in specific patterns.

1. Challenges with Multiple Sequence Alignments (MSAs)

While MSAs have been a powerful tool, they come with notable limitations:

- **Handling Variable Lengths**: MSAs require sequences to be aligned to a fixed length, often introducing **gap characters** to handle insertions and deletions. This can distort alignment and introduce artifacts, especially for sequences of varying lengths.
- **Limited Flexibility with Novel Sequences**: Models trained on fixed-length alignments struggle with longer or unique sequences not present in the training data, often requiring retraining or sequence trimming to make predictions.
- **Quality Variability**: Some alignments are inherently poor due to evolutionary distance or structural variability among proteins, making them less reliable as training data.

- **Difficulty with Certain Protein Families**: Certain proteins, like antibodies, present a unique challenge. Antibodies have a **highly variable complementary determining region (CDR3)**, essential for their binding flexibility. This variability makes alignment difficult and can reduce the effectiveness of MSA-based approaches.

2. Advantages of Protein Language Models

Protein language models bypass these constraints by training directly on **unstructured protein sequences** without the need for alignment. This shift offers several benefits:

- **Broader Applicability Across Protein Families**: Protein language models can be trained on diverse sets of proteins, regardless of evolutionary relationships or structural similarities. This allows them to generalize across unrelated proteins with varying lengths and characteristics.
- **Unified Model Across Protein Families**: Rather than building separate models for each protein family, a single protein language model can capture patterns and structures relevant across the protein universe, learning from broader biological diversity.
- **Alignment-Free Training**: By eliminating the need for alignment, these models can learn directly from raw sequences, making them suitable for proteins like antibodies, which exhibit high variability and poor alignment properties in traditional MSAs.

3. Recent Advances in Protein Language Models

In recent years, there has been significant progress in developing large-scale language models for proteins. Notable examples include **ProGen** and **ESM (Evolutionary Scale Modeling)**, which have demonstrated success in capturing structural and functional information directly from sequence data. These models are trained using **self-supervised learning**, meaning they rely solely on the sequences themselves without external labels or experimental measurements of function. By learning from the natural sequences found in protein databases, these models can infer biologically relevant features without needing annotations.

4. Training Methods in Protein Language Models

There are two primary methods for training protein language models: **autoregressive** and **masked modeling**.

A. Autoregressive Modeling. In the **autoregressive approach**, the model is trained to predict the next amino acid in a sequence based on preceding residues. This is similar to how natural language models fill in missing words within a sentence. For example, in natural language, given the input "The brown fox jumped over the ____," the model predicts the most likely word ("fence" or "log") to complete the sentence. In the protein context, given a sequence of amino acids, the model tries to predict the next amino acid based on the current sequence, gradually building an understanding of sequence patterns and functional domains.

- **Advantages**: Autoregressive models excel in tasks where sequential relationships matter, making them well-suited for applications like protein sequence completion or de novo sequence generation.
- **Limitations**: Since predictions depend on preceding amino acids, autoregressive models might struggle with long-range dependencies and context beyond the local sequence window.

B. Masked Language Modeling. The **masked modeling approach** works by presenting the model with a complete sequence, masking certain amino acids, and then asking it to predict the identity of the masked residues. This strategy is akin to **BERT (Bidirectional Encoder Representations from Transformers)** in NLP, where random words are masked within sentences, and the model learns to reconstruct them.

- **Advantages**: Masked modeling enables the model to learn from context on both sides of a masked residue, capturing long-range dependencies and structural context within the protein.
- **Limitations**: While masked models capture sequence-wide relationships, they may be computationally intensive, especially for large protein sequences, due to their bidirectional architecture.

Both approaches have unique strengths and are frequently used in combination to leverage both local and global contextual information.

Summary

Protein language models are opening new avenues in **computational biology** by enabling sophisticated predictions of structure, function, and interactions from raw sequence data. By moving beyond MSAs, these models overcome alignment limitations and provide a unified framework for understanding diverse protein families. The two main training approaches—**autoregressive** and **masked modeling**—offer complementary advantages in learning sequence dependencies and context. These advancements have significant implications for **protein engineering**, **drug discovery**, and **precision medicine**, empowering researchers to

explore uncharted territories in protein science and synthetic biology.

28:12 Training Masked Language Models for Protein Prediction

Masked modeling is a pre-training architecture that enables protein language models to capture the intricate relationships between amino acids within a sequence by learning to predict masked elements. This technique originates from **BERT (Bidirectional Encoder Representations from Transformers)**, a natural language model known for its success in various NLP tasks. For proteins, masked modeling allows the model to learn from large, unlabeled protein databases by predicting missing or masked amino acids within sequences, thereby capturing meaningful structural and functional patterns.

1. Overview of Masked Language Modeling (MLM)

In the **masked modeling approach**:

- **Randomly Masked Tokens**: For training, around **15% of tokens** (in NLP, these are words; in protein models, they are amino acids) are masked. The model must then predict these masked tokens based on the context of the remaining sequence.
- **Prediction Mechanism**: The model provides a likelihood score for each possible token (amino acid) that could fill the masked position. The token with the highest probability is then chosen as the predicted amino acid.

For protein models, this approach allows the model to learn contextual relationships between amino acids and implicitly understand properties such as **secondary structure, stability, and interaction sites**.

2. ESM Model Training with Masked Language Modeling

A prominent example of masked language modeling applied to proteins is **ESM (Evolutionary Scale Modeling)**, developed by Meta:

- **Input Protein Sequence**: The model is trained by masking certain amino acids in protein sequences.
- **Embedding Layer**: Instead of a one-hot encoding for each amino acid, ESM employs a **learned embedding** for each amino acid, represented as a vector of 1,280 values. Each amino acid (such as methionine (M) or lysine (K)) is embedded into a unique vector, and these embeddings evolve through training to enhance prediction accuracy.
- **Learning Contextual Properties**: Through training, the embeddings naturally cluster amino acids based on properties (such as hydrophobicity or charge), despite not being explicitly trained on these biochemical characteristics.

This approach helps the model discern amino acid properties that contribute to a protein's functional and structural characteristics.

3. Training Data: UniRef and Clustering for Effective Model Performance

Protein language models require large databases to effectively learn biological patterns. One of the primary datasets for training these models is **UniRef**, which compiles millions of protein sequences:

- **Non-IID Nature of Protein Data**: Protein sequences are not independent and identically distributed (IID). Instead, some protein families are highly represented in databases, while others are rare. This imbalance can lead models to overfit on more common protein families, limiting their generalization to underrepresented ones.
- **Clustering (UniRef50 and UniRef90)**: To address overfitting, datasets are often **clustered** at different sequence identity thresholds:
    - **UniRef50**: Sequences are clustered such that any two sequences in the dataset share no more than 50% identity. This encourages the model to learn from diverse protein families.
    - **UniRef90**: Clustering at 90% identity allows for more redundancy, providing more sequence variation within clusters. This is beneficial for tasks like mutation effect prediction, where slight variations within a family are relevant.
- **Dataset Composition**: Most models trained on UniRef datasets consist primarily of **bacterial sequences**, with much smaller proportions of eukaryotic and viral sequences. This bias influences model performance on different types of proteins, as models tend to be better at tasks involving protein families more abundant in the training set.

This curated clustering approach allows for both diversity and redundancy, depending on the task, and has proven essential for achieving balanced performance across various protein classes.

4. Data Requirements and Model Performance with Larger Datasets

Protein language models, particularly those used for structure and function prediction, are **data-intensive**. Models perform better with larger datasets, as more training sequences expose them to a broader range of sequence contexts and co-evolving residues. For example:

- **Improved Secondary Structure Prediction**: Studies have shown that model performance in predicting secondary structure (like alpha helices and beta sheets) increases with the number of training sequences.
- **Scaling Data for Better Results**: While models like UniRef50 contain around 53 million sequences, larger databases such as **BFD (Big Fantastic Database)**, with billions of sequences, offer even more diverse representations. These large-scale datasets contribute to more robust models capable of generalizing across a wide array of protein families.

5. Mutation Effect Prediction with Masked Language Models

One valuable application of protein language models is **predicting the effects of mutations** on protein function:

- **Masking Specific Mutations**: In mutation effect prediction, instead of masking random amino acids, the model specifically masks the site of the mutation (e.g., a mutation from phenylalanine (F) to alanine (A) at position 3).
- **Log Probability Ratios**: The model computes the **negative log probabilities** of each amino acid at each position, then calculates the log ratio between the mutated and wild-type amino acids to assess the mutation's impact.
    - A **positive log ratio** suggests that the mutation is more favorable than the wild type, while a negative ratio indicates the opposite.
- **Self-Supervised Prediction**: This approach aligns well with the model's training objective, as masked modeling inherently conditions the model to predict missing elements in the sequence. This natural alignment between training and prediction tasks enhances accuracy in evaluating mutation effects.

Through this mechanism, protein language models can help predict mutations' functional consequences, offering insights into **disease-causing mutations**, **drug resistance** mechanisms, and more.

Summary

Masked language modeling has transformed protein analysis by enabling models like **ESM** to learn from unstructured protein sequences, bypassing the limitations of MSAs. By training on large, curated databases and employing self-supervised techniques, these models capture rich sequence patterns, making them adept at tasks such as **secondary structure prediction** and **mutation effect analysis**. With data-intensive approaches and sophisticated architectures, masked language models open new frontiers in **protein engineering**, **therapeutics**, and **evolutionary biology**, providing a powerful tool for analyzing the vast and diverse protein universe.

37:57 Model evaluation for Protein Language Models

Evaluating the performance of protein language models involves assessing their predictive accuracy across a variety of protein-related tasks. These tasks include **mutation effect prediction**, **contact prediction**, **structure prediction**, and other functions critical to understanding protein behavior. The effectiveness of these models can vary significantly based on the type of task, the availability of evolutionary sequence data, and the specific architecture or training strategy used. Here, we'll explore some of the primary evaluation methodologies and metrics, as well as the results from benchmarking studies.

1. Key Evaluation Tasks

Protein language models are evaluated on several important tasks:

- **Mutation Effect Prediction**: One of the most direct applications, mutation effect prediction assesses how specific mutations alter a protein's properties, such as **stability**, **binding affinity**, or **functionality**. Mutation effect predictions can help in identifying **disease-causing mutations** or mutations that might enhance protein function.
- **Contact Prediction**: Using attention maps within models, contact prediction identifies pairs of amino acids likely to be spatially close in the protein's 3D structure. This task is similar to using co-evolving sites in a multiple sequence alignment (MSA) to infer contact points, but language models can achieve this without an MSA.

- **Structure Prediction**: Models may also predict secondary or tertiary structure elements, which are crucial for understanding a protein's functional form.
- **Other Functional Predictions**: Protein language models can predict properties like **solubility**, **stability**, **subcellular localization**, and **binding affinity**. These predictions support a range of applications, from drug design to synthetic biology.

2. Benchmarking with ProteinGym

One prominent benchmarking study, **ProteinGym**, systematically evaluated over 50 models on mutation effect prediction using **deep mutational scanning** data. Deep mutational scans provide **experimental measurements of single and multiple mutations** across various proteins, giving a comprehensive basis for model evaluation.

In deep mutational scans, each mutation's effect on specific phenotypes—such as **thermostability**, **binding potential**, or **expression level**—is measured. ProteinGym utilized these datasets to compare different models across these effects, using metrics such as **Top-K Recall** to quantify performance on mutation predictions.

3. Comparison of Model Types and Historical Performance Trends

ProteinGym's findings reveal both historical progress in model capabilities and task-specific strengths across different model architectures:

- **Site-Independent Models**: These early models treated each site independently and achieved moderate accuracy.
- **Pairwise Models (e.g., EVmutation)**: By considering pairwise interactions between amino acid sites, these models improved mutation effect predictions significantly compared to site-independent models.
- **Higher-Order Models**: More advanced models like **DeepSequence** and **EVE** incorporated **higher-order constraints** (beyond pairwise) using approaches like **Variational Autoencoders (VAEs)** and **Transformers**. These models demonstrated further improvements, especially for tasks involving complex epistatic interactions (where the effect of one mutation depends on the presence of another).
- **Transformer-Based Models**: Models such as **TransPro** and **ESM** (trained with attention-based architectures) have set new benchmarks for accuracy across multiple protein tasks, capturing nuanced interactions across long-range amino acid pairs.

These advancements demonstrate a clear trend of increasing model sophistication and accuracy over time.

4. Task-Specific Strengths and Limitations

Different models excel at different tasks depending on their architecture and training data:

- **Stability Prediction**: Models trained with an emphasis on structural information, like **inverse folding models**, are particularly good at predicting stability. These models focus on how likely a protein structure is to remain intact under different conditions.
- **Activity and Expression Predictions**: Models that learn from broader datasets (e.g., language models trained on large, unaligned datasets) generally perform well on predicting a protein's activity or expression level, as they capture more diverse functional constraints beyond structure alone.

These differences underscore the need for specialized model architectures depending on the desired task.

5. Effect of Evolutionary Data Availability

The availability of evolutionary sequence data plays a significant role in model performance:

- **Evolutionary Sequence Density**: Proteins with extensive evolutionary data (i.e., proteins that appear frequently across diverse organisms) enable models to learn more effectively. For instance, proteins like **proteases**—common across species—have higher predictive performance compared to proteins with limited evolutionary examples, like the SARS-CoV-2 spike protein.
- **Improved Performance with Abundant Data**: Models trained on proteins with high evolutionary sequence density (e.g., with more homologs in databases) consistently outperform those trained on sparse data. This pattern holds across both MSA-based models and general protein language models.

Ultimately, the more evolutionary data available, the more accurately models can capture functional constraints and predict effects on structure and function.

6. General Observations from Benchmarking

- **Increase in Predictive Accuracy**: Models show steady improvement in accuracy as training data and model architectures become more advanced.

- **Task-Specific Optimizations**: While general protein language models perform well across a range of tasks, specialized models for tasks like stability prediction or mutation effect prediction yield even greater accuracy in specific domains.
- **ProteinGym's Contribution**: The systematic comparisons conducted by ProteinGym provide invaluable insights into the strengths and limitations of various models. This dataset-driven approach helps guide future model development, highlighting areas where additional data or architectural adjustments could lead to further gains.

Summary

Protein language models are evaluated through their performance on critical tasks like mutation effect prediction, contact prediction, and structure prediction. Benchmarking studies like ProteinGym offer insights into the evolving landscape of protein models, illustrating how architectural advances and the availability of evolutionary data have enhanced model accuracy. With these evaluations, we gain a deeper understanding of which models are best suited for specific protein-related tasks, enabling better application in fields such as **biomedicine, drug discovery, and synthetic biology**.

44:14 Model Embeddings in Protein Language Models

Protein language models generate embeddings that capture essential information about amino acids and sequences. These embeddings, unlike the simplistic one-hot encoding, contain **contextual sequence information** that is sensitive to the specific role each amino acid plays within a protein structure. Let's delve into the nature of these embeddings, their properties, and how they serve as valuable tools in **predicting protein behavior** and **guiding downstream applications**.

1. Amino Acid Embeddings vs. Contextual Sequence Embeddings

The most basic embeddings in protein language models are **amino acid embeddings**—vectors that uniquely represent each amino acid based on its properties. For example, an **alanine** would have a specific vector, independent of its context in a sequence. However, these are static and don't account for how an amino acid's environment can affect its role within the protein.

Contextual sequence embeddings go further by integrating the surrounding sequence information, which evolves as the model progresses through layers. With each layer, the model captures a deeper representation of each amino acid in the **context of the entire protein**, effectively embedding functional and structural dependencies. For a final layer representation, each amino acid is described by a vector (often of high dimensionality, such as 1,280 values), creating a **rich representation of the entire protein**.

2. Generating Protein-Level Embeddings

To understand the entire protein, one common approach is to aggregate these **amino acid-specific embeddings** into a single **protein-level embedding**. This can be done by **averaging** the vectors across all amino acids, creating a single representation that encapsulates the protein's overall characteristics. These aggregated embeddings are incredibly informative and can cluster proteins by their structural or functional properties.

- **Clustering by Structural Features**: When these embeddings are visualized using techniques like **t-SNE**, distinct structural classes such as **alpha-dominant** and **beta-dominant** proteins can be observed, reflecting the embeddings' ability to capture secondary structural patterns.
- **Clustering by Localization**: The embeddings also reveal protein localization, such as clustering proteins that localize in the **nucleus**, **cytoplasm**, or other organelles. This shows that the model learns functionally relevant attributes, even though it was trained without explicit labels for localization.

3. Incorporating Structural Information into Embeddings

Certain models enhance standard embeddings by incorporating **structure-aware tokens**. For instance, models like **SA Pro** extend the typical amino acid tokens by combining them with tokens representing the **local 3D structural environment** of each position. This helps the model capture detailed spatial arrangements, improving its ability to distinguish between structurally unique protein types (e.g., **alpha- vs. beta-rich proteins**).

- **Enhanced Structural Clustering**: With structure-aware tokens, embeddings show even clearer clustering by structure, as they capture nuances of **local folding** and **neighboring interactions** that are crucial for structural integrity and function.

4. Embedding Models for Specialized Protein Families

In cases of specialized proteins, such as **antibodies**, models specifically trained on relevant families

outperform generic models. For example, **AbLang**, an antibody-specific language model, provides embeddings that capture key features of antibody **V gene families** (variations contributing to immune diversity) and differentiates between **naive** and **memory B cells**. This precision illustrates the value of training models on domain-specific sequences for enhanced interpretability and performance.

5. Using Embeddings for Downstream Tasks

The protein embeddings generated by these models are versatile inputs for **supervised learning tasks**. For example:

- **Predicting Protein-Protein Interactions**: For a task like predicting the interaction between a **T-cell receptor (TCR)** and a **major histocompatibility complex (MHC)**, the embeddings from each protein can be averaged and fed into a neural network, trained to classify binding or non-binding pairs. The clustering of embeddings into **binding vs. non-binding groups** confirms that these representations capture essential interaction features.
- **Modeling Binding Affinity and Functional Prediction**: By leveraging embeddings trained on unsupervised protein language models, supervised models can more accurately predict **binding affinities**, **localization**, and **functional roles**. These embeddings act as highly informative inputs, improving predictive accuracy compared to traditional approaches like **one-hot encoding** or **BLOSUM-based embeddings**.

6. Active Learning with Protein Language Model Embeddings

For tasks like **protein engineering** and **mutational optimization**, an **active learning** approach utilizing protein embeddings can streamline experimental validation:

- **Initial Mutation Testing**: Begin by testing a small set of mutations and measure their effects. Use this initial data to train a supervised model that uses the protein language model embeddings to predict mutation impacts.
- **Iterative Model Refinement**: The trained model then proposes additional mutations, which are experimentally tested, creating a feedback loop. Each cycle refines the model's accuracy, directing experiments toward **highly impactful mutations**.
- **Efficiency in High-Dimensional Mutation Spaces**: This iterative approach is particularly effective in scenarios requiring **combinatorial mutation exploration** (e.g., double or triple mutations). Instead of testing all possible mutations, the model prioritizes those likely to yield functional improvements, optimizing the exploration of mutational space without exhaustive experimental resources.

Conclusion

Protein language model embeddings provide a comprehensive and contextually sensitive representation of proteins, capturing both structural and functional nuances. By aggregating these embeddings, researchers can gain insights into **protein structure**, **localization**, and **interaction potential**. Furthermore, these embeddings empower supervised models in tasks such as **protein-protein interaction prediction**, **activity optimization**, and **binding affinity estimation**. Through active learning, these embeddings also support **efficient protein design**, enabling systematic exploration of mutational landscapes for optimal outcomes. The use of embeddings from protein language models marks a significant advancement in protein science, offering a robust toolkit for both predictive and experimental applications.

56:24 Attention Maps in Protein Language Models

Attention mechanisms in protein language models, like those found in **Transformer architectures** (e.g., ESM), play a significant role in capturing interactions between amino acid positions across a protein sequence. These mechanisms are instrumental in **identifying structural contacts**, enabling models to predict 3D structural features and interactions within proteins.

1. Understanding Attention Maps in Protein Models

In a Transformer model, attention maps represent **interactions between sequence positions** by calculating how much focus one amino acid position places on another. This is achieved through **query**, **key**, and **value** matrices, which determine the relationships and dependencies across the sequence.

- **Position-to-Position Mapping**: In the context of a protein sequence, each row and column in an attention map corresponds to a specific **amino acid position**. If the sequence has length LLL, the attention map is an L×LL \times LL×L matrix, where each cell indicates the importance of one position relative to another.

- **Interpretation of High Attention Values**: When a high attention score appears between two positions, it suggests that these positions are interdependent, potentially indicating that they are **in close proximity within the protein's 3D structure**.

2. Using Attention Maps to Infer Structural Contacts

The **attention scores** between positions can be interpreted as signals for **structural contact** in the protein's folded form. By examining these attention patterns across various layers and heads within the Transformer model, researchers can identify specific **spatial relationships** between amino acids, even if the model was trained solely on sequence data.

- **3D Structural Correlation**: Many pairs with high attention values correspond to **physically interacting residues** in the protein's 3D structure, such as hydrogen bonds or van der Waals interactions.
- **Layer-Specific and Head-Specific Attention**: Not all attention heads or layers may focus on structural aspects. Specific **heads** or **layers** might excel at capturing contacts, while others may focus on different aspects, like sequence motifs or functional sites.

3. Selecting Relevant Attention Maps for Structural Prediction

Because not every attention map is optimized for structural insights, a **secondary model** can be trained to refine which maps contribute most to structural contact prediction:

- **Filtering Important Attention Heads and Layers**: Using labeled data from proteins with known structures, a secondary model learns which **heads and layers** are most relevant for contact prediction. This allows the model to selectively emphasize attention maps that correlate well with physical contacts.
- **Weighted Combination of Maps**: The secondary model can also assign **weights** to each attention map based on their relevance, combining them to improve accuracy in **predicting structural contacts**.

4. Applications of Attention-Derived Structural Contacts

The insights gleaned from attention maps extend beyond contact prediction and can be used in several structural and functional applications:

- **Contact Prediction in 3D Structure Modeling**: Using attention maps as an initial approximation for contact maps enables models to **predict 3D protein structures** without requiring multiple sequence alignments. This makes attention maps particularly useful for modeling novel or less-studied proteins where little evolutionary data is available.
- **Functional Site Identification**: By analyzing which residues consistently receive high attention, researchers can locate **functionally important sites** or **binding pockets** that are crucial for a protein's activity.
- **Validation of Evolutionary Models**: Attention-derived contacts can complement traditional **evolutionary contact prediction** (based on co-evolving residues), offering an alternative approach for proteins with limited evolutionary data or those outside of conventional protein families.

5. Visualization and Interpretation of Attention Maps

Attention maps can also be visualized for more intuitive interpretation:

- **Single Head and Layer Visualization**: By selecting a single attention head within a specific layer, researchers can observe **localized contact patterns** that may correspond to particular structural domains or secondary structures (e.g., alpha helices or beta sheets).
- **Global Aggregation of Contacts**: Alternatively, combining attention maps across layers provides a comprehensive view of **inter-residue interactions** throughout the entire protein sequence, highlighting key structural and functional relationships.

Summary

Attention maps in protein language models offer a powerful tool for uncovering **structural contacts** within proteins. Through **layered and head-specific attention**, these maps reveal meaningful interactions, making them effective for **3D structure prediction** and **functional site identification**. By training secondary models to highlight the most structurally relevant attention maps, researchers can enhance the predictive capabilities of protein language models, bridging the gap between sequence-based models and spatial structure insights. Attention maps thus serve as an invaluable resource for exploring the **intricate architecture of proteins**, guiding both basic research and practical applications in protein engineering and drug design.

59:05 Protein Design

The field of **protein design** has emerged as a transformative approach, leveraging computational models to create proteins with novel structures and functions. With the recent Nobel Prize awarded to **David Baker** for his pioneering work in this area, the importance and potential of protein design have gained global recognition. Computational protein design allows for the creation of entirely new proteins with **specific desired characteristics**—from targeted binding sites to complex assemblies and dynamic shapes.

1. Overview of De Novo Protein Design

In de novo protein design, researchers use **computational methods** to generate protein sequences and structures that have not previously existed in nature. Unlike natural proteins, which evolve over time through genetic selection, de novo proteins are constructed **from scratch**, designed to fulfill specific tasks or exhibit certain structures.

- **Examples from the Baker Lab**: The Baker lab has produced a variety of synthetic proteins, showcasing the range and potential of de novo design:
    - **Complex Protein Assemblies**: Designed complexes where multiple proteins—sometimes as many as 120—assemble into intricate structures. These complexes could serve as molecular machines or act as highly structured scaffolds.
    - **Molecular Rotors**: Proteins engineered to function as moving parts within molecular machinery, capable of rotating or changing conformation based on environmental factors.
    - **Binding Proteins**: Proteins with engineered binding affinity to specific molecules, such as fentanyl, which could lead to targeted therapeutic applications.
    - **Nanoparticles for Vaccines**: Protein-based nanoparticles designed to present antigens in a controlled and repetitive manner, enhancing immune response and opening up new possibilities for vaccines.

2. Applications of Protein Design Across Domains

Protein design has vast applications that span across several industries, from healthcare to sustainability. By enabling scientists to precisely control protein structure and function, protein design can address critical issues in **environmental sustainability, healthcare, and bioengineering**.

- **Sustainability**: Proteins can be engineered to tackle environmental challenges.
    - **Plastic Degradation**: Designing enzymes, such as PETases, that can efficiently break down plastics into environmentally benign components.
    - **Carbon Capture**: Proteins designed to capture and convert CO_2 more efficiently could play a role in mitigating climate change by reducing atmospheric carbon dioxide levels.
- **Healthcare**: Custom-designed proteins have transformative potential in diagnostics and therapy.
    - **Antibodies and Inhibitors**: Synthetic antibodies and protein inhibitors can be created to target specific pathogens or disease-related proteins, offering precise and potent therapeutic interventions.
    - **Vaccine Development**: Protein nanoparticles that mimic virus structures can serve as highly effective vaccine components, presenting antigens in an organized fashion to stimulate a robust immune response.

3. Role of Protein Language Models in Design

Protein language models, trained on extensive protein sequence data, provide valuable insights into **sequence-function relationships** and **mutation effects**. These models assist in predicting which sequences will lead to stable, functional proteins and which modifications will yield desired behaviors.

- **Generating Novel Sequences**: By leveraging embeddings and mutational predictions, protein language models can help generate new sequences that satisfy certain structural or functional criteria.
- **Structure-Informed Design**: Integrating language models with structure-aware models allows for the development of proteins that have been computationally optimized for both **sequence and 3D conformation**, enhancing stability and function.

4. Future Directions and Challenges in Protein Design

While advances in protein design are promising, several challenges remain to be addressed to fully unlock the potential of this field:

- **Complexity of Protein-Protein Interactions**: Designing proteins to interact predictably and specifically with other proteins remains a challenging task, as it requires precise knowledge of binding interfaces and interaction dynamics.
- **Function in Variable Environments**: Proteins designed for therapeutic or industrial applications must function reliably across diverse conditions. This demands models that can predict behavior under varying temperatures, pH levels, and chemical environments.
- **Efficiency in Prediction and Validation**: While computational design accelerates protein discovery, experimental validation remains essential. Bridging computational predictions with high-throughput experimental validation methods will be key to scaling up practical applications of protein design.

In summary, protein design stands at the intersection of **computational biology, chemistry, and bioengineering**, offering unprecedented opportunities to craft proteins with bespoke properties. As computational methods continue to evolve, supported by protein language models and structure-aware models, the field of protein design will likely expand its impact, transforming fields from **medicine and materials science to sustainability**. With ongoing developments, the scope and accuracy of protein design will continue to grow, promising a new era of custom-built proteins tailored to address some of the most pressing challenges in science and society.

1:00:51 Case Study on Viral Evolution: Predicting SARS-CoV-2 Spike Protein Mutations

This case study explores the use of **protein language models and evolutionary modeling** to understand and predict viral evolution, specifically focusing on **SARS-CoV-2** (COVID-19). The goal is to develop computational models that could predict viral mutations, particularly those contributing to **antibody escape**, even before such mutations appear in the population.

1. Viral Evolution and Mutation Waves

The SARS-CoV-2 pandemic saw distinct **mutation waves** corresponding to variants like Alpha, Delta, and Omicron, each with significant differences in the **Spike protein**—the viral protein responsible for binding to host cells. The ongoing emergence of these variants has posed significant challenges for **immunity** because new mutations can escape previously developed antibodies, whether from natural infection or vaccination.

- Early in the pandemic, it was believed that SARS-CoV-2 did not mutate rapidly, leading to hopes for stable, long-lasting vaccine protection. However, this assumption proved incorrect as variants emerged, accumulating numerous mutations. Omicron, for instance, had **37 mutations** in the spike protein compared to the original strain, many of which contributed to **antibody escape**.

2. The Role of Machine Learning in Predicting Viral Escape

The primary research question was whether **machine learning models** could predict these viral mutations early, enabling proactive development of vaccines and therapies. Given the substantial data generated during the pandemic, from millions of SARS-CoV-2 sequences to structural data on antibody-antigen interactions, researchers aimed to develop models that could **forecast likely escape mutations**.

- **Data Availability and Model Constraints**: Researchers intentionally restricted their models to **pre-2020 data** to assess if these models could have predicted future mutations without relying on pandemic-era data. This restriction also avoided overfitting to known pandemic variants, testing the model's ability to generalize from related viruses like **SARS-CoV-1** and **MERS-CoV**.

3. Components of Mutation Prediction

Three main factors influence the likelihood of a mutation leading to antibody escape:

- **Accessibility**: Whether the mutation site on the spike protein is accessible to antibodies. Protein structure data can reveal regions where antibodies are more likely to bind.
- **Dissimilarity and Impact on Binding**: The degree to which a mutation alters the chemical properties (such as charge or hydrophobicity) at a binding site. Significant changes here can prevent antibodies from binding effectively.
- **Viral Fitness**: The mutation's effect on the virus's overall fitness, especially regarding its ability to bind to the host cell receptor, ACE2. Mutations that compromise the virus's infectivity are unlikely to persist, as they would reduce the virus's ability to spread.

4. Use of Protein Language Models for Predictive Modeling

By employing **protein language models** trained on sequences from the **UniRef database** (UNIF50, UNIF90, and UNIF100) or specific **coronavirus alignments**, researchers built models that could predict which spike

protein mutations would likely persist and evade antibodies.

- **Sequence Similarity Considerations**: For virus-specific tasks, models trained on UNIF100 sequences, which retain more closely related viral sequences, were more useful than models trained on UNIF50, which removed closely related sequences, limiting the representation of specific viral families.

5. Comparing Model Predictions with Experimental Data

These models were validated against **deep mutational scanning** data—experimental assays where every possible mutation is tested to assess its impact on antibody binding and viral fitness. Comparisons revealed that **pre-trained models on pre-pandemic data** could predict immune escape mutations as effectively as experimental methods that used **post-pandemic antibodies**, underscoring the model's utility for **early warning and vaccine development**.

6. Case Study Applications

- **Early Warning for Variants of Concern**: By ranking thousands of emerging strains, models could identify those with the highest escape potential. This early detection could help researchers flag and prepare for future variants of concern.
- **Evaluating Future Vaccine Efficacy**: Rather than testing vaccines against only existing variants, models could generate "future-oriented" variant mutations to assess if new vaccines would remain effective. This could prevent the need for continual vaccine updates by ensuring that antibodies elicited by vaccines bind to regions of the virus that are less likely to mutate.
- **Designing Future-Proof Vaccines**: A long-term approach involves designing vaccines that focus antibody responses on **conserved regions** of the virus that are less prone to mutation. By computationally mutating non-essential regions, vaccines could theoretically encourage immunity that targets the virus's "weak points"—regions it cannot mutate without compromising infectivity.

7. Safety and Ethical Considerations

Given the sensitivity of mutation prediction in pathogens, safety protocols and ethical considerations are paramount. **Biosafety protocols** are critical to ensuring that models are only accessible to vetted users in academic or industry settings. Such safeguards prevent the misuse of predictive insights for harmful purposes.

Conclusion

This case study exemplifies the **power of protein language models** in real-world applications, showcasing their ability to predict evolutionary pathways in viruses with high accuracy. By combining structural data, evolutionary constraints, and sophisticated model embeddings, these tools can significantly aid public health efforts in preparing for and responding to viral threats. This approach represents a proactive shift in **infectious disease management**—leveraging computational predictions to stay ahead of viral evolution and enhance vaccine efficacy in an ever-evolving landscape.

Lecture 12 - DNA Language Models

Video: Lecture12 DNA language models and Convolution

Slides: Lecture12_DNALanguageModels.pdf

00:00 Intro to DNA language models

Today's session dives into **DNA language models**, with a continued focus on Transformer-based approaches and an exploration of **convolutional neural networks (CNNs)** as a mainstay model type used for DNA sequence analysis.

Context and Goals:

- **DNA Language Models**: Similar to how protein language models are developed, DNA language models aim to interpret the "language" of DNA sequences, understanding patterns, motifs, and the relationships between nucleotide sequences.
- **Transformers in DNA Analysis**: Given the success of Transformers in sequential data analysis, these models are increasingly applied to DNA, capturing dependencies across varying sequence lengths.
- **Convolutional Neural Networks**: Particularly effective for local pattern recognition, CNNs help detect important motifs or regulatory elements in DNA sequences, making them valuable for certain DNA-specific tasks where local sequence features are crucial.

In this chapter, we'll explore how these models apply to DNA, their respective strengths, and how they work

together to interpret complex genomic data.

1:00 RNA Splicing Models and Splice AI

Introduction to RNA Splicing Models: The first problem tackled today is RNA splicing, which is the process of transforming a pre-RNA sequence into fully spliced RNA. Splicing accurately identifies the intron-exon boundaries, an essential step in producing functional proteins. This task is challenging because splicing signals, particularly **splice sites**, are subtle and not as easily identifiable compared to other genomic signals.

Complexity of RNA Splicing:

- **Splice Sites**: Key components in splicing are the **donor site** (marked by "GT") and the **acceptor site** (typically "AG").
- **Weakly Conserved Signals**: Additional sequences around the splice sites, often weakly conserved, contribute to the complexity of identifying genuine splice sites.
- **Functional Impact**: The splicing process can yield different isoforms of a gene, affecting the protein's function. Proper identification of splice sites is crucial for understanding which isoforms are biologically relevant.

Splice AI: A Convolutional Neural Network Approach One model that has significantly advanced RNA splicing prediction is **Splice AI**, a model based on convolutional neural networks (CNNs). CNNs are particularly well-suited for this task due to their ability to detect local patterns, which helps in recognizing the subtle splice site signals amidst the broader DNA sequence.

How Splice AI Works:

- **Input Sequence**: Splice AI takes an input DNA sequence and feeds it through a series of **convolutional layers**. These layers scan across the sequence to detect patterns relevant to splicing.
- **Convolutional Layer Insights**: Each convolutional layer focuses on different aspects of the sequence, gradually building a hierarchical understanding of splice sites and surrounding signals.

In summary, **Splice AI** leverages CNNs to enhance the prediction of splice sites, setting a new standard in RNA splicing models. Its ability to recognize subtle sequence signals amidst complex data structures has proven valuable in accurately predicting functional splicing events.

3:30 Understanding Convolutions in Neural Networks and Their Application

What is a Convolution?

- **Definition**: A convolution is an operation where two functions interact to produce a third function that represents their combined effects. In neural networks, convolutions are commonly used to detect patterns, especially in spatially structured data like images or DNA sequences.
- **Application in Research**: Convolutions have broad applications beyond image processing, as illustrated by a COVID-19 wastewater surveillance study. In this research, convolutional methods helped estimate the prevalence of infections by analyzing viral shedding patterns over time.

Example of Convolution in Practice

In the example discussed, each infected individual sheds a certain amount of virus daily over a five-day infection period. To calculate the virus concentration in wastewater over time, a convolution operation combines:

- **Infection Function (I(t))**: The number of new infections per day.
- **Shedding Function**: Describes the virus shed by an individual daily.

By convolving these functions, researchers derived the expected viral concentrations, illustrating how convolutions can reveal hidden patterns in complex data.

Key Terms in Convolutions:

1. **Input**: The data sequence being analyzed, such as a DNA sequence or viral particle counts.
2. **Kernel**: A smaller matrix or function that slides over the input, identifying features or patterns.
3. **Feature Map**: The output resulting from the convolution operation, highlighting detected patterns.
4. **Width**: The range over which the kernel operates.
5. **Stride**: The step size as the kernel moves across the input, allowing the model to reduce data size and computation.
6. **Padding**: Adds borders to the input data to maintain the original size of the output after convolution.

Convolutions in 2D

In image processing, convolutions help extract features by detecting patterns such as edges or textures. For example:

- **Kernel for Vertical Lines**: A specific kernel detects vertical lines in an image by highlighting contrasts.
- **Convolution Process**: The kernel slides across the image, element-wise multiplying values to create a new "convolved" image that emphasizes specific patterns.

Stride and Pooling

- **Stride**: Defines how much the kernel moves with each step. A higher stride reduces the size of the output, simplifying data.
- **Pooling (e.g., Max Pooling)**: Reduces the spatial size of the representation by taking the maximum value in a defined region, allowing the model to focus on essential features.

Historical Context and Importance

Convolutions became a cornerstone of deep learning with breakthroughs like:

- **LeNet-5 (1989)** by Yann LeCun: Pioneered convolutional networks in digit recognition, using backpropagation to optimize kernel parameters.
- **AlexNet (2012)** by Alex Krizhevsky: Trained on a large dataset with GPUs, it demonstrated the power of deep CNNs in image recognition, revolutionizing the field and solidifying convolutional neural networks as key tools in AI.

In conclusion, convolutions play a critical role in detecting spatial patterns within data, enabling applications ranging from image analysis to genomic research. Convolutional neural networks (CNNs) leverage these principles to efficiently process and interpret complex data by reducing model parameters, maintaining spatial relationships, and highlighting important features.

26:49 1D Convolutions for DNA Sequence Analysis

In genomic modeling, particularly with DNA, the sequence lengths are massive, making it computationally challenging and costly to use Transformers alone. Convolutional neural networks (CNNs), particularly 1D CNNs, are frequently utilized for DNA sequence analysis due to their efficiency and their ability to detect patterns or motifs in linear sequences like DNA.

How 1D Convolutions Work on DNA Sequences

1. **One-Hot Encoding**: DNA sequences are first converted into one-hot encoded vectors. Each nucleotide (A, T, C, G) is represented by a unique vector with a single '1' in the corresponding position and '0's elsewhere.
    - **Example**: A DNA sequence "ATCG" would convert into a matrix where each row represents a nucleotide in one-hot encoding:
        - A → `[1, 0, 0, 0]`
        - T → `[0, 1, 0, 0]`
        - C → `[0, 0, 1, 0]`
        - G → `[0, 0, 0, 1]`
2. **Channels and Convolutions**:
    - **Input Channels**: The initial one-hot encoded DNA sequence has four channels, corresponding to the four nucleotide types.
    - **Convolution Process**: As the sequence passes through convolutional layers, the CNN can generate additional channels. Each output channel can be connected to all input channels, enabling the model to learn complex patterns across nucleotides.
    - **Detecting Motifs**: The CNN layers can be designed to detect sequence motifs (e.g., "ATG") by assigning weights that identify specific nucleotide combinations across adjacent positions.
3. **Capturing DNA Sequence Patterns**:
    - **Layer Connections**: Each channel in the convolutional layer connects to all prior channels, allowing the network to "mix and match" patterns across positions. For example, a specific convolutional filter could detect the pattern "ATG" by activating on sequences where A is in the

first position, T in the second, and G in the third.

4. **Model Output**: Each layer can represent specific genomic features. Depending on how the model is trained, output channels can indicate regions related to chromatin structure, enhancer or repressor binding sites, or other regulatory elements within the DNA sequence.

Application to RNA Splicing Models

In models like **Splice AI**, which predicts RNA splicing sites, these 1D convolutional layers are essential for recognizing sequence motifs that signal splicing events. Convolutional layers analyze the DNA sequence to detect splice donor and acceptor sites, even with the complex and often weakly conserved signals involved in RNA splicing.

By using CNNs in this way, models can efficiently process extensive DNA sequences, extracting valuable patterns that are crucial for understanding regulatory functions and gene expression mechanisms within the genome.

30:08 SpliceAI: Advancements in RNA Splicing Prediction

Overview

SpliceAI is a deep learning model designed to predict RNA splice sites with high accuracy. RNA splicing involves cutting out non-coding regions (introns) and joining coding regions (exons) in a pre-mRNA sequence, a process that can alter gene function. Predicting accurate splice sites is challenging because splice signals in DNA are often short and weakly conserved. SpliceAI's approach represents a significant advancement in accurately predicting these sites.

Model Structure

1. **Input Sequence Representation**:
    - The input DNA sequence is first one-hot encoded. Each nucleotide (A, T, C, G) is represented as a vector where only the corresponding nucleotide's position is marked with a '1', and all others are '0'.
    - The model considers 80 nucleotide-long sequences in the **80 nucleotide model**, while larger versions like the **10K nucleotide model** can handle sequences up to 10,000 nucleotides, integrating more long-range dependencies.
2. **Convolutional Layers and Kernel Parameters**:
    - The SpliceAI model uses convolutional layers with varied kernel widths and dilation parameters to capture both local and long-range sequence features.
    - **Width (W)**: Defines the number of nucleotides considered in each convolution operation. For SpliceAI, the width is typically set to 11 nucleotides, capturing enough local context to account for one full DNA helix turn—a meaningful segment for DNA-binding proteins.
3. **Dilation Parameter (D)**:
    - Dilation, similar to stride in CNNs, expands the effective range of the convolution without increasing kernel size by "skipping" positions in the sequence.
    - For example, a kernel of size three with a dilation of two would capture every second nucleotide, effectively covering five nucleotides without increasing the computational complexity of the kernel. This enables SpliceAI to consider both closely spaced and more distant nucleotides, critical for recognizing splicing patterns that may span large genomic regions.
4. **Skip Connections and Batch Normalization**:
    - Skip connections (also called residual connections) enable the model to retain initial input information by adding outputs from earlier layers directly to later layers. This structure stabilizes training and enhances SpliceAI's ability to recognize both local and long-range signals.
    - Batch normalization adjusts the mean and variance of neuron activations across batches, ensuring stable training even with a deep model structure.

Output and Predictions

- **Softmax-Processed Output**:
    - SpliceAI's output layer generates logits, or raw scores, for each position in the sequence. These logits are converted to probabilities via a softmax function to classify each position as either:

        - **Donor**: Likely a donor splice site.
        - **Acceptor**: Likely an acceptor splice site.
        - **Neither**: Not involved in splicing.
- **Prediction Quality**:
    - SpliceAI's predictions have been shown to align closely with experimentally verified splice sites, reducing false positives and capturing more accurate splice site locations than previous models.

Impact and Applications

SpliceAI's sophisticated approach to convolution and dilation has made it possible to predict RNA splicing with a level of precision previously unattainable. By leveraging a combination of local and long-range sequence information, SpliceAI allows researchers to infer splicing patterns across a genome with greater confidence, potentially without RNA-seq data. This has broad implications in understanding gene expression regulation, genetic disease mechanisms, and personalized medicine.

37:50 DNA Foundation Models: Training and Applications

DNA foundation models are powerful, large-scale neural networks trained on extensive genomic data. Similar to language models in NLP, these models are designed to learn general DNA sequence patterns and principles by predicting masked-out segments within the DNA sequence. Here's a breakdown of how these models work and their potential applications:

Training DNA Foundation Models

- **Masked Language Modeling**:
    - DNA foundation models are typically trained by masking specific nucleotides or nucleotide sequences within a DNA segment and then predicting the masked-out sections. This training method is analogous to tasks used in natural language processing (NLP), where words or phrases are masked, and the model predicts the missing pieces.
    - Through this process, the model learns the probabilistic structure of DNA, including regulatory motifs, coding patterns, and structural features.
- **Learning Core DNA Principles**:
    - To accurately predict masked sequences, the model must internalize various biological principles governing DNA sequence composition. This requires large-scale genomic data, which fortunately is readily available.
    - As a result, the model learns to recognize a broad range of genomic features beyond just the sequence itself, including elements such as **regulatory motifs**, **enhancers**, **promoters**, **chromatin states**, and more.

Applications and Advantages

- **Rich Internal Representation**:
    - Once trained, DNA foundation models have embeddings or internal representations that encapsulate a wide range of DNA information, not limited to just the immediate task. These representations are thought to contain:
        - **Splicing information** (similar to the SpliceAI model's focus)
        - **Regulatory elements** that control gene expression
        - **Chromatin states** indicating accessibility and binding potential
    - This makes them highly versatile, with embeddings that capture diverse biological insights.
- **Transfer Learning**:
    - The internal representations from these models can be repurposed for various specific tasks, a concept known as **transfer learning**.
    - Instead of training models from scratch on each task, researchers can leverage the foundational knowledge embedded in DNA foundation models to achieve more accurate predictions in areas like:
        - **Gene regulation**: Predicting enhancer or promoter activity based on DNA sequence.
        - **Genomic annotation**: Identifying functionally relevant regions within a genome.

        - **Disease association studies**: Linking certain DNA motifs or patterns to disease states.

DNA Foundation Models in Practice

- **Example: ProT5**:
    - ProT5, which was discussed in the context of protein language models, serves as a parallel example of how sequence-based models can generalize across tasks. Similarly, DNA foundation models can apply their learned representations across diverse genomic applications, potentially identifying novel genomic elements or regulatory patterns.

In essence, DNA foundation models function as **general-purpose sequence interpreters**, equipped to handle a range of tasks in genomics through their highly structured and informative embeddings. These models are transforming how researchers analyze and interpret complex genomic data, allowing for rapid adaptation to new, specific questions while leveraging a robust foundational understanding of DNA.

41:15 DNABERT Architecture: Tokenization, Embedding, and Attention Layers

Tokenization with K-mers

DNABERT tokenizes DNA sequences into **k-mers** (typically **3-mers**), where each k-mer represents a string of three consecutive nucleotides. For instance, a DNA sequence "ATGCGA" would be broken down into overlapping 3-mers like "ATG," "TGC," and "CGA." This k-mer approach is chosen for two main reasons:

1. **Meaningful Biological Units**: K-mers capture biologically relevant motifs and sequences, which may correlate with regulatory sites or protein-binding domains.
2. **Reduced Vocabulary Size**: By grouping nucleotides, DNABERT reduces the complexity of DNA sequence modeling, making it more manageable and computationally efficient.

Embedding and Positional Encoding

Each k-mer is then embedded into a high-dimensional **vector space** that reflects its unique characteristics. In addition to this **semantic embedding**, DNABERT employs **positional encoding** to maintain the order of k-mers, which is crucial as the location of motifs and patterns within a DNA sequence often impacts its biological function. Positional encoding can be achieved using sinusoidal functions or learned positional vectors, enabling the model to keep track of k-mer locations.

Self-Attention Layers

Following tokenization and embedding, DNABERT applies **12 layers of self-attention and feed-forward networks**. Each k-mer can now interact with every other k-mer, regardless of its position within the sequence. This **self-attention mechanism** enables the model to capture complex dependencies and interactions across the entire DNA sequence, which is essential for understanding regulatory interactions that span large genomic distances.

Training Objectives: Masked Language Modeling (MLM) for DNA

The core training objective for DNABERT is **masked language modeling (MLM)**, where parts of a sequence are masked, and the model learns to predict these masked k-mers based on context. This approach compels DNABERT to develop a nuanced understanding of DNA sequence patterns, relationships, and context by predicting missing segments.

During training, the model masks a portion of k-mers within a sequence and then learns to predict these masked segments. By iterating through extensive genomic datasets, DNABERT learns intrinsic DNA sequence patterns and motifs, resulting in a powerful foundation model for DNA that understands underlying sequence structures.

Fine-Tuning and Downstream Applications

Once pre-trained, DNABERT is versatile and can be fine-tuned for various **genomic prediction tasks**, such as:

1. **Transcription Factor Binding Site Prediction**: DNABERT can identify binding motifs for transcription factors, aiding in the understanding of gene regulation. By focusing on sequence regions crucial for transcription factors, DNABERT learns to highlight potential binding sites, which can then be verified or annotated in genomic studies.
2. **Genomic Annotation**: The model can predict regulatory elements, such as enhancers, promoters, and repressors, within DNA sequences. These elements play vital roles in gene expression and regulation, making them critical targets in epigenetics and functional genomics research.
3. **Disease Association and Variant Analysis**: By analyzing the effects of sequence variations,

DNABERT can help link specific motifs or mutations to disease phenotypes. This application is particularly useful in genomics and precision medicine, where understanding the genetic basis of disease is crucial.

4. **Splicing Prediction**: DNABERT can be employed to predict splice sites within genes, which is critical in understanding gene structure and alternative splicing mechanisms.

Attention Maps and Interpretability

One of the remarkable aspects of DNABERT is the **interpretability of its attention maps**. The attention mechanism within DNABERT reveals where the model is focusing its attention within a sequence, layer by layer. For instance, in predicting transcription factor binding sites, the final attention layers often converge on sequence regions that correspond to known binding motifs.

The **attention maps** offer a transparent view of how DNABERT processes genomic data, allowing researchers to understand the biological basis behind its predictions. Each layer of attention can be visualized to see where the model "focuses," often highlighting essential regulatory regions, such as enhancers or promoter regions, that are key to gene regulation.

DNABERT in Context: Comparison with Other Models

DNABERT, being a Transformer model, offers significant advantages over traditional **recurrent neural networks (RNNs)** and **convolutional neural networks (CNNs)** by providing **global context** and enabling **long-range sequence dependencies**. Where RNNs struggle with sequence length due to information bottlenecking, DNABERT excels by allowing each part of the sequence to freely share information with all other parts. Compared to CNNs, DNABERT can capture both local and global interactions, making it well-suited for tasks where regulatory sequences are interspersed throughout extensive genomic regions.

Summary

DNABERT represents a transformative application of **Transformer models in genomics**, capable of learning complex DNA sequence features and transferring this knowledge across various tasks. By leveraging **k-mers, self-attention, and transfer learning**, DNABERT has emerged as a powerful tool in genomic analysis, advancing our understanding of DNA structure, regulatory elements, and the genetic basis of diseases. Its capability to focus on biologically relevant motifs while retaining contextual relationships across sequences positions DNABERT as an essential model in the era of genomic deep learning. Through pre-training on extensive DNA datasets and fine-tuning for specialized tasks, DNABERT opens new avenues for genomic research, disease prediction, and personalized medicine.

46:48 The Nucleotide Transformer: A Deep Foundation Model for DNA Analysis

The **Nucleotide Transformer** is an advanced DNA sequence model based on the **Transformer architecture**, similar to DNABERT but with key distinctions that enhance its capacity to analyze larger and more complex DNA sequences. Like other Transformer-based models, the Nucleotide Transformer is pre-trained on vast amounts of DNA data, learning to capture the underlying structure, motifs, and dependencies across the genome. This model's versatility allows it to excel in tasks ranging from identifying regulatory elements to assessing the impact of specific mutations. Below, we delve into the architecture, training strategy, and applications of the Nucleotide Transformer, highlighting its robustness and adaptability in genomic research.

Training Methodology: Masked Prediction and Embedding Extraction

The **pre-training process** for the Nucleotide Transformer follows the **masked language modeling (MLM)** approach, similar to BERT-based models but tailored specifically for DNA. The workflow begins with raw DNA sequences, which are broken down into overlapping **6-mers** (sequences of six nucleotides). These 6-mers serve as the fundamental units, capturing more context than single nucleotides and allowing the model to recognize short functional elements within the sequence.

In the pre-training phase, portions of the sequence are masked (randomly replaced by a placeholder), and the model is trained to predict these hidden k-mers based on the surrounding context. This task encourages the model to learn the structural and sequence-based dependencies within DNA, building an **internal representation** of various motifs, regulatory sequences, and biologically significant patterns across large genomic regions.

After completing the pre-training task, the model can generate **rich embeddings** for each position within a DNA sequence. These embeddings—high-dimensional vectors representing each nucleotide's context and biological significance—form the basis for transfer learning, where the Nucleotide Transformer can be fine-tuned for specialized tasks.

Transfer Learning and Fine-Tuning: Versatility in Genomic Tasks

Once pre-trained, the Nucleotide Transformer's parameters are mostly **frozen**, and the model is used as a feature extractor. Its embeddings capture a wealth of information about DNA sequences, which can then be leveraged by secondary networks for specific genomic tasks. The model excels in:

1. **Binary Classification Tasks**: For example, it can determine whether a sequence acts as a **promoter** (indicating the starting region of a gene) or not. During fine-tuning, a small classifier network is trained on the Nucleotide Transformer embeddings to classify sequences. Since the majority of the Transformer's parameters are frozen, fine-tuning only requires adjusting the classifier network parameters, making this a computationally efficient process.
2. **Protein-Level Predictions**: The Nucleotide Transformer can also assist in evaluating DNA mutations and predicting their impact on protein function. For example, if a specific DNA mutation is suspected to affect protein structure or stability, the model can analyze the sequence and assess potential **deleterious effects**. This approach involves processing both the wild-type (original) and mutated sequences through the Nucleotide Transformer and obtaining embeddings for each. The embeddings are then averaged across positions to create a summary representation of the sequence. By comparing these embeddings, researchers can predict the functional impact of the mutation on the protein.
3. **Zero-Shot Prediction**: Remarkably, the Nucleotide Transformer can make accurate predictions without explicit training on a specific task, using a **zero-shot learning** approach. In zero-shot prediction, the model relies solely on the similarity or dissimilarity between embeddings for different DNA sequences. This feature is particularly valuable for researchers working with limited labeled data or attempting novel tasks. For instance, when analyzing rare mutations or uncharacterized regulatory regions, the model's embeddings provide a biologically meaningful measure that can guide preliminary assessments.

Performance and Benchmarking: Comparison with Task-Specific Models

The Nucleotide Transformer has demonstrated strong performance when fine-tuned for tasks like **splicing site prediction**. In particular, it competes closely with **splice AI**, a specialized model designed specifically for splicing prediction, achieving similar levels of accuracy. This outcome is significant because splice AI is purpose-built for splicing, whereas the Nucleotide Transformer was trained as a generalist model on a broader range of DNA sequences. The ability to match task-specific models in accuracy underscores the power of **foundation models** like the Nucleotide Transformer. Its comprehensive embedding space captures not only splicing information but also many other genomic features, allowing it to be adapted for diverse applications in genomics.

Key Strengths of the Nucleotide Transformer

The Nucleotide Transformer offers unique advantages that make it a valuable tool in genomics:

- **Contextual Depth**: By training on 6-mers, the Nucleotide Transformer learns a nuanced understanding of nucleotide context, capturing dependencies over a longer range compared to models based on single nucleotides. This depth is essential for accurately modeling interactions that are vital to regulatory functions, such as enhancer-promoter interactions.
- **Flexibility Across Genomic Scales**: The model can perform fine-grained, nucleotide-level predictions as well as broader, protein-level assessments. This flexibility enables researchers to use the model across tasks ranging from localized motif recognition to systemic assessments of mutation impacts.
- **Transferable Embeddings for Downstream Applications**: The Nucleotide Transformer's embeddings serve as rich features that can be applied to numerous prediction tasks, enabling robust **transfer learning**. This property minimizes the need for task-specific model re-training, which is especially useful in applications like gene expression prediction, variant annotation, and regulatory element identification.
- **Interpretability via Attention Mechanisms**: As with DNABERT, the attention layers within the Nucleotide Transformer allow for **interpretability**. Attention maps can be visualized to highlight areas of focus within the DNA sequence, often aligning with known functional sites, such as transcription factor binding sites or regulatory elements. This transparency aids researchers in validating the model's predictions against biological knowledge.

Summary

The **Nucleotide Transformer** is a groundbreaking model that pushes the boundaries of DNA sequence

analysis through advanced Transformer-based techniques. By pre-training on vast genomic datasets and leveraging k-mer tokenization, masked prediction tasks, and self-attention mechanisms, it provides a robust, adaptable platform for a wide range of genomic tasks. With its flexibility in fine-tuning for specialized applications, ability to handle both local and long-range dependencies, and potential for zero-shot prediction, the Nucleotide Transformer exemplifies the power of **foundation models** in genomics. This model not only matches but also rivals task-specific models like splice AI in performance, while retaining the versatility to be applied to novel challenges in genomic research and medicine. Its role as a multi-purpose genomic model highlights a future where such foundational architectures underpin much of DNA-related computational analysis, helping drive breakthroughs in understanding gene function, regulation, and disease mechanisms.

50:20 SegmentNT: Leveraging the Nucleotide Transformer with UNet Architecture for Genomic Segmentation Tasks

SegmentNT represents a powerful extension of the **Nucleotide Transformer** model by incorporating an additional **segmentation head**. This segmentation head is designed to improve performance on various **genomic downstream tasks** by leveraging a **UNet architecture**. While SegmentNT may not yet be as widely used as other DNA foundation models, its innovative architecture provides a unique approach to identifying and segmenting regions of interest within DNA sequences. Below, we explore the functionality of the UNet architecture in SegmentNT, its application in genomics, and the specific tasks it addresses.

UNet Architecture: A Deep Dive into Its Role in SegmentNT

The **UNet** is a specialized **convolutional neural network (CNN)** architecture commonly employed in tasks like **image segmentation**. It segments images by identifying regions of interest, and in the case of genomic data, it helps pinpoint specific regions within long DNA sequences. SegmentNT applies this UNet structure to the DNA domain, allowing it to process genomic data with high precision.

1. **Downsampling and Bottlenecking**: The UNet begins with a **downsampling process** in which the original DNA sequence, represented by a long sequence of nucleotides, is progressively **compressed**. This downsampling reduces the sequence length by half at each layer while increasing the number of **channels** (features) through convolutional layers. This gradual reduction reaches a **bottleneck** layer, where the DNA sequence is represented in a compact, lower-resolution format. The bottleneck allows the model to capture essential features with reduced noise, facilitating the extraction of high-level abstractions in the DNA sequence.
2. **Upsampling and Reconstruction**: After reaching the bottleneck, the UNet enters an **upsampling phase**, which involves reconstructing the sequence back to its original resolution. This step essentially mirrors the downsampling, gradually expanding the sequence length and decreasing the number of channels. The upsampling process allows the model to recover detailed spatial information while maintaining the simplified, denoised representation achieved in the bottleneck.
3. **Skip Connections**: An essential feature of the UNet is its **skip connections** (also known as residual connections). These connections enable the model to **retain high-resolution information** from the original input sequence by directly passing it to the corresponding upsampled layers. This design choice preserves essential spatial and contextual information that may be lost during downsampling, improving the model's ability to generate accurate segmentations. Skip connections also play a critical role in **denoising** tasks by smoothing out noise and enhancing the clarity of structural details.

In the context of SegmentNT, this UNet architecture enables the model to effectively segment genomic regions by reducing noise and maintaining high-resolution information across the sequence, which is crucial for identifying intricate patterns in DNA data.

Applications and Performance of SegmentNT

SegmentNT is adept at tackling a variety of genomic tasks, including **splice site prediction** and **identification of intronic and exonic regions**. Here's a closer look at its applications and how it outperforms traditional methods:

1. **Splice Site Prediction**: SegmentNT is highly effective in predicting **splice donors and acceptors** (splice sites where RNA splicing occurs). By integrating the UNet's segmentation capabilities, SegmentNT can accurately identify not only individual splice sites but also differentiate between true splice sites and nearby false positives. This precision in splice site prediction is crucial for accurate gene annotation and understanding alternative splicing.
2. **Intron and Exon Segmentation**: With accurate splice site predictions, SegmentNT goes a step further

to delineate entire **introns** and **exons**. Identifying these segments is more challenging than simply locating splice sites because it requires integrating multiple splice predictions across the sequence. This task is vital for gene structure annotation, as it reveals the functional and non-functional segments of genes.

3. **Prediction of Regulatory Elements**: SegmentNT extends its predictive capacity to recognize various regulatory elements, such as **polyadenylation (polyA) sites**, **enhancers**, and **promoters**. By combining the nucleotide-level embeddings from the Nucleotide Transformer with the segmentation accuracy of UNet, SegmentNT excels at identifying these elements, which play crucial roles in gene expression regulation. These predictions provide insights into how genes are controlled and contribute to a deeper understanding of complex gene regulatory networks.

Importance of Pre-Training in SegmentNT

The model's impressive performance across these tasks is largely attributed to the **pre-training phase** of the Nucleotide Transformer, where it undergoes a **masked prediction task** on a vast DNA corpus. During this pre-training, the model learns general DNA sequence features that are not limited to one specific genomic task. When these embeddings are used as input to the SegmentNT model, they carry rich biological information, enabling SegmentNT to achieve high accuracy across diverse tasks.

By comparing SegmentNT's performance with models trained from scratch (i.e., models not pre-trained on large DNA datasets), it becomes clear that the **pre-trained embeddings** significantly enhance prediction accuracy. This approach highlights the advantage of using **foundation models** as a base, followed by specialized training for particular downstream applications.

Summary

SegmentNT is an advanced genomic model that builds upon the Nucleotide Transformer by incorporating a **UNet-based segmentation head**. This model's architecture—characterized by downsampling, upsampling, and skip connections—enables it to capture detailed spatial information, making it especially suited for tasks requiring precise segmentation within DNA sequences. SegmentNT's applications range from splice site prediction to intron-exon identification and regulatory element recognition. It demonstrates that **combining Transformer embeddings with CNN-based segmentation** architectures can yield high-performing models capable of nuanced genomic analysis. SegmentNT's success underscores the value of pre-trained foundation models and the adaptability of UNet for high-resolution, task-specific segmentation, solidifying it as a promising tool for genome research and annotation.

55:25 Hyena Models: Efficient Long-Range Processing in DNA Sequence Analysis

The **Hyena models** represent an innovative approach in DNA sequence modeling, addressing the challenges of computational expense and scalability inherent in **Transformer-based models**. While Transformers have shown success in handling sequence data due to their **self-attention mechanisms**, their application to long DNA sequences remains computationally intensive. Hyena models overcome this by integrating aspects of **convolutional neural networks (CNNs)** with the **information flow properties** of **recurrent neural networks (RNNs)**, resulting in a model that efficiently handles single-nucleotide resolution across vast DNA sequences with significantly reduced computational demands.

Key Innovations in Hyena Models

1. **Single Nucleotide Resolution with Long-Range Dependencies**: Hyena models achieve **single nucleotide resolution** while maintaining the capacity to capture long-range dependencies. This is crucial for DNA sequence analysis, as functional elements may interact over substantial distances. Unlike Transformers, which face computational bottlenecks with sequence length due to their quadratic complexity, Hyena models achieve this with a **logarithmic complexity (L log L)**. Here, **L** represents the sequence length, making Hyena models substantially more efficient for large DNA sequences.
2. **Hybrid Architecture with Gated Convolutional Layers**: The architecture of Hyena combines the strength of **convolutional layers** with the **gating mechanisms** often seen in RNNs. This structure allows for sequential information processing without requiring each position in the sequence to explicitly attend to every other position. Instead, the model sequentially gates and processes information from each **convolutional stack**, enabling it to selectively pass along only the most relevant features. The convolutional layers capture local sequence patterns, while the gated structure allows for **controlled, long-range information flow** across large distances in the DNA sequence.
3. **Reduced Computational Complexity**: Transformers, with their attention-based mechanisms, face a quadratic computational complexity (L^2) with respect to sequence length, which becomes prohibitive

with long DNA sequences. Hyena's design reduces this to **L log L** by avoiding full pairwise attention between tokens, relying instead on a combination of local convolutional processing and selective gating. This streamlined complexity makes Hyena models both computationally efficient and scalable.

Visualization of Hyena's Embedding Space

The **latent space** created by the Hyena model provides a powerful view into the underlying structure of different gene types. Using **t-SNE** (t-distributed Stochastic Neighbor Embedding) visualizations, the latent space of Hyena models exhibits **tight clustering** for genes of similar function. When compared with other models, such as **DNABERT** and **Nucleotide Transformer**, Hyena's embeddings form **more homogeneous and distinct clusters**, indicating that genes with similar functional roles are positioned closely in the embedding space.

- **Comparison with DNABERT and Nucleotide Transformer**: In contrast to the more diffused clusters seen in DNABERT and Nucleotide Transformer embeddings, Hyena's latent space clusters gene types more distinctly. This tighter grouping suggests that Hyena captures more **specific, functionally relevant features** in its representations, producing embeddings that could offer clearer insights into gene functionality and regulatory elements.

EvoDNA: Building on Hyena's Architecture with Attention Mechanisms

EvoDNA is an extension of the Hyena model architecture, introduced as another **DNA foundation model** that builds on the efficient information processing principles of Hyena. EvoDNA incorporates **attention layers** alongside the **Hyena architecture's convolutional and gating mechanisms**. This hybrid model, referred to as a **striped Hyena model**, integrates **selective attention** with Hyena's efficient processing structure to enhance performance across a range of genomic tasks.

- **Enhanced Gating Structure**: While Hyena primarily relies on convolutional layers with gating, EvoDNA introduces **attention-based gating** in specific sections of its architecture. This setup allows EvoDNA to further refine which parts of the DNA sequence to emphasize, especially in complex tasks requiring more nuanced differentiation between nucleotide interactions.
- **Performance Across Genomic Tasks**: EvoDNA demonstrates strong performance across various genomic tasks, similar to SegmentNT, such as predicting **splice sites**, identifying **promoters**, and recognizing **enhancer regions**. By combining the computational efficiency of Hyena with targeted attention, EvoDNA excels in extracting functionally significant patterns in DNA sequences while remaining scalable for extensive datasets.

Summary

The **Hyena models** offer a significant advancement in genomic data processing by combining the strengths of CNNs, RNN-inspired gating mechanisms, and efficient convolutional operations. With their ability to process **single nucleotide resolution** across long sequences with **logarithmic computational complexity**, they serve as a scalable alternative to Transformers for DNA sequence analysis. The resulting latent space in Hyena models is both dense and functionally meaningful, enhancing the interpretability of gene types and regulatory elements. **EvoDNA** builds upon this foundation, adding attention layers that refine the model's focus on key DNA sequence features, thereby broadening the applicability of foundation models in genomic research. Together, Hyena and EvoDNA underscore the importance of hybrid architectures in pushing the boundaries of computational biology, making large-scale genomic analysis more feasible and insightful.

1:00:30 Borzoi Model for Gene Expression Prediction

The **Borzoi model** (referred to as Boro in some contexts) presents a unique approach to the challenging task of **predicting mRNA expression** across the genome. Unlike previous models focused primarily on **DNA sequence data**, Borzoi integrates a **large corpus of RNA expression data** with genomic sequences to predict **transcriptional activity** across various genomic regions. This capability provides valuable insight into **gene regulation** and **expression patterns**, addressing a crucial aspect of functional genomics.

Key Features and Structure of the Borzoi Model

1. **Prediction Across 32-Base Pair Windows**: The Borzoi model operates by dividing the genome into **32 base pair windows**. For each window, the model predicts **RNA-seq coverage**—essentially estimating the transcriptional output or **expression level** for that genomic region. This windowed approach enables the model to capture fine-grained variation in transcriptional activity across different genomic locations.
2. **UNet Architecture with Transformer Integration**: The model structure is fundamentally a **UNet**

architecture, a powerful framework widely used for segmentation tasks in **computer vision** and increasingly in genomics. The UNet architecture in Borzoi follows a typical **downscaling and upscaling** pipeline:

    - The input **high-resolution DNA sequence** is progressively downscaled, first analyzing sequences at **16 nucleotide (mer) resolution**, then further aggregating into chunks of **32, 64, and 128 base pairs**.
    - After the downscaling step, the sequence passes through several **Transformer blocks**. The Transformer layers allow the model to capture complex, long-range dependencies within the sequence, which is essential for understanding broader regulatory contexts in DNA.
    - Finally, the **upscaling process** reconstructs the sequence resolution back to the 32 base pairs, facilitating direct mRNA expression predictions for each window.

3. **Residual Skip Connections**: To retain essential high-resolution information, Borzoi employs **residual or skip connections**—a hallmark of UNet models. These connections link the downscaling and upscaling paths, enabling the model to incorporate both **fine-grained sequence details** and **higher-level abstractions**. This approach aids in predicting intricate transcriptional patterns by maintaining continuity across different resolutions.

Predictive Power and Insights Gained from Borzoi

The Borzoi model demonstrates impressive performance in predicting **mRNA expression patterns** across the genome. For example, visualizations of predicted expression profiles closely align with actual **RNA-seq data** for specific genes, such as **EGFR (Epidermal Growth Factor Receptor)**. This alignment suggests that the Borzoi model successfully integrates essential genomic features, including:

- **Chromatin Structure**: Borzoi appears to internalize aspects of chromatin accessibility, which influences transcriptional activity by modulating gene accessibility to transcriptional machinery.
- **Promoter and Enhancer Regions**: Through training, the model learns to recognize promoter sequences, enhancer regions, and other regulatory elements critical to transcription initiation.
- **RNA Splicing**: Remarkably, Borzoi can infer splicing patterns, a complex problem often requiring specific model designs, by predicting which exons and introns will be included in the final mRNA transcript.

Practical Applications and Future Use

Borzoi's ability to predict transcriptional activity genome-wide has substantial implications for **functional genomics**, regulatory **element discovery**, and understanding **gene expression mechanisms**. Additionally, the embeddings generated by Borzoi offer a **rich representation of genomic features**, making them valuable for further research and exploration in downstream tasks.

- **Access to Embeddings**: Researchers can use Borzoi embeddings, as generated across the entire human genome, for custom analyses. For instance, these embeddings can support research in gene regulatory mechanisms, assist in annotation of non-coding regions, or aid machine learning models targeting specific regulatory functions.

Borzoi's Contribution to Gene Expression Prediction

The Borzoi model exemplifies the **integration of deep learning with genomic data** to tackle the complexities of gene expression prediction. By combining **UNet architecture**, **Transformer blocks**, and skip connections, Borzoi brings a sophisticated approach to predicting RNA expression with high fidelity. This model not only demonstrates the capacity to generalize across various tissues and conditions but also offers a framework for addressing intricate regulatory tasks in genomics with **high-resolution, sequence-level accuracy**.

Future Directions: Transitioning to Small Molecule Drug Prediction

With Borzoi's success in expression prediction, the next steps in deep learning for genomics move towards **small molecule drug development**. The following sessions will cover **drug development principles** and introduce **tools for molecular analysis**, expanding from nucleotide-based models to molecular models, marking a pivotal transition in computational biology applications.

Lecture 13 - Drug Development Intro

Video: Lecture13 Drug Development Intro

Slides: Lecture13_IntroDrugDevelopment.pdf

0:00 What is a drug?

To effectively understand **drug development**, we must begin by defining what qualifies as a **drug** and how it fits into the broader framework of medical science. Today, drugs are substances that have undergone extensive testing and regulatory scrutiny to be approved for **therapeutic use**. Let's explore the technicalities of what defines a drug, its historical context, and the general pathway from idea to market.

Definition of a Drug

In technical terms, a **drug** is defined as a **substance recognized by an official pharmacopoeia** or formulary. It must meet stringent standards, detailing its **composition, production process, and purity levels**. Key aspects of this definition include:

- **Purpose**: A drug must be intended for **diagnosis, cure, mitigation, treatment, or prevention** of diseases. This purpose distinguishes it from other substances or compounds.
- **Regulatory Distinction**: Drugs are regulated separately from foods and medical devices. Although there are specialized categories such as **medical foods**, which are designed to address specific dietary needs in diseases, these are not classified as drugs. Additionally, medical devices, although integral to healthcare, follow separate regulations.

Drugs fall into two main categories:

1. **Small Molecule Drugs**: These are chemically synthesized compounds, traditionally the focus of **pharmaceutical research** and commonly taken in oral or injectable form.
2. **Biologics**: These are larger, more complex molecules produced by living cells or organisms, including **antibodies, vaccines, and gene therapies**. Biologics often require more sophisticated manufacturing processes and delivery systems.

The Role of Drugs in Human History

Drug use and development are deeply rooted in **human history**, with nearly every culture exploring medicinal substances. Historically, early drug discoveries often emerged from **natural sources**, especially plants, with traditional knowledge guiding usage and benefits. Some of today's standard medications have their origins in these ancient practices:

- **Poppy Extracts**: Used historically for pain relief, poppies led to the discovery of **opiates** (e.g., morphine and codeine), which remain central in pain management.
- **Willow Bark**: This was traditionally used to treat fever, and it later led to the isolation of **salicylic acid**, which served as the foundation for **aspirin**—one of the most widely used anti-inflammatory and pain-relieving drugs.
- **Artemisia (Sweet Wormwood)**: Recognized in Chinese medicine for treating fevers, it eventually provided the basis for **artemisinin**, a powerful antimalarial drug.

These examples highlight how **natural products** served as starting points for drug discovery, a trend that continues in the search for new drugs today. However, the modern approach to drug development has vastly evolved, characterized by complex scientific, regulatory, and technological frameworks.

The Modern Framework for Drug Discovery and Development

The modern drug development process is a **multifaceted, highly regulated** pathway involving stages from **initial discovery** to **clinical testing** and **market approval**. Unlike the traditional trial-and-error methods, today's drug discovery is driven by rigorous **scientific methodologies** and **technological advancements** in chemistry, biology, and computational methods. This contemporary framework ensures that drugs are not only effective but also safe for human use.

While we will delve deeper into **machine learning** and **deep learning** contributions to this process in future discussions, understanding the **holistic journey of a drug**—from conceptualization to commercialization—is essential for appreciating the complexities and challenges involved.

This foundation sets the stage for exploring **how modern tools, including AI**, are revolutionizing drug discovery and enabling **new therapeutics** that address unmet medical needs more efficiently than ever before.

3:00 Modern drug discovery framework

The **modern drug discovery framework** is a highly structured, intentional, and interdisciplinary process that aims to transition an **initial scientific insight** into a **commercially viable drug**. Unlike traditional methods rooted in serendipity or simple trial and error, today's drug discovery is based on a well-defined framework

driven by our understanding of **molecular pathways**, **disease mechanisms**, and **targeted therapeutic design**.

From Basic Science to Therapeutic Agent

The process of drug discovery begins with a **basic scientific idea** about a molecular pathway that plays a role in disease. Researchers identify a **disease target**—a protein, gene, or cellular mechanism crucial to the disease's pathology. By understanding how this target operates, they can begin a **deliberate search for therapeutic agents** designed to interact with it, modifying or inhibiting its function to produce a beneficial effect. These therapeutic agents can range from **small molecules** to **biologics** like antibodies or even natural products with bioactivity against the target pathway.

Interdisciplinary Nature of Drug Discovery

This process is profoundly **interdisciplinary**, involving contributions from:

- **Biologists and Chemists**: Essential for understanding the disease and designing molecules with the desired properties.
- **Engineers and Bioinformaticians**: Play roles in optimizing delivery systems and computational models that predict drug behavior.
- **Computational Biologists**: Use bioinformatics to understand complex biological data, select promising targets, and analyze molecular interactions.
- **Business and Legal Experts**: Ensure the process aligns with regulatory standards and is commercially viable, navigating intellectual property (IP) and market strategy.

The collaborative nature of modern drug discovery requires **thousands of people** and can cost **up to or over a billion dollars** for a single drug. This high cost is partly due to stringent standards set by **regulatory agencies like the FDA**, which require extensive evidence demonstrating both **efficacy** and **safety** of a potential drug. These standards help ensure patient safety but significantly raise the **time, complexity, and cost** of the development process.

The Role of Business Models in Drug Development

Pharmaceutical companies are motivated to invest in drug development because of the **high potential for profit** from patented drugs. This incentive structure is designed to promote innovation, but it can sometimes **conflict with therapeutic needs**. The ideal case aligns **therapeutic need** and **market incentives**; however, this alignment doesn’t always hold, especially when the business model may not support development in areas with limited commercial appeal.

A prominent example is the **development of antibiotics**. The **rise of antimicrobial resistance** has led to an urgent need for new antibiotics, yet the economic incentives for producing them are weak. For instance, if a pharmaceutical company develops a new antibiotic that effectively combats resistant bacteria, the optimal usage scenario may involve **restricting its distribution** to preserve its efficacy and reduce the likelihood of resistance emerging. However, **limited usage reduces sales**, making such a product less profitable than drugs that require regular, sustained use. As a result, there has been **minimal investment in novel antibiotics** over recent decades despite their critical importance for global health.

Aligning Business and Therapeutic Needs: New Models

This misalignment between **business models and public health needs** in cases like antibiotics has prompted researchers and policymakers to explore alternative models. These new models may involve:

- **Government Funding and Subsidies**: To offset development costs and encourage research even in areas with low commercial returns.
- **Subscription Models**: Where healthcare systems pay for access to antibiotics, regardless of how often they are prescribed, encouraging companies to invest in antibiotics without needing high volume sales.
- **Patent Extensions and Market Exclusivity**: To provide longer periods of exclusivity and thus greater potential for return on investment.

The **modern drug discovery framework** is thus as much a **scientific and technological enterprise** as it is a **regulatory and economic challenge**. In upcoming sections, we will explore how advances in **machine learning** and **deep learning** are enabling more efficient, targeted, and potentially less costly ways to design new drugs, opening doors to treat diseases with unmet therapeutic needs while reshaping the economics of drug development.

7:00 Types of drugs

In modern medicine, drugs can be broadly classified into **small molecule drugs** and **biologics**, each with distinct characteristics, manufacturing requirements, and regulatory pathways. Beyond these primary categories, there are also innovative drug classes such as **live biotherapeutic agents** and **cell-based therapies** that represent emerging frontiers in therapeutic development.

Small Molecule Drugs

Small molecule drugs are **low molecular weight compounds** (typically under 500 Dalton) that are generally synthesized through chemical processes. Often, these drugs are optimized derivatives of natural compounds extracted from various sources, including plants. Their **small size** allows them to diffuse across cell membranes, making them effective for targeting intracellular proteins. Small molecule drugs usually bind to a specific protein target, inhibiting or modifying its function to achieve therapeutic effects.

Characteristics:

- **Synthesis**: Typically produced synthetically after initial natural product identification.
- **Administration**: Predominantly oral, though other routes like topical or injectable are possible in certain cases.
- **Examples**: Aspirin, metformin, and statins.
- **Advantages**: Ease of production, stability (often stored as a dry powder), and long shelf-life, usually without refrigeration.
- **Disadvantages**: Small molecule drugs can sometimes produce **toxic metabolites** upon degradation, potentially leading to side effects.

Biologics

Biologics represent a class of drugs derived from **biological sources** and generally include **larger molecular structures**, such as proteins, antibodies, and other cellular products. Unlike small molecules, biologics are primarily administered via **injection or intravenous (IV)** due to their complex structure, which would be broken down in the digestive system if taken orally.

Characteristics:

- **Production**: Derived through **biological processes** such as fermentation or cell culture.
- **Administration**: Typically non-oral, requiring IV or injection.
- **Examples**: Herceptin for breast cancer and insulin for diabetes.
- **Advantages**: High specificity due to targeted protein binding, potentially resulting in fewer side effects.
- **Disadvantages**: Sensitive to environmental conditions (often requiring refrigeration), expensive production, and administration challenges as they often require medical supervision.

Live Biotherapeutic Agents

Live biotherapeutic agents represent an exciting class of **biologic drugs** that utilize **living organisms** (often bacteria) as therapeutic agents. This field, closely linked to **microbiome research**, aims to exploit beneficial bacteria that may influence health positively. For example, specific bacteria found in the guts of healthy individuals may be absent in individuals with certain diseases, leading researchers to hypothesize that restoring these bacteria could have therapeutic benefits.

Characteristics:

- **Administration**: Often as capsules containing live bacteria meant to colonize the patient's gut.
- **Advantages**: Leveraging naturally occurring organisms, these agents potentially have a lower risk of toxicity.
- **Disadvantages**: Complex to standardize and regulate due to variability in how they colonize and interact with each patient's microbiome. Dosing can be challenging since colonization may lead to bacterial growth over time, creating discrepancies between initial and effective dose levels.

Live biotherapeutic agents have shown promise in treating recurrent **Clostridium difficile infections** through **fecal microbiota transplantation**. Although microbiome-based therapeutics are still in early stages, they represent a promising avenue for precision and personalized medicine.

Cell-Based Therapies

Cell-based therapies involve the use of **live cells**, often derived from the patient or a donor, to treat diseases.

Examples include **CAR T-cell therapy** for cancer and **stem cell therapies** for tissue regeneration. These therapies leverage the innate properties of living cells to target disease mechanisms with high specificity.

Characteristics:

- **Administration**: Requires precise injection, often into specific tissues or through IV.
- **Advantages**: High personalization potential, especially in targeting specific cells like cancer cells.
- **Disadvantages**: Complex to manufacture, standardize, and regulate due to the need for living cells. The logistics of cell therapies often require specialized storage, transport, and clinical administration.

The field of cell-based therapies is rapidly expanding, driven by advancements in genetic engineering, cell culture techniques, and personalized medicine approaches. However, regulatory challenges remain due to the complexities inherent in using live human cells.

Companion Diagnostics

While not a type of drug, **companion diagnostics** represent a significant advancement in **personalized medicine** by allowing healthcare providers to tailor treatment based on individual patient profiles. This approach can optimize drug efficacy by identifying patients who are most likely to benefit from specific treatments.

Concept:

- **Purpose**: Companion diagnostics are used to screen patients for specific genetic, molecular, or cellular markers that predict a drug's effectiveness or likelihood of adverse effects.
- **Examples**: HER2 testing in breast cancer to determine the suitability of Herceptin, and PD-L1 expression testing for immunotherapy responsiveness.
- **Advantages**: This approach allows for targeted treatment, potentially reducing adverse effects and improving outcomes by matching the right drug to the right patient group.

By helping to identify patient subpopulations within broader disease categories, companion diagnostics can sometimes **revive drugs that might otherwise fail clinical trials** due to heterogeneous patient responses. These diagnostics are instrumental in advancing precision medicine, especially in complex diseases where patient populations may have varied underlying molecular mechanisms.

Emerging and Specialized Drug Types

1. **Phage Therapy**: A biologic approach using bacteriophages (viruses that target bacteria) to treat bacterial infections, particularly useful against antibiotic-resistant bacteria. Phage therapy is still experimental and faces unique regulatory hurdles.
2. **Vaccines**: Although traditionally focused on preventing infectious diseases, recent advancements are exploring vaccines in **cancer immunotherapy** and **autoimmune conditions**, where the immune system's activity is modulated to treat or prevent disease.

These various **types of drugs and therapies** underscore the diversity in modern treatment options. Each class comes with unique **advantages, disadvantages, and challenges**, from manufacturing and storage to administration and regulatory approval. In the coming sections, we will delve deeper into the **drug development process**, focusing on **small molecule drugs** and exploring how machine learning and deep learning are transforming discovery, testing, and optimization in this field.

23:27 Pharmacokinetics (PK): How the Body Processes Drugs

Pharmacokinetics (PK) is a core concept in drug development that examines what the **body does to a drug** after it is administered. In contrast, **pharmacodynamics (PD)** focuses on what the **drug does to the body**. PK is crucial in determining **dosing schedules, administration routes, bioavailability,** and **potential side effects**. The study of PK involves understanding four main phases: **absorption, distribution, metabolism,** and **excretion** (often abbreviated as **ADME**).

Absorption

Absorption is the process through which a drug **enters the bloodstream** or its **site of action**. Factors influencing absorption include:

- **Route of administration** (e.g., oral, intravenous, or inhalation), with some routes allowing for faster or more direct absorption.
- **Chemical properties** of the drug, such as hydrophilicity (water solubility) or hydrophobicity (fat solubility).

- **Environmental conditions** like blood flow and pH at the absorption site, which can enhance or inhibit the drug's ability to cross cellular membranes.

The extent to which a drug is absorbed effectively determines its **bioavailability**, or the fraction of the drug that reaches systemic circulation in an active form.

Distribution

Distribution describes the movement of the drug from the bloodstream into **various tissues** and **compartments** within the body. Key considerations include:

- **Blood flow** to different organs and tissues, as areas with high blood flow (e.g., heart, liver) typically receive drugs more quickly.
- **Protein binding**, where drugs may bind to plasma proteins in the blood, limiting the amount of free drug available to exert a therapeutic effect.
- The ability to cross **selective barriers**, such as the **blood-brain barrier**.

A critical metric in distribution is the **volume of distribution (Vd)**, which reflects how extensively a drug disperses throughout the body relative to its concentration in the blood.

The Blood-Brain Barrier

The **blood-brain barrier (BBB)** is a selective, semi-permeable barrier that separates circulating blood from the brain's extracellular fluid. This **tight junction** of endothelial cells allows only specific molecules to pass through, protecting the brain from potentially harmful substances. The BBB is particularly impermeable to **large or hydrophilic molecules**, creating a challenge for delivering drugs aimed at treating central nervous system disorders.

Strategies to overcome the BBB include:

- **Designing drugs** with properties that increase BBB permeability.
- **Nanoparticle-based delivery systems** to shuttle drugs across the barrier.
- **Intranasal administration**, which allows drugs to bypass the BBB by directly accessing nerve pathways in the nasal cavity.

Metabolism

Metabolism transforms drugs into more water-soluble forms for easier excretion. This occurs mainly in the **liver** through two phases:

1. **Modification (Phase I)**: Involves enzymes like the **cytochrome P450** family, which oxidize or reduce the drug, making it more reactive.
2. **Conjugation (Phase II)**: Adds a polar group (e.g., glucuronidation) to the drug, increasing its solubility and facilitating excretion.

Metabolism can **activate** drugs, convert them into **inactive metabolites**, or sometimes create **toxic intermediates**. For instance, acetaminophen is metabolized to a toxic intermediate that can cause liver damage, especially in individuals with high alcohol intake who upregulate specific liver enzymes, enhancing this toxic pathway.

Excretion

Excretion is the process of removing drugs and their metabolites from the body. This typically occurs via:

- **Kidneys (urine)**: The most common route for excreting water-soluble metabolites.
- **Liver (bile)**: Some metabolites are secreted into bile and eventually excreted in feces.
- **Other routes**: Sweat, breath, and breast milk also facilitate drug excretion, though to a lesser extent.

The efficiency of excretion impacts the drug's **half-life**, which is crucial in determining the **dosing frequency** to maintain therapeutic levels without causing toxicity.

Modeling Pharmacokinetics

PK models help predict how drugs behave in the body, often using **compartmental models**:

- **One-compartment model**: Simplifies drug behavior to a single space (e.g., bloodstream), with drug levels decaying over time as it is metabolized and excreted.
- **Two-compartment model**: Accounts for an additional compartment, such as tissues or organs, where the drug can move temporarily before returning to the bloodstream for excretion.

These models inform dosing schedules by predicting **peak concentrations (Cmax)**, **minimum concentrations (Cmin)**, and **half-life**, which represents the time required for the drug's concentration in the bloodstream to reduce by half.

Practical Considerations in Pharmacokinetics

1. **Absorption limitations**: Poor absorption leads to low bioavailability, necessitating higher doses, which may increase side effects.
2. **Tissue accumulation**: Some drugs may accumulate in specific tissues (e.g., lipophilic drugs in fat), influencing their effectiveness and safety.
3. **Toxic metabolites**: Certain drugs can metabolize into harmful byproducts, making it crucial to monitor specific pathways, especially in patients with liver or kidney impairments.
4. **Patient variability**: Age, health status, and concurrent medications can significantly alter PK properties, necessitating personalized dosing strategies.

Conclusion

Pharmacokinetics is foundational in **drug development and therapeutic management**, determining how often, in what quantity, and by which route a drug should be administered to maximize its therapeutic benefits while minimizing potential risks. Understanding PK enables researchers and clinicians to **optimize drug design and delivery** to better meet patient needs, paving the way for safer, more effective treatments.

42:00 Pharmacodynamics (PD): What Drugs Do to the Body

Pharmacodynamics (PD) investigates the effects drugs have on the body, examining the **mechanisms of drug action, dose-response relationships,** and **therapeutic vs. toxic effects**. PD focuses on how drugs interact with **receptors**, **enzymes**, or **other cellular targets** to produce a biological response. A detailed understanding of PD enables the design of drugs with **maximal therapeutic effects** while minimizing **adverse side effects**.

Drug-Receptor Interactions

Most **small molecule drugs** exert their effects by **binding to specific receptors**—typically **proteins** involved in cellular signaling pathways or enzymatic processes. Key types of drug interactions include:

- **Agonists**, which activate their target receptors to produce a biological response.
- **Antagonists**, which bind to receptors but **block or inhibit** their activation, effectively dampening the response.
- **Partial agonists**, which bind to and activate receptors but produce a **weaker response** than full agonists.

One essential class of receptors is **G protein-coupled receptors (GPCRs)**, highly "druggable" due to their role in a broad range of physiological processes. Other common targets include **ion channels** and **enzymes**.

Affinity and Potency

The **affinity** of a drug for its receptor represents the **strength of the interaction** between the two. A high-affinity drug binds more readily, leading to stronger or more sustained effects. **Potency** is often characterized by the **effective concentration (EC50)**—the concentration at which the drug achieves **50% of its maximal effect**.

The **therapeutic effect** of a drug is often visualized through **dose-response curves**, which show the relationship between dose and the resulting biological effect. Key terms include:

- **Emax**: The maximum possible effect a drug can produce.
- **Therapeutic window**: The range of doses that produce a therapeutic effect without causing toxicity.

Mechanism of Action (MOA)

A drug's **mechanism of action (MOA)** describes how it exerts its effects at the molecular level. For example:

- **Beta blockers** are drugs that **inhibit beta-adrenergic receptors** involved in the body's "fight-or-flight" response. They are used in cardiovascular medicine to slow heart rate and reduce blood pressure.
- Some beta blockers are highly **selective** for particular receptor subtypes, while others have broader effects, impacting other receptors and leading to **side effects**.

Understanding a drug's MOA is essential in predicting its **therapeutic effects** and **side effects**. Drugs targeting **enzymes** often block enzymatic activity, whereas drugs targeting **receptors** may initiate complex

signaling cascades, which can have broader, context-dependent effects based on other molecules in the pathway.

Dose-Response Relationships and Therapeutic Window

Dose-response curves illustrate how increasing drug doses correlate with biological effects. At higher concentrations, the drug **saturates its target receptors**, meaning additional doses will not increase the therapeutic effect but may heighten **toxicity**.

The **therapeutic window** is the range between an effective dose and a dose that causes toxicity. A narrow therapeutic window means careful dosing is essential, as even small increases in dose can lead to adverse effects.

Tolerance and Sensitization

Drugs often exhibit changes in effectiveness over time:

- **Tolerance**: The body adapts, requiring **higher doses** to achieve the same effect. This adaptation may involve **increased drug metabolism** or **downregulation** of receptors.
- **Sensitization**: In rare cases, repeated drug exposure **enhances responsiveness** to the drug, often through **upregulation of receptors** or **increased pathway sensitivity**.

Understanding these dynamics is critical in managing **long-term treatment regimens** and reducing **risks of dependency**.

Toxicology

Toxicology focuses on the **adverse effects** of drugs, studying both **acute toxicity** (effects from a single dose) and **chronic toxicity** (effects from repeated exposure). Toxicity can arise from **off-target effects**, where the drug binds to unintended receptors, or from the **metabolic byproducts** of the drug, which may be toxic. For example:

- **Acetaminophen (Tylenol)** is safe at low doses but can produce **toxic metabolites** in high doses, especially in individuals with compromised liver function.

The balance of **therapeutic effects** and **toxicity** is fundamental in pharmacodynamics, shaping decisions on **dosage**, **frequency**, and **administration routes** for safe and effective drug use.

Conclusion

Pharmacodynamics provides a detailed map of **how drugs exert their effects** on the body, from binding interactions to downstream signaling pathways and physiological outcomes. Understanding PD is crucial for optimizing **efficacy**, **selectivity**, and **safety** in drug development, ensuring that therapeutic goals are met while minimizing adverse reactions. This depth of insight allows for the development of precision therapies tailored to individual patient needs and more effective management of diverse medical conditions.

49:39 Toxicology: Assessing the Safety and Side Effects of Drugs

Toxicology is the study of a drug's **potential adverse effects** and is crucial for determining its **safety profile** before it can be approved for use. This field assesses the **risk of toxicity** across various systems and scenarios, ensuring that the drug achieves therapeutic goals without posing unacceptable risks to health.

Types of Toxicology Studies

Toxicology studies assess multiple dimensions of drug safety:

- **Acute Toxicity**: Observes immediate reactions following drug administration. These studies help identify **early side effects** or **toxic responses**.
- **DNA Damage**: Tests if a drug can cause **genotoxic effects**, potentially leading to mutations or cancer.
- **Reproductive Health**: Evaluates effects on **fertility, fetal development,** and the overall health of offspring, critical for drugs intended for reproductive-age populations.
- **Carcinogenicity**: Studies if long-term exposure to the drug increases the risk of **cancer**, which is a costly and time-intensive process, often involving **extended observation periods**.

Additionally, **organ-specific toxicity** is examined, targeting essential systems like the **central nervous system, respiratory system, and cardiovascular system**. Such studies are often performed in **animal models** before human trials to ensure that no severe organ damage occurs.

Therapeutic Index and Its Importance

The **therapeutic index (TI)** is a key measure of a drug's safety. It reflects the **gap between the effective dose** (where the drug has a therapeutic effect) and the **toxic dose** (where adverse effects appear):

- **High TI**: Indicates a wide margin between therapeutic and toxic doses, allowing for safer administration without close monitoring.
- **Low TI**: Shows a narrow margin, making dosing challenging, as small variations can lead to toxicity.

For example, **warfarin**, a blood thinner, has a narrow TI. Too low a dose renders it ineffective, while too high a dose can cause dangerous bleeding. The **microbiome** can further complicate dosing by **metabolizing warfarin differently** in each individual, introducing variability that makes accurate dosing even harder.

Drug Interactions

Drugs can interact in ways that enhance or diminish their effects:

- **Synergistic Interactions**: Two drugs combined may produce a stronger effect than either alone. For instance, **alcohol** and **sedatives** both depress the central nervous system. When taken together, they can produce a profound, potentially dangerous sedative effect.
- **Antagonistic Interactions**: One drug may counteract or block the action of another. **Naloxone**, for example, binds to **opioid receptors**, effectively blocking opioid drugs from binding, which is why it's used as an antidote for opioid overdose.
- **Additive Effects**: Some drugs combine in an additive way, where their effects are simply the sum of each drug's individual effects. For instance, **antihypertensive drugs** used together may collectively lower blood pressure more effectively, but this requires careful balancing to avoid excessive drops.

Conclusion

Toxicology provides a comprehensive view of the potential risks associated with drug use, from acute responses to long-term effects on genetic stability and organ health. A strong **therapeutic index** and careful **assessment of drug interactions** are central to ensuring a drug's safety in diverse patient populations. Through these studies, toxicology helps ensure that medications provide the maximum therapeutic benefit with the least risk, guiding their safe and effective use in clinical settings.

54:15 Steps in the drug development process

The drug development process is a **highly structured, multi-phase endeavor** that begins with a promising idea for **targeting a disease** and ends—if successful—with a **marketable therapeutic**. This pathway involves intense **scientific validation, optimization of candidate compounds**, and extensive testing for safety and efficacy. Let's break down each of the stages involved:

1. Target Identification

The first step is **identifying a target**—typically a protein or gene that plays a central role in the disease's mechanism. **Target selection** is rooted in **bioinformatics, genomics, and proteomics**, often using high-throughput screening or bioinformatic tools. Key considerations include:

- **Disease connection**: How integral is this target to the disease?
- **Druggability**: Can a small molecule or biologic agent effectively bind to this target?
- **Structural availability**: Are there structural data or predictions (e.g., using AlphaFold) that can guide binding predictions?

Targets generally include proteins like **enzymes, receptors**, or signaling proteins essential to the disease mechanism. Some targets, however, may be challenging due to their **lack of stable binding sites**, as with disordered proteins or large, flat surfaces on transcription factors.

2. Target Validation

After identifying a target, **target validation** ensures it plays a causative role in the disease. Techniques for validation include:

- **CRISPR screens** and **overexpression studies** to manipulate gene activity and observe outcomes.
- **Animal models** that replicate disease states.
- **Biomarker analysis** in human samples, offering early indications of therapeutic efficacy.

Target validation is critical to confirm that modulating this target will indeed produce a **therapeutic effect**, as opposed to targeting an unrelated or ineffective component of the disease mechanism.

3. Hit Identification

With a validated target, the next step is **identifying potential hit compounds** that interact with the target. Methods include:

- **High-throughput screening (HTS)**: Screens a vast chemical library using biochemical or cell-based assays to find initial compounds showing activity.
- **Fragment-based screening**: Involves smaller chemical fragments that bind weakly at high concentrations, which can then be expanded into more potent compounds.
- **Virtual screening**: Uses computational methods to predict compounds likely to bind effectively based on the target's structure, leveraging tools from molecular docking to machine learning.

Hit identification is a winnowing process, often beginning with **millions of compounds**, from which a subset with desirable binding properties is selected.

4. Hit to Lead Optimization

Once a set of hits has been identified, the next stage is **hit to lead** optimization. Here, chemists refine **chemical structures** and study their **structure-activity relationships (SAR)** to improve binding affinity, specificity, and pharmacokinetics:

- **Structural modifications** are iteratively applied and tested to increase efficacy, minimize off-target effects, and improve the compound's stability and solubility.
- **Initial pharmacokinetic assessments** are conducted to assess absorption, distribution, metabolism, and excretion (ADME) characteristics.

This step transitions promising hits into **lead compounds** that have undergone preliminary improvements and are ready for more focused development.

5. Lead Optimization

The lead compounds are now ready for **extensive refinement**. This stage is where **medicinal chemistry** and **biophysical studies** intensify, optimizing compounds for:

- **Increased binding affinity** to the target, enhancing efficacy.
- **Reduced off-target effects**, minimizing potential toxicity.
- **Optimal pharmacokinetic properties**, including half-life and bioavailability.
- **Efficacy in disease models**, ensuring the therapeutic effects seen in early tests translate effectively in more complex models.

Lead optimization typically reduces the number of candidate compounds to a **handful of leads** that balance therapeutic efficacy with acceptable safety profiles, setting the stage for preclinical and clinical studies.

Key Concepts in Drug Targeting

- **Structure-Activity Relationship (SAR)**: The relationship between a molecule's structure and its biological activity, guiding chemical modifications to enhance activity or reduce side effects.
- **Drugability**: The potential for a target to be modulated by a drug, influenced by factors like the presence of a well-defined binding pocket.
- **High-Throughput Screening (HTS)**: Allows the rapid testing of large libraries of compounds against a target to find initial hits, using automated systems.

Conclusion

The drug development process is **incremental and multi-faceted**, involving deep **biological validation, chemical optimization**, and early pharmacological testing. Each stage refines and narrows the field of candidates, moving from thousands of possibilities to a few carefully engineered compounds. By the time a candidate emerges for clinical trials, it has been rigorously tested and optimized, ensuring that it has the best possible chance of success in treating the targeted disease.

1:05:18 Structure-Activity Relationships (SAR)

The study of **structure-activity relationships (SAR)** is foundational in drug development, guiding the optimization of lead compounds to enhance **efficacy, specificity, and safety**. SAR is the study of how a drug's **chemical structure** influences its **biological activity**, allowing researchers to pinpoint and modify parts of the molecule to achieve desired properties.

Key Goals in SAR Analysis

1. **Identify essential structural features** necessary for biological activity.
2. **Enhance potency** by making modifications that improve binding affinity to the target.

3. **Reduce off-target effects** to increase specificity and reduce potential side effects.
4. **Minimize toxicity** by identifying structural components that contribute to harmful effects.
5. **Optimize pharmacokinetics (PK)** properties, including absorption, distribution, metabolism, and excretion (ADME).

The SAR Process

The SAR process is iterative and data-driven, involving the following steps:

- **Starting with Lead Compounds**: Researchers begin with a promising lead compound identified in earlier stages of drug development. The lead's **interaction with its target** is examined, ideally using structural data such as **crystal structures** of the target protein bound to the compound.
- **Predicting Modifications**: Based on the initial structure, researchers hypothesize how chemical modifications might affect binding, activity, and other properties. This might include changing side chains, adding or removing rings, or modifying bond types.
- **Empirical Testing**: Modified compounds are synthesized and tested to observe changes in activity, potency, and selectivity. Over time, the accumulation of SAR data allows researchers to refine their predictions and develop an increasingly accurate understanding of how structural changes impact the molecule's properties.

Common Strategies for Structural Modifications

To enhance the drug's efficacy, specificity, and stability, a range of structural modifications can be applied:

- **Alkyl Chain Variations**: Adjusting the length and branching of alkyl chains can impact how the molecule fits within the binding site and affects interactions.
- **Linker Length Adjustments**: Changing the length and flexibility of linkers between parts of the molecule can influence its orientation and binding.
- **Incorporation of Rings and Double Bonds**: Introducing rings and double bonds can reduce the molecule's **rotational freedom**. By restricting conformations, these modifications reduce **entropy loss** upon binding, enhancing the compound's binding affinity.
- **Use of Conformational Constraints**: Adding structural constraints, such as ring systems, can lock the molecule into preferred conformations, improving target binding.

Quantitative SAR (QSAR) and Computational Methods

Quantitative Structure-Activity Relationship (QSAR) methods use **statistical and computational tools** to model the relationship between structural properties and biological activity quantitatively. QSAR models often involve machine learning and deep learning algorithms to predict a compound's **biological activity** based on its structural features.

These models can significantly speed up the SAR process by:

- **Predicting toxicological properties** before synthesis, thereby identifying compounds with higher safety profiles.
- **Screening for activity** across large virtual libraries, identifying promising compounds without the need for extensive experimental testing.

Challenges in SAR

Despite advances, several challenges remain in SAR analysis:

1. **Balancing Multiple Properties**: Modifications to improve one aspect, such as binding affinity, may negatively impact others, such as PK properties or toxicity.
2. **Off-Target Effects**: A compound may interact with unintended proteins, leading to adverse effects. **Kinase inhibitors**, for example, often show broad activity, which can result in off-target interactions.
3. **Complexity of Biological Systems**: Biological systems are complex and interconnected, meaning changes to structure can have unpredictable downstream effects.

Conclusion

SAR is a central component of the drug development process, transforming initial hits into highly refined lead compounds ready for preclinical evaluation. By integrating **empirical testing, computational modeling**, and **quantitative SAR (QSAR)**, researchers can systematically optimize compounds to achieve maximal efficacy with minimal adverse effects. The SAR process is one of continuous learning and refinement, balancing

various chemical and biological factors to develop safe, effective therapeutic agents.

1:10:17 Drug development timeline

The **drug development timeline** is extensive and rigorous, reflecting the high standards set to ensure both efficacy and safety. The journey from initial **drug discovery** to an approved product on the market often spans **decades** and involves multiple distinct phases, each with significant hurdles and regulatory oversight.

1. **Drug Discovery Phase**:
    - This initial stage can take years or even decades, involving **target identification**, **hit discovery**, and early **lead optimization**.
    - Only a fraction of the initial candidates move forward, with many potential compounds failing due to lack of efficacy or issues with safety and drugability.
2. **Preclinical Testing**:
    - Once promising leads are identified, they undergo preclinical testing in **cell cultures** and **animal models**. This stage evaluates basic **efficacy, toxicity, pharmacokinetics (PK)**, and **pharmacodynamics (PD)**.
    - At this stage, scientists gather data to support the **Investigational New Drug (IND) application** required by regulatory bodies like the **FDA**.
3. **IND Application**:
    - Submitting an IND application to the FDA is a critical step before clinical trials can begin. This comprehensive document includes **safety data**, **manufacturing information**, and the proposed **clinical trial plan**.
    - If the FDA does not object within a specified timeframe, the drug sponsor can proceed to **Phase I clinical trials**.

Clinical Trial Phases

The clinical trials are divided into **three main phases**, each designed to answer specific questions about the drug.

1. **Phase I – Safety and Dosage**:
    - Conducted in a small group of **20-80 healthy volunteers** or patients, Phase I focuses on **safety** and determining an appropriate **dosage range**.
    - The goal is to assess the **initial PK/PD** profile and identify any adverse effects. Dosing begins very low and is increased gradually to observe tolerability and PK characteristics.
2. **Phase II – Efficacy and Side Effects**:
    - Phase II involves a larger group of **a few dozen to several hundred patients** who have the disease or condition being targeted.
    - This phase refines dosing regimens and looks for **preliminary signs of efficacy** while continuing to monitor safety. Companies may test various doses and schedules to determine an optimal dosing strategy.
    - While still focused on safety, Phase II trials also provide initial efficacy data, crucial for deciding whether to invest further in costly Phase III trials.
3. **Phase III – Large-Scale Testing**:
    - Phase III trials are **large-scale studies** involving hundreds to thousands of patients, providing the data required for **regulatory approval**.
    - These studies confirm efficacy, monitor side effects, and collect comprehensive safety data across a broader, more diverse patient population.
    - Due to the scale of these trials, Phase III is costly and time-intensive, but it provides the critical evidence needed for the final regulatory decision.

Post-Market Surveillance (Phase IV)

After approval, **Phase IV studies** or **post-market surveillance** continues to monitor the drug's performance in the general population. This phase assesses **long-term safety** and may reveal rare side effects not observed during clinical trials.

Probability of Success and Attrition Rates

- **Attrition is high** throughout this process. Notably, only about **14% of drugs** entering clinical trials are ultimately approved.
- Oncology drugs, for instance, have an especially low success rate, with an estimated **3% success** from Phase I through approval. The high risk reflects the complexity of cancer biology and the stringent requirements for oncology treatments.

FDA and Regulatory Standards

The **FDA** and other regulatory agencies play a central role in drug development, overseeing each phase and setting the standards for **safety, efficacy, and quality**. The agency enforces regulations covering everything from the design of clinical trials to the manufacturing standards required to ensure drug consistency.

Financial Implications

- Drug development is incredibly costly, with estimates around **$1 billion** to bring a single drug to market. These costs include not only successful drugs but also the cumulative expenses of failed trials.
- Additionally, the FDA approval process itself involves substantial industry fees. For example, fees are associated with IND and New Drug Application (NDA) filings, which fund part of the regulatory review process.

Types of FDA-Approved Drugs

The types of drugs approved by the FDA vary widely, with a significant focus on **oncology, psychiatry, and infectious diseases**. However, many approved drugs are modifications or improvements of existing therapies rather than entirely new molecular entities.

In summary, drug development is a **complex, multi-phase process** demanding significant time, resources, and regulatory compliance. The rigorous timeline and high attrition rate highlight the difficulty and expense of developing safe and effective treatments, underscoring the role of precision and innovation—particularly the potential of **AI and machine learning**—in improving this process. In subsequent discussions, we will explore how modern computational methods can help streamline and potentially shorten the drug development timeline, improve the likelihood of success, and reduce overall costs.

Lecture 14 - Chemistry GNNs

Video: Lecture14 Chemistry GNNs

Slides: Lecture14_Intro_Small_Molecules.pdf

00:00 Common molecular representations

In computational chemistry, **molecular representations** play a foundational role, enabling chemists and data scientists to encode complex chemical structures in ways that allow for machine parsing, data storage, and computation. These representations vary significantly in their format, ease of use, and applications, each having unique advantages and limitations for specific types of analyses.

1. SMILES (Simplified Molecular Input Line Entry System)

- **Overview**: SMILES is a string-based representation, originally developed by the EPA, that captures the structure of molecules in a compact, linear format. It has become one of the most widely used representations for small molecules in cheminformatics.
- **Format**: For example, **ethanol** is represented simply as `CCO` (omitting hydrogens for simplicity). The letters and numbers denote atoms and bond types (single, double), while parentheses indicate branching.
- **Advantages**: SMILES is highly compact, easily read by most cheminformatics software, and suitable for database storage.
- **Limitations**:
    - **Synthetic Ambiguity**: SMILES strings are flexible and easy to generate, but this flexibility means that many syntactically valid SMILES strings cannot be decoded into actual molecules.
    - **Stereochemistry**: While SMILES can include stereochemical information, it is often omitted in practice, which can be problematic when stereochemistry is critical to the molecule's function.

2. SMARTS (SMILES Arbitrary Target Specification)

- **Overview**: Building on SMILES, SMARTS allows for **pattern matching** within SMILES strings, enabling

the identification of specific molecular substructures.

- **Applications**: SMARTS is particularly useful for **substructure searching** in cheminformatics, such as identifying aromatic rings or other functional groups within a molecule.
- **Limitations**: Although SMARTS is powerful for substructure searches, it is more complex than SMILES, requiring a deeper understanding of chemical patterns to use effectively.

3. SELFIES (Self-Referencing Embedded Strings)

- **Overview**: SELFIES is a newer representation designed to overcome some limitations of SMILES. It functions more like a **programming language for molecules**, allowing for systematic molecule generation.
- **Key Advantage**: Every syntactically valid SELFIES string corresponds to a chemically valid molecule. This feature is especially valuable for applications in **generative chemistry**, where researchers aim to produce and screen large numbers of potential molecular candidates computationally.
- **Limitations**: SELFIES, while innovative, still requires additional work to ensure unbiased sampling across the chemical space.

4. InChI (International Chemical Identifier)

- **Overview**: InChI is a unique, standardized representation for each molecule, ensuring one-to-one correspondence between a molecule and its InChI string. For example, the InChI for **ethanol** is a complex string, making it difficult to interpret manually.
- **Application**: InChI is valuable for **database searches** and comparisons because it allows for direct molecular comparisons, avoiding redundancy from different SMILES representations for the same molecule.
- **Limitations**: The complexity of InChI makes it hard to read and interpret. Many prefer SMILES for visualization due to its simplicity.

5. Molecular Graphs

- **Overview**: Molecular graphs are **graph-based representations** where atoms are vertices and bonds are edges. These graphs capture all information about the molecule, including atom types, bond types, and even stereochemistry.
- **Advantages**: Molecular graphs are comprehensive, providing a complete picture of a molecule's structure, and are especially useful for **computational analyses** such as graph-based machine learning.
- **Limitations**: Unlike string-based formats, molecular graphs are not as compact or suitable for quick searches in databases. They are better suited for computational applications and visualizations than for data storage.

Comparison of Molecular Representations

Representation	Format	Primary Use	Key Strengths	Key Limitations
SMILES	String	General cheminformatics applications	Compact, widely supported	Ambiguous, limited stereochemistry
SMARTS	Pattern-based	Substructure searching	Allows detailed pattern matching	Complexity in structure matching
SELFIES	String	Generative chemistry	Every string corresponds to valid molecule	Requires careful sampling of chemical space
InChI	String (unique)	Database indexing, comparisons	Unique representation for each molecule	Complex, hard to read manually

Molecular Graphs	Graph	Computational analyses, ML	Complete structural information	Not compact for databases

Each representation has its **niche** in cheminformatics and molecular modeling, with SMILES as the most popular for general applications, InChI preferred for ensuring unique identification, and molecular graphs and SELFIES advancing in machine learning and generative applications. Understanding and choosing the appropriate representation is crucial for accurately modeling, simulating, and screening chemical compounds computationally.

13:05 Encoding Molecules as SMILES Strings

The **SMILES (Simplified Molecular Input Line Entry System)** format allows us to translate a molecule's structure into a single line of text that computational tools can process efficiently. This linear notation leverages simple syntax to represent complex chemical structures, offering a compact, widely-used format in cheminformatics. Understanding how SMILES encodes atoms, bonds, rings, charges, and stereochemistry is fundamental to its effective use.

1. Atoms and Bond Types

- **Atoms**: Each atom in a SMILES string is represented by its **chemical symbol**. Most atoms are single uppercase letters (e.g., `C` for carbon, `O` for oxygen). Some, like sodium (`Na`) or chlorine (`Cl`), require a two-letter combination where the first letter is uppercase and the second is lowercase.
- **Aromatic Atoms**: In SMILES, lowercase letters represent atoms in **aromatic compounds**. For example, lowercase `c` indicates an aromatic carbon, commonly found in benzene rings. This distinction helps encode resonance and ring structures simply.
- **Bonds**:
    - **Single bonds** are generally omitted for simplicity, although a single dash (`-`) can be used if desired.
    - **Double bonds** are denoted with an equals sign (`=`), and **triple bonds** with a hash (`#`).
    - **Aromatic bonds** can be omitted or represented with an asterisk (`*`), though omitting them is more common in practice, as the lowercase notation for atoms already implies aromaticity.
- **Disconnected Components**: The period (`.`) symbol is used to denote disconnected components of a molecule, often seen in salts or ion pairs. For example, `Na.Cl` represents sodium chloride.

2. Chains and Branching

- **Linear Chains**: Simple chains of atoms are represented in a straightforward sequence. For example, the SMILES for ethanol, `CCO`, sequentially lists the atoms as carbon, carbon, and oxygen.
- **Branches**: SMILES uses **parentheses** to denote branching from the main chain. For instance, if a side chain branches off the main molecule, everything within the parentheses represents that branch. The main chain continues from the atom preceding the branch. Complex molecules can have **nested branches** within branches, using multiple sets of parentheses to specify each level of branching.
- **Example**: The SMILES string `CC(C)O` represents isopropanol, where the `(C)` denotes a methyl branch off the main `C-C-O` chain.

3. Rings and Cyclic Structures

- **Ring Closures**: SMILES uses **numerical markers** to represent ring closures, ensuring the parser understands which atoms connect to close the ring. For instance, **benzene** is represented as `c1ccccc1`, where the `1` markers indicate that the first and last carbons are connected, completing the ring.
- **Multiple Rings**: In cases of more complex ring systems, additional numbers are used to represent each ring closure within the molecule. For example, **naphthalene**, with two fused benzene rings, would be represented as `c1ccc2ccccc2c1`, where the `1` and `2` markers specify the fused ring structure.
- **Nested and Fused Rings**: SMILES can handle more complex structures, such as nested or fused rings, by extending the numbering system. This flexibility is crucial for encoding polycyclic compounds accurately.

4. Charges

- **Charge Notation**: SMILES uses **square brackets** with `+` or `-` signs to indicate the charge on an atom. Charges are particularly relevant for ions or complex molecules with formal charges.
- **Examples**: For **sodium ion** (Na^+), SMILES would be `[Na+]`, and for **chloride ion** (Cl^-), `[Cl-]`.
- **Multiple Charges**: If an atom has multiple charges, the count is specified next to the charge symbol (e.g., `[Fe+2]` for Fe^{2+}).

5. Stereochemistry (Not Covered Here but Essential)

- **Chirality and Configuration**: While basic SMILES does not inherently include stereochemistry, extended versions of SMILES allow for chirality indicators (e.g., using `@` symbols for chiral centers). These additions help denote three-dimensional configurations, which are critical for bioactivity in many drugs.

6. Disambiguation with Brackets

- **Ambiguity Resolution**: When symbols could be interpreted in multiple ways, **square brackets** clarify the intended structure. For example, `[Sc]` represents **scandium**, whereas `SC` would be interpreted as sulfur (`S`) followed by a carbon (`C`), likely in an aromatic setting if lowercase is used.
- **Specificity**: Using brackets is particularly useful for representing **uncommon elements** or when certain atoms appear in unusual oxidation states or configurations.

Example Constructions

1. **Simple Chain**: Ethanol
   - SMILES: `CCO`
   - Explanation: This string represents a two-carbon chain (ethane) with a terminal hydroxyl group, forming ethanol.
2. **Branched Molecule**: Isopropanol
   - SMILES: `CC(C)O`
   - Explanation: The `(C)` branch represents a methyl group attached to the second carbon in the main chain (`CCO`), forming isopropanol.
3. **Ring Structure**: Benzene
   - SMILES: `c1ccccc1`
   - Explanation: The lowercase `c` denotes aromatic carbons, and the `1` markers at the beginning and end signify the ring closure.
4. **Charged Ion**: Ammonium Chloride
   - SMILES: `[NH4+].[Cl-]`
   - Explanation: This compound is represented as a disconnected pair (`.`) of ammonium ion (`[NH4+]`) and chloride ion (`[Cl-]`), showing both components of the ionic compound.
5. **Complex Ring System**: Naphthalene
   - SMILES: `c1ccc2ccccc2c1`
   - Explanation: The `1` and `2` markers indicate fused rings, with two six-membered aromatic rings sharing carbons.

Summary

Encoding molecules in SMILES requires a precise understanding of **atom notation, bond types, branching, ring closures, and charges**. The SMILES format, despite its limitations, remains a central tool in computational chemistry and cheminformatics due to its compactness and compatibility with various software tools. However, because SMILES lacks a unique representation format (except for canonical SMILES), ensuring **correct structural representation** often involves converting SMILES to other formats, such as InChI or molecular graphs, particularly for complex molecules where ambiguity must be minimized.

19:30 Morgan fingerprints: Encoding Molecular Structure for Machine Learning

Morgan fingerprints, also known as **circular fingerprints**, are a powerful tool for transforming molecular

structures into fixed-length binary vectors, making them highly suitable for machine learning applications. Unlike full molecular graphs, which require detailed structural data, Morgan fingerprints provide a more computationally efficient representation, capturing essential structural features around each atom and storing them in a format that is easily fed into machine learning models.

Purpose and Advantages of Morgan Fingerprints

Morgan fingerprints are a widely-used method in cheminformatics for encoding molecular information into **fixed-length vectors**. By hashing molecular substructures to predefined bit positions, they capture **topological and structural details** of a molecule. These fingerprints are particularly useful for tasks like **quantitative structure-activity relationship (QSAR)** modeling, which aims to predict properties such as hydrophobicity, solubility, and potential bioactivity. They're also effective for **high-throughput screening** in drug discovery.

The advantage of Morgan fingerprints lies in their balance between **simplicity and utility**:

1. **Fixed-Length Representation**: Each molecule, regardless of its size, is represented by a vector of predetermined length, making Morgan fingerprints directly compatible with many machine learning models.
2. **Encapsulation of Structural Information**: By encoding atom-centered substructures up to a defined radius, Morgan fingerprints capture the local chemical environment.
3. **Computational Efficiency**: Encoding molecules in this way is faster than generating full molecular graphs, and the resulting fingerprints can be rapidly processed in machine learning workflows.

How Morgan Fingerprints are Generated

The generation of a Morgan fingerprint involves a series of steps that focus on each atom in the molecule and examine its local structure within a specified radius.

1. **Atom-Centric Encoding**: Each atom in the molecule serves as a **center point**, around which the substructure is examined. For each atom, the fingerprinting process includes details of surrounding atoms within a certain **radius**.
2. **Defining the Radius**: The **radius** determines the extent of the local neighborhood to include around each atom:
    - **Radius 0**: Only the atom itself is encoded.
    - **Radius 1**: The atom and its immediate neighbors are encoded.
    - **Radius 2**: The atom, its immediate neighbors, and the neighbors of those neighbors are included.
    - Typically, a **radius of 2** is used, capturing sufficient detail for many applications without overwhelming the bit vector with excessive information.
3. **Hashing Substructures to Bit Positions**: Each substructure identified around an atom is **hashed** into a specific position within the fingerprint vector. If a particular hashed position is encountered, the corresponding bit in the vector is set to **1**. This results in a **binary vector** where each bit represents the presence or absence of certain structural features.
4. **Handling Collisions**: Since multiple substructures can hash to the same bit position, **collisions** can occur where different substructures map to the same index. This ambiguity is one limitation of Morgan fingerprints, as it introduces potential overlap of information. Increasing the length of the bit vector (e.g., using 2048 instead of 1024 bits) can mitigate this by providing more unique bins.

Example of Morgan Fingerprint Construction

Consider a simple molecule, like ethanol (`CCO`). Each carbon and the oxygen are treated as center points, and substructures within a radius of 2 are evaluated. This would yield:

- Atom 1: First carbon and its immediate neighbors (second carbon and oxygen).
- Atom 2: Second carbon and neighbors (first carbon and oxygen).
- Atom 3: Oxygen and its neighboring carbon.

Each atom and substructure are hashed into the fingerprint, resulting in a bit vector unique to ethanol's structure. This encoding captures local molecular features, providing a useful summary of the molecule's structure.

Balancing Radius and Bit Vector Size

Increasing the radius enables more detailed substructures, but also demands a larger vector to avoid collisions. For example:

- A **radius of 2** provides a moderate level of detail and is widely used in cheminformatics.
- Larger radii (3 or more) are generally only used if very fine-grained structural detail is required, which can significantly increase computational demand and complexity.

The length of the bit vector can also be adapted depending on the dataset and task:

- Commonly used lengths are **1024** or **2048 bits**.
- Larger vectors help avoid hash collisions but may introduce sparsity in smaller datasets.

Interpretation and Use in Machine Learning

Morgan fingerprints translate complex molecular structures into a vector space representation, where molecules with similar topological features have similar fingerprints. The resulting vectors can then be used in machine learning models such as:

- **Random Forests**: Good for feature-rich datasets, where the binary vector directly represents molecular features.
- **Multi-layer Perceptrons**: These can effectively process Morgan fingerprints by learning non-linear patterns in the data.
- **Other Algorithms**: Any algorithm that can process binary or numerical vectors can utilize Morgan fingerprints, including support vector machines and logistic regression.

Limitations and Considerations

While Morgan fingerprints provide a robust, compact representation of molecular structures, they do have limitations:

- **Non-Reversible**: The hashed vector cannot be reversed back to reconstruct the original molecule, meaning Morgan fingerprints are lossy.
- **Collision-Prone**: Different molecular structures may map to the same vector positions, especially in smaller bit vectors or large molecules with complex structures.
- **Insufficient for Large Biomolecules**: For very large biomolecules like proteins, Morgan fingerprints may not be suitable due to the complexity of their structure and the emergent properties at such scales.

In summary, Morgan fingerprints are a powerful tool for molecular representation, providing a fixed-length, computationally efficient vector ideal for machine learning applications in drug discovery and cheminformatics. By transforming complex molecular graphs into simplified yet informative vectors, Morgan fingerprints allow rapid analysis and property prediction, making them a mainstay in the field of computational chemistry.

33:00 Molecular graphs: Advanced Representation for Chemical Structures

Molecular graphs are a sophisticated and flexible way to represent molecules mathematically, particularly valuable when fingerprints alone fail to capture the full structural complexity required for accurate predictions. Unlike bit-vector fingerprints, molecular graphs provide a **direct mapping of a molecule's structure**, where each **atom is a node** and each **bond is an edge**. This structure-based representation allows for greater specificity and a richer dataset, making it an ideal input for **graph neural networks (GNNs)**, which excel at tasks that require in-depth relationship modeling, such as **quantitative structure-activity relationships (QSAR)** and **structure-based drug design**.

Structure of Molecular Graphs

1. **Nodes and Edges**: In a molecular graph, each **node** corresponds to an atom, and each **edge** represents a bond. This structure captures both the **types of atoms** and the **types of bonds** (single, double, triple, or aromatic) between them, enabling detailed representation of chemical relationships within the molecule.
2. **Labels and Features**:
    - **Node Labels**: Each node (atom) can carry a label or type (e.g., C for carbon, N for nitrogen), allowing the graph to differentiate between atom types.
    - **Edge Labels**: The edges can also be labeled to denote **bond types** (e.g., single, double, triple). This is particularly useful in molecular graphs, where bond types influence the chemical

behavior and properties of the molecule.

3. **Undirected vs. Directed Graphs**: In most cases, molecular graphs are **undirected** because a chemical bond typically has no direction. However, in cases such as **chemical reaction networks**, graphs can be **directed** to represent the flow of reactions from reactants to products.
4. **Weighted Graphs**: Although not commonly used in basic molecular graphs, **weighted edges** can be introduced to indicate bond strengths, interaction intensities, or other prior knowledge about the molecule's structure or behavior.

Types of Graph Representations in Molecular Graphs

1. **Adjacency Matrix**:
    - An **adjacency matrix** provides a tabular representation of a graph, where each cell denotes the presence or absence (or type) of a bond between atoms. If atoms iii and jjj are bonded, the cell at (i,j)(i, j)(i,j) would be filled with a 1 (or an integer representing bond type); otherwise, it would be 0.
    - **Symmetry**: For undirected graphs, the adjacency matrix is symmetric, while directed graphs will have an asymmetric matrix reflecting the direction of each edge.
2. **Adjacency List**:
    - When dealing with large graphs (e.g., in social networks or large datasets), **adjacency lists** become more efficient. Here, each atom has a list of directly bonded atoms, significantly reducing storage needs in sparse networks.
3. **Sparse Matrix Representation**:
    - Another option for handling large graphs is to use sparse matrices, which only store **non-zero values** (bonds in this case), reducing memory usage.
4. **Graph Visualizations**:
    - Graphs can also be visualized, providing an intuitive way to **inspect** molecular structures, understand their complexity, and troubleshoot any issues in the representation.

Applications and Benefits of Molecular Graphs

The **molecular graph representation** offers several advantages over simpler methods like fingerprints:

- **Complete Structural Encoding**: Molecular graphs retain the **full topological structure** of a molecule, allowing for a more thorough examination of its properties and interactions.
- **Compatibility with Graph Neural Networks (GNNs)**: By using molecular graphs, we can leverage GNNs, which are specifically designed to handle and learn from complex relationships in graph data.
- **Enhanced Predictive Modeling**: GNNs built on molecular graphs can predict various properties such as **bioactivity**, **toxicity**, **solubility**, and **binding affinity** with greater accuracy, given that the entire molecular structure informs the model.

Graph Neural Networks (GNNs) in Molecular Modeling

Graph neural networks are deep learning architectures tailored for graph-structured data, making them ideal for modeling molecular graphs. By learning from the relationships between nodes (atoms) and edges (bonds), GNNs can identify patterns and predict properties that are otherwise difficult to capture with simpler machine learning models.

- **Message Passing**: In GNNs, each node in the graph communicates with its neighbors in an iterative process called **message passing**. This allows each node to gather and aggregate information from nearby nodes, effectively capturing the influence of its molecular environment.
- **Layered Structure**: Like traditional neural networks, GNNs have layers, with each layer allowing the graph to capture increasingly distant relationships within the molecule.
- **Customizability for Directionality and Edge Weights**: GNNs can incorporate **directionality** and **edge weights**, which is especially useful for molecular structures with specialized interactions.

Types of Graphs Beyond Molecular Context

While molecular graphs are a natural fit for chemistry, the principles and techniques apply broadly to other fields where **graph-based data** is relevant, such as:

- **Social Networks**: Where individuals are nodes, and connections are edges, often directed to represent

the flow of influence.

- **Communication Networks**: Routing information where nodes are servers or routers and edges represent data flow paths.
- **Biological Networks**: Gene and protein interaction networks, where nodes represent genes or proteins, and edges represent interactions or regulatory relationships.

By understanding molecular graphs, students and researchers gain a foundation applicable to any domain with **relational data structures**. As such, the skills developed in working with molecular graphs and GNNs in cheminformatics extend far beyond chemistry, offering versatile tools for a wide range of applications in data science and machine learning.

In summary, molecular graphs offer a **comprehensive representation** of molecular structures, capturing nuances that fingerprints might overlook. This detailed representation enables GNNs to analyze molecules at an in-depth level, opening new possibilities for **property prediction** and **drug discovery**. By effectively leveraging molecular graphs, researchers can extract richer insights from chemical data, advancing both theoretical understanding and practical applications.

41:00 Applications of molecular graphs, and review of CNNs

Applications of Molecular Graphs in Drug Discovery

Molecular graphs are indispensable in various stages of **drug discovery and development**, enabling researchers to capture and manipulate the full structure of molecules with precision. Here's how molecular graphs can significantly enhance drug development workflows:

1. **Chemical Similarity Search**:
    - When a **lead compound** shows promising binding affinity to a specific target, researchers often search for **structurally similar compounds** that may have similar activities. With molecular graphs, we can encode the structural characteristics of known compounds and **search within a vast chemical library** (often containing millions of compounds) for similar structures.
    - By doing so, researchers can prioritize a smaller, more focused set of molecules for initial testing, reducing costs and resource use in large-scale screens. This approach is particularly useful when limited resources are available for direct screening and when there are other concurrent drug discovery efforts within the organization.
2. **Quantitative Structure-Activity Relationship (QSAR) Modeling**:
    - QSAR models use molecular graphs to predict various **biochemical and pharmacokinetic properties**. For instance, properties like **solubility, permeability across the blood-brain barrier, ADME (Absorption, Distribution, Metabolism, Excretion), and toxicity** can be predicted based on the molecular structure.
    - QSAR models leverage graph-based inputs to find correlations between molecular features and desired properties, facilitating **virtual screening** of compounds before experimental testing.
3. **Graph Neural Networks (GNNs)**:
    - Molecular graphs serve as the primary data format for **graph neural networks** (GNNs), which are well-suited to learn complex relationships within graph-structured data.
    - GNNs applied to molecular graphs can predict specific molecular attributes, binding affinities, and even identify potential **side effects or adverse reactions** by modeling interactions at the molecular level. This capability makes GNNs an essential tool in modern **computational drug discovery**.
4. **Structure-Based Drug Design**:
    - Molecular graphs enable detailed **structure-based drug design** by representing the complete molecular structure, which is necessary for accurate simulations and predictions of how a molecule might interact with a target protein.
    - By leveraging the graph-based representation, researchers can model the **binding energies and interaction dynamics** using AI-driven tools or traditional energy functions, allowing for the design of compounds optimized to bind specifically to their target.

Convolutional Neural Networks (CNNs) as a Foundation for GNNs

To understand **graph neural networks** (GNNs) more intuitively, it's helpful to consider the structure and

function of **convolutional neural networks** (CNNs), which are widely used in image processing.

1. **Processing 2D Grids**:
    - CNNs were initially designed to operate on structured 2D grids, making them highly effective for analyzing **images**. In CNNs, each **pixel** is treated as a unit that is connected to neighboring pixels, either vertically or horizontally, allowing the network to learn local patterns such as **edges, gradients, and textures**.
    - As the layers progress, CNNs aggregate these lower-level features into **higher-order patterns**, which ultimately enable the network to recognize complex objects (e.g., cats, dogs, cars).
2. **Hierarchical Feature Learning**:
    - CNNs employ a hierarchical approach, where **early layers** focus on low-level features like edges, and later layers progressively identify more complex shapes and objects by combining lower-level information.
    - This concept of **feature aggregation** through layered learning is a foundational idea that also applies to graph neural networks when working with molecular graphs.

Moving from CNNs to Geometric Deep Learning with GNNs

With the foundation of CNNs in mind, we can appreciate the **challenges and solutions in geometric deep learning** using GNNs:

1. **Geometric Deep Learning**:
    - In geometric deep learning, the aim is to generalize the success of CNNs on structured data (like images) to **unstructured, non-Euclidean data**, such as **graphs and manifolds**.
    - Graphs are inherently less structured than grids or matrices because they lack a fixed arrangement of nodes and can have varying connections, making them ideal for molecular representation, but they require specialized neural network architectures.
2. **Graph Neural Networks (GNNs)**:
    - GNNs extend the principles of CNNs by using a process called **message passing** or **neighbor aggregation**, where each node in a graph (representing an atom in a molecule) gathers information from its neighboring nodes. This process is repeated over multiple layers, enabling nodes to learn both local and global patterns.
    - Just as CNNs process pixels and learn hierarchical features, GNNs process atoms and learn about molecular structure through successive layers of aggregation, progressively capturing interactions across the molecular structure.
3. **Directed and Undirected Information Flow**:
    - While CNNs traditionally operate on undirected grids, GNNs can handle both **directed and undirected graphs**, which is essential for accurately modeling molecular interactions.
    - In directed graphs, such as reaction networks, GNNs can model the **directional influence** of nodes (such as reactants to products). For undirected molecular graphs, where bonds generally have no inherent direction, information flows bidirectionally.
4. **Labeled and Weighted Graphs**:
    - GNNs can incorporate **node and edge labels** to differentiate between types of atoms and bonds, and they can also handle **weighted edges** if certain bonds or interactions have different strengths or probabilities.
    - This flexibility enables GNNs to handle diverse graph structures, from simple molecules with single and double bonds to complex networks with directional and weighted connections.

In summary, **molecular graphs** are essential tools in drug discovery and development, providing a foundation for detailed analysis and prediction. By leveraging **graph neural networks** and the concepts of **geometric deep learning**, we can analyze the complex, relational structure of molecules, capturing nuances that are critical for effective drug design. This transition from CNNs to GNNs reflects a broader movement towards adaptable neural networks capable of operating on varied data types, expanding the scope of machine learning in molecular and chemical informatics.

44:50 Geometric deep learning: Extending Deep Learning to Complex Structures

Geometric deep learning is an approach that generalizes traditional deep learning methods to work with data beyond conventional grids, like images or sequences, by adapting to structures such as **graphs, point clouds, and manifolds**. This is particularly useful for tasks involving non-Euclidean data, where relationships and connections are critical and can't be easily flattened into a 2D grid.

Complex Structures in Geometric Deep Learning

In many applications, data does not fit neatly into a simple grid or sequence. Consider, for example:

1. **Graphs**: Common in social networks, molecular structures, and knowledge graphs, where nodes represent entities and edges define relationships or interactions.
2. **3D Point Clouds**: Found in computer vision tasks (like 3D object recognition) and in molecular or structural biology, where points represent atoms or molecules in space.
3. **Manifolds**: Often appear in high-dimensional spaces, where the data lies on a lower-dimensional but continuous and smooth surface.

In these cases, **spatial relationships** are often more meaningful than a strictly structured layout, and the connectivity itself holds valuable information.

Concepts of Symmetry, Invariance, and Equivariance

A foundational aspect of geometric deep learning is its focus on **symmetries** and properties like **invariance** and **equivariance**. These concepts help ensure that models respond predictably to transformations in data, which is especially relevant in fields like molecular modeling and physical simulations.

- **Invariance**: A model is invariant to a transformation if its output remains unchanged regardless of the transformation applied to its input. For example, a model that predicts the **solubility** of a molecule should be invariant to the molecule's rotation or translation in space, as these do not affect solubility.
- **Equivariance**: A model is equivariant if a transformation applied to the input results in a corresponding transformation in the output. This property is useful in tasks like **molecular dynamics simulations**, where a rotation of the reference frame should yield a similarly rotated output for the positions and momenta of atoms.

Implementing Invariance and Equivariance in Models

There are several strategies to ensure these properties in models:

1. **Designing Invariant or Equivariant Layers**: The structure of neural network layers can be crafted so that transformations are inherently handled. For instance, in a **graph neural network (GNN)**, layers can be designed so that they aggregate information based on node connections without depending on the node's spatial arrangement.
2. **Penalty-Based Regularization**: Another approach involves adding a penalty in the **objective function** if the model's output changes undesirably under transformations. This incentivizes the model to learn invariant or equivariant features as needed.
3. **Data Augmentation**: For image-based models, transformations such as rotations, translations, and flipping can be applied to the training data. This forces the model to learn features that remain robust to these changes. However, in some domains, especially with high-dimensional graphs, this approach can be impractical or insufficient.
4. **Learning through Constraint-Based Design**: Constructing models with strict mathematical constraints or using group theory can ensure that they inherently respect desired symmetries.

Application of Geometric Deep Learning in Molecular Graphs

In molecular modeling, **geometric deep learning** provides tools that are directly applicable to **molecular graphs**, where the relationships between atoms (nodes) and bonds (edges) define the molecule's structure. By treating each atom as a node and each bond as an edge, we can apply geometric deep learning principles to predict molecular properties, interactions, and behavior in a spatially-aware manner.

1. **Graph Neural Networks (GNNs)**: GNNs leverage the structure of graphs to pass information between nodes, enabling the model to understand relationships that may influence molecular properties, such as **binding affinity** or **toxicity**.
2. **Handling Rotation and Translation**: In the context of molecular properties like **binding affinity** or **reaction potential**, the model needs to be invariant to rotations and translations of the molecule. GNNs can be designed to aggregate information in ways that respect this invariance, making them well-suited

for molecular analysis.

3. **Capturing Hierarchical Patterns**: Just as CNNs identify increasingly complex patterns across their layers (edges, shapes, objects), GNNs in molecular settings can learn hierarchical relationships from individual atoms to complex molecular substructures. This hierarchical learning is essential for modeling **multi-scale phenomena**, such as how local molecular interactions influence broader pharmacokinetic behaviors.
4. **Equivariance in Physical Simulations**: In molecular dynamics or simulations of molecular interactions, equivariance is crucial as spatial transformations in one atom should propagate appropriately to other connected atoms. Geometric deep learning allows for building models that respect this, making them valuable for high-fidelity simulations of physical systems.

Advantages of Geometric Deep Learning in Drug Discovery

The utility of geometric deep learning in drug discovery and design is vast, as it enables a structured way of understanding and predicting molecular properties through models that inherently respect the spatial and relational nature of molecular data. Benefits include:

- **Improved Prediction Accuracy**: By respecting the structural characteristics of molecules, geometric deep learning methods often yield better predictive accuracy for molecular properties and interactions.
- **Efficient Representation of Molecular Space**: The flexibility of GNNs to handle complex graph-based data, such as variable bond types and atomic characteristics, allows for more **comprehensive modeling of molecular space**.
- **Enhanced Generalization to Novel Compounds**: Geometric deep learning models, by being invariant and equivariant where necessary, can better generalize to novel compounds, which is critical in the discovery of new drugs.

In summary, **geometric deep learning** represents an evolution of traditional deep learning, adapting methods to handle complex and structured data like graphs and point clouds. When applied to molecular graphs, it provides powerful tools for **analyzing, predicting, and simulating molecular properties and behaviors**, advancing the field of drug discovery by offering models that are robust, interpretable, and physically meaningful.

51:28 Overview of Graph Neural Networks (GNNs): Structure, Workflow, and Applications

Graph Neural Networks (GNNs) are specialized deep learning models tailored for directly operating on graph structures. Unlike standard neural networks designed for data in grids (like images or time sequences), GNNs handle data represented as nodes (e.g., atoms) and edges (e.g., bonds), making them invaluable for complex structures such as **molecular graphs, social networks, and knowledge graphs**. The goal of GNNs is to capture and leverage the relationships and dependencies within a graph to predict specific properties or behaviors associated with nodes, edges, or even the entire graph.

Key Applications of GNNs

In **drug discovery and molecular analysis**, GNNs can predict a range of properties:

1. **Node-level predictions**: For example, predicting if a specific atom (node) within a molecule will bind at an active site or participate in a reaction.
2. **Edge-level predictions**: Determining if a particular bond (edge) in a molecule is likely to break under the influence of an enzyme.
3. **Graph-level predictions**: Classifying the entire molecule by predicting properties such as its ability to cross the blood-brain barrier or its solubility in water.

GNNs support both **classification tasks** (e.g., binary decisions like solubility) and **regression tasks** (e.g., predicting a continuous value like the molecule's solubility level).

General Workflow of a GNN

The workflow for a GNN typically involves the following stages:

1. **Input Data and Initial Embedding**: Each node in the graph starts with some initial input features. In molecular applications, this could include atomic properties like atomic number, charge, or hybridization state. These features are embedded into a **latent representation** (a vector in a high-dimensional space) that captures the initial state of each node.
2. **Message Passing Through GNN Layers**: After embedding, the GNN performs a **series of**

message-passing steps across several GNN layers. This process is akin to the way convolutional layers in CNNs aggregate information from neighboring pixels, but in GNNs, the information is aggregated from connected nodes in the graph structure.

- **Aggregation and Update**: During each GNN layer, each node receives information (often called messages) from its neighbors. It aggregates this information, combines it with its own latent vector, and updates its state. This aggregation can be **sum, mean, or max pooling**, depending on the task requirements.
- **Multiple Layers**: Information is passed across multiple layers, effectively allowing each node to "see" further in the graph with each layer. For example, after two layers, each node can incorporate information from its two-hop neighborhood.

3. **Final State and Output Generation**: Once the graph has been processed through the GNN layers, the nodes reach a **final state**, which encapsulates the learned information from their surroundings. This state can then be used to generate predictions:
   - For node-level tasks, each node's final state serves as the basis for the prediction.
   - For edge-level tasks, the pairwise relationship between the final states of connected nodes can be analyzed.
   - For graph-level tasks, the final states of all nodes can be pooled into a single graph-level vector for classification or regression.

Example Workflow of a GNN

To illustrate, let's consider a **hypothetical molecular prediction task** using a GNN:

- We start with a molecular graph where each atom is a node with features like **atomic mass, electronegativity, and hybridization**.
- Each edge represents a bond, characterized by **bond type** (single, double, aromatic, etc.).
- After embedding, each node has a **latent vector** representing its initial features.
- This latent vector is iteratively updated over multiple GNN layers as information is passed and aggregated from connected nodes.
- At the end, the aggregated node representations are either directly used or further combined to make predictions, such as determining the likelihood of a molecule binding to a specific protein.

Types of GNN Operations

The operations within GNN layers are designed to generalize convolutional operations to graphs, where the spatial structure is defined by the edges connecting nodes rather than fixed spatial coordinates:

- **Convolution on Graphs**: In CNNs, convolution is performed in fixed grids; in GNNs, the convolution operates over the neighborhood of each node based on the edges. This local aggregation of features enables each node to learn from its neighbors while respecting the graph's inherent structure.
- **Pooling and Readout**: Similar to pooling layers in CNNs, pooling in GNNs helps reduce dimensionality and aggregate information across nodes. Pooling can be applied to nodes within subgraphs or across the entire graph, making it useful for **graph-level outputs**.
- **Normalization**: Due to varying node degrees, normalizing the aggregated messages (like normalizing pixel intensities in CNNs) helps prevent the model from over-relying on densely connected nodes and ensures balanced information flow.

Benefits of GNNs in Molecular Modeling and Drug Discovery

The structured message passing in GNNs enables **effective learning from complex relationships**, a crucial advantage in molecular and biological applications. Key benefits include:

- **Flexibility with Complex Structures**: GNNs can accommodate the irregular, non-Euclidean structures of molecular graphs, where atoms and bonds do not follow a grid-like structure.
- **Efficient Use of Structural Information**: By embedding and passing information according to the molecular structure, GNNs inherently respect the spatial and chemical relationships within molecules.
- **Scalability**: GNNs scale well with large graphs by using message-passing protocols, enabling models to handle large datasets like molecular libraries in virtual screening.

In summary, **GNNs provide a powerful framework for learning from graph-based data** by combining deep

learning techniques with the relational structures of graphs. Their versatility makes them suitable for a broad range of applications in **drug discovery, material science, and network analysis**, where understanding the intricate relationships within data is essential.

56:30 Message passing in Graph Neural Networks (GNNs)

Message passing is the fundamental operation in Graph Neural Networks (GNNs) that enables nodes to exchange and aggregate information from their neighbors. This process allows each node to develop a richer, contextually aware representation by incorporating features from connected nodes. This mechanism is essential for graph-based tasks such as **node classification, link prediction, and graph-level property prediction**.

Steps in Message Passing

Each GNN layer involves several key steps in the message-passing process. Let's break them down with a focus on how this affects the embedding and update process of each node.

1. **Message Creation**: For each node in the graph, GNNs create messages from its neighbors. This is typically done by taking the **latent vector (hidden representation)** of each neighboring node and transforming it using a **learnable weight matrix**. This transformation allows each node to convey information in a way that is meaningful to the task.
    - For example, if we're looking at the **yellow node** in a graph, it receives messages from its connected neighbors (e.g., the pink and gray nodes). Each neighbor's latent vector is transformed (e.g., by multiplying it by a weight matrix), producing a **message vector** for each connection.
2. **Aggregation**: The next step is to aggregate these messages into a single vector that represents the combined influence of all the neighbors. **Aggregation** is crucial because it determines how a node interprets the information it receives. Several aggregation strategies can be used, including:
    - **Sum aggregation**: Simply adds up all incoming messages. This approach is effective but can bias nodes with a high degree (nodes connected to many others) as they receive a larger combined influence.
    - **Mean aggregation**: Takes the average of incoming messages, which normalizes for the number of connections and gives each neighbor an equal weight in the final aggregated vector.
    - **Max pooling**: Retains only the most significant elements from all incoming messages, which can be useful in some applications but may lose granularity.
3. The choice of aggregation method impacts the GNN's behavior and performance. **Normalization** is often necessary, especially if there are nodes with widely varying numbers of connections, to ensure that highly connected nodes do not disproportionately affect the final representation.
4. **Permutation Invariance and the Challenges of Directionality**: In a GNN, there is no inherent "direction" or position for neighbors (unlike grids in CNNs, where pixels have specific positions relative to each other). This absence of structure in node ordering requires the GNN to be **permutation invariant**—the order in which messages from neighbors are received should not affect the result.
    - Unlike CNNs, where filters detect specific spatial patterns like edges or gradients based on pixel location, GNNs lack spatial orientation in the graph structure. Thus, GNNs need to aggregate messages in a way that is independent of their order. This is achieved by using symmetric functions (like sum, mean, or max) during aggregation to maintain **invariance**.
5. **Update Function**: After aggregating messages from neighbors, each node updates its latent vector. This step combines the node's **current latent vector** with the **aggregated message vector**. The update function often consists of multiplying the node's latent vector by another learnable weight matrix and combining it with the aggregated messages (e.g., by adding or concatenating). This updated latent vector represents the node's new state, enriched with contextual information from its neighborhood.
    - The update step can vary in complexity. Some GNN architectures may use simple addition, while others may apply non-linear functions (like ReLU or tanh) to introduce non-linearity, making the model more expressive.
6. **Repeating for Multiple Layers**: After each message-passing layer, the updated latent vectors can be fed into the next GNN layer, allowing information to propagate over longer distances in the graph. With each additional layer, a node can incorporate information from a broader neighborhood, eventually

reaching nodes multiple hops away. This multi-layer stacking enables the GNN to capture complex dependencies in the graph.

7. **Learning Parameters**: The weight matrices and other parameters used in message creation, aggregation, and update functions are learnable parameters in the GNN. The model learns these parameters during training to optimize the target task, such as predicting a molecular property or classifying nodes. This learning allows the GNN to adapt its message-passing operations to the specific characteristics of the data.

Example of Message Passing in Action

Suppose we're analyzing a molecular graph to predict the **solubility** of a compound. In this case:

- Each **node** represents an atom with features like atomic number and electronegativity.
- Each **edge** represents a bond with features like bond type (single, double, etc.).

During message passing:

- Each atom (node) gathers messages from its bonded neighbors. For example, the **carbon atom** (yellow node) receives information from **oxygen** and **hydrogen atoms**.
- After applying the aggregation function (e.g., mean aggregation), the carbon atom's latent vector now reflects an "average" influence from its neighbors.
- This updated latent vector is then used in the next layer, allowing the carbon atom to eventually learn from atoms that are further away in the molecular structure.

Importance of Message Passing in Molecular Analysis

The message-passing mechanism is particularly beneficial for tasks in **molecular analysis and drug discovery**:

- **Local Context**: Allows each atom to integrate information about nearby atoms and bonds, which is critical for understanding local chemical environments.
- **Scalability**: By using neighborhood-based aggregation, GNNs can efficiently handle large graphs without requiring a fully connected structure.
- **Expressivity**: Through multiple layers of message passing, nodes can learn not only about direct neighbors but also about more distant parts of the molecule, which is essential for capturing long-range dependencies.

In summary, **message passing** is the core process that enables GNNs to learn meaningful representations of nodes, edges, and entire graphs. By carefully designing each step—message creation, aggregation, and updating—GNNs can adapt to the complexities of molecular graphs, providing robust insights into properties and behaviors across diverse chemical structures.

1:02:50 Node Updates in a Basic Graph Neural Network (GNN)

In a basic Graph Neural Network (GNN), node updates occur as part of each layer's **message-passing process**. This update mechanism is crucial because it allows each node to build an increasingly sophisticated representation by iteratively integrating information from its neighbors.

Step-by-Step Breakdown of the Update Mechanism

1. **Initialize Node Embeddings**: Each node uuu in the graph has an initial embedding, denoted as Hu(0)H^{(0)}_uHu(0), which typically comes from input features or an embedding layer.
2. **Compute Latent Representations**: For each layer kkk in the GNN, we update the latent representation of each node based on the information from its neighbors. At layer kkk, the latent state of node uuu, denoted $H_u^{(k)}$, is computed by incorporating messages from neighboring nodes. The general update equation is as follows:

$$H_u^{(k)} = \sigma\left(W^{(k)} H_u^{(k-1)} + \sum_{v \in \mathcal{N}(u)} W_{neighbor}^{(k)} H_v^{(k-1)} + b\right)$$

   - **W**: This is a learnable weight matrix applied to node uuu's own latent vector.

- **$W_{neighbor}$**: Another learnable weight matrix, typically applied to each neighboring node v in u's neighborhood N(u).
- **Aggregation**: The summation $\sum_{v \in N(u)}$ aggregates messages from all neighbors. This aggregation step enforces **permutation invariance**, meaning that it doesn't matter in what order the messages arrive from neighbors.
- **Bias** b: A bias term may be added to allow for more flexible transformations.

3. **Non-Linearity Application**: After combining the information from the node's own latent vector and the aggregated neighbors, we apply a **non-linear activation function** (e.g., ReLU or tanh) to the result. This non-linearity is crucial in making the GNN more expressive, allowing it to model complex interactions between nodes.
4. **Increasing Receptive Field**: Each time we apply a GNN layer, we effectively increase the **receptive field** for each node. Initially, a node's receptive field includes only itself. However, after the first layer, the node's representation will incorporate information from its direct neighbors. After the second layer, it will include information from two-hop neighbors (neighbors of neighbors), and so on. This expansion means that nodes can accumulate information from progressively larger parts of the graph.
    - For example, in a molecular graph, the receptive field of a carbon atom might initially only include directly bonded atoms (such as an adjacent oxygen). After two GNN layers, this carbon could incorporate information from atoms that are two bonds away, providing a broader chemical context.
5. **Layer Depth and Information Spread**: While it might seem advantageous to use many layers to maximize the receptive field, deeper layers can lead to certain issues:
    - **Information Dilution**: With too many layers, node representations can become overly blended, losing unique characteristics and making it harder to distinguish between nodes. This phenomenon, often called **oversmoothing**, can lead to poor performance on complex tasks.
    - **Computational Costs**: More layers require additional computation and increase the risk of issues like vanishing or exploding gradients during backpropagation, making the training process more difficult.
6. Therefore, for many practical applications, it's beneficial to use a limited number of GNN layers (e.g., 2-4 layers). This approach balances computational efficiency with the depth of information each node can incorporate.

Summary of a Basic GNN Layer Workflow

1. **Embedding**: Initialize each node with a feature embedding.
2. **Message Passing**: Neighbors send messages based on their latent representations, weighted by a transformation matrix.
3. **Aggregation**: Aggregate the messages in a permutation-invariant way, typically by summing or averaging.
4. **Update with Non-Linearity**: Update each node's latent vector by combining its previous state with the aggregated neighbor messages and applying a non-linear function.
5. **Repeat**: Pass through additional GNN layers as needed, increasing the receptive field at each step.

By combining these steps, each node in a GNN builds a detailed, context-aware representation that encodes information from its surrounding subgraph. This iterative update mechanism is central to the GNN's ability to handle graph-structured data, making it well-suited for tasks in molecular modeling, social network analysis, recommendation systems, and more.

1:08:28 Utilizing the Output of a GNN

Once we've completed the layers and message-passing steps in a **Graph Neural Network (GNN)**, we're left with a graph where each node has a final latent representation—a vector that encodes information from both the node's features and those of its neighboring nodes. But how do we transform these node-level representations into useful predictions, especially when our objective may vary widely, from classifying individual nodes to predicting properties for the entire graph?

The output interpretation and aggregation methods differ based on the specific task, and these determine the final steps in GNN processing.

Key Output Tasks in GNNs

1. **Node-Level Predictions**:
    - For tasks where we're interested in classifying or predicting attributes for individual nodes, we can directly use each node's final latent representation.
    - For instance, if we have a graph representing molecules, and we know the molecular properties of each atom but are missing information for a specific atom (node), we can feed the latent vector for this unknown node into a classifier to predict the desired attribute, such as **whether it participates in a binding site or interacts with a certain molecule**.
2. **Edge-Level (Link) Prediction**:
    - Link prediction is about predicting the existence or strength of connections (edges) between nodes, which is useful in areas like **social networks** or **biological networks** where you want to infer new relationships.
    - For example, in genomics, we may know a set of genes associated with a disease and want to predict other potential genes by examining their connectivity within a biological pathway. This is done by taking the latent vectors for two nodes and applying a similarity measure, such as **cosine similarity or a learned function**, to predict whether an edge exists.
    - Alternatively, we could feed the combined representations of two nodes into a multi-layer perceptron (MLP) for a more complex relationship prediction.
3. **Graph-Level Predictions**:
    - In scenarios where we need a single prediction for the entire graph, such as the **solubility of a molecule** or **its toxicity**, we need to aggregate information from all nodes into a single, fixed-length vector that captures the overall structure and characteristics of the graph.

Aggregation Techniques for Graph-Level Predictions

Since each graph might have a different number of nodes, we need a way to **combine node-level latent vectors into a single representation**. Here are some common aggregation techniques:

- **Sum Aggregation**:
    - By summing all node representations, we get a single vector that reflects cumulative information from all nodes. This approach works well when node count is similar across graphs, but it can lead to bias if some graphs are much larger than others.
- **Mean Aggregation**:
    - This normalizes the aggregation by taking an average of node vectors, ensuring that the graph size doesn't impact the result. It's particularly useful when dealing with graphs of varying sizes, as it adjusts for node count disparities.
- **Max Pooling**:
    - Here, we take the maximum value from each position across all node vectors, capturing the most prominent feature values. This method highlights dominant characteristics in the graph but may ignore subtleties if other nodes contain essential lower values.
- **Attention Mechanisms**:
    - Attention-based mechanisms allow us to weigh each node's contribution differently based on its importance to the task. The model learns these weights, which adaptively highlight critical nodes, offering a powerful way to prioritize certain graph regions. For example, in a molecule, attention mechanisms can weigh critical functional groups more heavily when predicting chemical properties.
- **Global Nodes or Virtual Nodes**:
    - In some advanced GNNs, a "global" or "virtual" node is introduced, connected to all other nodes. During message-passing, this node aggregates information from across the graph, acting as a central hub. By the end, the virtual node's latent vector becomes the graph-level representation, capturing the entire network's features.

Using the Aggregated Output

Once the node representations have been aggregated into a single vector, we can apply the appropriate

classifier or regressor to generate a prediction:

- For **classification tasks** (e.g., predicting if a molecule crosses the blood-brain barrier), the aggregated vector is passed into a classifier, which outputs a label.
- For **regression tasks** (e.g., estimating a molecule's solubility), the vector is fed into a regression model to predict a continuous value.

Recap and Considerations

Using the output of a GNN involves multiple considerations, including the nature of the task, the structure of the data, and the size and diversity of the graphs. Each choice in the process—from aggregation technique to model type—affects the GNN's ability to make accurate, generalizable predictions.

By tuning these steps and selecting the appropriate method for each type of prediction, GNNs become powerful tools for complex datasets, especially in fields like drug discovery, molecular modeling, social network analysis, and more.

Lecture 15 - Generating New Molecules

Video: Lecture15 - Generating New Molecules - MLCB24

Slides: Lecture15_GeneratingSmallMolecules.pdf

0:00 Basic autoencoders

In today's lecture, we explored the use of **generative models** for the **creation of novel molecules** aimed at drug discovery and development. The focus was on understanding different classes of generative models, particularly **autoencoders**, and how they can be adapted to generate new chemical structures. This lecture covered foundational concepts in autoencoders, variational autoencoders, and practical applications in **chemoinformatics** and **hit optimization**.

1. Introduction to Generative Models

Generative models are a class of machine learning models designed to **generate new data points** that resemble the training data. These models can create **novel images, text, proteins, or chemical compounds** by learning an underlying **data distribution**.

- **Examples of Generative Models:**
  - **Transformers**: Used in text generation (e.g., ChatGPT) and protein sequence generation.
  - **Diffusion Models**: Used for image generation (e.g., Stable Diffusion).
  - **Autoencoders**: The focus of this lecture, used for representation learning and generating new data by reconstructing input data through a **latent space**.

Generative models have broad applications in **drug discovery**, such as generating candidate molecules, optimizing drug leads, and designing new proteins with desired properties.

2. Basic Autoencoders: Overview and Limitations

An **autoencoder** is a neural network designed to learn a **compressed representation** of input data by encoding it into a **latent space** and then reconstructing it.

- **Structure of an Autoencoder**:
  - **Encoder**: Compresses the high-dimensional input data into a lower-dimensional latent space.
  - **Latent Space**: A reduced representation of the input data.
  - **Decoder**: Reconstructs the original input data from the latent representation.
- **Example**: For an image with a resolution of 1000x1000 pixels (3 million values for RGB), the encoder compresses this into a latent representation, such as a vector of 100 dimensions, which is then decoded back to a similar image.

Limitations of Basic Autoencoders

- **No Structure in Latent Space**: The latent space may not have a meaningful structure, making it difficult to **sample new data**. Neighboring points in the latent space may correspond to **unrelated inputs**, limiting the utility of the model for generating similar yet novel molecules.

- **Reconstruction Objective Only**: The loss function focuses solely on reconstructing the input, without encouraging the latent space to have a smooth or meaningful structure.

To address these limitations, we need more advanced models, such as **variational autoencoders**.

3. Variational Autoencoders (VAEs)

A **Variational Autoencoder (VAE)** is a type of autoencoder designed to create a **structured latent space**, making it easier to generate new samples that are meaningful and similar to the training data.

Key Concepts in VAEs

- **Latent Space Regularization**: In a VAE, the latent space is regularized to follow a known probability distribution (e.g., a Gaussian distribution). This ensures that points sampled from this space correspond to realistic data points.
- **Probabilistic Encoding**: Instead of mapping an input to a single point in the latent space, the encoder maps the input to a **distribution** (e.g., mean and variance of a Gaussian). The decoder then samples from this distribution to reconstruct the input.
- **KL Divergence**: A key component of the VAE loss function is the **Kullback-Leibler (KL) divergence**, which measures the difference between the learned latent distribution and a standard Gaussian. This regularization encourages the latent space to have a smooth structure.

Benefits of VAEs in Drug Discovery

- **Smooth Latent Space**: The structured latent space allows for **interpolation** between data points, making it easier to explore variations of a given molecule.
- **Hit and Lead Optimization**: In the drug development process, small changes to a candidate molecule can be explored by making small perturbations in the latent space, leading to molecules with slightly altered properties.

4. Application of VAEs in Chemistry

VAEs have been adapted for generating **chemical structures** and **drug-like molecules**. Here's how:

- **Input Representation**: Molecules are often represented as **SMILES strings** (Simplified Molecular Input Line Entry System) or **graph-based representations**.
- **Encoder-Decoder Architecture**:
    - The **encoder** compresses the molecular representation into a latent vector.
    - The **decoder** generates a new molecule based on this latent vector, which can then be evaluated for its chemical properties.

Chemical Space Exploration

- By sampling from the latent space, researchers can generate **new chemical entities** that are structurally similar to known molecules but potentially have **improved properties** (e.g., better binding affinity, solubility, or stability).

Case Study: Work at MIT

- Pioneering work by researchers at MIT, including **Regina Barzilay** and **Tommi Jaakkola**, has demonstrated the power of VAEs for molecular generation. Their models have successfully generated **novel drug-like compounds** and optimized known leads for desired properties.

5. Sparse Autoencoders

Sparse autoencoders are a variant of autoencoders where the latent representation is encouraged to be **sparse**, meaning most of the neurons are inactive (i.e., have values close to zero).

Benefits of Sparse Autoencoders

- **Feature Separation**: Sparse autoencoders help disentangle the features, making it easier to interpret the model. Each neuron may capture a distinct, meaningful feature of the input data.
- **Model Interpretability**: In complex models like protein language models or large-scale generative models (e.g., ChatGPT), sparse autoencoders can help researchers understand what each part of the model is capturing, making the model more interpretable.

6. Summary of Key Points

- **Autoencoders** provide a way to learn a compressed representation of data, but basic autoencoders lack structured latent spaces, limiting their utility for generating new data.
- **Variational Autoencoders (VAEs)** introduce a probabilistic approach, regularizing the latent space to follow a known distribution, making it suitable for sampling and generating novel data points.
- VAEs have significant applications in **chemoinformatics**, allowing researchers to generate and optimize new drug molecules by exploring the structured latent space.
- **Sparse Autoencoders** enhance interpretability by encouraging a sparse latent representation, making it easier to understand the features captured by the model.

This lecture laid the groundwork for **systematically generating new molecules**, leveraging generative models like VAEs. In the next session, we will delve deeper into specific applications and discuss how these models integrate with **experimental pipelines** for drug development and optimization.

Next Steps

- We will continue exploring how to use these generative models in conjunction with **reinforcement learning** to fine-tune the generated molecules based on desired properties (e.g., binding affinity, toxicity).
- We will also discuss **evaluation metrics** for assessing the quality of generated molecules and how these models can be validated experimentally.

8:00 Variational AutoEncoders (VAEs) from a Probabilistic Perspective

Variational Autoencoders (VAEs) build on the concept of basic autoencoders but introduce a probabilistic framework that allows us to structure the latent space more effectively. Let's dive into the probabilistic foundations of VAEs, understand how they address limitations of basic autoencoders, and explore why this approach is powerful for tasks like generating new molecules or images.

1. Joint Probability and Latent Representation

In a VAE, we model the joint probability of the input data X (e.g., an image, a molecule) and the latent representation Z (a vector in the latent space). This joint probability is expressed as:

$$P(X,Z)=P(Z|X)\times P(X)=P(X|Z)\times P(Z)$$

Here, $P(Z|X)$ represents the probability of a latent vector given the input, and $P(X|Z)$ is the probability of generating the input given the latent vector.

- **Challenge**: We don't know $P(Z|X)$ or $P(X)$ directly. This is because we don't have a way to model the entire data distribution explicitly.

Instead, we work with $P(X|Z)$, which is our decoder, and $P(Z)$, which we define based on our assumptions about the latent space.

2. Defining the Latent Space Distribution

To solve the problem of not knowing $P(Z)$, we make a key assumption: we assume that the latent vectors Z follow a **multivariate Gaussian distribution**. This simplifies the problem by defining a known probability density function for the latent space:

- **Multivariate Gaussian Distribution**: We assume most of the density is concentrated around the origin, and the probability density falls off as we move farther from the origin. This distribution is characterized by a mean of zero and a unit variance for each dimension.
- **High-Dimensional Latent Space**: In high-dimensional spaces, the majority of points tend to lie on the surface of a hypersphere rather than close to the origin. This is a result of the geometry of high-dimensional spaces, where the volume is concentrated far from the center.

This assumption allows us to impose structure on the latent space, making it easier to sample new points and generate meaningful variations of the data.

3. Understanding the Decoder and Latent Sampling

The decoder $P(X|Z)$ generates data points (e.g., images or molecules) from the latent representation. However, the quality of these samples depends on how well the latent space is structured:

- **Decoder Training**: The decoder is trained on a dataset (e.g., images of specific objects, molecules with desired properties). The latent space is shaped by this training data, and it is not guaranteed to contain equal proportions of all possible samples. For example, if the training set contains mostly images of frogs, then the decoder will be biased towards generating frogs.
- **Latent Space Sampling**: To generate new samples, we take a point in the latent space (a vector Z), pass it through the decoder, and obtain a corresponding output. The effectiveness of this process depends on how well the latent space has been structured to capture the diversity of the training data.

4. Challenges with Estimating Probability Distributions

We face several challenges when dealing with probability distributions in VAEs:

- **Prior Probability of Latent Vectors (P(Z))**: We define this as a Gaussian distribution, which we can control and sample from easily.
- **Data Probability (P(X))**: This is the probability of observing any specific data point in the training set. However, it is difficult to estimate directly because it depends on the entire dataset and the decoder's learned behavior.
- **Decoder Output Probability (P(X|Z))**: This is known and represents the likelihood of generating a specific data point given its latent vector. This is literally what the decoder computes during training.
- **Posterior Probability (P(Z|X))**: This is the probability of the latent vector given the observed data. It is complex to compute directly because it requires knowledge of both P(X)P(X)P(X) and the entire data distribution. Instead, we approximate this using a neural network, which is known as the **encoder** in the VAE.

5. Applying Bayes' Rule

We can use Bayes' rule to relate these probabilities:

$P(Z|X)=P(X|Z)\times P(Z) / P(X)$

- **Posterior Approximation**: We approximate P(Z|X) with a learned neural network (the encoder), denoted as q(Z|X). This is crucial because computing the true posterior is intractable.
- **Evidence Lower Bound (ELBO)**: Instead of maximizing the likelihood of the data directly (which is hard to compute), we maximize a surrogate objective called the **Evidence Lower Bound (ELBO)**. This involves two main terms:
    1. **Reconstruction Loss**: Measures how well the decoder can reproduce the original data from the latent vector.
    2. **KL Divergence**: Ensures that the learned posterior distribution q(Z|X) is close to the prior P(Z), promoting a well-structured latent space.

This probabilistic framework enables the VAE to learn a smooth, well-organized latent space that can be easily sampled for generating new data.

6. Importance of Latent Space Structure

The key advantage of VAEs is that they impose a **probabilistic structure** on the latent space. This makes it more likely that small perturbations to the latent vector will result in valid and meaningful outputs. For example, in drug discovery, a slight shift in the latent vector could yield a new molecule with improved properties, facilitating lead optimization.

This structured approach contrasts with basic autoencoders, where there is no guarantee that nearby points in the latent space correspond to similar outputs. The VAE's probabilistic framework helps ensure continuity and coherence in the generated samples.

In summary, Variational Autoencoders use a probabilistic approach to model the latent space, imposing a Gaussian prior and optimizing the decoder's output through a combination of reconstruction loss and regularization. This method provides a powerful tool for generating new, diverse samples that are close variations of the training data, making VAEs highly effective for applications like drug molecule generation and image synthesis.

15:50 VAEs from a probabilistic perspective

In understanding **variational autoencoders** (VAEs), we need to delve into some fundamental concepts from

information theory, including **information, entropy, and KL divergence** (Kullback-Leibler divergence). This section introduces these concepts, clarifies their relevance, and explains how they become crucial in the optimization of a VAE.

1. Entropy and Information Theory

Entropy, in the context of information theory, is a measure of uncertainty or unpredictability in a probability distribution. If we have a probability distribution P(X), the entropy H(P) is defined as:

$$H(P) = -\sum_{x} P(x) \log P(x)$$

For continuous variables, this sum is replaced by an integral. The interpretation here is straightforward: if an event is highly predictable (e.g., a biased die always landing on six), the entropy is low. Conversely, if all outcomes are equally likely (e.g., a fair die roll), the entropy is higher. Entropy quantifies the amount of uncertainty or information contained in the distribution. In a biological context, think of entropy as the variability in possible states a system can adopt.

2. KL Divergence: Measuring Differences Between Distributions

While entropy measures the uncertainty within a single distribution, **KL divergence** (or Kullback-Leibler divergence) measures how one probability distribution diverges from a second, reference probability distribution. Mathematically, the KL divergence between two distributions PPP and QQQ is defined as:

$$D_{KL}(P||Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

Or for continuous variables:

$$D_{KL}(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} \, dx$$

The KL divergence is **not symmetric**, meaning that $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. It is not a true distance metric but rather a measure of how much information is lost when QQQ is used to approximate PPP. In other words, it tells us how different the two distributions are.

To understand this intuitively, think about **biological systems** like cellular ATP and ADP balance. The cell maintains a far-from-equilibrium state with a high concentration of ATP relative to ADP. KL divergence could be used to quantify how far this state is from equilibrium. In physics and thermodynamics, this divergence, when multiplied by kTkTkT (Boltzmann constant times temperature), corresponds to the **free energy** available in the system.

3. KL Divergence in Variational Autoencoders

In the context of VAEs, we use KL divergence to regularize the latent space representation. Recall that a VAE aims to learn a probabilistic mapping from input data to a **latent space** and then back to the data space. We approximate the complex true posterior distribution P(z∣x) (which represents the latent variable given the input) using a simpler, parameterized distribution Q(z∣x), often assumed to be Gaussian.

The KL divergence $D_{KL}(Q(z|x)||P(z))$ measures how much the approximate posterior Q(z∣x) diverges from the prior distribution P(z). This divergence term acts as a **regularizer**, encouraging the learned latent space to remain close to the prior distribution, typically chosen as a standard normal distribution. By minimizing this divergence, we ensure that the latent space is structured in a way that facilitates easy sampling and smooth transitions between different points, making the generated outputs more coherent.

4. Balancing Reconstruction and Regularization

In training a VAE, the **objective function** consists of two competing terms:

- **Reconstruction Loss**: This measures how well the decoder can reconstruct the input data from the latent representation. It ensures that the latent space captures meaningful features of the input data.
- **KL Divergence Loss**: This regularizes the latent space by penalizing deviations of Q(z∣x)Q(z|x)Q(z∣x) from the prior P(z)P(z)P(z). It prevents the model from simply memorizing the data and encourages generalizability.

The overall loss function, known as the **evidence lower bound (ELBO)**, is given by:

$$\text{ELBO} = \mathbb{E}_{Q(z|x)}[\log P(x|z)] - D_{KL}(Q(z|x)||P(z))$$

The first term is the **reconstruction term**, ensuring the model captures the data well. The second term is the **regularization term**, ensuring the latent space remains structured and coherent.

5. Why is KL Divergence Important?

The use of KL divergence in VAEs serves a crucial purpose:

- It **prevents overfitting** by encouraging the latent representations to conform to a prior distribution.
- It ensures that similar input data points are mapped to nearby regions in the latent space, allowing for smooth sampling and meaningful interpolation.
- It provides a principled way to incorporate prior knowledge about the latent space distribution, improving the generative capabilities of the model.

In summary, KL divergence helps the VAE learn a latent space that not only encodes the input data effectively but also remains well-structured and easy to sample from, facilitating the generation of new, coherent data points.

6. Connecting KL Divergence to Free Energy

As an analogy, in a physical system, moving away from equilibrium requires energy. Similarly, in a VAE, moving the approximate posterior Q(z∣x) away from the prior P(z) incurs a "cost," quantified by the KL divergence. This cost is akin to a **free energy difference**, pushing the model to find a balance between accurate reconstruction (low energy state) and maintaining a structured latent space (low divergence).

This dual objective captures the essence of why variational autoencoders are such powerful generative models. By balancing these competing forces, VAEs can effectively learn a smooth, continuous latent space that is well-suited for **sampling, interpolation, and generation** of new data.

In the next section, we will dive deeper into how this latent space is leveraged in **drug discovery**, exploring how small perturbations in the latent representation can lead to new candidate molecules with potentially improved therapeutic properties. This is particularly relevant for applications like **lead optimization**, where we seek to fine-tune molecular structures for enhanced efficacy and safety.

22:10 Information, entropy, and the KL divergence

In this section, we're diving into **information theory**, entropy, and the **Kullback-Leibler (KL) divergence**, which are essential concepts for understanding how **variational autoencoders (VAEs)** operate. These concepts help explain the competing objectives in VAEs and why certain optimization techniques are used.

Entropy and its Role in Information Theory

Entropy, first introduced by Claude Shannon, is a measure of **uncertainty** or **randomness** in a system. In a probability distribution p(x)p(x)p(x), the entropy H(p)H(p)H(p) is defined as:
H(p)=−∑xp(x)logp(x)H(p) = -\sum_{x} p(x) \log p(x)H(p)=−x∑p(x)logp(x)
For continuous variables, this becomes an integral. Entropy provides a way to quantify the amount of information or uncertainty associated with a probability distribution.

If we consider a **fair die**, where each of the six faces has an equal probability of 1/6, the entropy is maximized because there is maximum uncertainty about the outcome. The entropy would be log(6). In contrast, if the die is heavily biased towards a particular outcome (e.g., it always rolls a six), the entropy is minimized (approaching zero) because there is little uncertainty.

Interpretation: Entropy measures the **average information content** of a random variable. If the distribution is highly predictable (e.g., biased die), entropy is low. If the outcomes are highly uncertain (e.g., fair die), entropy is high.

Relative Entropy and the KL Divergence

The **Kullback-Leibler (KL) divergence**, or **relative entropy**, is a way to measure the difference between two probability distributions. Specifically, it quantifies how one probability distribution p(x)p(x)p(x) diverges from a reference distribution q(x)q(x)q(x). The KL divergence DKL(p∣∣q)D_{KL}(p || q)DKL(p∣∣q) is defined as:
DKL(p∣∣q)=∑xp(x)logp(x)q(x)D_{KL}(p || q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)}DKL(p∣∣q)=x∑p(x)logq(x)p(x)
For continuous variables, this becomes an integral. Unlike entropy, the KL divergence is not symmetric, meaning that:

$D_{KL}(p||q) \neq D_{KL}(q||p)$
This asymmetry reflects the fact that $D_{KL}(p||q)$ tells us **how much information is lost** when we use qqq to approximate ppp.

In a **biological context**, consider the balance between ADP and ATP in a cell. At equilibrium, the cell would contain mostly ADP because ATP tends to break down to release energy. However, cells are far from equilibrium because they maintain a high level of ATP. We can use the KL divergence to measure how far this **biological state** is from its natural equilibrium. The further the system is from equilibrium, the more energy it has available for cellular processes.

In an **information theory context**, the KL divergence measures the **inefficiency** of assuming a distribution qqq when the true distribution is ppp. It's analogous to measuring the **energy cost** of changing a natural probability distribution to match a desired one.

Connecting KL Divergence to Free Energy

There's a useful analogy between KL divergence and **free energy** in thermodynamics. If you multiply the KL divergence by a factor RTRTRT (where RRR is the gas constant and TTT is temperature), it gives a measure of the **free energy** available to the system. This concept connects the **probability shifts** seen in information theory with the **energy changes** in physical systems.

In **thermodynamics**, the free energy of a system reflects its ability to do work. Similarly, in a probabilistic model, the KL divergence represents the **"cost"** of deviating from the natural distribution.

Properties of the KL Divergence
The KL divergence has some important properties:

- It is always **non-negative**, meaning $D_{KL}(p||q) \geq 0$. The divergence is zero if and only if $p = q$ everywhere.
- It is **not symmetric**, so $D_{KL}(p||q) \neq D_{KL}(q||p)$.
- It does not satisfy the **triangle inequality**, so it is not a true distance metric but rather a measure of **divergence**.

These properties make the KL divergence particularly useful for tasks where we want to **minimize the difference** between an observed data distribution and a model's prediction. For example, in variational autoencoders, we use KL divergence to ensure that the learned latent representation $q(z|x)$ is close to a simple prior distribution $p(z)$, typically a Gaussian.

Application in Variational Autoencoders (VAEs)
In the context of VAEs, the KL divergence plays a key role in the **loss function**. The VAE loss function consists of two parts:

- The **reconstruction loss**, which ensures that the decoder can accurately reproduce the input data.
- The **KL divergence term**, which acts as a regularizer, ensuring that the learned latent space distribution $q(z|x)$ does not deviate too much from the prior distribution $p(z)$.

This balancing act between **reconstruction fidelity** and **latent space regularization** is what gives VAEs their power in generating realistic samples while maintaining a well-structured latent space.

In summary, the **KL divergence** is a fundamental concept that helps measure how one probability distribution diverges from another, and it plays a critical role in optimizing variational autoencoders by ensuring a structured and interpretable latent space. Understanding entropy and KL divergence provides a deeper insight into how **generative models** like VAEs achieve their goal of generating new and diverse samples while staying grounded in learned data distributions.

28:50 Rewriting Bayes' Law in the Context of Variational Autoencoders

In this section, we are going to delve into how we rewrite Bayes' Law for use in **Variational Autoencoders (VAEs)**. This is critical because it forms the foundation for how we optimize the VAE model, allowing it to generate new samples from a learned distribution. Up to this point, we've introduced the terminology and fundamental components of VAEs. Now, we will move into the mathematical formulation that connects these pieces.

Revisiting Bayes' Law and the Latent Space

Recall that in our VAE framework, we are working with two main distributions:

- **P(x)**: The probability of the observed data x.
- **P(z)**: The prior distribution over the latent variable z, which we often set to a **multidimensional Gaussian** centered at the origin.
- **P(x|z)**: The likelihood of the data given a particular latent representation.
- **Q(z|x)**: The variational approximation of the posterior, which serves as our encoder.

The goal of training the VAE is to maximize the likelihood of the observed data P(x), but this is computationally intractable because it involves integrating over all possible latent variables z. Instead, we use a tractable approximation via **variational inference**.

Introducing an Expectation over the Encoder Distribution

We start by taking an **average over the encoder distribution** Q(z|x). This is essentially integrating over all possible values of z based on the encoder's output. Formally, we write:
EQ(z|x)[logP(x)]
This expectation allows us to simplify the original equation, as P(x) itself does not depend on z. Thus, this integral is simply logP(x).

Decomposing the Evidence Lower Bound (ELBO)

We can decompose logP(x) into three main terms by applying algebraic manipulations and leveraging the expectation over Q(z|x):
logP(x)=EQ(z|x)[logP(x|z)]−KL(Q(z|x)||P(z))+KL(Q(z|x)||P(z|x))\log P(x) = \mathbb{E}_{Q(z|x)}[\log P(x|z)] - \text{KL}(Q(z|x) || P(z)) + \text{KL}(Q(z|x) ||
P(z|x))logP(x)=EQ(z|x)[logP(x|z)]−KL(Q(z|x)||P(z))+KL(Q(z|x)||P(z|x))
Let's break down these components:

- **Reconstruction Term** (EQ(z|x)[logP(x|z)]\mathbb{E}_{Q(z|x)}[\log P(x|z)]EQ(z|x)[logP(x|z)]):
  This term represents the **reconstruction error**. It measures how well the decoder can reconstruct the input data xxx given a sample from the latent space zzz. If this term is high, it means the model is successfully reconstructing the input data from its latent representation.
- **KL Divergence between the Encoder Output and Prior** (KL(Q(z|x)||P(z))\text{KL}(Q(z|x) || P(z))KL(Q(z|x)||P(z))):
  This term measures the **distance between the encoder's output distribution** and the prior distribution we assumed for the latent space (e.g., a standard Gaussian). If the encoder output deviates significantly from the prior, this term becomes large, and the model is penalized. This encourages the learned latent space to resemble a well-behaved, simple distribution (like a Gaussian), making it easier to sample from during generation.
- **KL Divergence between the Encoder Output and the True Posterior**
  (KL(Q(z|x)||P(z|x))\text{KL}(Q(z|x) || P(z|x))KL(Q(z|x)||P(z|x))):
  This term quantifies the difference between the approximated posterior Q(z|x)Q(z|x)Q(z|x) (from the encoder) and the true posterior P(z|x)P(z|x)P(z|x). While this term is theoretically informative, it is not directly computable because P(z|x)P(z|x)P(z|x) is intractable. However, by minimizing the first two terms, we indirectly minimize this third term as well.

Understanding the KL Divergence Terms

The **KL divergence** is a measure of how one probability distribution differs from another. Here, it plays two crucial roles:

  - The first KL divergence term (KL(Q(z|x)||P(z))\text{KL}(Q(z|x) || P(z))KL(Q(z|x)||P(z))) acts as a **regularization term**, ensuring that the learned latent distribution Q(z|x)Q(z|x)Q(z|x) does not diverge too far from the prior P(z)P(z)P(z). This maintains a smooth and consistent latent space.
  - The second KL divergence term (KL(Q(z|x)||P(z|x))\text{KL}(Q(z|x) || P(z|x))KL(Q(z|x)||P(z|x))) indicates how well our encoder Q(z|x)Q(z|x)Q(z|x) approximates the true posterior P(z|x)P(z|x)P(z|x). While we cannot compute this directly, minimizing the

other terms indirectly helps approximate it.

Optimizing the Evidence Lower Bound (ELBO)

To train the VAE, we aim to **maximize the Evidence Lower Bound (ELBO)**, which is the sum of the reconstruction term and the negative KL divergence. Maximizing ELBO is equivalent to maximizing the log likelihood of the data while also ensuring that the latent space remains well-structured. The objective function for the VAE becomes:
$ELBO = E_{Q(z|x)}[\log P(x|z)] - KL(Q(z|x) \| P(z))$
This formulation balances two objectives:

- **Accurate reconstruction of the input data** (high log-likelihood).
- **Regularization of the latent space** (low KL divergence), encouraging it to follow the assumed prior distribution.

Intuition Behind the Optimization Process

The optimization of the VAE can be thought of as a two-step process:

Step 1: Reconstruction: The encoder learns to map the input data xxx to a point in the latent space zzz, and the decoder learns to reconstruct xxx from zzz. This process minimizes the reconstruction error.

Step 2: Regularization: The KL divergence term ensures that the latent space distribution $Q(z|x)Q(z|x)Q(z|x)$ remains close to the prior $P(z)P(z)P(z)$. This regularization prevents the model from overfitting and allows for smooth interpolation between points in the latent space.

In summary, by rewriting Bayes' Law in this context and introducing the ELBO, we derive an efficient training objective for VAEs. This formulation allows us to balance accurate data reconstruction with regularization of the latent space, enabling the model to generate new, realistic samples from the learned distribution. VAEs thus provide a powerful framework for generative modeling, particularly in applications like drug discovery, where exploring the latent space can lead to novel and diverse candidate molecules.

35:00 Evidence lower bound (ELBO)

In the context of variational autoencoders (VAEs), the **Evidence Lower Bound (ELBO)** is a key concept that helps us approximate the complex problem of maximizing the probability of the observed data. Let's break this down step by step:

1. **Maximizing the Data Likelihood:**
   - The primary goal of generative modeling with a VAE is to maximize the likelihood of the observed data, denoted as $\log P(X)\log P(X)\log P(X)$. However, directly computing this quantity is intractable because it involves integrating over the entire latent space ZZZ. This is where the evidence lower bound (ELBO) comes in as a useful proxy.
2. **KL Divergence and the ELBO:**
   - The key to understanding the ELBO is recognizing the relationship between the actual data likelihood $\log P(X)$, the encoder approximation $Q(Z|X)$, and the true posterior $P(Z|X)$.
   - We can express the data likelihood in terms of two components: the ELBO and an **encoder error** term that represents the KL Divergence between $Q(Z|X)$(the approximate posterior) and $P(Z|X)$ (the true posterior).
   - The ELBO comprises two terms:
     - **Reconstruction Loss:** This term measures how well the decoder can reconstruct the input data from the latent space representation ZZZ. It captures the probability of the input data given the latent variables, $\log P(X|Z)$.
     - **KL Divergence Term:** This term measures the divergence between the approximate posterior $Q(Z|X)$ and the prior distribution $P(Z)$. It effectively regularizes the latent space to ensure it follows a desired distribution (e.g., a standard Gaussian).
3. **Maximizing the ELBO:**
   - Instead of directly maximizing $\log P(X)\log P(X)\log P(X)$, which is difficult to compute, we maximize the ELBO. This is advantageous because:

        - **Maximizing the ELBO** implicitly increases the likelihood of the observed data logP(X).
        - It simultaneously minimizes the KL Divergence between the approximate posterior Q(Z|X) and the true posterior P(Z|X), which reduces encoder error without requiring us to explicitly compute the intractable posterior.
4. **Forces Acting on the Latent Variables (μ\muμ and σ\sigmaσ):**
    - Within the ELBO, there are two opposing forces on the mean (μ\muμ) and variance (σ\sigmaσ) of the latent variables:
        - The **reconstruction loss** encourages μ\muμ to map data points to specific locations in the latent space that allow for accurate reconstructions.
        - The **KL Divergence** term regularizes μ\muμ and σ\sigmaσ to match the prior distribution, typically a standard Gaussian centered around the origin. This prevents the latent variables from spreading out too much and ensures the learned representations are well-structured.
5. **Balancing Reconstruction and Regularization:**
    - The model must balance two conflicting objectives:
        - **Accurate Reconstruction:** To minimize the reconstruction loss, the latent space representations (μ) should ideally be far apart for different data points (e.g., cats, cars, frogs), so they can be reconstructed accurately by the decoder.
        - **Latent Space Regularization:** To minimize the KL Divergence, the representations (μ and σ) should follow a standard Gaussian distribution, which may involve compressing different classes closer together in the latent space.
6. **Fast Computation of KL Divergence:**
    - Since the prior distribution is chosen as a standard Gaussian (mean of 0, standard deviation of 1), the KL Divergence term can be computed efficiently using a closed-form expression. Given the predicted μ\muμ and σ\sigmaσ from the encoder, we use the formula for KL Divergence between two Gaussians, which simplifies the computation during training.

In summary, the ELBO provides a practical and efficient way to train variational autoencoders by balancing accurate reconstructions with regularization of the latent space. By maximizing the ELBO, we indirectly maximize the likelihood of the observed data while maintaining a structured and continuous latent space representation, crucial for generating new samples and interpolating between known data points. This balance enables the VAE to be a powerful generative model capable of producing realistic, diverse outputs across various domains.

39:50 The opposing forces on a VAE

In Variational Autoencoders (VAEs), there are opposing forces shaping the latent space representation, resulting in a dynamic tension that balances reconstruction fidelity and regularization.

KL Divergence Regularization:

The KL Divergence term acts as a **regularization force**, pulling the learned mean (μ\muμ) of the latent variables towards the origin (i.e., a mean of zero). Additionally, it pushes the variance (σ2\sigma^2σ2) towards one, enforcing a **unit Gaussian** structure for the latent space. This regularization encourages the latent space to be smooth and consistent across samples, making it easier to interpolate between different points in the space and sample new data effectively.

By pulling the variance towards one, the VAE avoids the scenario where the model overfits to specific training examples by reducing the uncertainty (variance) to a very small value. Instead, the model is encouraged to maintain a generalizable latent representation that can encode diverse data points without collapsing the variance.

Reconstruction Loss:

On the other hand, the reconstruction loss acts as a **competing force**. This term aims to accurately reconstruct the input data from the latent representation, driving the encoder to place different input examples into distinct regions of the latent space. Essentially, the reconstruction loss encourages the separation of

μ\muμ values in the latent space to ensure that different inputs are mapped to different parts of the space, preserving their unique characteristics.

If the reconstruction loss dominates, σ2\sigma^2σ2 would ideally get very small, allowing the VAE to memorize the training set by creating highly precise, localized representations. This would lead to a highly specialized latent space but at the cost of **poor generalization**.

Tension Between Regularization and Memorization:

The interplay between these two opposing forces creates a dynamic balance within the VAE. The KL Divergence regularizes the latent space by constraining the variance and pulling the means towards the origin, while the reconstruction loss seeks to maximize fidelity by separating the latent representations of different samples.

This tension typically results in **blurry reconstructions** because the VAE avoids highly precise memorization of the training data. Instead, it prioritizes a smoother, more generalizable latent space representation that merges similar data points. This may cause the output images or generated data to appear less crisp but increases the robustness and generative capability of the model.

Blurriness and Generalization:

The trade-off imposed by the KL Divergence can make the VAE reconstructions less detailed or slightly blurry, especially compared to models that prioritize reconstruction loss alone. However, this blurriness is a side effect of the regularization, which ultimately allows the VAE to **generalize better** and generate plausible samples from the latent space that were not part of the training set.

The benefit of this approach is that the VAE learns a **smooth and coherent latent space**, where interpolation between latent points leads to meaningful and realistic outputs. This is particularly valuable in applications like drug discovery, where exploring small perturbations in the latent space can yield novel molecules with similar properties to known candidates.

In summary, the opposing forces in a VAE — regularization via the KL Divergence and separation via reconstruction loss — create a delicate balance. The model trades off perfect reconstruction for a more structured and generalizable latent space, leading to slightly less precise but more versatile generative capabilities. This compromise is crucial for applications that require exploration of the latent space, such as generating new chemical structures or optimizing lead compounds in drug discovery.

41:15 Issues with computing in a variational model

In a variational autoencoder (VAE), we encounter specific challenges related to backpropagation, particularly because of the **stochastic sampling step**. Backpropagation is the fundamental algorithm used to update model parameters in nearly every neural network model we've discussed. It involves traversing the computational graph in reverse to adjust weights based on the model's errors.

The Backpropagation Process

1. **Forward Pass**: We start by feeding data through the network. The network makes predictions by processing the input through its layers, and at the end, it outputs a prediction (e.g., the probability that an image contains a cat).
2. **Error Calculation**: The prediction is then compared to the true label. If the prediction is incorrect (e.g., it predicts 99% cat, but the image is actually not a cat), we need to update the parameters.
3. **Backward Pass**: This is where backpropagation occurs:
    - We go to the last layer and evaluate the error.
    - We look at the neurons feeding into this layer and ask: "If I slightly increase this parameter, will it raise the probability of the correct label or lower it?"
    - We adjust the parameter accordingly and move backward to previous layers, repeating this process.
    - This involves calculating gradients using the **chain rule of derivatives**, allowing us to propagate the error through every parameter in the model.

The Problem with Stochastic Sampling

In a VAE, we introduce a **stochastic sampling step**. When the encoder produces the latent variables, it estimates the **mean (μ\muμ)** and the **standard deviation (σ\sigmaσ)** of a Gaussian distribution. The latent representation is then sampled from this Gaussian distribution. This step is probabilistic, which poses a problem for backpropagation:

- The process of sampling a value based on the estimated mean and standard deviation is equivalent to a **coin flip**. However, this randomness cannot be directly differentiated, making it challenging to backpropagate the gradients through this stochastic sampling step.

Without the ability to backpropagate through the sampling, the model cannot update the encoder's parameters effectively. If we were to remove the stochastic component and avoid sampling altogether, we would be left with a **basic autoencoder**, which lacks the structured latent space that is a key advantage of the VAE.

The Reparameterization Trick

To solve this issue, we use a technique known as the **reparameterization trick**, a clever solution that allows us to bypass the non-differentiable sampling step:

1. Instead of directly sampling a value from the Gaussian distribution, we first generate a random value (ϵ\epsilonϵ) from a standard normal distribution (mean = 0, variance = 1).
2. We then transform this random value using the estimated mean (μ\muμ) and standard deviation (σ\sigmaσ): z=μ+σ×ϵz = \mu + \sigma \times \epsilonz=μ+σ×ϵ
3. By doing this, the randomness is encapsulated in the ϵ\epsilonϵ, which is independent of the model's parameters. The transformation (μ+σ×ϵ\mu + \sigma \times \epsilonμ+σ×ϵ) is now a deterministic function of μ\muμ and σ\sigmaσ.
4. This transformation allows us to treat μ\muμ and σ\sigmaσ as standard parameters, which can be differentiated and optimized through backpropagation.

Why the Reparameterization Trick Works

- The key idea here is that while we still introduce randomness, it is done **outside** the differentiable part of the model. The random noise (ϵ\epsilonϵ) is generated independently and does not rely on the model's parameters.
- By scaling and shifting this random noise using μ\muμ and σ\sigmaσ, we can produce the desired sample without interrupting the flow of gradients.
- This trick enables the VAE to maintain its **probabilistic nature** while allowing efficient optimization through gradient descent.

Summary

The reparameterization trick is a fundamental innovation in variational autoencoders. It:

- **Overcomes the challenge of non-differentiable sampling**, enabling backpropagation through the entire network.
- **Maintains the probabilistic modeling of the latent space**, allowing the VAE to generate meaningful and structured representations.
- Has become a standard approach in many models involving stochastic components, facilitating robust learning in generative tasks.

By employing this technique, VAEs effectively combine **probabilistic modeling** and **differentiable optimization**, making them a powerful tool for generating new data and exploring complex latent spaces.

46:30 Structure of the latent space and b-VAEs

In a well-trained **Variational Autoencoder (VAE)**, the latent space is ideally structured so that **similar data points are mapped to nearby regions**. This organization enables smooth interpolation, meaning that small movements in the latent space yield minor, coherent changes in the reconstructed output. For instance, if the input data are images of faces, then moving slightly in the latent space might correspond to changes in facial orientation, expression, or lighting conditions while preserving identity.

However, the ideal scenario described above doesn't always naturally emerge. While the KL divergence term

in the VAE loss function regularizes the latent space by enforcing a standard Gaussian prior, this alone may not always create a sufficiently disentangled or interpretable latent representation. This is where **β-VAEs** come into play.

Introducing β-VAEs: Scaling the KL Divergence

The **β-VAE** is a variant of the VAE where we introduce a scaling factor, **β**, to the KL divergence term in the loss function. The standard VAE loss function is:

$$\text{ELBO} = E_{Q(z|x)}[\log P(x|z)] - KL(Q(z|x) \| P(z))$$

In a β-VAE, we modify this by introducing the **β parameter**:

$$\text{ELBO}_\beta = E_{Q(z|x)}[\log P(x|z)] - \beta \cdot KL(Q(z|x) \| P(z))$$

Effect of the β Parameter:

1. **When β = 1**:
    - This is equivalent to the standard VAE. The regularization term, derived from first principles using Bayes' law, is balanced with the reconstruction loss.
2. **When β > 1**:
    - We increase the weight of the KL divergence term. This means the model prioritizes **adhering to the Gaussian prior** more heavily.
    - The model faces a greater "cost" (analogous to **free energy** in physical systems) for deviating from the Gaussian distribution. This results in a more constrained latent space.
    - As a consequence, the model is forced to focus on learning **high-level, descriptive features** of the data. The latent variables must efficiently encode the most meaningful aspects of the input data, leading to a more disentangled representation.
3. **When β < 1**:
    - We reduce the emphasis on the KL divergence term, allowing the model to prioritize reconstruction fidelity over regularization.
    - As β approaches zero, the VAE effectively behaves like a **basic autoencoder**, where the primary focus is on minimizing reconstruction error without any imposed structure on the latent space.

The Benefits of β-VAEs: Disentangled Latent Representations

The main advantage of using β-VAEs is the potential to learn a **disentangled latent space**. In a disentangled representation, each dimension of the latent space captures an **independent factor of variation** in the data. For example, in the case of facial images:

- One latent dimension might represent the angle of the head (e.g., left or right tilt).
- Another dimension might capture facial expression (e.g., smiling or neutral).
- A different dimension could correspond to lighting conditions.

Comparison to Standard VAEs:

In standard VAEs (β = 1), the latent dimensions may not have a clear, interpretable meaning. Nearby points in the latent space might correspond to drastically different inputs, such as the transition from one person's face to another's. The lack of clear structure can make the interpolation between data points less meaningful.

In contrast, β-VAEs (with β > 1) tend to produce a **more organized latent space**. When visualizing the latent dimensions, we often see smoother transitions. For example, in generative models of faces:

- Moving along one dimension might result in a face gradually turning left or right.
- Moving along another might change the expression without altering identity.

This disentanglement is valuable because it allows for **better control** over the generative process, making it easier to manipulate specific aspects of the generated data.

Trade-offs and Considerations

While increasing β encourages a disentangled and structured latent space, there are trade-offs:

- **Loss of Reconstruction Accuracy**: When β is increased beyond 1, the model places less emphasis

on accurately reconstructing the input data. As a result, the reconstructed samples may appear less precise or more blurry.

- **Balance Between Fidelity and Generalization**: β-VAEs achieve a balance between memorizing the training data (as in autoencoders) and generalizing well to unseen data (as in VAEs). The optimal choice of β depends on the application: for tasks requiring strong generalization and disentangled features, a higher β may be preferable.

Visualizing the Latent Space

When comparing the latent spaces of a standard VAE versus a β-VAE:

- In a **standard VAE**, the latent space often appears cluttered, with no clear organization or interpretation of dimensions.
- In a **β-VAE**, the latent space is more structured. Each dimension might clearly correspond to a distinct, meaningful feature of the input data, making it easier to explore variations systematically.

For example, in image generation tasks, if we traverse the latent space of a β-VAE along one dimension, we might observe a smooth change in head orientation (e.g., a face turning left to right). In a standard VAE, however, the same traversal might result in a random morphing of facial features, making it difficult to interpret the changes.

Summary: Why Use β-VAEs?

β-VAEs provide a simple yet powerful modification to standard VAEs that **encourages disentangled representations** and a more structured latent space. By adjusting the weight of the KL divergence term, we can control the trade-off between reconstruction accuracy and latent space regularization:

- **Higher β values** lead to more regularization, better disentanglement, and smoother, more interpretable latent spaces.
- **Lower β values** prioritize reconstruction fidelity, but may result in less meaningful or entangled latent dimensions.

In practice, the choice of β depends on the specific needs of the task. For applications like **drug discovery**, where exploring the latent space to generate new candidate molecules is crucial, a β-VAE with a higher β can help provide a clearer and more navigable latent representation, enabling researchers to systematically explore variations and identify promising leads.

This concludes our discussion on β-VAEs. Next, we will shift our focus to another generative model that has revolutionized the field of generative modeling: **Generative Adversarial Networks (GANs)**.

50:50 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of generative models known for producing high-quality, realistic samples. Although their popularity has waned somewhat with the rise of diffusion models, GANs have been a foundational technique in generative modeling and continue to offer valuable insights into machine learning architectures. Let's explore the structure and key concepts behind GANs and compare them to Variational Autoencoders (VAEs).

The Core Idea Behind GANs

The central concept of GANs is the **adversarial game** between two neural networks:

1. **Generator (G)**:
    - The generator's job is to produce fake samples that resemble the training data. It takes in random noise (typically sampled from a simple distribution like a Gaussian) and transforms it into a synthetic data point, such as an image or molecule.
    - The goal of the generator is to create samples that are **indistinguishable** from the real data, effectively "fooling" the second network, the discriminator.
2. **Discriminator (D)**:
    - The discriminator acts as a **classifier** that distinguishes between real samples (from the training dataset) and fake samples (generated by the generator).
    - The discriminator's objective is to correctly identify whether a given sample is real or fake.

The Adversarial Game: A Zero-Sum Competition

The training process of GANs involves a **zero-sum game** between the generator and the discriminator:

- The **generator** tries to minimize the probability that the discriminator correctly identifies its outputs as fake.
- The **discriminator** tries to maximize its accuracy in distinguishing real samples from generated ones.

The objective functions for the generator and discriminator can be expressed as follows:

- **Discriminator Loss**:
  $$L_D = -\mathbb{E}_{x\sim\text{data}}[\log D(x)] - \mathbb{E}_{z\sim\text{noise}}[\log(1 - D(G(z)))]$$
  Here, D(x)D(x)D(x) is the probability assigned by the discriminator that a real sample xxx is real, and D(G(z))D(G(z))D(G(z)) is the probability that a generated sample G(z) is real.
- **Generator Loss**:
  $$L_G = -\mathbb{E}_{z\sim\text{noise}}[\log D(G(z))]$$
  The generator aims to maximize the probability that the discriminator classifies its fake samples as real.

Training Dynamics: A Simultaneous Optimization

During training:

- The **discriminator** is optimized to accurately classify real versus fake samples. It learns to recognize the subtle differences between genuine data points and the synthetic samples generated by the generator.
- The **generator** is optimized to improve its ability to create realistic data points that can deceive the discriminator.

Ideally, this adversarial training leads to a state of **Nash equilibrium**, where:

- The generator produces samples that are **indistinguishable** from real data.
- The discriminator cannot distinguish between real and generated samples, yielding a classification accuracy of 50% (pure chance).

Comparison to Variational Autoencoders (VAEs)

GANs and VAEs both belong to the class of generative models, but they differ fundamentally in their objectives and outputs:

- **VAEs**:
  - Aim to model the **underlying data distribution** using a probabilistic framework.
  - Optimize for **reconstruction fidelity** and a well-structured latent space using the evidence lower bound (ELBO).
  - Often produce samples that are slightly **blurry**, due to the regularization imposed by the KL divergence term. This regularization helps ensure generalization but may sacrifice some visual quality.
- **GANs**:
  - Directly optimize for **realism**, focusing on producing samples that look indistinguishable from the real data.
  - Often generate **crisp, high-quality images**, as the generator is specifically trained to fool the discriminator.
  - May suffer from issues like **mode collapse**, where the generator only learns to produce a limited variety of samples, sacrificing diversity for quality.

The Trade-offs Between VAEs and GANs

1. **Sample Quality**:
   - GANs typically generate **crisper and more realistic samples** compared to VAEs. This is because the adversarial objective forces the generator to refine its outputs to deceive the discriminator.
2. **Sample Diversity**:
   - VAEs tend to cover a **wider diversity of samples**, even if some of them appear less realistic or are blurred. This is due to the probabilistic nature of VAEs, which encourages exploration of the

entire latent space.
    - GANs, on the other hand, may exhibit **mode collapse**, where the generator focuses on producing a small subset of high-quality samples at the expense of variety.
3. **Training Stability**:
    - GANs can be challenging to train because of the delicate balance required between the generator and the discriminator. If one model becomes too strong, it can overpower the other, leading to issues like **vanishing gradients** or **oscillatory behavior** during training.
    - VAEs generally have more stable training dynamics because their optimization is guided by a well-defined probabilistic framework.

Applications of GANs

Despite their challenges, GANs have been highly successful in various applications:

- **Image Generation**: GANs are capable of generating realistic images, making them popular in fields like computer vision and art. Techniques like StyleGAN have pushed the boundaries of photorealistic image synthesis.
- **Data Augmentation**: In medical imaging, GANs can be used to generate synthetic training data, augmenting limited datasets and improving the robustness of machine learning models.
- **Molecular Generation**: GANs can be adapted to generate molecular structures for drug discovery, although the focus on realism might limit their diversity compared to VAEs or diffusion models.

The Rise of Diffusion Models

Recently, **diffusion models** have gained prominence, often outperforming GANs in generating high-quality, diverse samples. While we won't go into diffusion models in detail in this lecture, it's important to note that they offer a different approach, based on gradually transforming noise into a structured sample through a sequence of iterative steps.

Diffusion models tend to have more stable training dynamics and can produce both realistic and diverse samples, making them a strong alternative to GANs and VAEs in many applications.

Summary: Why GANs Matter

Despite the growing popularity of diffusion models, GANs remain an influential and foundational technique in generative modeling. Their **adversarial framework** introduced a new way of training models to produce realistic samples by pitting two networks against each other. While they come with challenges, such as training instability and mode collapse, the crisp and high-quality samples they generate continue to make them valuable in applications requiring photorealism and fine detail.

In the next section, we will explore more advanced generative models and their applications in **small molecule generation**, including innovations like the **Junction Tree VAE**, which builds on both the VAE framework and molecular graph representations to design novel chemical compounds.

54:20 A Junction tree VAE for small molecules

In this section, we explore a novel generative model designed specifically for small molecule generation, called the **Junction Tree Variational Autoencoder (Junction Tree VAE)**. This model was developed to address the limitations of sequence-based molecular representations, such as SMILES strings, by utilizing **graph-based representations** that capture the structural nuances of molecules more effectively.

Challenges with Sequence-Based Representations

Earlier approaches to molecule generation often relied on sequence-based representations like **SMILES strings**:

- **SMILES strings** encode molecules as sequences of characters, similar to how we might represent text. While they are convenient for input into sequence models (e.g., RNNs, Transformers), they present significant challenges:
    - **Non-uniqueness**: Different SMILES strings can represent the same molecule, leading to ambiguities during training.
    - **Instability**: Small changes in the SMILES string (e.g., a single character alteration) can result in drastically different molecular structures. For instance, modifying a single methyl group placement might yield a completely different SMILES representation, despite the underlying chemical similarity.

This instability and lack of a coherent mapping between similar molecules in sequence space hinder the model's ability to generate valid, chemically meaningful structures. Generative models operating directly on SMILES strings often produce invalid or unrealistic molecules, as the space of valid chemical structures is sparsely represented in the sequence domain.

Graph-Based Representations: A More Natural Fit

To overcome these issues, researchers turned to **graph-based representations** of molecules. In a molecular graph:

- **Nodes** represent atoms.
- **Edges** represent chemical bonds.

This representation aligns closely with the true structural properties of molecules, making it easier to capture local and global chemical interactions.

Junction Tree VAE: The Core Idea

The **Junction Tree VAE** leverages this graph-based representation while introducing a new way to decompose and model molecular structures. Instead of generating molecules **atom by atom**, which often leads to invalid intermediates, the Junction Tree VAE constructs molecules using **larger structural motifs** or **substructures**, forming a tree-like representation.

Why Not Atom-by-Atom Generation?

Generating molecules atom by atom has a significant drawback:

- Intermediate structures often represent **invalid molecules**, making it difficult for the model to remain in the space of valid chemical entities throughout the generation process.
- This problem becomes pronounced when making small modifications to existing molecules. For instance, changing a single methyl group's position might require traversing through multiple invalid intermediates.

The **Junction Tree VAE** circumvents this issue by **clustering atoms into meaningful substructures** and modeling the connections between these clusters rather than individual atoms. This approach ensures that the model operates in a space constrained to valid chemical structures.

Junction Tree VAE Architecture

The Junction Tree VAE consists of two main components:

1. **Graph Encoder-Decoder**:
    - The **graph encoder** takes a molecular graph as input and compresses it into a **latent space representation**, capturing the global structure of the molecule. This representation includes a mean vector (μ\muμ) and a standard deviation vector (σ\sigmaσ), as in a standard VAE.
2. **Junction Tree Encoder-Decoder**:
    - The molecular graph is also decomposed into a **junction tree**, where each node represents a **cluster of atoms** (e.g., a functional group or a ring system).
    - The **junction tree encoder** captures the hierarchical structure of the molecule by encoding the relationships between these substructures, again producing a latent space representation (μ\muμ and σ\sigmaσ) for the tree.

These two representations (the graph and the junction tree) are **concatenated** to form the final latent vectors for both the mean and standard deviation, which are then used in the VAE framework.

Constructing the Junction Tree

The process of creating the junction tree involves:

- **Decomposing the molecule** into clusters of atoms based on chemical substructures (e.g., rings, functional groups).
- **Connecting these clusters** in a tree-like structure using an adapted version of the **Junction Tree Algorithm**, a method commonly used in graphical model analysis. This algorithm determines the optimal way to segment the graph into subcomponents while preserving the overall chemical validity.

By leveraging this hierarchical representation, the Junction Tree VAE captures both the **local connectivity** (within substructures) and the **global connectivity** (how substructures are linked) of the molecule, leading to a

more robust generative model.

Latent Space Representation

The Junction Tree VAE generates two separate latent space embeddings:

- **Graph Latent Space**: Encodes the overall molecular graph structure.
- **Tree Latent Space**: Encodes the hierarchical relationships between molecular substructures.

These two embeddings are **concatenated** before being fed into the VAE's decoder. This dual representation allows the model to capture both fine-grained atomic details and broader structural patterns, enabling more accurate generation of chemically valid molecules.

Generating New Molecules

The generation process in a Junction Tree VAE involves:

1. **Sampling from the Latent Space**: Points are sampled from the concatenated latent space, representing potential new molecules.
2. **Decoding the Junction Tree**: The tree structure is decoded first, ensuring that the substructures and their connections are chemically valid.
3. **Reconstructing the Molecular Graph**: The full molecular graph is then reconstructed based on the decoded junction tree, yielding a new molecule.

This hierarchical decoding process maintains chemical validity at every step, reducing the risk of generating nonsensical or unstable molecules.

Results and Observations

The Junction Tree VAE demonstrates several key advantages over previous methods:

- **Higher Validity**: Molecules generated using the Junction Tree VAE are more likely to be chemically valid because the model operates within a constrained space of feasible substructures and connections.
- **Diversity and Novelty**: The model can explore a wide range of chemical structures by sampling different points in the latent space, leading to the generation of novel molecules with diverse properties.
- **Smooth Latent Space Interpolation**: By representing the molecule at multiple levels (graph and tree), the Junction Tree VAE enables smooth interpolation in the latent space. Small perturbations in the latent representation result in slight, interpretable changes in the molecular structure, facilitating tasks like lead optimization.

For example, when visualizing samples from the latent space:

- **Random samples** drawn from the prior distribution yield molecules that are structurally coherent and chemically plausible.
- **Interpolated samples** near a specific point in the latent space exhibit gradual modifications, such as adding or repositioning functional groups, while preserving the core molecular scaffold.

Summary

The **Junction Tree VAE** is a powerful extension of the standard VAE framework tailored for molecular generation:

- It addresses the limitations of sequence-based models like SMILES by using **graph-based representations**.
- It introduces a hierarchical decomposition of molecules into **substructures**, enhancing the model's ability to generate valid and diverse chemical entities.
- The dual latent space representation (graph and tree) provides a comprehensive encoding of both local and global molecular features, leading to smoother and more interpretable latent space dynamics.

In conclusion, the Junction Tree VAE offers a promising approach for tasks like **lead optimization** and **drug discovery**, where exploring slight structural variations of candidate molecules is crucial. This model exemplifies the cutting-edge work being done at MIT and serves as a stepping stone towards more sophisticated molecular generative models.

Next, we will move on to explore the application of AI in identifying **candidate antibiotics**, where similar generative techniques can play a pivotal role in discovering new therapeutics.

1:02:30 Using AI to identify candidate antibiotics

In this final section of the chapter, we explore how advanced AI techniques, particularly those involving **molecular graph representations**, have been applied to the field of **antibiotic discovery**. This work is part of a collaborative effort between researchers at MIT and the Broad Institute, including notable contributions from Jim Collins's lab in the Institute for Medical Engineering and Science (IMES). The approach leverages graph neural networks and novel message-passing strategies to accelerate the identification of promising antibiotic candidates.

Challenges in Traditional Antibiotic Discovery

Historically, antibiotic discovery has relied on **small molecule screening**, a process where vast libraries of chemical compounds are tested for their ability to inhibit bacterial growth:

- **Screening Process**: Researchers test each compound in the library against a target organism (e.g., *E. coli*) to identify **hits**, which are molecules that show antibacterial activity.
- **Lead Optimization**: Once hits are identified, they are further validated and refined through chemical modifications to optimize efficacy and reduce toxicity.
- **Limitations**: This process is **expensive** and time-consuming, requiring extensive laboratory testing. The chemical diversity of available compound libraries is limited, making it challenging to explore the full space of potential antibiotics, especially for **drug-resistant bacteria**.

Graph Neural Networks for Antibiotic Discovery

To overcome these challenges, the researchers developed a **graph neural network (GNN)** model specifically tailored for molecular graph representations. This model introduces a unique **message-passing strategy** that enhances its predictive capabilities.

Key Innovation: Bond-Centric Message Passing. In standard graph neural networks applied to molecular data, each **node** typically represents an atom, and edges represent bonds between atoms. However, this approach may overlook the critical role of **chemical bonds** in determining molecular properties. The innovation in this model lies in shifting the focus from atoms to bonds:

- **Latent Vectors for Bonds**: Instead of assigning a latent vector to each atom, the model assigns latent vectors to **bonds**, effectively capturing the connectivity and interactions between atoms. The bonds serve as the primary carriers of information, with atoms acting as intermediaries that connect different bonds.
- **Message Passing**: The latent vectors associated with bonds are updated through a message-passing mechanism, where bonds exchange information according to predefined rules. This approach aligns with the chemical intuition that the properties of a molecule are heavily influenced by the nature and arrangement of its chemical bonds.
- **Graph Convolutional Layers**: The model employs multiple layers of graph convolution (typically three layers), allowing it to aggregate information from neighboring bonds and build a comprehensive representation of the entire molecular structure.

Training the Model on a Drug Repurposing Library

The researchers applied their GNN model to a **drug repurposing library** at the Broad Institute, consisting of approximately **10,000 compounds**:

- **Library Composition**: Many of the compounds were either **FDA-approved** drugs or had reached advanced stages of clinical trials, making them strong candidates for repurposing as antibiotics.
- **Training Objective**: The model was trained to predict the extent to which each compound inhibits the growth of *E. coli*. This was formulated as a binary classification task: distinguishing compounds that inhibit bacterial growth from those that do not.
- **Performance Metrics**: The model achieved a high **Area Under the ROC Curve (AUC)** of nearly **0.9**, indicating strong predictive performance. The results demonstrate that the model can effectively identify compounds with antibacterial activity, even when these compounds were not part of the training set.

Validation of Top Predictions

To validate the model's predictions, the researchers conducted further experimental testing:

1. **Experimental Setup**: They selected the **top 100 predicted compounds** from the test set, focusing on those not included in the training data to ensure unbiased evaluation.

2. **Results**: Nearly **50% of the top predictions** were found to inhibit *E. coli* growth, a significant improvement compared to random screening, which typically yields much lower hit rates. This highlights the model's ability to accurately predict antibiotic activity and prioritize promising candidates for further investigation.

Case Study: Identification of a New Antibiotic Candidate

Among the top-ranked compounds identified by the model was a **kinase inhibitor** previously known for its activity against human targets. The researchers repurposed this molecule for its antibacterial properties and conducted additional tests:

- **Inhibition of *E. coli* Growth**:
  - The compound, later named **Halicin**, demonstrated strong inhibitory effects on *E. coli* in dose-response experiments, with bacterial growth declining as the concentration of the drug increased.
- **Activity Against Drug-Resistant Bacteria**:
  - Halicin was also tested against **antibiotic-resistant strains**, including *Acinetobacter baumannii*, a highly concerning pathogen known for its resistance to most available antibiotics.
  - The compound effectively inhibited the growth of *A. baumannii*, suggesting potential as a broad-spectrum antibiotic.
- **In Vivo Efficacy**:
  - The researchers further validated Halicin's efficacy in a **mouse model** of infection, where it showed significant activity against *Clostridioides difficile* (commonly known as *C. diff*), a major cause of hospital-acquired gut infections.

Broader Implications and Future Directions

The success of this approach demonstrates the potential of **AI-driven drug discovery**:

- **Efficiency**: By using a graph neural network with bond-centric message passing, the model can explore the chemical space more effectively than traditional sequence-based or atom-centric methods.
- **Scalability**: The application of AI models to large compound libraries can drastically reduce the time and cost associated with screening, enabling the rapid identification of promising leads for further development.

The Growing Role of AI in Drug Discovery

The increasing adoption of AI methods in pharmaceutical research is reshaping the landscape of drug development:

- **Startups and Industry Impact**:
  - There has been a notable rise in **AI-based drug discovery startups**, many of which focus on leveraging generative models and graph-based learning. These companies are now catching up to, and in some cases surpassing, traditional pharmaceutical research programs in terms of the diversity of candidates in their pipelines.
- **Industry Adoption**:
  - On the right side of the comparison, we see the **top 20 pharmaceutical companies** gradually integrating AI-based methods into their research programs, indicating a shift towards more data-driven, AI-enabled approaches across the industry.

This represents an exciting time in antibiotic discovery and drug development, as AI models continue to improve in their predictive capabilities and become a crucial part of the drug discovery toolkit.

Conclusion

The application of AI to antibiotic discovery, particularly through the use of **graph neural networks** with innovative message-passing strategies, showcases the power of combining machine learning with chemical intuition. By focusing on **bond-centric representations** and leveraging large-scale drug repurposing libraries, the researchers have identified promising new antibiotic candidates, including Halicin, which shows potential against drug-resistant pathogens.

This work exemplifies the cutting-edge research being conducted at MIT and the Broad Institute, highlighting the transformative impact of AI on accelerating the drug discovery process. As these methods continue to

evolve, they hold promise for addressing critical public health challenges, such as the growing threat of antibiotic resistance.

In the next session, we will delve into practical examples and hands-on applications of these techniques, providing insights into how you can implement similar models in your own projects. See you on Thursday for a walkthrough of these tools and tips for training generative models effectively!

Lecture 16 - Training Neural Networks

Video:
https://www.youtube.com/watch?v=ea0g2gS6YLE&list=PLypiXJdtIca4gtioEPLIExlAKvu64z7rc&index=17

Slides: Lecture16_TrainingDeepNetworks.pdf

0:00 Training Neural Networks – A Case Study on Blood-Brain Barrier Prediction

In this chapter, we'll explore the step-by-step process of **training a neural network model** to predict the ability of small molecules to cross the **blood-brain barrier (BBB)**. This hands-on example, which draws on real-world data and machine learning practices, will provide a detailed overview of model training, data handling, and optimization strategies essential in practical applications.

1. Problem Definition: Predicting Blood-Brain Barrier Transmission

The blood-brain barrier (BBB) serves as a selective barrier, preventing certain molecules from entering the brain. The ability of molecules to cross the BBB depends on factors such as **hydrophobicity** and **molecular size**. Here, we aim to predict whether a given molecule can cross the BBB based on its molecular structure. We use a **binary classification model** with data from a **Kaggle dataset** containing around 2,000 molecules labeled as either capable (1) or not capable (0) of crossing the BBB.

Key insights before modeling include:

- **Class Imbalance**: Approximately 75% of the molecules can cross the BBB. Therefore, a naive classifier could achieve 75% accuracy simply by predicting "yes" for all inputs.
- **Baseline**: This 75% represents our **baseline accuracy**; the model must surpass it to be useful.

2. Data Loading and Preprocessing

The data includes **SMILES (Simplified Molecular Input Line Entry System) strings**, which provide a compact representation of a molecule's structure. To train our model, we need to transform these strings into a numerical format that a neural network can interpret.

Steps for data loading and transformation:

1. **Download and Inspect Data**: The dataset includes SMILES strings, labels (1 or 0), and molecule names. The data format is a simple spreadsheet.
2. **SMILES to Embeddings**: Directly using SMILES strings as input to the neural network isn't feasible. Instead, we transform them into **fixed-length vector embeddings** using a pre-trained model (e.g., **KMTA from Hugging Face**). KMTA, based on the BERT architecture, outputs an embedding for each SMILES string.
3. **Embedding Tokenization**: SMILES strings are tokenized before passing into KMTA. We use KMTA's tokenizer to segment SMILES strings into appropriate tokens, reducing the complexity of the embedding process.

3. Dataset and DataLoader Setup

To streamline data handling in PyTorch, we implement a **custom Dataset class** for our BBB data:

- **Dataset Class**: This class reads the SMILES strings and labels, generates embeddings, and provides methods to access individual samples.
- **DataLoader**: The DataLoader further simplifies batching and shuffling, providing batches of data to the model during training and evaluation. **Batch size** impacts both memory usage and the accuracy of gradient estimates, with larger batches providing more accurate gradient estimates but requiring more memory.

4. Model Architecture

Our model is a **simple feedforward neural network**. Given the output dimension of the KMTA embedding (768), our neural network has three linear layers, progressively reducing dimensionality:

1. **Linear Layers**: A sequence of linear layers, gradually reducing dimensionality from 768 to 128, 64, and

finally to a single output.

2. **Activation Functions**: **ReLU** activations after each linear layer introduce **non-linearity**, allowing the model to learn complex patterns.
3. **Sigmoid Output**: The final layer outputs a single value, passed through a **sigmoid** function to convert it to a probability, suitable for binary classification.

This design follows the **NN.Sequential** framework in PyTorch, which organizes the model layers into a straightforward pipeline.

5. Training the Model

We train the model using a **binary cross-entropy (BCE) loss** function, which is suitable for binary classification. Training involves calculating gradients, adjusting weights, and minimizing loss over multiple iterations:

1. **Loss Calculation**: The BCE loss computes the error between the predicted and actual labels.
2. **Gradient Computation and Backpropagation**: Using `loss.backward()`, we backpropagate through the model to compute the gradient of the loss with respect to each weight.
3. **Weight Update**: Each weight is updated by subtracting a small portion (learning rate) of the gradient from its value. This process, known as **gradient descent**, gradually optimizes the model to minimize loss.
4. **Epochs and Shuffling**: Multiple passes (epochs) over the dataset allow the model to improve iteratively. Shuffling data between epochs ensures the model generalizes well rather than memorizing patterns from batch order.

Learning Rate and Optimization: A fixed learning rate of 0.001 is used here. In more complex models, an adaptive or decreasing learning rate might improve performance by preventing the model from overshooting the optimal weights.

6. Evaluation and Testing

With the model trained, we evaluate its performance on a **test set**. Important evaluation metrics include:

- **Accuracy**: The percentage of correct predictions, which we compare against the baseline of 75%.
- **Loss Monitoring**: We observe the training loss over epochs, expecting a decreasing trend as the model learns.

Finally, we test our trained model on unseen data to gauge its ability to generalize. The test data provides insight into whether the model's performance is due to genuine learning or overfitting.

7. Debugging and Improvements

Iterative Debugging: Throughout the process, debugging and tuning are crucial. Common issues include syntax errors, incorrect dimension matching, or insufficient model complexity.

Further Improvements:

- **Model Complexity**: For higher accuracy, we could experiment with **deeper or more complex architectures** (e.g., adding dropout layers to prevent overfitting).
- **Learning Rate Scheduling**: Implementing a dynamic learning rate schedule may help the model converge faster and more accurately.
- **Cross-Validation**: For robust validation, we might use cross-validation to ensure that performance is consistent across different subsets of data.

Summary

This chapter has illustrated how to set up, train, and evaluate a neural network model for **binary classification**. By taking a molecular structure representation (SMILES) and transforming it into embeddings, we successfully trained a model to predict BBB transmission. While the model developed here is straightforward, the principles of data preparation, model structuring, and optimization apply broadly, from **drug discovery** applications to other domains requiring molecular predictions. As we advance, refining these models with more sophisticated architectures and data-handling techniques can further enhance their predictive power, especially for complex biomedical challenges.

36:00 Training a Neural Network: Techniques, Optimization, and Best Practices

In this section, we delve into the mechanics of **training neural networks** with a focus on improving

convergence speed, reducing error, and achieving model robustness. We cover **gradient descent**, **momentum-based optimizations**, and **adaptive learning rate methods**. These techniques are crucial for enhancing model performance, particularly when training on large and complex datasets.

1. Basic Gradient Descent and Stochastic Gradient Descent (SGD)

At the core of neural network training lies **gradient descent**, a process where the model's weights are iteratively adjusted to minimize a given **loss function**. Each adjustment aims to make the model's predictions closer to the true labels. This process involves:

- **Computing the Gradient**: The gradient of the loss with respect to each weight is computed, representing the direction in which the weight should be adjusted to reduce loss.
- **Weight Update Rule**: Each weight is updated by subtracting a small fraction of the gradient (scaled by a **learning rate**) from its current value.

This approach works best in theory when applied over the entire dataset, often called **batch gradient descent**. However, for computational efficiency, we typically use **mini-batches**—small subsets of the data—to approximate the gradient. This practice, known as **stochastic gradient descent (SGD)**, has several advantages:

- **Speed**: Smaller batches make each iteration faster, allowing for quicker adjustments.
- **Regularization Effect**: Mini-batch SGD introduces noise, which can help the model generalize better by avoiding overfitting to specific data points.

The choice of **batch size** (e.g., 32 in our example) impacts memory usage and gradient accuracy. Larger batches tend to yield more accurate gradient estimates but require more memory.

2. Momentum Methods

Momentum-based optimization methods introduce a concept of **momentum** to the weight updates. The idea is similar to physical momentum: once the update process starts moving in a particular direction, it maintains that direction, smoothing out oscillations and enabling faster convergence.

- **Momentum in Optimization**: For each weight, the update doesn't rely solely on the current gradient but also includes a fraction of the previous update. Mathematically, this is represented by adding a momentum term to the weight update equation. This term is scaled by a **momentum coefficient**, usually between 0.5 and 0.9.
- **Advantages**:
    - **Faster Convergence**: Momentum helps accelerate gradients in directions with consistent progress, which can lead to quicker convergence.
    - **Stability on Rough Loss Surfaces**: For optimization landscapes with many small "bumps," momentum helps the model avoid getting stuck, effectively smoothing out the optimization path.

Momentum is particularly useful in deep networks with complex loss surfaces, where standard SGD can struggle to make consistent progress.

3. Adaptive Learning Rate Methods

Another class of optimization methods is **adaptive learning rate methods**, which adjust the learning rate individually for each parameter based on historical gradient information. These methods include **Adagrad**, **RMSprop**, **Adam**, and **AdamW** (Adam with weight decay).

Adagrad (Adaptive Gradient)

- **Learning Rate Adjustment**: Adagrad modifies the learning rate based on the sum of squared gradients for each parameter. Parameters with frequently updated gradients receive smaller learning rates, while those with less frequent updates receive larger learning rates.
- **Strengths**: Useful for sparse data and features since it naturally scales the learning rate for each feature.
- **Limitations**: Adagrad's adaptive learning rate continuously decreases, which can sometimes cause the model to stop learning prematurely.

RMSprop (Root Mean Square Propagation)

- **Improvement over Adagrad**: RMSprop keeps a moving average of the squared gradients rather than

a cumulative sum, allowing the model to adapt the learning rate without diminishing it to zero.

- **Strengths**: Maintains a balance between high learning rates for infrequent updates and stability over time, especially useful in deep networks.

Adam (Adaptive Moment Estimation)

- **Combination of Momentum and RMSprop**: Adam combines the advantages of momentum (tracking moving averages of past gradients) and RMSprop (adjusting the learning rate based on past squared gradients).
- **Parameter Updates**: Adam maintains two moment estimates: the first for the gradient itself and the second for the square of the gradient. These estimates allow Adam to adapt the learning rate for each parameter.
- **Advantages**:
    - **Robust to Noisy Gradients**: Adam performs well on tasks with sparse or noisy gradients.
    - **Consistent Learning Rates**: Adjusts learning rates based on parameter-wise historical gradients, making it effective for complex optimization tasks.

AdamW is a variant of Adam that includes weight decay, a form of regularization that penalizes large weights, improving generalization by preventing overfitting.

4. Learning Rate Scheduling

While adaptive learning rates handle some adjustment needs, many models benefit from a **learning rate schedule** that varies the learning rate over time:

- **Step Decay**: The learning rate is reduced at specific intervals (e.g., every few epochs) by a fixed factor, slowing down updates as the model converges.
- **Exponential Decay**: The learning rate decreases exponentially over time, helping to stabilize the model as it nears optimal weights.
- **Cosine Annealing**: The learning rate oscillates within a range, gradually reducing as training progresses but allowing brief periods of higher learning rates.

Learning rate scheduling is particularly useful for preventing overshooting and settling the model near optimal weights.

5. Practical Implementation with PyTorch: Using Optimizers and Schedulers

In PyTorch, optimization is handled by classes that encapsulate various strategies, making it easy to implement momentum or adaptive learning rates. Let's go over setting up a **PyTorch optimizer** with AdamW and weight decay:

python

```
# Define the optimizer and specify hyperparameters
optimizer = torch.optim.AdamW(model.parameters(), lr=0.001, weight_decay=1e-5)


# Define a learning rate scheduler
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.5)
```

In this example:

- The **AdamW optimizer** handles the adaptive learning rate and momentum, while weight decay prevents overfitting.
- The **scheduler** halves the learning rate every 10 epochs, balancing stability and convergence speed as training progresses.

6. Avoiding Overfitting with Regularization

Overfitting occurs when the model performs well on training data but poorly on unseen data. Techniques to counteract overfitting include:

- **Weight Decay**: Adds a penalty proportional to the square of the weights, encouraging smaller weights and thus reducing overfitting.

- **Dropout**: Randomly "drops out" (sets to zero) neurons during training, preventing the model from relying too heavily on any specific set of neurons.
- **Batch Normalization**: Normalizes the input of each layer, stabilizing learning and enabling higher learning rates.

Summary

Effective neural network training requires balancing computational efficiency, convergence speed, and generalization to unseen data. Techniques like **momentum**, **adaptive learning rates**, and **regularization** allow us to train more robust models on complex datasets. By carefully selecting optimizers, tuning learning rates, and applying regularization, we can enhance the performance and stability of our models. This foundation is essential as we move into more advanced architectures and real-world applications.

40:30 Momentum Methods and Advanced Optimization Techniques for Training Neural Networks

In this section, we'll explore **momentum-based optimization techniques** and how they are integrated with adaptive methods, specifically **AdaGrad**, **RMSprop**, and **Adam**, to enhance neural network training. These methods build on basic gradient descent by using historical information to refine weight updates, making training more efficient and robust, especially in complex, high-dimensional spaces.

1. Momentum in Gradient Descent

Momentum is a technique designed to accelerate gradient descent by incorporating a fraction of the previous update direction into the current one. This effectively smooths the optimization path and speeds up convergence, especially on irregular surfaces.

Formula for Momentum Update:

$vt=\beta \cdot v_{t-1}+(1-\beta) \cdot g_t$

$\theta=\theta-\eta \cdot v_t$

Where:

- v_t is the momentum term at time t,
- g_t is the current gradient,
- β is the momentum coefficient (typically around 0.9),
- η is the learning rate,
- θ represents the parameters (weights) being updated.

In this formulation, the **momentum term** v_t is updated by combining a portion of the previous momentum v_{t-1} and the current gradient g_t. The parameter β\betaβ controls how much of the previous momentum is retained versus the current gradient's influence:

- **High Momentum (e.g., 0.9)**: Smooths out small fluctuations and preserves the direction, ideal for moving across rough surfaces.
- **Low Momentum (e.g., 0.1)**: Provides more direct following of the current gradient but lacks the smoothing effect.

This approach effectively applies an **exponential decay** on past gradients, meaning gradients from very early time steps lose influence quickly.

2. AdaGrad: Adapting Learning Rates per Parameter

AdaGrad (Adaptive Gradient Algorithm) modifies the learning rate for each parameter individually based on the accumulated gradients. Parameters with frequently large gradients have reduced updates over time, while infrequently updated parameters get larger updates. This behavior is particularly useful for sparse data where certain features or dimensions are updated less frequently.

AdaGrad Update Rule:

$$\theta = \theta - \frac{\eta}{\sqrt{G_{ii} + \epsilon}} \cdot g_t$$

Where:

- G_{ii} is the accumulated sum of squared gradients for parameter iii,

- ϵ is a small constant to prevent division by zero.

Advantages:

- Effective for sparse data since it emphasizes rare features.
- Helps convergence in early training steps by adapting the learning rate based on gradient magnitude.

Drawbacks:

- **Learning Rate Diminishes Over Time**: As the squared gradients accumulate, the learning rate can effectively become close to zero, slowing down training. This makes AdaGrad unsuitable for models that require a constant learning rate over long training sessions.

---

3. RMSprop: Addressing AdaGrad's Diminishing Learning Rate

RMSprop (Root Mean Square Propagation) addresses the limitation of AdaGrad by introducing an **exponential moving average** of past squared gradients, instead of a simple cumulative sum. This allows the learning rate to adapt based on recent gradients while avoiding the issue of vanishing learning rates.

RMSprop Update Rule:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

Where:

- $E[g^2]_t$ represents the moving average of squared gradients at time ttt,
- γ is the decay rate for the moving average, typically set around 0.9.

Key Benefits:

- Keeps the adaptive learning rate stable over time by focusing only on recent gradient information.
- Prevents the learning rate from diminishing to near zero, enabling the model to continue learning over long training sessions.

Applications: RMSprop is particularly effective in **recurrent neural networks** (RNNs) and **non-convex optimization problems**, where gradients may vary significantly.

---

4. Adam: Combining Momentum and RMSprop

Adam (Adaptive Moment Estimation) combines the advantages of **momentum** (tracking moving averages of past gradients) and **RMSprop** (adjusting learning rate based on recent squared gradients). Adam maintains both the first and second moment (mean and variance) estimates of gradients, making it one of the most widely used optimizers in deep learning.

Adam Update Rule:

1. Calculate first and second moments:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$$

2. Bias correction:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

3. Update parameters:

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

Where:

- m_t is the exponentially decaying average of past gradients (momentum term),
- v_t is the exponentially decaying average of past squared gradients,
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of m_t and v_t,
- β_1 and β_2 are decay rates for the moment estimates (default values are 0.9 and 0.999, respectively).

Advantages of Adam:

- **Automatic Adjustment**: Adam adapts learning rates for each parameter individually, making it well-suited for complex tasks with diverse parameter scales.
- **Resilient to Noisy Gradients**: The momentum term stabilizes training, especially when gradients are erratic.
- **Widely Effective**: Performs well across a variety of models and applications, from convolutional neural networks (CNNs) to transformers.

Practical Tips for Using Momentum and Adaptive Methods

1. **Choose Adam for General Applications**: Adam is widely used because it combines momentum and adaptive learning rates, making it highly versatile.
2. **Adjust Learning Rates**: While Adam performs well with default settings, specific tasks may benefit from adjusting η, β_1, and β_2.
3. **Consider Weight Decay (AdamW)**: For models prone to overfitting, **AdamW** adds a weight decay term to Adam, penalizing large weights and enhancing generalization.

By integrating **momentum** with **adaptive learning rates**, methods like Adam allow for efficient training, especially on deep and complex models. These advanced optimization techniques help to overcome the limitations of standard gradient descent, enabling better convergence and improved performance across diverse applications in neural networks.

47:30 Fitting a Simple Linear Model and Avoiding Overfitting

To illustrate the basics of model fitting, let's walk through a simple example of fitting a **linear model** to data, then discuss how using overly complex models can lead to **overfitting**.

Setting Up the Linear Model

In this example, we start with synthetic data generated from a **linear function** with some added noise:

y=ax+b+noise

where:

- a is the slope,
- b is the intercept,
- and the noise represents random variation around the line.

This kind of data is **linearly separable**, meaning a simple linear model should fit it well. Ideally, our model would have two parameters: one for the slope and one for the intercept.

Model Complexity and the Risk of Overfitting

When we fit models to data, the complexity of the model should align with the complexity of the data. If we use a model that's much more complex than the underlying data, we risk **overfitting**, where the model learns not only the underlying trend but also the noise or random fluctuations in the data.

Example:

- A simple linear model has two parameters (slope and intercept) that define a straight line, which is appropriate for our noisy linear data.
- If we instead fit a complex model (e.g., a neural network with many layers and nodes), the model has enough capacity to memorize each individual point, including noise. This leads to overfitting.

Loss and Overfitting in Training vs. Validation Sets

To understand overfitting, let's explore how **training loss** and **validation loss** behave over time:

1. **Training Loss**:
    - The training loss usually decreases steadily as the model learns to fit the training data.
    - If the model is complex, it can continue to decrease the training loss further by fitting not just the signal (true pattern) but also the noise, which drives down the training loss even more.
2. **Validation Loss**:
    - The validation loss, which measures how well the model generalizes to unseen data, typically decreases initially as the model learns general patterns.
    - However, if the model starts to overfit, the validation loss will stop decreasing and may start to increase, indicating that the model is no longer generalizing well.

This difference in loss behavior between training and validation sets is a key signal of overfitting. When training loss keeps decreasing, but validation loss increases, the model is likely memorizing specific data points rather than learning generalizable patterns.

Choosing the Right Model Complexity

To prevent overfitting and ensure that our model captures only the relevant patterns, it's important to:

- **Match Model Complexity to Data Complexity**: Use a simple model for simple data. For our linearly separable data, a linear model with just two parameters is sufficient.
- **Evaluate on a Validation Set**: Always set aside part of the data for validation to monitor overfitting.
- **Regularize the Model if Necessary**: Use regularization techniques, which we'll cover later, to penalize overly complex models, especially in cases where simpler models may not suffice.

Practical Implications of Overfitting

In more complex settings, such as deep learning models, overfitting becomes a prominent challenge due to the high capacity of these models to learn specific details. Techniques like **early stopping**, **dropout**, and **weight regularization** are often used in these cases to counteract overfitting. For simpler problems like linear regression, however, the best solution is to use a model with the appropriate complexity.

By understanding the balance between **model complexity** and **generalization**, we can select models that perform well not only on training data but also on unseen data, which is the ultimate goal of machine learning.

50:45 Bias-Variance Tradeoff in Model Complexity

In machine learning, understanding the **bias-variance tradeoff** is essential to finding the right balance between model complexity and performance. The **total error** in a model's predictions can be dissected into three main components: **bias error**, **variance error**, and **irreducible error**.

Bias

Bias refers to the error introduced by assumptions made in the model as it tries to approximate the true function underlying the data. In simple terms, bias represents how closely the model's predictions align with the actual values on average.

- **High Bias**: When a model has high bias, it consistently makes similar errors, tending to predict outcomes that are far from the true values. This usually occurs in **underfitted models**, which are often overly simple, as they lack the flexibility to capture the underlying patterns of the data. Imagine a low-complexity model like a linear regression trying to fit highly nonlinear data; it will produce high-bias predictions.
- **Low Bias**: A model with low bias closely aligns its predictions with the true values on average. For example, a more complex model with a high degree of flexibility, such as a deep neural network, can typically reduce bias by capturing intricate patterns in the data.

Variance

Variance measures how much the model's predictions fluctuate with different training sets. High-variance models are sensitive to the specific data they are trained on, often producing a wide spread of predictions across different samples.

- **High Variance**: When variance is high, predictions vary significantly with slight changes in the data. This often occurs in **overfitted models** that are highly complex and adapt too closely to the training

data, capturing noise along with signal. Such models will perform well on the training data but poorly on unseen data, as they fail to generalize.

- **Low Variance**: A model with low variance produces stable predictions that do not fluctuate drastically with changes in the training data. Simple models tend to have low variance because they are less flexible and less sensitive to individual data points.

Irreducible Error

The **irreducible error** is the inherent noise or randomness in the data that cannot be eliminated, even with a perfect model. This component often stems from **measurement errors** or **unpredictable variation** in the data-generating process.

- **Sources of Irreducible Error**: This error may come from limitations in the precision of measurement instruments or variability in the process being measured. For instance, if an instrument used to collect data has slight inaccuracies, these imperfections add irreducible error, which no model can eliminate.
- **Impact on Predictions**: Irreducible error sets a baseline for the best possible performance. Even with an optimal model, predictions will deviate from true values by this unavoidable margin.

The Tradeoff

Balancing **bias** and **variance** is crucial because increasing model complexity typically reduces bias but increases variance, and vice versa. This balance determines the model's **generalization capability** on unseen data:

1. **High Bias, Low Variance**: The model is too simplistic, failing to capture the complexity of the data (underfitting). This may produce consistently inaccurate predictions that remain stable across different datasets.
2. **Low Bias, High Variance**: The model is too complex and overfits the training data, resulting in predictions that are accurate on the training set but fluctuate wildly on new data (overfitting).
3. **Optimal Balance**: The goal is to find a model with enough complexity to minimize bias without incurring high variance, achieving a balance that minimizes total error and generalizes well to new data.

Understanding and managing the bias-variance tradeoff is essential in **model selection and tuning**. By adjusting model complexity and leveraging techniques like regularization, cross-validation, and ensemble methods, we can navigate this tradeoff to build models that achieve robust performance on both training and unseen data.

54:00 Regularization and Managing Overfitting in Neural Networks

In the context of neural networks, **regularization** techniques are essential for reducing overfitting by **adding constraints** to models with high complexity. These methods balance bias and variance to help the model generalize better to new data, addressing the classic **bias-variance tradeoff**.

Purpose of Regularization

When we have complex models with many parameters, there's a risk of the model becoming too good at fitting the training data, capturing noise along with the underlying pattern. Regularization techniques help manage this by slightly increasing the model's bias but significantly reducing variance. This adjustment helps avoid **overfitting** while guiding the model toward a more generalized form.

Common Regularization Techniques

1. **Early Stopping**
   Early stopping is a straightforward and effective approach. During training, we monitor the **validation loss** rather than just the training loss. When the validation loss begins to rise after initially decreasing, it suggests the model is starting to overfit. At this point, we can stop training to preserve the model state that best balances fit and generalization.
2. **L1 Regularization**
   L1 regularization (or **Lasso**) adds a penalty to the **absolute values of the model parameters**. This approach encourages the model to reduce unimportant weights to exactly zero, effectively simplifying the model by removing features with little impact. The L1 term is added to the loss function, penalizing the model for large parameter values and leading to a sparse model where only the most impactful parameters remain.
3. **L2 Regularization**
   L2 regularization (or **Ridge regression**) applies a penalty on the **squared values of the model**

parameters. Unlike L1, L2 does not drive parameters to zero but rather reduces their magnitude, preventing any single parameter from dominating the model. This "weight shrinkage" effect encourages the model to spread the learned information across all parameters, producing a more stable model.

4. **Elastic Net (Combination of L1 and L2)**
   By combining L1 and L2 regularization, **Elastic Net** balances sparsity with stability. This method includes both an L1 and L2 term in the loss function, leveraging the benefits of each. L1 forces some weights to zero, simplifying the model, while L2 keeps important features by spreading the influence across multiple parameters.
5. **Dropout**
   Dropout is a form of regularization where a random subset of **neurons is temporarily ignored** during each training iteration. This dropout rate (e.g., 20%) prevents the model from relying too heavily on any single neuron, making the network more resilient and reducing overfitting. Dropout can be applied to any layer in a neural network, fostering a level of redundancy that supports more generalization.
6. **Batch Normalization**
   Originally developed to improve training efficiency, **Batch Normalization** also acts as a regularizer by **normalizing the activations** within each mini-batch. This normalization helps stabilize the training process by smoothing the **loss landscape**, reducing the risk of gradient explosion or vanishing. Batch normalization indirectly improves generalization, allowing the model to handle higher learning rates and faster convergence.

Managing Hyperparameters in Regularization

With regularization comes additional **hyperparameters**, such as the L1 or L2 penalty strengths and dropout rates. **Cross-validation** is a key tool for tuning these hyperparameters, as it allows us to partition the training data into several subsets to iteratively validate and optimize these settings. By evaluating on different validation sets, we ensure the model's settings perform well across various data partitions, leading to better overall performance.

Visual Impact of Regularization on Model Fit

Regularization's effects on overfitting can be visualized as follows:

1. **Overfitted Model without Regularization**
   A model with no regularization will often fit the training data too closely, with complex, wavy predictions that interpolate exactly between training points. However, this complexity leads to poor generalization and nonsensical extrapolations outside the training data range.
2. **Early Stopping**
   With early stopping, we halt training as soon as the validation loss stabilizes, producing a model that captures the main trends without overfitting. The model's predictions are smoother and more stable across the training data range.
3. **L1 and L2 Regularization**
   - **L1 Regularization**: Applying L1 tends to zero out many weights, leading to a sparse model where only the most essential parameters remain. The model performs well on key data points but may appear piecewise linear due to the limited number of active weights.
   - **L2 Regularization**: With L2, the model still captures the main trends but spreads the effect across multiple weights, creating a more balanced and smoother fit than L1.

Choosing and Combining Regularization Techniques

Each regularization method has unique advantages, and combining them can yield robust models. For example, L2 regularization combined with dropout and batch normalization often produces highly stable, generalizable models. The choice of technique and parameters will depend on the data, the model's complexity, and the performance tradeoffs relevant to the application.

1:05:20 Adam Optimizer, L2 Regularization, and Weight Decay

The **Adam optimizer** has become one of the most widely used optimization algorithms in deep learning due to its effective combination of **momentum** and **adaptive learning rates**. The addition of **L2 regularization** with Adam introduces unique considerations, especially due to the optimizer's reliance on past gradients.

Why L2 Regularization Works Differently with Adam

1. **L2 Regularization in Gradient Descent**

    - Traditionally, L2 regularization is added to the **loss function**, and the resulting gradient is then used to update the parameters.
    - In simpler gradient descent scenarios, this approach is straightforward and effective: it reduces the parameter magnitudes uniformly, discouraging complex patterns and high variance.
2. **Challenges with Momentum-Based Optimizers**
    - Adam, and other momentum-based optimizers, maintain **momentum terms**—which store historical gradients to smooth out updates.
    - If L2 regularization is applied to the **loss function** directly with Adam, each step amplifies the regularization effect because the momentum accumulates, pushing parameters in specific directions and diluting the intended uniform penalization of L2.
    - This accumulation leads to **undesirable momentum build-up** in certain directions, effectively distorting the regularization effect.
3. **Weight Decay: An Improved Approach**
    - **Weight decay** addresses these issues by applying L2 regularization **directly to the parameters after computing the gradient**, rather than modifying the loss function.
    - This approach prevents the interaction with the momentum term, ensuring that the L2 penalty doesn't accumulate in unintended ways.
    - When using **AdamW** (Adam with weight decay), weight decay is implemented after the gradient computation, achieving a stable regularization effect without interfering with Adam's momentum dynamics.
4. **Practical Implementation**
    - In frameworks like PyTorch, using `AdamW` allows for easy inclusion of weight decay by specifying a `weight_decay` parameter, effectively simplifying the regularization process while preserving the optimizer's efficiency.

Dropout as a Regularization Technique

Dropout is another powerful regularization method, designed to prevent over-reliance on specific neurons and promote **distributed learning**:

- **Mechanism**: Dropout randomly ignores a subset of neurons during training. For each forward pass, a fraction (e.g., 50%) of neurons in a given layer are "dropped out" or temporarily deactivated.
- **Effect**: By dropping neurons, dropout forces the network to learn more generalized patterns across various subsets of neurons, reducing the risk of overfitting on particular neurons or connections.
- **Interpretation**: This introduces redundancy, as different neurons must learn to represent the same information, making the model more resilient and capable of handling new data.
- **Implementation**: Dropout is applied by inserting a `Dropout` layer with a specified dropout rate (e.g., `Dropout(0.5)` for 50%) after other layers in the model.

In practice, dropout tends to add a smoothing effect to predictions, as it prevents the model from overly complex fits. Below, you can see:

- **Dropout Application**: Adding 50% dropout to a model typically results in a smoother, more generalized fit within the data range.
- **Parameter Behavior**: Dropout creates a "cloud" of parameter values, representing the distributed learning across various paths. This can look like a dispersed range of parameter magnitudes, though the precise interpretation varies.

Batch Normalization as Implicit Regularization

Batch normalization not only **stabilizes training** but also acts as an implicit form of regularization:

- **Normalization**: Batch normalization standardizes layer outputs to have a mean of zero and a variance of one within each mini-batch. This prevents sharp changes in distributions between layers, which could otherwise cause instability and slow learning.
- **Regularization Effect**: By normalizing each batch independently, batch normalization introduces slight noise to the model, effectively regularizing it and reducing overfitting.

- **Practical Application**: Batch normalization can be applied after each layer in a model to achieve smoother, faster convergence and improved generalization.

Summary

1. **AdamW** integrates L2 regularization effectively without disturbing momentum-based optimization, using weight decay directly on the parameters post-gradient computation.
2. **Dropout** prevents overfitting by randomly deactivating neurons, forcing the model to rely on a distributed set of features.
3. **Batch Normalization** contributes both to training stability and model regularization by normalizing each mini-batch, limiting overfitting through added stochasticity.

Each of these regularization techniques offers specific advantages in controlling model complexity and enhancing generalization, especially when training deep neural networks on complex datasets. The choice of method should align with model architecture, dataset characteristics, and specific application goals to achieve an optimal balance between training efficiency and predictive performance.

1:09:30 Batch Normalization: Mechanism, Impact on Gradient Flow, and Empirical Success

Batch normalization (BatchNorm) is a critical technique in deep learning that stabilizes and accelerates training by normalizing the input of each layer, providing several core benefits including reduced sensitivity to hyperparameters and improved gradient flow.

The Gradient Flow Problem

In deep neural networks, backpropagation distributes gradients backward from the loss function to earlier layers. However, during this process, gradients can experience two major issues:

1. **Vanishing gradients**: As gradients propagate backward, they can diminish exponentially, particularly in deeper layers, making it difficult for early layers to learn effectively.
2. **Exploding gradients**: Conversely, gradients can also increase exponentially, causing instability and making it challenging to converge to an optimal solution.

Batch normalization counters these issues by normalizing the activations in each layer, setting the **mean** and **standard deviation** of each batch to zero and one, respectively. This controlled scaling and shifting of activations help maintain stable gradient magnitudes across the network, improving overall model training.

Mechanics of Batch Normalization

1. **Standardization**: Within each batch, BatchNorm normalizes each feature by subtracting the batch mean and dividing by the batch standard deviation. This process transforms the distribution to have a mean of zero and a variance of one across the batch.
2. **Scaling and Shifting**: To maintain representational flexibility, BatchNorm introduces two learnable parameters for each feature: a scaling factor (**gamma**) and a shift factor (**beta**). These parameters enable the model to retain or adjust the range of the normalized values as needed, allowing for **learned feature representations** that are not overly constrained by strict normalization.
3. **Empirical Effectiveness**: BatchNorm allows networks to use higher learning rates without sacrificing stability, often resulting in faster convergence. Additionally, it enables larger, deeper networks to train successfully without careful initialization.

BatchNorm and the Internal Covariate Shift Hypothesis

Initially, batch normalization was proposed to address **internal covariate shift**—the idea that layer inputs change distribution during training due to parameter updates in preceding layers, causing instability and necessitating frequent re-adjustment of learning rates.

However, further research, including a notable study from MIT, has suggested that this explanation may not fully capture BatchNorm's actual impact. Instead, the empirical success of BatchNorm is increasingly attributed to its ability to **smooth the loss landscape**:

- **Smoothing the Loss Landscape**: BatchNorm effectively reduces the sharpness and irregularity of the loss surface, making it easier for optimization algorithms to traverse without getting stuck in local minima. By stabilizing the learning process, it minimizes oscillations and fluctuations in gradient descent, allowing faster and more reliable convergence.

Practical Implementation and Usage

- **Where to Apply BatchNorm**: Batch normalization layers are typically added after linear or

convolutional layers and before activation functions. This placement maximizes the regularization effect while stabilizing activations.

- **Increased Robustness to Hyperparameters**: BatchNorm reduces sensitivity to hyperparameter choices such as learning rate and weight initialization, making it easier to design models with fewer adjustments.

Summary

Batch normalization is a powerful normalization technique that:

- Improves gradient flow through normalization, reducing both vanishing and exploding gradients.
- Allows stable training of deep networks, even with high learning rates.
- Smooths the loss landscape, facilitating better convergence and preventing models from getting stuck in local minima.

Although the precise reasons behind BatchNorm's effectiveness are still under investigation, its practical benefits for accelerating and stabilizing training make it a staple in neural network architectures, particularly for complex models. By understanding and implementing BatchNorm correctly, you can enhance model performance, reduce training time, and improve the network's generalization capabilities across diverse tasks and data distributions.

1:12:00 Final Evaluation and Reflections on Blood-Brain Barrier Model Performance

After multiple **training epochs**, our model achieved approximately **90% test accuracy** in predicting **blood-brain barrier (BBB) permeability**. This represents a significant improvement over a simple majority classifier (which would yield around 75% accuracy). However, the model is still not perfect, suggesting potential areas for further refinement.

Loss and Accuracy Trends

During training, the **training loss** steadily decreased across epochs, indicating the model's ability to minimize the difference between predicted and actual values within the training set. Observing the **test accuracy** reveals that our model was consistently improving in generalization, reaching a plateau around 90%. This indicates that, while the model generalizes well, there remains room to explore further optimization.

- **Loss Patterns**: As expected, the training loss decreased continuously, though it exhibited occasional spikes. These spikes are often due to stochastic variations introduced by mini-batch sampling, a standard effect in training.
- **Accuracy Patterns**: While accuracy generally increased, it did not follow a perfectly smooth trend, mirroring the fluctuations in loss.

Potential Model Improvements

To push beyond the current accuracy, several approaches could be explored:

1. **Model Architecture**:
    - The current model is a **simple three-layer neural network**, which may not capture complex structural nuances of molecular data.
    - Increasing the number of layers or incorporating **convolutional layers** might improve performance.
    - **Graph Neural Networks (GNNs)** could provide a more nuanced understanding of molecular structure by representing molecules as graphs rather than simple embeddings, potentially capturing relational and spatial information between atoms.
2. **Embedding Refinement**:
    - The input embeddings, derived from a transformer trained on **SMILES strings**, may lack detailed structural insights. Alternate representations, such as **fingerprint-based embeddings** or **3D molecular representations**, could capture molecular features more effectively, enhancing the model's decision-making capacity.
3. **Enhanced Regularization**:
    - Adding regularization techniques such as **Dropout** or further tuning of **L2 weight decay** may prevent the model from overfitting to training data, particularly if a more complex model is employed.

Interpreting Confidence in Predictions

In classification tasks, models often output probabilities indicating confidence in their predictions. Ideally, the model should avoid **overconfident predictions**—e.g., assigning probabilities near 1 or 0 for every prediction—unless it has substantial evidence for such certainty.

- **Confidence vs. Accuracy**: Even if a model achieves high accuracy, the loss function will continue to decrease only if the model becomes increasingly confident in its predictions. This quest for high confidence is a potential source of **overfitting**, as the model may adjust its weights to fit specific samples too precisely, thereby reducing generalization to new data.
- **Impact on Loss Metrics**: The loss function penalizes the model based on the difference between predicted probabilities and true labels. Thus, achieving a low loss requires not just correct predictions but high-confidence predictions. This balancing act explains why **loss** may decrease while **accuracy** fluctuates, as accuracy measures only correctness, whereas loss also considers confidence.

Next Steps and Closing

This concludes our in-depth exploration of **small molecule modeling for BBB prediction**. In the coming sessions, we will transition to topics such as **Genome-Wide Association Studies (GWAS)** and **deep learning applications in genomics and life sciences** led by Professor Kellis. These sessions will expand our understanding of how machine learning and neural networks are revolutionizing our approach to complex biological and genetic data.

This wraps up our journey through the fascinating application of deep learning for drug discovery, offering a solid foundation and insights into the challenges and future potential in this domain.

Lecture 17 - Genetics, Disease, GWAS, PRS, Mechanism

Video: Lecture17 Disease GWAS PRS Mechanism

Slides: Lecture17_Genetics_GWAS_PRS_Mechanism.pdf

00:00 Genetics: From Ancient Times to Today

Introduction to Genetic Understanding in Disease

In this lecture, we bridge previous insights into protein modeling, chemical structure, and genomic understanding to address a critical question: how do genetic mechanisms contribute to disease? By synthesizing these domains, we can begin to dissect the genetic basis of human diseases and uncover pathways that lead to effective therapeutic strategies. The focus is to leverage genetic information not only to identify regions associated with disease but to reveal actionable mechanisms that can be targeted or modified to reverse disease pathways.

Genetics and the Role of Genome-Wide Association Studies (GWAS)

Genome-wide association studies (GWAS) enable an **unbiased exploration of genetic regions** across the genome that are statistically associated with disease traits. GWAS scans the genome to identify common genetic variants linked to specific diseases, delivering a region-based view of disease risk. This method, while powerful in its scope and scale, identifies **associations without mechanistic insight**. Thus, while GWAS can pinpoint regions of interest, understanding the underlying biological mechanism remains a complex and necessary challenge for translating these associations into therapeutic insights.

Core Topics in Genetic Variation, GWAS, Polygenic Risk Scores, and Mechanistic Analysis

1. **Genetic Variation**: A foundational understanding of genetic diversity across human populations, encompassing how this diversity manifests in phenotypic traits and disease susceptibility.
2. **GWAS Techniques**: Approaches and statistical frameworks used to identify regions of the genome with genetic variants associated with diseases, paving the way for understanding genotype-phenotype relationships.
3. **Polygenic Risk Scores (PRS)**: Calculation and utility of PRS involve aggregating small contributions from multiple genetic variants to predict an individual's overall genetic predisposition to certain diseases. PRS operates independently of the mechanisms of individual variants, focusing solely on cumulative risk.
4. **Mechanistic Dissection**: Translating GWAS findings into mechanistic insights requires additional analysis and functional validation. The goal is to move from associative data to **causal understanding**

of disease pathways, enabling interventions at a molecular or cellular level.

Historical Context: Ancient Genetic Manipulation Through Selective Breeding
From early agricultural societies, humans have influenced genetic selection, reshaping crops, animals, and ecosystems through selective breeding practices. By consciously selecting plants for taste, resilience, or productivity, and animals for specific traits like strength or temperament, humans have long driven the evolutionary trajectories of these species. **Coevolution between plants and animals**—such as the relationship between apple trees and horses, where animals help spread seeds in exchange for nutrition—demonstrates early human influence over genetic outcomes. This historical context underscores the long-standing human understanding of inheritance and the power of selection, even before formal genetic principles were established.

The Revolution of Mendelian Genetics
With Gregor Mendel's experiments in pea plants, the principles of **discrete inheritance** emerged, providing the basis for modern genetics. Mendel's work identified dominant and recessive traits and demonstrated that traits are inherited independently, a foundational insight that would later define the concept of genes and alleles. Although Mendel's findings initially went unrecognized, they laid the groundwork for understanding how traits could be inherited without blending, despite the continuous variation observed in traits like height or eye color. His experiments revealed that continuous traits could arise from the combined effects of multiple discrete genetic factors, thus bridging the gap between **phenotypic continuity and genetic discreteness**.

This section sets the stage for understanding how genetic variation, combined with advanced tools like GWAS, enables modern scientists to explore the genetic architecture of diseases with unprecedented detail and precision.

17:07 Disease-associated Variants

Identifying Genetic Variants Linked to Disease
Disease-associated variants are genetic mutations that increase or decrease an individual's risk for specific conditions. For instance, certain mutations in the *ARMS2*, *TP53*, *SIN3*, and *C2* genes have been linked to an elevated risk for age-related macular degeneration (AMD), a condition that leads to progressive central vision loss as individuals age. Understanding the genetic basis of such diseases, and identifying specific variants that contribute to increased susceptibility, enables us to target these pathways for potential treatments. This knowledge serves as the foundation for precision medicine, where interventions can be tailored to an individual's genetic risk profile.

From Association to Treatment
An example of the potential impact of these insights is in therapeutic development. By identifying genetic variants that elevate AMD risk, researchers and companies, like the one led by Dr. Leon Sad, have developed compounds specifically designed to address the molecular mechanisms associated with these genetic variants. Thus, genetic information does more than just predict risk—it can directly inform treatment options. This pathway from genetic association to targeted therapy demonstrates the value of understanding disease mechanisms at the genetic level.

Methods for Identifying Disease-associated Variants
The process of identifying these disease-associated variants involves analyzing the genome across a large number of individuals. In studies examining AMD or other traits, researchers use genome-wide scans to compare genetic variation in affected individuals (cases) against those without the condition (controls). By testing each variant across the 23 pairs of human chromosomes, scientists evaluate the statistical likelihood that specific variants are associated with the condition in question, rather than occurring by random chance.

Case-Control Studies and Variant Analysis
In practice, these analyses rely on extensive **case-control studies**, where genetic data from patients with the condition are compared against a control group matched by age and other relevant factors. Differences in variant frequencies between cases and controls help highlight **loci** that are potentially causal or at least significantly associated with disease risk. This approach also allows researchers to classify variants as either **risk-increasing** or **protective**, based on whether their presence is associated with higher or lower incidence of the disease.

By understanding disease-associated variants through these comparative studies, we can not only assess an individual's risk for diseases but also identify points of intervention that can lead to effective therapeutic options.

19:19 Challenge of Mechanism

Understanding Variant Function Beyond Association

Identifying genetic variants associated with diseases presents an immediate follow-up challenge: deciphering *how* these variants contribute to disease mechanisms. Most variants reside in a complex genomic context that includes both coding and non-coding regions, each carrying unique functions and regulatory roles. For example, within protein-coding regions, variants might be directly involved in altering protein structure or function. But in non-coding regions—where many disease-associated variants are found—their roles are less straightforward, often influencing gene expression, splicing, or other regulatory processes without directly coding for proteins.

Annotating Functional Genomic Regions

Functional annotations help clarify the role of variants. For example, a variant within an exon may change the amino acid sequence of a protein, whereas variants in introns or intergenic regions may alter regulatory elements, such as enhancers or promoters. In a sequence where *ATG* denotes a start codon for an exon and adjacent splice sites signal boundaries for intron removal, we can begin to parse out coding from non-coding influences. Some disease-associated variants, however, lie in regions like upstream regulatory sequences or intronic spaces within unrelated genes, making their functional implications more ambiguous. A variant upstream of *TP53* or within an intron of *SIN3* could potentially impact nearby or distant gene expression or interact with non-coding RNAs, such as microRNAs or tRNAs, affecting various levels of gene regulation.

Key Questions and Steps in Mechanism Exploration

To address these challenges, we aim to:

1. **Catalog Genetic Variants**: Establish a comprehensive map of single nucleotide polymorphisms (SNPs) and other genetic variants in the human genome.
2. **Associate Variants with Disease**: Use genome-wide association studies (GWAS) to link these variants with specific diseases, identifying loci that significantly correlate with disease traits.
3. **Translate Association to Mechanism**: Leverage GWAS results and functional annotations to dissect disease mechanisms. This includes examining variant enrichments in functional regions and integrating multi-omic data to pinpoint regulatory networks or pathways affected by these variants.
4. **Pathway to Therapeutic Development**: With an understanding of variant impact, we can explore therapeutic strategies aimed at modifying the affected pathways or reversing the underlying disease processes.

Each of these steps moves us closer to translating genetic associations into actionable insights for understanding, diagnosing, and treating complex diseases.

21:25 Variation and Disease

Human Genetic Variation

Each human cell contains two copies of the genome, one from each parent, divided into 23 chromosomes with a total of 3.2 billion DNA letters. Despite only 1.5% of the genome encoding approximately 20,000 genes, millions of polymorphisms—variations in DNA sequence—exist across individuals, providing unique genetic "fingerprints." These polymorphisms, or alleles, arise from different forms of a gene or DNA sequence across populations.

Types of Genetic Variation

1. **Single Nucleotide Polymorphisms (SNPs):** These occur when a single nucleotide, such as cytosine (C), mutates to another base, like guanine (G). SNPs happen roughly once per 1,000 bases in the genome, allowing vast genetic diversity and potential disease associations.
2. **Insertions and Deletions (Indels):** These variations involve the addition or removal of one or more nucleotides, occurring approximately once every 10,000 bases. Indels can disrupt gene sequences or regulatory regions, impacting gene function.
3. **Short Tandem Repeats (STRs):** STRs involve repeating sequences, like GTC repeated multiple times. Variability in STR length can lead to disease if it alters protein structure or expression, especially within coding regions.
4. **Structural Variants or Copy Number Variants (CNVs):** Large-scale structural changes, including deletions, duplications, or inversions, occur roughly every million bases. CNVs can affect gene dosage, leading to an increased or decreased number of gene copies, often with significant phenotypic effects.

Impact of Genetic Variants on Disease
Genetic variants contribute to disease by altering protein structure or expression. For example:

- **Sickle Cell Anemia:** A single amino acid substitution in hemoglobin—replacing glutamic acid with valine—causes red blood cells to deform, leading to severe health issues if inherited from both parents. However, carrying one copy is protective against malaria, illustrating how a disease-associated variant in one population context can be beneficial in another.
- **Huntington's Disease:** Caused by an expanded repeat of the CAG trinucleotide within the *HTT* gene. Individuals with more than 30 CAG repeats develop Huntington's, leading to neurodegeneration, movement issues, and dementia. Repeat expansions like these can profoundly affect protein function and stability.
- **Cystic Fibrosis:** In the *CFTR* gene, a deletion of a single amino acid results in cystic fibrosis, a disorder that impairs lung function and causes chronic infections.

Mapping Variants to Disease Mechanisms
Studying these types of variations helps us link genetic changes to specific diseases by examining how variations affect protein function, gene regulation, or cellular pathways. Disease mechanisms, once identified, can reveal therapeutic targets and intervention strategies that address the root cause rather than merely managing symptoms.

25:49 Defining Variants

Understanding Variant Frequency
To study genetic variation and its implications in disease, it is essential to classify variants by frequency. Variants can be categorized as:

- **Common Variants:** Present in more than 5% of the population.
- **Low-Frequency Variants:** Found in less than 5% of individuals.
- **Rare Variants:** Extremely uncommon, sometimes observed in only one individual.
- **Private or De Novo Variants:** Unique mutations that do not appear in the individual's parents; these occur spontaneously in the germline or during somatic cell divisions.

Allele Types and Population Specificity
Variants can also be classified based on population data:

- **Major vs. Minor Allele:** A variant's status as "major" or "minor" depends on its frequency within a particular population. For example, an allele with 0.3% frequency in Europeans would be the minor allele.
- **Reference vs. Alternate Alleles:** Refers to the version found in a selected reference genome.
- **Ancestral vs. Derived Alleles:** By comparing the human genome to our closest evolutionary relatives (e.g., chimpanzees), we can identify which allele likely represents the ancestral state. Derived alleles represent genetic changes from this ancestral version.

The Human Genome and Diverse Genetic Backgrounds
The first human genome sequence, finalized in 2003, represented one individual's DNA and served as a reference. However, to understand global genetic diversity, scientists now sequence genomes from a broad range of populations, revealing distinct regional allele patterns. Most human genetic diversity originates from African populations, reflecting humanity's origin. Migration out of Africa around 50,000 years ago introduced this variation globally, with new variants arising as humans spread across continents.

Cataloging Variants with SNP Arrays
Once common variants, such as SNPs, are cataloged, genotyping can be streamlined. Rather than resequencing entire genomes, researchers use SNP arrays, which contain probes specific to known variant sites. Each probe can detect the presence of variant forms (e.g., "C" or "G" at a particular SNP) through hybridization, making it possible to determine an individual's genotype at thousands of sites efficiently.

This foundational understanding of variant classification and cataloging enables targeted genetic studies, essential for linking specific genetic changes to disease and furthering the study of human evolution and population genetics.

31:18 Common Variants and Haplotype Blocks

Profiling Common Variants with DNA Microarrays

Advances in DNA microarray technology from the 1990s enabled the efficient profiling of genetic variation. By hybridizing DNA samples to probes, researchers can identify which version of an allele a person carries. For instance, a particular SNP may show red if it's "GG," green if "AA," and yellow if "AG," simplifying genotyping.

Haplotype Blocks and Co-inheritance Patterns
Genomic variants are often inherited as groups known as **haplotype blocks**. These are chunks of DNA within which genetic variants are frequently co-inherited due to limited recombination events, called hotspots, that break up these blocks across generations. The pattern of co-inheritance within a block reveals regions of the genome where variants are closely linked, meaning that if an individual has one variant in a block, they likely carry others within that same block.

Population Differences in Haplotype Structure
The structure of haplotype blocks can differ significantly between populations. For instance, African, European, and Asian populations may show variations in haplotype size and recombination frequency. For example, East Asian populations might display additional recombination events within a haplotype that aren't observed in European populations, leading to smaller blocks and more genetic diversity.

Advantages and Challenges of Haplotype Blocks in Disease Association
Haplotype blocks offer both benefits and drawbacks in genetic studies:

- **Advantage:** Researchers can use a single SNP within a haplotype to infer the presence of other variants, reducing the number of SNPs that need to be measured directly. This approach allows for efficient genotyping, especially useful in large-scale studies.
- **Challenge:** Haplotype blocks complicate pinpointing causal variants. When a disease is associated with a haplotype block, it becomes difficult to identify the exact variant responsible since all variants in the block are inherited together and thus appear to be co-associated with the disease.

New Mutations Within Existing Haplotype Blocks
Mutations continue to arise within existing haplotype blocks. These new variants accumulate as populations diverge geographically, contributing to regional genetic diversity. For example, specific alleles may define lineages unique to certain populations or regions, such as Asia or the Americas. Thus, genetic diversity expands even within stable haplotype structures, with new mutations creating additional layers of complexity.

Understanding haplotype blocks provides essential insights into the patterns of human genetic variation, helping researchers interpret genetic associations and their implications across different populations. This framework allows for more targeted exploration of disease-linked variants while acknowledging the evolutionary context in which they arise.

38:03 GWAS Genome-wide Association Studies: Genotype-Phenotype

Objective of GWAS
A genome-wide association study (GWAS) seeks to correlate specific alleles with traits or diseases across the genome. It does this by testing each common genetic variant for associations with a particular phenotype. The statistical significance of these associations is quantified, typically represented as a **negative log10 p-value**. Higher values indicate stronger evidence that an association is non-random, and thus more likely to be biologically meaningful. Results are often visualized as **Manhattan plots**, where peaks represent regions with strong associations.

Understanding Recombination and Haplotype Blocks
In GWAS, peaks in the Manhattan plot often reflect haplotype blocks that carry an association signal. Due to co-inheritance within these blocks, nearby SNPs exhibit correlated associations, creating "skyscraper" peaks. Recombination hotspots mark the boundaries of these blocks, and understanding the co-inheritance patterns between variants helps researchers interpret the structure of associated regions. Some hotspots can further fragment a region, potentially allowing finer resolution in pinpointing causal variants, though often, haplotype structure limits this precision.

Adjusting for Statistical Inflation
Correcting for the sheer number of tests in GWAS is essential to avoid false positives. With approximately a million independent tests (considering haplotype blocks), the accepted genome-wide significance threshold is a p-value of 5×10^{-8}, correcting for multiple hypotheses. This ensures that only robust associations surpass this stringent threshold.

Addressing Confounders in GWAS
Several confounding factors can introduce bias in GWAS. **Population structure** can cause associations linked

to ancestry rather than function. Similarly, **relatedness** between individuals, **sex biases**, and **population genetics factors** like Hardy-Weinberg equilibrium deviations must be controlled to ensure accurate results. Historically, lack of stringent control led to a "GWAS crisis," with numerous associations failing replication. Improved methodologies, including standardized significance thresholds, have since enhanced the reliability of GWAS findings.

Limitations of GWAS: Bias Toward Common Variants
GWAS has traditionally focused on common variants due to the limited number of genomes available for early studies. This means associations are often biased toward common variants with higher minor allele frequencies, generally over 5%. Although capturing nearly all common variants, these studies often miss rare variants, leading to an ascertainment bias. Thus, while GWAS is powerful, it has inherent limitations in detecting associations with rare genetic variants, which may also play crucial roles in disease.

50:52 Common/weak-effect vs. Rare/Strong-effect variants

Nature of Common and Rare Variants
Common genetic variants typically have weak effects on traits or diseases. This is because natural selection filters out highly deleterious mutations over generations; any variant with a strong negative impact on fitness is less likely to be passed down. For example, a variant that drastically increases the risk of schizophrenia would likely diminish in prevalence as it affects social functioning and, consequently, reproductive success.

Exception to the Rule
Some common variants have strong effects, such as the APOE ε4 allele, associated with a significantly increased risk of Alzheimer's disease. Despite its strong effect, APOE ε4 remains common (around 5-10% in certain populations). This could be due to **genetic drift** or **historical selection** that did not account for longer lifespans, where Alzheimer's typically manifests. Environmental shifts, like the prevalence of high-calorie diets in modern societies, can also amplify effects of certain alleles, as seen in variants that correlate with obesity post-World War II in industrialized nations. These shifts illustrate how environmental context can alter the impact of genetic variants over time.

Evolutionary Dynamics and Disease Association
The interaction between genetic variation and environment can change how variants are selected for or against. For instance, the sickle cell mutation confers a survival advantage against malaria in heterozygous carriers, leading to its higher prevalence in malaria-endemic regions despite its association with sickle cell disease when homozygous. This demonstrates how alleles beneficial in one environment may become detrimental in another, illustrating the complex relationship between **selection, environmental pressures, and genetic variation**.

Implications Beyond Disease
This concept extends to non-disease traits. A variant affecting an outwardly visible trait (phenotype) without impacting fitness might persist or even become prevalent if it influences **sexual selection**. For example, a culturally favored trait—like a particular hair color—could rise in frequency within certain populations due to mating preferences, illustrating that visible phenotypic consequences can impact allele frequencies independent of survival advantage.

Limitations of GWAS for Rare Variants
While GWAS effectively identifies common variants with weak effects, it struggles with rare, strong-effect variants due to limited sample sizes in initial studies and the bias of microarrays toward common alleles. Whole-genome sequencing can detect rare variants, but the sheer number of rare alleles introduces noise, making it challenging to discern causal associations. Identifying rare variants with strong effects requires large cohorts to achieve sufficient power and to address the high rate of false positives from **multiple hypothesis testing**.

In summary, the study of genetic variation reflects a balance between selective pressures, environmental changes, and population dynamics, highlighting the nuanced complexity of both common and rare variants in human health and traits.

59:06 Family studies (Linkage Analysis) vs. Unrelated Individuals (GWAS)

Linkage Analysis for Rare Variants in Families
Linkage analysis, or traditional genetics, focuses on rare, strong-effect variants by studying families with a high incidence of a disease, such as cystic fibrosis. This approach uses the **shared inheritance patterns** among affected family members to narrow down the genomic regions likely containing causal variants. By examining siblings who are affected and those who are not, researchers can progressively eliminate regions of the

genome that are not associated with the disease. The more affected individuals within the family, the more precisely linkage analysis can pinpoint potential regions. By aggregating data from multiple families with similar genetic conditions, researchers can identify the exact genes involved, which may harbor the rare, disease-causing mutations.

GWAS for Common Variants in Unrelated Individuals
GWAS, on the other hand, is well-suited for identifying common variants that increase disease risk across unrelated individuals. By examining a large population and correcting for multiple hypothesis testing, GWAS identifies **statistically significant associations** between specific genetic variants and disease prevalence. This approach, however, is typically limited to common variants due to the large sample size required to detect associations with rare variants. For common variants, a study with around 400 cases might achieve genome-wide significance for a three-fold increase in risk.

Challenges in Detecting Protective Variants
Protective variants, which decrease disease risk, present a unique challenge. Identifying families or large cohorts that lack a specific disease is much more complex because these protective traits may not be as readily identifiable. Consequently, studies on protective variants often require **significantly larger sample sizes** (e.g., 30,000 families) to reach statistical power, as it's harder to assemble enough cases where the absence of the disease can be confidently linked to specific genetic traits.

Isolated Populations and Specific Disease Frequency
One strategy for both linkage analysis and GWAS is to study isolated populations where certain diseases are either more or less prevalent. By examining genetic variants within these populations, researchers can identify alleles that either increase or decrease disease risk. This approach benefits from **population-specific allelic distributions** and can reveal insights into both risk-increasing and protective variants.

In summary, linkage analysis excels in identifying rare, high-impact variants within families, while GWAS is optimal for mapping common variants in diverse populations.

1:04:05 Population Differences, Ancestry Painting, PCA, Multi-Ethnic Fine-Mapping

Population-Specific Genetic Variants
The variation in genetic diversity among populations affects how many variants we capture when sequencing genomes. For example, populations with historical isolation, such as Finland, exhibit fewer genetic variants per genome compared to more genetically diverse populations, such as the Yoruba in Africa. The degree of variation also differs within populations, influenced by historical migration patterns, genetic drift, and environmental adaptation.

Tracing Population History and Genetic Bottlenecks
By examining genetic variants and calculating coalescence times—points at which ancestral genetic lines converge—researchers can estimate effective population sizes at various historical points. Many human populations experienced genetic bottlenecks in the distant past, followed by expansions. These bottlenecks and expansions are recorded in genetic data, revealing population history embedded within our genomes.

Ancestry Painting and Principal Components Analysis (PCA)
To understand the ancestral origins of genome segments, we use ancestry painting, which colors different parts of the genome based on inferred ancestral components. By performing PCA on genetic data, researchers can separate genome diversity into principal components that correlate with geographic and ancestral backgrounds. For instance, principal components often reveal continental and regional differences, with distinct genetic clusters corresponding to geographic locations, effectively mapping populations based on genetic similarity.

Mapping Individual Genomes
Ancestry painting allows us to decompose an individual's genome into segments linked to specific ancestral backgrounds, such as subsaharan African, European, or Native American heritage. This process visualizes recombination events over generations, showing the origins of specific genome segments. The ancestral contributions reveal complex histories of population mixing and migration.

PCA and Geographic Correlation
Within Europe, for example, PCA can place individuals from different countries along axes that correspond to geographical directions. Principal component one often correlates with north-south variation, while principal component two aligns with east-west differences. This distribution suggests that human migration is relatively slow, resulting in geographically localized genetic variation.

Multi-Ethnic Fine-Mapping
To refine disease associations identified in genome-wide studies, researchers use fine-mapping, which becomes particularly powerful when incorporating multiple ethnic groups. By comparing associations across populations, such as Africa and Europe, researchers can identify overlapping genomic regions more likely to contain causal variants. Multi-ethnic mapping thus improves accuracy in pinpointing disease-related variants by leveraging differences in linkage patterns across diverse populations.

Functional Fine-Mapping and Bayesian Methods
In addition to purely genetic data, functional annotations—such as conservation levels or regulatory roles of genomic regions—aid in fine-mapping efforts. Bayesian models combine genetic association data with functional annotations to assign posterior probabilities, narrowing down candidate regions or even identifying single, likely causal variants for diseases.

1:12:41 Polygenic Scores (PGS) and Polygenic Risk Scores (PRS)

Overview of Polygenic Risk Prediction
Polygenic risk scores, once viewed skeptically, have now gained significant traction in clinical settings. With advancements in data and statistical approaches, **PRS enables prediction of disease risk by aggregating the effects of multiple common genetic variants across the genome**. As of 2022, organizations like the American Heart Association advocate for using genetic profiling to assess individual risk, particularly in conditions like cardiovascular disease, suggesting its applicability in clinical decisions.

Individual vs. Population-Level Predictive Power
Historically, rare variants with strong effects were the focus of genetic predictions for individuals, given their significant but infrequent impacts on conditions like monogenic diseases. **PRS shifts focus to common, weak-effect variants**—individually, they contribute minimally to disease risk, but collectively, they can be powerful predictors. For a population, the cumulative effect of these weak variants can surpass that of rare variants, making PRS highly valuable in assessing the genetic risk across diverse individuals.

Calculating Polygenic Risk Scores
Each variant in the genome has an associated **effect size, or beta coefficient**, which quantifies its impact on disease risk. To calculate an individual's PRS, the beta coefficients of their risk-increasing and protective variants are summed across the genome. This **summed value represents the polygenic score** for the individual, positioning them within a distribution of risk across the population. Individuals with higher PRS values fall into higher risk categories, while those with lower scores are at reduced risk.

Polygenic Scores in Cardiovascular Risk Prediction
When PRS is combined with traditional risk factors—such as high cholesterol, hypertension, and lifestyle choices—the combined model enhances prediction accuracy. Notably, **PRS alone can outperform any single conventional risk factor** and, when added to these factors, improves the precision of risk estimates for conditions like atrial fibrillation and heart disease. Consequently, PRS is now considered a valuable addition to clinical risk assessments.

PRS and Family History
PRS also complements family history, an established but limited predictor. For instance, individuals with a **negative family history but a high polygenic risk score** may still have a substantial disease risk, potentially greater than those with a positive family history but a low PRS. This interaction highlights how PRS captures **genetic variation at the individual level** that family history alone may miss.

Implications for Lifetime Risk and Clinical Use
PRS can help estimate lifetime disease risk, adjusting for age and other individual factors. As data accumulates, PRS is expected to inform **personalized prevention and intervention strategies** across medical practice, moving genetic risk prediction from academic to practical applications in healthcare.

1:18:53 Mechanistic Insights from Genome-Wide Functional Enrichments

Leveraging Functional Genomic Annotations to Decode Mechanisms
When identifying disease-associated loci, the primary challenge is determining how these genetic regions contribute to disease. For example, the **FTO locus**, one of the strongest genetic associations with obesity, contains multiple variants across a large span, any of which might be causal. By integrating **RNA expression data, epigenomic profiles, and computational models**, researchers can dissect which variants are most likely to disrupt regulatory motifs, affect enhancer activity, or influence specific gene expression patterns in relevant tissues.

Building a Functional Circuitry Map
The power of combining genome-wide data with regulatory insights allows for detailed mapping from variant to effect. Instead of a broad association with a genetic region, one can pinpoint specific **variants overlapping enhancers** and other functional elements. This approach includes **annotating enhancers by cell type**, identifying which variants disrupt binding sites for regulatory proteins, and linking these enhancers to their target genes. By leveraging this enriched functional data, researchers can build a **mechanistic map**, identifying how specific variants in the FTO locus influence gene expression and ultimately phenotype.

Trait-Specific Enrichment Patterns Across Tissues
Genome-wide functional enrichments reveal that **genetic variants associated with specific traits often map to enhancers active in related tissues**. For example, genetic variants linked to height are enriched in enhancers active in embryonic stem cells, while immune-related traits are enriched in enhancers active in T-cells and B-cells. This tissue-specific enrichment extends across various traits, illustrating a robust relationship between trait-associated variants and tissue-specific enhancer activity. This functional insight is particularly crucial as the majority of disease-associated variants reside in **non-coding regions**, where they exert regulatory rather than protein-coding effects.

Surprising Associations and Implications for Disease Mechanisms
Notably, this approach has led to unexpected findings, such as the association of Alzheimer's disease variants with **immune-related enhancers** rather than brain-specific regions. This finding, initially identified through combined genetic and epigenomic analysis, suggests an **immune component to Alzheimer's disease**, which has since been validated by further studies on microglial cells.

By **linking genome-wide association data with functional annotations**, researchers can derive mechanistic insights that not only clarify how specific variants contribute to disease but also provide new directions for potential therapeutic interventions. This integration of genetic, epigenomic, and computational data offers a transformative approach to understanding complex diseases at the mechanistic level.

1:25:18 Summary
In this lecture, we covered a comprehensive framework for understanding **human genetic variation** and its implications for disease mechanisms. We discussed how **ancestry and population differences** influence both common and rare variants and explored the distinctions between different ancestries, as well as methods like **fine mapping** that help pinpoint genetic influences on disease. Additionally, we introduced **polygenic risk scores (PRS)**, illustrating their predictive capabilities for assessing disease susceptibility across populations.

We also examined **genome-wide functional enrichments**, which allow us to connect genetic associations with specific tissues and pathways, shedding light on **global disease mechanisms**. This approach is foundational in understanding how various traits are linked to specific tissue functions and biological processes, particularly in non-coding regions.

Looking forward, we will delve deeper into **disease-specific circuitry**, tracing the path from genetic variants to affected cell types, target genes, specific nucleotides, regulators, and ultimately to the **phenotypic outcomes at both cellular and organismal levels**. This progression will further enrich our understanding of genetic architecture and its applications in personalized medicine and therapeutic development.

Lecture 18 - Disease Mechanism, circuitry, eQTLs, heritability

Video: Lecture18 - Disease Mechanism - MLCB24

Slides: Lecture18_DiseaseMechanism_Circuitry_eQTLs_Heritability.pdf

00:00 Lecture Overview
In today's lecture, we transition from broad genetic analyses to the **mechanistic understanding of disease** using individual loci and biological pathways. We will cover:

1. **Disease Mechanism and Circuitry**: From genome-wide associations to the functional interpretation of individual loci.
2. **Expression Quantitative Trait Loci (eQTLs)** and **Methylation Quantitative Trait Loci (mQTLs)**: Using molecular traits to link genetic variants to phenotypes.
3. **Mediation Analysis and Mendelian Randomization**: Distinguishing the causal pathways between molecular phenotypes and diseases.

4. **Heritability Analysis**: Quantifying the genetic contribution to disease and attributing it to specific biological pathways.

This lecture aims to provide a deeper understanding of how genetic differences translate into molecular changes and ultimately manifest as disease phenotypes.

From Genome-Wide Insights to Individual Mechanisms

In previous discussions on **Genome-Wide Association Studies (GWAS)**, we identified **genetic loci** associated with diseases using statistical associations across the entire genome. We also introduced **Polygenic Risk Scores (PGS)**, which aggregate the effects of multiple genetic variants to predict an individual's genetic predisposition to a disease.

These analyses provide **global insights** by highlighting genetic regions enriched for associations with specific diseases. However, this global view needs to be refined to understand **individual mechanisms** and how **specific loci** contribute to the biology of the disease.

Key Question: How do we transition from identifying disease-associated loci to understanding their functional role in disease mechanisms?

To address this, we will:

- Investigate **individual loci** to uncover the underlying biological processes.
- Use **functional annotations** to interpret the impact of genetic variants on regulatory elements, such as enhancers, promoters, and non-coding RNAs.
- Explore **molecular phenotypes** (e.g., gene expression and methylation) to link genetic variants to cellular functions.

Understanding Quantitative Trait Loci (QTLs)

Quantitative Trait Loci (QTLs) are genomic regions associated with quantitative traits, which can vary continuously rather than being binary (e.g., height, cholesterol levels). Unlike case-control studies that categorize individuals into disease versus no disease, QTL analyses focus on **traits with continuous distributions**.

- **eQTLs**: Identify loci where genetic variants influence the expression levels of genes. These loci provide insights into how genetic differences modulate **gene regulation** and expression, impacting cellular functions.
- **mQTLs**: Identify loci affecting DNA methylation levels, a key epigenetic modification influencing gene activity and regulation.

Key Insight: Both eQTLs and mQTLs provide a **local view** of genetic influence, linking nearby genetic variants to molecular phenotypes. This local linkage, often called **cis-regulation**, offers a direct path to understanding how specific genetic changes impact gene function.

Using Molecular Traits to Understand Disease

Molecular traits like gene expression can serve as **intermediary phenotypes**, bridging the gap between genetic variation and disease. By associating genetic variants with changes in gene expression (eQTLs), we can:

- Identify **regulatory variants** that alter gene function.
- Link these changes to broader **biological pathways** and ultimately to disease phenotypes.

This approach allows us to move from a **statistical association** to a **mechanistic understanding**, where we can trace how a specific genetic variant impacts gene expression and contributes to the disease process.

Mediation Analysis and Mendelian Randomization

When studying the relationship between genetic variants, molecular phenotypes, and disease, we often encounter scenarios where multiple factors are interconnected. For example, we might observe genetic variants associated with both **lipid metabolism** and **obesity**. This raises a fundamental question:

- **Is the genetic association with obesity mediated through lipid metabolism, or is lipid metabolism a consequence of obesity?**

To address this, we use:

1. **Mediation Analysis**: This technique helps us test whether the effect of a genetic variant on a disease phenotype is **mediated** through an intermediate trait (e.g., gene expression or lipid levels).

    - We examine whether controlling for the intermediate trait (e.g., lipid levels) reduces the association between the genetic variant and the disease (e.g., obesity).
    - If the association is reduced, it suggests that the intermediate trait mediates the genetic effect on the disease.
2. **Mendelian Randomization**: This method leverages genetic variants as **instrumental variables** to infer causal relationships. It relies on the principle that genetic variants are randomly inherited and can be used to test causality, distinguishing between direct effects and mediated effects.
    - For instance, if genetic variants associated with lipid metabolism also predict obesity when controlling for lipid levels, this suggests a direct effect of these variants on obesity.
    - Conversely, if the association diminishes when adjusting for lipid levels, it suggests that the effect is mediated through lipid metabolism.

Key Insight: Mediation analysis and Mendelian randomization help us **disentangle complex biological relationships**, clarifying whether molecular changes are a cause or consequence of disease.

Heritability and Pathway-Specific Contributions

Heritability quantifies the proportion of variation in a trait that can be attributed to genetic differences. In the context of complex diseases:

- We use **partitioned heritability** to determine the **contribution of specific biological pathways** to the overall genetic risk of a disease.
- For example, we might ask, "What fraction of the heritability of obesity can be explained by genetic variants affecting lipid metabolism?"

This approach allows us to connect **global genetic risk** to specific **biological processes**, providing a systems-level view of how genetic factors drive disease.

Key Insight: By estimating the heritability attributed to different pathways, we can identify the most relevant biological mechanisms and prioritize them for further investigation and therapeutic targeting.

Systems-Level Understanding of Disease Mechanisms

In summary, today's lecture builds on the foundation of GWAS and PGS by delving deeper into the **circuitry of genetic regulation**:

1. We use **molecular QTLs** to link genetic variants to specific gene regulatory effects.
2. We apply **mediation analysis** and **Mendelian randomization** to infer causal relationships between molecular traits and disease.
3. We assess **heritability partitioning** to attribute genetic risk to specific biological pathways.

Together, these tools allow us to move beyond statistical associations and towards a comprehensive, **mechanistic understanding of disease**, paving the way for novel therapeutic strategies and personalized medicine.

05:39 Review: Genetics, Variation, GWAS, Polygenic scores (PGS)

In our last lecture, we delved into the **fundamental principles of genetics and variation**, focusing on the identification of disease-associated loci through **genome-wide association studies (GWAS)**, the interpretation of these findings, and the use of **polygenic scores (PGS)** to predict individual risk.

Understanding Genetic Variants and Disease Association

Genetic variants are differences in the DNA sequence between individuals, often represented as **single nucleotide polymorphisms (SNPs)**. These can have **protective effects** (shown in green) or **increase disease risk** (shown in red). Using GWAS, we perform a statistical analysis across the entire genome to identify variants associated with specific diseases.

GWAS Results are typically visualized using a **Manhattan plot**, where:

- The **x-axis** represents the genomic coordinates across chromosomes.
- The **y-axis** shows the statistical significance of the association (e.g., p-value) for each SNP.

Peaks in the Manhattan plot indicate **non-random associations** between genetic variants and the disease. For example, a peak might suggest a strong association with age-related macular degeneration. These signals highlight regions of interest but do not immediately reveal the **mechanisms** by which these variants affect disease.

Challenges in Interpreting GWAS Results

The main challenge lies in translating these associations into **biological mechanisms**:

1. **Cataloging Variants**: Identifying all genetic variants associated with a disease and creating a comprehensive list.
2. **Systematic Association**: Analyzing these variants to determine their potential impact on biological processes and pathways.
3. **Understanding Mechanisms**: Deciphering how these variants influence gene expression, protein function, and cellular processes.
4. **Translating Insights into Action**: Using these findings to inform **therapeutic interventions**, such as selecting target proteins or developing specific drugs.

The goal is to move from **broad statistical associations** to **precise functional insights** that guide therapeutic strategies.

Recombination and Fine Mapping

We discussed how **genetic linkage** is shaped by the structure of the genome, specifically through **haplotype blocks**—regions of the genome that are inherited together due to limited recombination between them. Within these blocks, SNPs are often in **linkage disequilibrium (LD)**, meaning that they are correlated and inherited together.

To pinpoint the **causal variants** within these blocks, we use **fine mapping**, which can be achieved through:

1. **Genetic Fine Mapping**: Leveraging rare recombination events within haplotype blocks to narrow down the list of candidate variants.
2. **Transethnic Associations**: Analyzing GWAS data from diverse populations (e.g., African, Asian, European) to exploit differences in LD structure. If a variant shows consistent association across populations with different haplotypes, it is more likely to be the true causal variant.

This approach helps in **identifying the specific variants** driving the disease association, despite the complex LD patterns across populations.

Common vs. Rare Variants: A Spectrum of Genetic Effects

There is a fundamental distinction between **common variants** and **rare variants** in terms of their **effect size**:

- **Common Variants**: Found in a large proportion of the population, these variants typically have a **weak effect** on disease risk. If they had strong effects, they would be subject to **negative selection**, reducing their frequency.
- **Rare Variants**: These are often **high-effect variants**, but they remain rare because they are typically deleterious and subject to strong **selective pressure**, maintaining them at low frequencies.

GWAS primarily identifies **common, weak-effect variants**, but integrating data on rare variants can provide additional insights into the genetic architecture of diseases.

Polygenic Scores (PGS): Aggregating Genetic Risk

While individual variants may have modest effects, collectively they can provide substantial predictive power when combined into a **polygenic score (PGS)**. A PGS aggregates the effects of many SNPs across the genome, each weighted by its effect size, to generate an overall **risk score** for an individual.

The Predictive Power of Polygenic Scores. PGS can explain a significant portion of an individual's **genetic risk** for a given disease. However, PGS does not capture the entire genetic risk explained by **family history**, which combines both **common and rare variants** as well as **environmental factors**.

Key Question: Why does PGS leave some residual risk unexplained, even when accounting for family history?

The answer lies in **genetic inheritance**:

- Each individual inherits a **unique combination of alleles** from their parents. Even though two siblings share the same parents, the specific alleles they inherit can differ significantly.
- On average, siblings share **50% of their genome**. For any given locus:
    - **25% of the genome** will be identical (both siblings inherited the same allele from the same grandparent).

    - **25% of the genome** will be completely different (each sibling inherited a different allele from different grandparents).
    - **50% of the genome** will be partially identical (one shared and one non-shared allele).

This variation between siblings means that **family history** provides a general risk estimate based on shared genetics, but **PGS** offers a more precise estimate by directly examining the specific alleles an individual inherited.

Integrating PGS into Disease Risk Prediction. The use of PGS has become a powerful tool in **predictive genetics**, enabling us to:

- Identify individuals at **high genetic risk** for specific diseases.
- Enhance the precision of **preventive measures** and **therapeutic interventions**.
- Provide personalized insights that go beyond traditional family history.

However, PGS is only one part of the picture. To fully understand disease risk, we need to integrate PGS with other factors, such as **environmental exposures**, **lifestyle**, and **molecular phenotypes** (e.g., gene expression, epigenetic changes).

Summary

- **GWAS** helps identify genetic variants associated with diseases, but interpreting these associations requires further investigation to uncover the **functional mechanisms**.
- **Fine mapping** and **transethnic analyses** help pinpoint the causal variants, moving beyond simple statistical associations.
- **Polygenic scores (PGS)** aggregate the effects of multiple variants, providing a comprehensive estimate of genetic risk but still leave residual variation due to differences in inherited alleles.
- Understanding the interplay between common and rare variants, genetic inheritance patterns, and environmental factors is essential for translating these insights into **clinical practice**.

The next step is to go from these **global genetic insights** to understanding the **molecular mechanisms** and **biological pathways** that underlie complex diseases. This sets the stage for today's focus on **disease mechanism and circuitry**, where we will dive deeper into how genetic variants influence gene regulation, expression, and ultimately, disease phenotypes.

11:16 From GWAS to Biological Insights

Building upon our understanding of **genetic variation and GWAS**, we can use **genome-wide enrichment analyses** to gain insight into the specific biological pathways, tissues, and mechanisms that underlie complex traits and diseases. This approach enables us to move from a high-level association across the genome to more focused hypotheses about where and how these associations might act.

Enrichment Analysis Across Functional Annotations

Using GWAS results, we can aggregate all genetic variants associated with a trait (e.g., height, type 1 diabetes, blood pressure, or cholesterol) and investigate their **enrichment across various functional annotations**. Each annotation corresponds to a specific biological feature, such as:

- **Enhancers** active in certain cell types (e.g., stem cells, immune cells, heart cells, liver cells).
- **Epigenomic markers** such as histone modifications, DNA methylation states, or chromatin accessibility regions.

The idea here is to correlate the **genetic variants associated with a trait** (aggregated across the entire genome) with **functional genomic features** (enhancers or accessible chromatin regions in different tissues). This correlation reveals whether variants for a specific trait are enriched in regulatory elements specific to certain tissues, potentially implicating those tissues in the trait's mechanism.

Interpreting Rows and Columns in Enrichment Matrices

To interpret these data, we structure our matrix as follows:

- **Rows** represent all genetic variants associated with a given trait.
- **Columns** represent various epigenomic annotations, such as enhancers or other regulatory elements in specific tissues.

For instance, one column could include all enhancers active in **liver cells**, while another represents enhancers in **immune cells**. By calculating the **enrichment score** for each row-column combination, we can see how

genetic variants linked to specific traits overlap with tissue-specific enhancers. This alignment between genetic risk loci and tissue-specific regulatory elements helps prioritize tissues for further analysis.

However, this enrichment analysis can be complex due to **varied background expectations**. Calculating the overlap by chance requires careful consideration of the **denominator** for each column, which could be:

- **The entire genome** (a broad background).
- **Accessible genome regions** (focused only on regulatory regions).

These choices can introduce biases due to the differing distributions of SNPs and accessible chromatin regions across tissues. By comparing all tissues against one another in a comprehensive matrix, we can reduce such biases through **internal controls** and better isolate genuine signals.

Case Study: Alzheimer's Disease and Immune Cells

An example of this approach is evident in **Alzheimer's disease**, where GWAS initially suggested a strong association with genetic variants in **immune cell enhancers** rather than brain-specific enhancers. This unexpected finding prompted a deeper investigation:

- Using a **mouse model** of Alzheimer's, we analyzed gene expression changes across disease progression.
- The analysis showed that **early changes** in Alzheimer's predominantly occurred in **immune cells**, while **neuronal changes** emerged later.

Together, these findings led us to hypothesize an **immune basis for Alzheimer's disease**, where initial immune dysregulation might contribute to or exacerbate neurodegeneration. This hypothesis was further validated by studies from other research groups, supporting an immune involvement in Alzheimer's.

Building Networks of Traits and Tissues

Expanding this framework, we can construct **network diagrams** where nodes represent **traits** and **tissues**. In this network:

- **Circles** represent different tissues (e.g., brain, heart, liver).
- **Rounded rectangles** represent traits (e.g., Alzheimer's, ADHD, height).
- Edges between nodes indicate **significant enrichment** between a tissue and a trait.

For example, **Alzheimer's disease** shows connections with immune cell tissues, while **ADHD** connects with brain tissue, and **height** with stem cell-associated enhancers. Such a network captures the connections between **genetic architecture and biological pathways** across multiple traits, providing a comprehensive view of trait-specific tissue involvement.

Expanding to Comprehensive Genomic Datasets

We can also analyze more extensive datasets, such as **hundreds of GWAS studies** versus **thousands of reference epigenomes**. This broader view categorizes traits into three major groups:

1. **Multifactorial Traits**: These traits, such as bone mineral density and blood cell counts, show enrichment across multiple tissues, reflecting their widespread biological impact.
2. **Polyfactorial Traits**: Traits that enrich in a limited number of related tissues but are not universal. They display moderate tissue specificity.
3. **Unifactorial Traits**: These traits localize to specific tissues. For instance, **cognitive traits** localize to the brain, while **cholesterol** traits localize to the liver.

This categorization helps **partition traits by their tissue-specific enrichments**, providing a roadmap for selecting the appropriate tissues when investigating trait mechanisms.

Implications for Studying Molecular Traits

With tissue-specific insights, we can decide where to focus for **expression quantitative trait loci (eQTL)** and other molecular analyses. For example:

- **Cognitive and psychiatric traits** (e.g., schizophrenia, cognitive performance) may be best studied in **brain tissue**.
- **Cardiac traits** like heart repolarization localize to **heart-specific enhancers**.
- **Cholesterol traits** align with **liver tissue**.

By selecting the relevant tissue, we can more effectively study **gene expression changes** and other

regulatory modifications that link genetic variants to disease.

From Global Enrichment to Individual Variant Prioritization

Finally, we can return to **individual loci** within GWAS signals and prioritize specific variants likely to be **causal** based on tissue enrichment:

1. **Empirical Prior from Genome-Wide Enrichment**: Across the genome, we calculate empirical priors for variants based on their enrichment in particular tissues.
2. **Fine Mapping of Causal Variants**: For a trait associated with multiple tissues, we prioritize variants that overlap with the tissue showing the strongest enrichment (e.g., immune enhancers for Alzheimer's).

This empirical approach helps us focus on the variants most likely to contribute to disease in a given tissue, refining our understanding of **disease mechanism and circuitry**. Through these integrated analyses, we transition from genome-wide patterns to specific hypotheses about how genetic variants impact cellular function and ultimately influence disease.

22:55 Bayesian Fine-Mapping

With genome-wide association studies (GWAS), we identify **regions of the genome** associated with complex traits and diseases. However, due to **linkage disequilibrium (LD)** — the non-random association of nearby variants — it is challenging to pinpoint the exact causal variants within these associated regions. **Bayesian fine-mapping** provides a probabilistic framework to integrate multiple sources of information, helping us prioritize variants that are most likely to be **causal**.

Components of Bayesian Fine-Mapping

Bayesian fine-mapping combines three key pieces of information:

1. **Association Evidence** (Effect Size):
    - From GWAS, we obtain **association statistics** for each variant, typically represented as effect sizes or p-values. This information captures the strength of the association between each variant and the trait of interest.
2. **Linkage Disequilibrium (LD) Structure**:
    - Variants within a genomic region are often correlated due to **LD**, making it difficult to disentangle their individual effects. We use the **LD matrix**, which represents the correlation between pairs of variants, to account for this structure. The **co-association** of variants in LD provides clues about where the true causal signal might be.
3. **Empirical Priors from Functional Annotations**:
    - By leveraging **epigenomic data**, we can assign **empirical priors** based on the overlap of variants with specific functional annotations (e.g., enhancers, promoters). Variants that overlap with relevant annotations, such as immune cell enhancers for immune-related traits, are assigned higher priors, increasing their likelihood of being causal.

Combining Evidence: Posterior Probability

The goal of Bayesian fine-mapping is to compute a **posterior probability** for each variant, representing the likelihood that it is causal. This is achieved through the following steps:

1. **Likelihood Calculation**:
    - The likelihood is derived from the **genetic association evidence** alone, reflecting how well the variant's effect size explains the observed GWAS signal.
2. **Prior Assignment**:
    - The prior probability is informed by **functional annotations**, such as overlap with enhancers, evolutionary conservation, or eQTLs (expression quantitative trait loci). Variants overlapping relevant annotations receive higher priors.
3. **Posterior Computation**:
    - The **posterior probability** is calculated by combining the likelihood and the prior using Bayes' theorem. This posterior probability reflects the updated belief about a variant's causality after considering both the association evidence and functional annotation data.

Mathematically, this can be represented as:

Posterior Probability∝Likelihood×Prior\text{Posterior Probability} \propto \text{Likelihood} \times \text{Prior}Posterior Probability∝Likelihood×Prior

Examples of Bayesian Fine-Mapping

Let's look at a few examples where Bayesian fine-mapping has provided deeper insights:

1. **Crohn's Disease**:
    - In a genetic region associated with Crohn's disease, the raw association evidence highlights several potential causal variants. However, by integrating functional annotations specific to **immune enhancers**, we find that a subset of these variants overlaps immune-specific annotations. Bayesian fine-mapping then **prioritizes** these overlapping variants as the most likely causal ones, consistent with the immune-mediated nature of Crohn's disease.
2. **Schizophrenia**:
    - For schizophrenia, the functional annotations enriched for central nervous system (CNS) activity are most relevant. Bayesian fine-mapping highlights variants overlapping CNS enhancers, significantly narrowing down the list of candidates. This prioritization aligns with the known involvement of brain function in schizophrenia, suggesting these variants may impact gene regulation in neural tissues.

Validating Fine-Mapped Variants

A critical step in fine-mapping is validating that the prioritized variants have true biological relevance. Two common approaches for validation include:

1. **Evolutionary Conservation**:
    - Fine-mapped variants often overlap **evolutionarily conserved elements**, which are regions of the genome that have remained unchanged across species due to selective pressures. The rationale is that causal variants affecting important biological functions are more likely to be conserved. In practice, fine-mapped variants show **increased overlap with conserved elements**, suggesting they play essential roles in gene regulation.
2. **Expression Quantitative Trait Loci (eQTLs)**:
    - Fine-mapped variants are also more likely to overlap with **eQTLs**, which are genetic variants associated with changes in gene expression. This overlap indicates that the prioritized variants might exert their effects through altering gene expression levels, providing a direct molecular mechanism linking genotype to phenotype.

For example, using data from the **Genotype-Tissue Expression (GTEx) Project**, researchers have shown that fine-mapped variants are significantly enriched for eQTLs across relevant tissues. This supports the hypothesis that these variants contribute to disease risk by regulating gene expression in a tissue-specific manner.

Summary

Bayesian fine-mapping provides a powerful framework to integrate diverse sources of information — GWAS signals, LD structure, and functional annotations — to pinpoint causal variants with high confidence. The approach helps overcome the challenges posed by linkage disequilibrium and the sheer number of associated variants, enabling us to:

- **Prioritize variants** that are most likely to be functional.
- **Reduce the search space** for causal variants, facilitating experimental follow-up.
- **Gain deeper biological insights** into the regulatory mechanisms underlying complex traits.

By leveraging both statistical association evidence and functional genomic data, Bayesian fine-mapping moves us closer to understanding the **molecular mechanisms** driving complex diseases, paving the way for more targeted therapeutic interventions.

26:06 Sub-threshold Region Prioritization

While Bayesian fine-mapping helps pinpoint the most likely **causal variants** within genome-wide significant loci (regions that show a strong association with a trait), there remains a vast number of **sub-threshold regions** — regions that do not meet genome-wide significance but may still harbor variants with true biological relevance. By incorporating **additional biological evidence**, we can prioritize these sub-threshold regions, increasing our ability to detect meaningful signals that might otherwise be overlooked.

Challenge of Sub-threshold Regions

In GWAS, we often focus on loci that surpass a stringent **p-value threshold** (e.g., $p < 5 \times 10^{-8}$). However, many true signals might fall below this threshold, especially in the case of:

- **Complex traits** with polygenic architectures, where individual variants have small effect sizes.
- **Limited sample sizes**, which reduce statistical power to detect associations, particularly for variants with modest effects.

Thus, relying solely on the significance threshold may lead to missed opportunities for identifying important genetic regions. This is where **integrative methods** come into play.

Leveraging Epigenomic Annotations

To address this, we can use **epigenomic annotations** as additional lines of evidence to help prioritize sub-threshold regions. These annotations, derived from functional genomics data, capture information about regulatory activity across the genome. For instance:

- **Enhancer activity** (e.g., H3K27 acetylation marks indicating active enhancers).
- **DNA hypomethylation**, often associated with regulatory elements.
- **Primate conservation**, highlighting regions that are evolutionarily conserved and likely functional.

By integrating these annotations, we can build a model that predicts the **functional potential** of sub-threshold loci, allowing us to identify regions that are more likely to be **true positives** despite not reaching genome-wide significance.

Machine Learning Approach for Prioritization

To systematically prioritize sub-threshold regions, we employed a **machine learning classifier** trained on a diverse set of features derived from:

1. **Epigenomic annotations**: Including enhancer marks, histone modifications, DNA methylation states, and chromatin accessibility signals.
2. **Evolutionary conservation**: Using metrics of primate conservation to identify regions likely under selective pressure.
3. **Regulatory evidence**: Incorporating data on known transcription factor binding sites and gene regulatory elements.

The model was trained to distinguish between **genome-wide significant loci** (as positive examples) and random regions or regions showing no evidence of association (as negative examples). The output was a **predictive score** for each sub-threshold region, indicating its likelihood of being truly functional.

Experimental Validation of Prioritized Regions

We did not stop at computational predictions; we conducted **experimental validation** to confirm the biological relevance of the prioritized sub-threshold loci. This included:

1. **Luciferase Reporter Assays**:
    - We cloned candidate enhancer regions upstream of a luciferase gene. If the region is functional, it would drive **increased luciferase activity**, indicating enhancer potential.
2. **Promoter Capture Hi-C**:
    - This technique maps physical interactions between enhancers and promoters. We used it to identify **target genes** of the prioritized regulatory regions, providing further evidence of their role in gene regulation.
3. **CRISPR Knockout Experiments**:
    - Using **CRISPR/Cas9**, we knocked out the prioritized sub-threshold loci in mouse models. We then assessed **phenotypic changes**, such as alterations in heart repolarization interval (QRS interval), to determine the functional impact of the knocked-out regions.

Case Study: Heart Repolarization Interval

In one application focused on the **QRS interval** (a measure of heart repolarization), we used this approach to identify **sub-threshold loci** that were predicted to be functional based on their overlap with relevant enhancer annotations. Although these loci did not initially meet the genome-wide significance threshold, our integrative analysis prioritized them for follow-up studies.

- **CRISPR knockout** experiments in mice revealed that disrupting these loci led to significant changes in the heart repolarization interval, providing **causal evidence** for their involvement in cardiac function.
- This finding underscores the power of using **additional biological evidence** to uncover functional loci that might otherwise be missed by GWAS alone.

Building Interactive Tools for Prioritization

Given the utility of integrating various types of evidence, we have developed **interactive browsers** and computational tools that allow researchers to:

1. **Visualize epigenomic data**: Overlay functional annotations on the genome alongside GWAS signals.
2. **Apply machine learning models**: Automatically score and prioritize sub-threshold regions based on their functional potential.
3. **Nominate candidate variants and loci**: Facilitate hypothesis generation for experimental follow-up.

These tools enhance our ability to interpret the vast amount of **sub-threshold genetic data**, transforming how we prioritize regions for further study.

Summary

Sub-threshold region prioritization is a critical extension of GWAS that leverages **epigenomic data, machine learning, and experimental validation** to uncover functional loci beyond the strict genome-wide significance threshold. This integrative approach allows us to:

- Identify **additional loci** that contribute to complex traits.
- Gain deeper insights into the **regulatory mechanisms** underlying genetic associations.
- Enable **targeted experimental follow-up**, increasing our chances of discovering true biological signals.

By incorporating these methods, we move beyond a binary view of significant versus non-significant loci, embracing a more nuanced approach that captures the complexity of the genome and its regulation.

27:52 GWAS locus dissection: enriched tissues, driver SNPs, target genes

Genome-Wide Association Studies (GWAS) provide a powerful framework for identifying genetic loci associated with complex traits and diseases. However, translating these **associations** into mechanistic insights requires a deeper dissection of the implicated loci to pinpoint:

1. **Driver SNPs**: The specific variants likely causing the association.
2. **Enriched Tissues**: The tissues or cell types where the genetic variants are most likely to exert their functional effects.
3. **Target Genes**: The genes whose regulation is affected by these variants.

Systematic Annotation and Resource Development

To facilitate the dissection of GWAS loci, we have developed several resources and methodologies across multiple projects, such as:

- **HaploReg (HaoR)**: A tool designed for exploring the regulatory potential of SNPs within linkage disequilibrium (LD) blocks, focusing on how variants may impact regulatory elements like enhancers.
- **eQTL Mapping Projects (ePAP)**: Epigenomic mapping efforts that integrate functional genomic data across diverse tissues and cell types, enabling us to connect SNPs to regulatory elements and gene expression changes.

These resources allow us to systematically annotate and interpret individual SNPs, loci, and tissues, providing insights into the specific **biological context** of each association.

Example: Breast Cancer Locus Dissection

Consider a GWAS for **breast cancer**. To understand the tissue-specific relevance of the implicated SNPs, we first examine the **enriched tissues** for this trait. Using the epigenomic annotations from ePAP, we observe that:

- The **most enriched tissues** for breast cancer-associated SNPs are, as expected, breast-related tissues such as **breast epithelium** and **breast cancer cell lines**.
- This strong tissue-specific signal suggests that the variants are likely overlapping enhancers that are **active in breast tissues**, rather than enhancers from unrelated tissues like brain or liver.

By leveraging these tissue-specific annotations, we can **narrow down the candidate SNPs** most likely to be driving the association. Furthermore, we can identify potential **target genes** by examining the correlated activity between the enhancers overlapping the SNPs and nearby gene expression.

Example: Schizophrenia Locus Dissection

In the case of **schizophrenia**, the dissection process follows a similar approach but focuses on **brain-specific tissues** and cell types. Here, we find:

- The **most enriched tissues** include mid-frontal cortex, cerebellum, and various neural progenitor cells, indicating that the implicated variants are likely affecting brain-specific regulatory elements.
- For example, a particular SNP near the **DCP1B gene** is identified within an **enhancer** that is highly active in brain tissues. This SNP is linked to a gene that is also expressed specifically in the brain, suggesting a direct regulatory mechanism impacting neural gene expression.

Systematic Identification of Driver SNPs

To move from statistical associations to **causal insights**, we can use a combination of:

1. **Epigenomic Annotation Overlap**:
    - By mapping the SNPs to **active enhancers** in the enriched tissues, we prioritize variants that are more likely to affect gene regulation.
2. **Expression Quantitative Trait Loci (eQTL) Data**:
    - Integrating eQTL data helps us establish direct links between the prioritized SNPs and changes in gene expression. SNPs that are also eQTLs for genes expressed in the relevant tissues are strong candidates for **driver SNPs**.
3. **Linkage Disequilibrium (LD) Structure**:
    - We account for the **LD structure** of the loci, which helps us differentiate between variants that are simply correlated and those that are more likely to be **causal** based on their functional annotations.

Nominating Target Genes

Once we have identified the likely **driver SNPs**, the next step is to link these SNPs to their **target genes**. This involves:

- **Promoter Capture Hi-C**: Mapping the physical interactions between enhancers (where the SNPs are located) and gene promoters. This technique helps us identify the **target genes** regulated by the enhancer elements.
- **Correlated Gene Expression**: We look at the co-expression patterns between the implicated enhancers and nearby genes, providing further evidence of potential regulatory relationships.

Case Study: Systematic Prediction and Hypothesis Generation

By integrating these different lines of evidence, we can build a **comprehensive map** of SNP-to-gene relationships for each trait. For example:

- In breast cancer, we might identify a set of SNPs that overlap enhancers active in **breast epithelium** and link these SNPs to genes involved in **cell proliferation** and **DNA repair**, which are critical processes in cancer development.
- In schizophrenia, we might pinpoint SNPs located in enhancers active in the **mid-frontal cortex** and connect them to genes related to **synaptic signaling** or **neural development**, providing insights into the potential biological pathways disrupted in this psychiatric disorder.

These predictions can then be **shared with the scientific community**, serving as a basis for **experimental validation**. Researchers can use these nominated SNPs and target genes to design follow-up studies, such as:

- **CRISPR-based gene editing**: To test the functional impact of the identified SNPs and their regulatory elements.
- **Gene expression assays**: To confirm changes in gene expression predicted by the eQTL analysis.
- **Animal models**: To study the phenotypic effects of manipulating the nominated target genes.

Moving Beyond Association

This integrative framework goes beyond simple statistical associations by providing a **mechanistic**

understanding of how genetic variants contribute to disease. It allows us to:

1. **Identify enriched tissues** where the variants are most likely to be functional.
2. **Pinpoint driver SNPs** based on their overlap with regulatory elements and eQTL evidence.
3. **Nominate target genes** that are likely impacted by the regulatory changes caused by these SNPs.

Ultimately, this process transforms GWAS signals into actionable **biological insights**, guiding therapeutic development and informing strategies for precision medicine. By systematically dissecting the loci, we are moving closer to understanding the **causal mechanisms** underlying complex traits and diseases.

30:40 Partitioning complex traits into tissue-specific components

In complex traits and diseases, genetic associations often do not localize neatly within a single tissue. Instead, many traits show **multifactorial influences**, where genetic variants exert effects across multiple tissues, each contributing to the overall phenotype. In this section, we explore how we can decompose these complex associations and **partition the genetic architecture** of traits into tissue-specific components.

Unifactorial vs. Multifactorial Traits

Using the insights from the **enrichment map** of genetic associations across tissues, we can classify traits based on the **diversity of their tissue-specific signals**:

- **Unifactorial Traits**: These traits show strong enrichment in a single tissue type. For example:
    - **Heart Repolarization Interval (QRS Interval)**: Enriched primarily in heart tissues.
    - **Educational Attainment**: Enriched in brain tissues.
    - **Alzheimer's Disease**: Enriched in immune cells with some contributions from brain tissues.
- **Multifactorial Traits**: These traits exhibit **broad tissue-specific enrichments**, indicating a more complex genetic architecture. Examples include:
    - **Coronary Artery Disease (CAD)**: Displays associations across a wide range of tissues, including heart, liver, adipose tissue, and brain, reflecting the interplay between cardiovascular function, lipid metabolism, and systemic factors like obesity.
    - **Waist-to-Hip Ratio**: Shows contributions from multiple tissues such as kidney, muscle, adipose tissue, and liver, which reflect broader metabolic processes.
    - **Health Span**: A highly polyfactorial trait, with associations found across many tissues, including immune cells, muscle, blood, and embryonic stem cells, highlighting the diverse biological processes that impact overall health and longevity.

Example: Coronary Artery Disease (CAD)

Coronary Artery Disease is a quintessential example of a **multifactorial trait**. Genetic variants associated with CAD do not solely map to heart tissues. Instead, they show significant enrichments across:

- **Liver Enhancers**: These variants likely affect genes involved in **lipid metabolism**, such as those regulating cholesterol and triglyceride levels.
- **Coronary and Heart Enhancers**: Variants here may influence **vascular function**, including endothelial cell proliferation and smooth muscle cell activity.
- **Adipose Tissue Enhancers**: Variants in this category are linked to **fat distribution** and metabolic processes that influence the risk of atherosclerosis.

By examining the tissue-specific enrichments, we can start **partitioning the genetic variants** associated with CAD into different biological categories. For example:

1. **Variants in Liver Enhancers**: These variants are enriched for pathways like **phospholipid transport**, confirming the role of lipid metabolism in CAD.
2. **Variants in Heart Enhancers**: These are linked to **endothelial cell proliferation** and **vascular function**, pointing to direct effects on coronary arteries.
3. **Variants in Adipose Tissue Enhancers**: These show enrichments for **CDC42 signaling** and other pathways related to fat distribution and metabolism.

This partitioning process allows us to **decompose the genetic architecture** of CAD into components related to different tissues, each contributing distinct aspects of the disease process. It provides a **more nuanced understanding** of the mechanisms driving the disease and suggests potential targets for therapeutic

intervention based on the tissue of action.

Tissue-Specific Locus Dissection

With the framework of tissue-specific partitioning, we can go further and dissect individual **genetic loci** based on their tissue enrichments:

- **Heart-Specific Loci**: For example, a locus enriched in **coronary artery and heart atrium enhancers** suggests a direct role in cardiovascular tissues. This could involve genes affecting **vascular development** or **heart muscle function**.
- **Liver-Specific Loci**: Some loci, such as those near the **PCSK9** gene, show strong enrichment in **liver enhancers**. This makes sense because PCSK9 is involved in **cholesterol regulation**, and liver tissue plays a central role in lipid metabolism.
- **Multitissue Loci**: Other loci may show complex patterns, with enrichments in multiple tissues like **thyroid gland, adipose tissue, and liver**. These loci are likely involved in broader **endocrine and metabolic processes**, influencing multiple aspects of disease risk.

Hypothesis Generation and Experimental Validation

These tissue-specific dissection efforts have been made publicly available, allowing the scientific community to use them as **hypothesis generation engines**. By providing a detailed map of **tissue-enriched loci**, researchers can:

- **Formulate testable hypotheses** about the tissue-specific mechanisms underlying complex traits.
- **Design experiments** to validate the functional roles of candidate driver SNPs and their associated genes.

For example, a researcher studying **lipid regulation** in CAD might focus on the subset of liver-enriched variants, testing their effects on cholesterol-related genes through **CRISPR perturbation** or **reporter assays**.

"Eating Our Own Dog Food": Using Tools Internally

In the spirit of **self-validation**, our group has also applied these tools internally to generate and test our own hypotheses. This ensures a **high standard of quality** for the resources we develop and demonstrates their utility in driving **novel biological discoveries**.

For instance, by applying the tissue-partitioning methods we developed, we were able to:

- Identify a new set of **liver-specific variants** associated with CAD.
- Show that these variants impact **lipid transport genes**, leading to altered cholesterol levels in functional assays.
- Demonstrate in **mouse models** that disrupting these liver-specific genes can replicate aspects of the human CAD phenotype.

This kind of **integrative analysis** is key to understanding the complex interplay of genetic variants across multiple tissues, moving us beyond simple associations and toward a **mechanistic understanding** of how genetic variation contributes to disease.

Conclusion

The ability to **partition complex traits into tissue-specific components** is a significant advance in the field of human genetics. It provides a framework for:

1. **Decomposing the genetic architecture** of multifactorial traits.
2. **Prioritizing functional variants** based on their tissue context.
3. **Guiding experimental studies** and therapeutic development by highlighting the most relevant tissues and pathways.

Ultimately, this approach helps us **bridge the gap** between GWAS associations and biological mechanisms, offering a clearer path toward **precision medicine** and targeted interventions tailored to the underlying tissue-specific processes of disease.

35:53 From Individual GWAS Region to Mechanism and Circuitry

In this section, we dive into the **detailed dissection of individual GWAS regions**, moving from broad genetic associations to specific **mechanistic insights and regulatory circuitry**. The goal is to provide a comprehensive understanding of the **biological processes underlying complex traits**, using every available piece of evidence. This approach can turn a **statistical association** from a GWAS into a **causal**

understanding of disease mechanism.

The Integrative Framework

We start with a **region of genetic association**, often identified by GWAS, and aim to partition and interpret this region using the following steps:

1. **Identify Specific SNPs**: Determine which variants (SNPs) within the region are most likely to be **causally linked** to the trait. This involves fine-mapping approaches and prioritization of variants based on linkage disequilibrium, epigenomic annotations, and predicted functionality.
2. **Map Enhancer Overlaps**: Assess whether the prioritized SNPs overlap **enhancers** or other regulatory elements. Enhancers are non-coding regions that can modulate gene expression, and SNPs in these regions may alter regulatory activity.
3. **Determine Tissue and Cell Type Specificity**: Identify the **tissues and cell types** where these enhancers are active. Tissue-specific activity can help narrow down the **biological context** in which the genetic variant is exerting its effect.
4. **Identify Motifs and Disrupted Regulatory Elements**: Analyze the **motifs** present within these enhancers. Motifs are short DNA sequences recognized by transcription factors. SNPs can **disrupt these motifs**, altering the binding of regulatory proteins.
5. **Identify Upstream Regulators**: Determine the **transcription factors** and other regulatory proteins that recognize these motifs. These upstream regulators are the proteins likely impacted by the SNPs.
6. **Assess Downstream Gene Expression Changes**: Determine the **target genes** whose expression is modulated by these enhancers. Changes in gene expression can provide insights into the **molecular phenotype** associated with the genetic variant.
7. **Determine Biological and Organismal Implications**: Understand the broader **biological processes** impacted by these changes in gene expression and the **phenotypic consequences** at the organismal level.

This comprehensive approach allows us to **connect genetic variation to regulatory function**, tissue specificity, and ultimately the **pathophysiology of complex diseases**.

Case Study: Dissecting the FTO Locus

A concrete example of this framework is our work on the **FTO locus**, a region of the genome associated with **obesity**. The study, published in the **New England Journal of Medicine** in 2015, serves as a model for how to dissect a complex GWAS locus. Despite the seemingly narrow scope—focusing on a single SNP and its downstream effects—the study uncovered a **novel regulatory mechanism** that contributes to obesity.

Key Steps in the FTO Study:

1. **Tissue and Cell Type Identification**: We first needed to determine the **tissue context** where the FTO-associated variants are likely acting. Initial analyses pointed towards **adipose tissue**, specifically a type of fat involved in energy regulation.
2. **Target Gene Identification**: The FTO locus was initially believed to directly impact the **FTO gene**, a known obesity-associated gene. However, our analysis revealed that the true **target genes** were actually **IRX3** and **IRX5**, located several hundred kilobases away. This highlights the importance of considering long-range regulatory effects in non-coding regions.
3. **Causal Variant Identification**: Using fine-mapping techniques, we identified a **single causal SNP** that appeared to be disrupting an enhancer. This SNP was not located in the coding region of the FTO gene but in a regulatory element that influenced distant genes.
4. **Regulatory Motif Disruption and Upstream Regulators**: The SNP was found to disrupt a **motif recognized by the ARID5B transcription factor**. This disruption altered the binding of ARID5B, leading to changes in the regulatory activity of the enhancer.
5. **Cellular Implications: Adipocyte Browning**: The altered enhancer activity resulted in changes in the **expression of IRX3 and IRX5**, which are involved in the regulation of **adipocyte differentiation**. Specifically, the causal variant promoted a shift away from **beige adipocyte formation** (a process known as **browning**) and towards white adipocyte formation, which is associated with increased fat storage.
6. **Organismal Implications: Obesity Phenotype**: The disruption of this regulatory circuitry had a direct

impact on **energy balance and obesity**. By reducing adipocyte browning, the FTO variant favored the accumulation of white fat, thereby increasing the risk of obesity.

Experimental Validation

To confirm the causality of the identified SNP, we employed **CRISPR genome editing**. By **precisely editing the SNP** in human cells and in mouse models, we were able to **replicate the phenotypic effects**, providing strong evidence for the role of this variant in obesity risk.

Additionally, we used **reporter assays** to demonstrate the enhancer activity and **motif disruption**, confirming the role of the ARID5B transcription factor in this regulatory mechanism.

Broader Implications and Model for GWAS Locus Dissection

The dissection of the FTO locus exemplifies how to move from a broad GWAS association to a **detailed mechanistic understanding**. This approach can be generalized to other GWAS loci, offering a pathway for the **functional interpretation of non-coding variants** that are often challenging to study.

By integrating **epigenomic data, motif analysis, tissue-specific activity, and experimental validation**, we can:

- **Identify causal variants** even in complex, non-coding regions.
- **Determine the regulatory circuitry** that links these variants to gene expression changes.
- **Understand the tissue-specific effects**, providing insights into the biological context of the disease.
- **Guide therapeutic development** by targeting the identified regulatory pathways.

This integrative approach shifts the focus from merely identifying **statistical associations** to uncovering the **biological mechanisms** that drive complex traits and diseases, paving the way for more targeted and effective **precision medicine**.

38:53 Introduction to the FTO locus, the strongest Genetic Association with Obesity

The **FTO locus** is one of the most striking genetic associations identified in human genomics, with a **p-value of 10−6010^{-60}10−60**, an extremely significant signal. This association, robust and replicated across multiple studies, points to the FTO locus as a major contributor to **obesity** and related metabolic disorders. Let's dive deeper into the genomic and biological context of this region, the challenges it presented, and the process through which we dissected its underlying mechanisms.

Overview of the FTO Locus

1. **Genomic Context and LD Structure**:
    - The FTO locus spans a **large region of 50,000 nucleotides**. It sits between two **recombination hotspots**, which are genomic regions with high rates of recombination. However, **within this specific locus**, recombination rates are low, resulting in strong **linkage disequilibrium (LD)**. This creates a **haplotype block**, where many genetic variants are tightly correlated, making it difficult to pinpoint the specific causal variant.
    - The region contains **89 common variants** that are all in high LD, spanning the **first intron and a portion of the second intron** of the FTO gene. Importantly, there are **no protein-coding alterations** among these variants, suggesting that the genetic effects are likely due to **regulatory changes**, rather than changes in the protein sequence.
2. **Broad Phenotypic Associations**:
    - While the FTO locus is primarily associated with **obesity**, it also contributes to a spectrum of related **metabolic traits**, including **type 2 diabetes**, **cardiovascular diseases**, and other obesity-related phenotypes. This broad association highlights the systemic impact of obesity on various health conditions.
3. **Statistical Significance and GWAS Signal**:
    - On a **Manhattan plot** of GWAS results, the FTO locus stands out as a towering peak, well above the genome-wide significance threshold, indicating a very strong association with obesity risk. It remains the **strongest genetic signal for obesity** across diverse populations.

Challenges in Understanding the FTO Locus

Despite its strong genetic association, the **FTO locus presented several challenges** for mechanistic understanding:

- **Dense LD Structure**: Due to the extensive LD across this haplotype block, many SNPs are co-inherited. This makes it difficult to disentangle which specific SNP is causally related to the phenotype.
- **Regulatory vs. Coding Effects**: The absence of coding variants suggested a **regulatory role**, but pinpointing the exact mechanism was challenging without understanding which tissue, enhancer elements, and downstream genes were involved.
- **Multiple Hypotheses and Confusion**: There was significant debate in the scientific community about the **target gene** and the relevant tissue context. Initial studies speculated that the **FTO gene** itself was the causal target due to its proximity to the association signal. However, subsequent work hinted at other possible targets like **IRX3**, **IRX5**, and even genes further away.

The FTO Gene: A Brief History

The FTO gene was originally identified in a mouse screen for **fused toes**, part of a series of genes labeled Fuso-N, Fuso-O (FTO), and so on. When researchers discovered the association of the FTO locus with obesity, they whimsically named the gene **"Fatso,"** later officially renamed as **fat and obesity-associated gene** (FTO). This led to an initial bias towards implicating FTO in obesity, given its suggestive name and location at the peak of the association signal.

However, as studies progressed, it became clear that **knockout experiments targeting FTO** also affected the surrounding regulatory region, casting doubt on whether the effects were due to changes in the FTO gene itself or its regulatory landscape.

Divergent Hypotheses and Initial Missteps

Given the strong association, numerous studies attempted to identify the **causal gene** and the tissue context. The hypotheses varied widely:

- Some suggested it was **FTO itself**, given the proximity and the apparent connection to obesity.
- Others proposed that the locus might influence **IRX3**, a gene involved in **pancreatic function**.
- There were additional hypotheses involving genes like **RBL2** in **lymphocytes**, with speculation about a wide array of possible tissues, from the brain to the pancreas, indicating the lack of a coherent understanding.

This led to a **scattered approach** with conflicting findings. For example, a high-profile study from a Chicago group proposed that **IRX3** in the brain was the target, but their findings showed a minor effect, raising skepticism about its true role as the causal gene.

Our Approach: A Systematic Dissection

Faced with the uncertainty and the diversity of hypotheses, we set out to systematically dissect the FTO locus. Our goals were to:

1. **Identify the Relevant Tissues**:
    - Using **epigenomic annotations**, we aimed to pinpoint which tissues exhibited activity of the regulatory elements overlapping the FTO-associated SNPs.
2. **Determine the True Target Genes**:
    - Instead of assuming proximity-based effects, we leveraged **chromatin interaction data** and **long-range regulatory analysis** to test if distant genes might be involved.
3. **Identify the Causal Variant**:
    - We used **fine-mapping techniques** to isolate a single variant with the highest posterior probability of causality, focusing on regulatory SNPs rather than coding variants.
4. **Establish the Mechanism of Action**:
    - By analyzing **transcription factor motifs** and performing **functional assays**, we aimed to understand the **regulatory impact** of the causal variant.
5. **Validate with Experimental Evidence**:
    - Finally, we employed **CRISPR genome editing** and other experimental approaches to validate the causality of the identified variant and its impact on gene expression and metabolic traits.

This comprehensive and integrative strategy allowed us to **move beyond statistical associations** and provide a **mechanistic understanding** of the FTO locus, setting a precedent for dissecting other complex

GWAS regions.

In the next section, we will discuss the **findings and experimental results** that emerged from this dissection and how they transformed our understanding of the FTO locus and its role in obesity. This will include insights into the **novel regulatory pathways** uncovered, the identification of **IRX3 and IRX5 as the true target genes**, and the broader implications for metabolic disease and potential therapeutic avenues.

43:40 Step 1: Establishing the Relevant Tissue and Cell Type

To dissect the mechanisms underlying the **FTO locus** and its association with obesity, the first crucial step is to **identify the tissue or cell type** where the genetic variant exerts its effect. This step sets the stage for understanding the biological context and helps narrow down the potential pathways and mechanisms involved.

Mapping Epigenomic Activity Across Tissues

To determine the relevant tissue, we leveraged comprehensive **epigenomic maps** from projects like the **Roadmap Epigenomics Project** and the **ENCODE Project**. As the senior author of the **Roadmap Epigenomics paper**, I had led a large collaborative effort to profile the **epigenomic landscape** across a wide array of human tissues and cell types. This dataset included hundreds of diverse **epigenomic assays**, which are technically challenging and require substantial resources. These assays provide insights into:

- **Chromatin accessibility** (open chromatin regions),
- **Histone modifications** (e.g., H3K27ac for active enhancers),
- **DNA methylation patterns**, and
- **Transcription factor binding sites**.

With this wealth of data, we could systematically map the activity of **non-coding regions** across many different tissues, enabling us to identify where regulatory elements like enhancers are active.

Applying Epigenomic Mapping to the FTO Locus

When we applied these methods to the **FTO locus**, we observed a large region of active chromatin spanning **12,000 base pairs** at the beginning of the locus. This region exhibited strong enhancer activity in a specific type of precursor cells known as **mesenchymal stem cells**. These cells are **multipotent progenitors**, capable of differentiating into several important tissue types, including:

1. **Chondrocytes** (precursors to cartilage),
2. **Osteoblasts** (bone-forming cells),
3. **Cardiac muscle cells**, and most importantly,
4. **Adipocytes** (fat cells).

This finding was critical because it pointed us towards a **lineage** that could differentiate into various tissues, but particularly suggested a role in **adipocyte biology**, linking the enhancer activity directly to the development of fat cells.

Different Types of Adipocytes: White, Brown, and Beige

To understand the implications of the enhancer activity in mesenchymal stem cells, we need to distinguish between the different types of fat cells:

1. **White Adipocytes**:
    - These cells are specialized for **energy storage**. They have large lipid droplets and serve as the body's primary reserve of stored energy.
    - They are abundant in adults and are critical for maintaining energy balance.
2. **Brown Adipocytes**:
    - These cells are specialized for **thermogenesis**, or heat production. They contain numerous **mitochondria**, giving them a brown color.
    - Brown adipocytes play a key role in **calorie burning** through a process called **uncoupled respiration**, where the **mitochondrial membrane depolarizes**, leading to the dissipation of energy as heat instead of storing it as ATP.
3. **Beige or "Bright" Adipocytes**:
    - These cells are **intermediate** between white and brown adipocytes. They can **switch** between an energy-storing and an energy-burning state, depending on environmental cues like

temperature or dietary changes.
    - Beige adipocytes are thought to have a role in **adaptive thermogenesis**, helping to regulate body temperature and energy expenditure.

Given the potential for **adipocyte plasticity**, it was plausible that the regulatory activity in the FTO locus was involved in influencing the balance between **energy storage (white adipocytes)** and **energy burning (brown/beige adipocytes)**.

Identifying the Impact of the Obesity-Associated Allele

We next focused on determining whether the **obesity-associated allele** had a functional impact on enhancer activity. To do this, we measured **enhancer activity** across different segments of the 12,000 base pair region, comparing the **risk allele** (associated with obesity) to the **non-risk (lean) allele**.

Key Findings:

- **Increased Enhancer Activity with the Risk Allele**:
    - The **obesity-associated (risk) allele** exhibited **increased activity** in adipocytes, suggesting a **gain-of-function** effect. This was surprising because we often think of disease-associated variants as **loss-of-function** mutations that disrupt normal regulatory activity.
    - In this case, the risk allele seemed to **enhance the activation** of the regulatory region, possibly leading to **overexpression** of downstream genes.

Possible Mechanisms of Gain-of-Function

We considered several hypotheses to explain this gain-of-function effect:

1. **Creation of a New Binding Site for an Activator**:
    - The risk allele may have **introduced a new motif** or binding site for a transcriptional activator. This would lead to **increased recruitment** of activators, thereby boosting enhancer activity and subsequent gene expression.
2. **Disruption of a Repressor Binding Site**:
    - Alternatively, the risk allele might have **abolished a repressor binding site**, removing inhibitory effects and leading to increased enhancer activity. This type of **de-repression** is another form of gain-of-function where the absence of repression leads to heightened gene activation.

These two mechanisms are not mutually exclusive, and both could contribute to the increased enhancer activity observed with the risk allele.

Hypothesis: Role of Adipocyte Differentiation and Function

Given the strong enhancer activity in **adipocyte precursors** and the gain-of-function nature of the risk allele, we hypothesized that the FTO locus might influence the **balance between white and beige adipocyte differentiation**. This balance is crucial for determining whether adipocytes will primarily **store energy** or **burn energy**.

- **Increased Enhancer Activity** due to the risk allele could lead to **greater expression of genes** that promote white adipocyte differentiation, tipping the balance towards **energy storage**, thereby predisposing individuals to obesity.
- Conversely, reducing this enhancer activity might favor the differentiation towards **beige adipocytes**, promoting **energy expenditure** and potentially providing a protective effect against obesity.

Summary

The first step of our systematic dissection revealed critical insights into the **tissue context** and **regulatory dynamics** of the FTO locus:

- The relevant tissue and cell type were identified as **mesenchymal stem cells** with differentiation potential towards **adipocytes**, particularly those involved in energy storage and thermogenesis.
- The obesity-associated risk allele showed a **gain-of-function effect**, leading to **increased enhancer activity**, suggesting a novel regulatory mechanism contributing to obesity.

This set the stage for our next steps, where we aimed to identify the **target genes** of this enhancer and the **transcriptional regulators** involved. This comprehensive analysis laid the groundwork for uncovering a novel regulatory circuit with implications for obesity therapy and metabolic health.

49:29 Step 2: Establishing Downstream Target Genes

With the relevant tissue and cell type identified as **adipocyte precursors** and **preadipocytes**, the next step in dissecting the mechanism of the **FTO locus** was to identify the **downstream target genes** influenced by the obesity-associated variants. This is crucial for understanding the **regulatory circuit** and the biological processes through which these variants impact obesity.

Linking Enhancer Activity to Target Genes

To determine the target genes regulated by the enhancer region, we employed **three complementary strategies**:

1. **Functional Genomic Linking**:
    - This method involves analyzing the **correlation of gene expression** with enhancer activity. When the enhancer is active, what genes are also active?
    - The idea is that even if an enhancer is physically closer to one gene, it may exhibit stronger **co-activation** with another gene farther away. By tracking patterns of synchronized activation across samples, we can infer potential regulatory links between enhancers and genes.
2. **Expression Quantitative Trait Loci (eQTL) Analysis**:
    - **eQTLs** are genomic loci where genetic variants correlate with **gene expression levels**. This is analogous to GWAS but focuses on **molecular phenotypes** (gene expression) instead of clinical phenotypes (disease traits).
    - We look for variants within the FTO locus that show a strong correlation with changes in the expression of nearby genes. For example, if a **G-to-T change** at a specific variant leads to increased expression of a gene, this suggests a **regulatory link** between that variant and the gene.
    - This provides **genetic evidence** that a non-coding variant is influencing gene expression, helping us identify potential target genes.
3. **Chromatin Conformation Capture (Hi-C)**:
    - This method explores the **3D structure of the genome**, revealing physical interactions between distant genomic regions that are brought into proximity by chromatin looping.
    - Using **Hi-C data**, we can observe long-range interactions between the FTO enhancer and potential target genes that are **hundreds of thousands to millions of base pairs away**.
    - In our analysis, we found strong chromatin contacts linking the FTO locus, specifically the variant **rs1421085**, to distant genes such as **IRX3** and **IRX5**. These interactions spanned **long distances**, indicating a regulatory connection that would not have been apparent based solely on linear genome proximity.

Testing Candidate Target Genes

Armed with these three lines of evidence (functional genomic linking, eQTL analysis, and chromatin conformation data), we proceeded to experimentally test which genes were influenced by the risk variant. We selected a cohort of individuals who were **homozygous for the risk allele** and compared them with individuals who were **homozygous for the non-risk allele**.

Key Findings:

- **FTO Gene Expression**:
    - Despite its proximity to the associated variants, the expression of the **FTO gene itself** did **not change** in response to the genotype. This ruled out FTO as the direct target of the enhancer, contradicting earlier hypotheses that had focused solely on the nearest gene.
- **IRX3 and IRX5 Gene Expression**:
    - In contrast, we observed significant changes in the expression of **IRX3** and **IRX5**, two genes located far away from the FTO locus:
        - **IRX3** is approximately **600,000 base pairs** away.
        - **IRX5** is even farther, about **1.2 million base pairs** away.
    - These genes showed **increased expression** in individuals carrying the risk allele, indicating a **gain-of-function effect**. The risk allele appears to enhance the activity of the enhancer, leading to **upregulation** of IRX3 and IRX5.

Long-Range Regulatory Effects

The observed effects on IRX3 and IRX5 were particularly striking because these genes are separated from the FTO locus by multiple **linkage disequilibrium (LD) block boundaries** and **recombination hotspots**. Without the **3D chromatin interaction data**, it would have been nearly impossible to predict these genes as targets due to their linear distance from the FTO locus.

- The **Hi-C data** revealed a **looping interaction** between the enhancer region within the FTO locus and the distant IRX genes, explaining the long-range regulatory effect.
- This highlights the importance of considering the **3D genome architecture** when identifying target genes, especially in complex traits where regulatory elements may act over large distances.

eQTL Evidence Confirms Target Genes

Further supporting evidence came from **eQTL analysis**:

- The obesity-associated variant was identified as an **eQTL** for both **IRX3** and **IRX5**, meaning that changes in the genotype directly influenced the expression levels of these genes.
- Importantly, this eQTL effect was observed **specifically in early adipocyte differentiation**, not in fully mature adipocytes or in whole tissue samples. This finding underscores the importance of studying the **right developmental stage**, as gene regulation can be highly dynamic and stage-specific.

Specificity of the eQTL Effect

The eQTL effect was limited to **preadipocytes**, the precursors of mature adipocytes. This is consistent with our earlier finding that the enhancer activity was strongest in **mesenchymal stem cells** and their immediate derivatives.

- **Preadipocytes** are the transitional stage between mesenchymal stem cells and fully differentiated adipocytes. During this stage, the cells are actively undergoing changes that determine whether they will become **white adipocytes** (energy-storing) or **beige/brown adipocytes** (energy-burning).
- The obesity-associated variant appears to influence this decision by **upregulating IRX3 and IRX5**, which are involved in regulating the balance between **energy storage** and **thermogenesis**.

Summary of Step 2

The second step of our analysis provided critical insights into the **downstream regulatory effects** of the FTO locus:

- We identified **IRX3 and IRX5** as the primary target genes of the FTO-associated enhancer, based on a combination of **functional genomics**, **eQTL analysis**, and **chromatin interaction data**.
- The **risk allele** led to a **gain-of-function effect**, increasing the enhancer activity and, consequently, the expression of IRX3 and IRX5.
- The regulatory effects were **stage-specific**, manifesting only during **early adipocyte differentiation**, underscoring the importance of studying the **right developmental context**.

This step moves us closer to a comprehensive mechanistic understanding of how the FTO locus contributes to obesity, setting the stage for further exploration of the **upstream regulators** and the **biological processes** involved in the next steps. This detailed dissection provides a model for how to systematically approach the identification of target genes in complex trait loci, integrating multiple lines of evidence to uncover the underlying molecular circuitry.

55:27 Step 3 - Establishing causal nucleotide variant

With the relevant tissue (preadipocytes) identified and the downstream target genes (**IRX3** and **IRX5**) established, the next step in dissecting the **FTO locus** was to pinpoint the **causal nucleotide variant** responsible for the observed regulatory effects.

The Challenge of Finding the Causal Variant

The FTO locus spans a large region with **89 common variants** in high linkage disequilibrium (LD), meaning that these variants are inherited together and correlate strongly. However, **not all of these variants** are functional; only one or a few might be directly causing the change in enhancer activity. Our goal was to identify which specific **single nucleotide polymorphism (SNP)** was the driver.

Identifying Key Regulatory Motifs

To determine the causal variant, we needed to examine the **regulatory motifs** in the enhancer region. **Regulatory motifs** are specific DNA sequences recognized by transcription factors (TFs) that can activate or

repress gene expression. By identifying which motifs are **disrupted** by the risk allele, we can home in on the causal variant. There are **several strategies** for assessing the importance of these motifs:

1. **Experimental Testing**:
    - We can **mutate** each of the 89 common variants and experimentally test their effects on enhancer activity. While feasible today, at the time of this study, this approach was **technically challenging** due to the large number of variants and the need for precise, high-throughput assays.
2. **Motif Enrichment Analysis**:
    - We can look for **enriched motifs** in the GWAS loci associated with obesity. By examining the entire set of genetic regions linked to BMI, we can identify regulatory motifs that appear **repeatedly** across these loci. If a specific motif is frequently present in obesity-associated regions, it is likely functionally relevant.
3. **Evolutionary Conservation**:
    - We can analyze the **conservation** of motifs across different mammalian species. Regulatory elements that are **evolutionarily conserved** are more likely to be functionally important because they have been preserved through selective pressures. If a variant disrupts a conserved motif, it is more likely to have a functional impact.

Applying These Strategies

Using a combination of **motif enrichment**, **evolutionary conservation**, and **functional assays**, we identified a **regulatory motif** in the enhancer region that was:

- **Enriched** in other GWAS loci associated with BMI.
- **Evolutionarily conserved**, indicating its functional importance across species.
- **Disrupted by a specific SNP** within the FTO locus.

The identified motif was an **AT-rich interacting domain motif (AATA)**, a sequence recognized by specific transcription factors. This motif was both **overlapping** with the risk SNP and **conserved** across mammals, making it a strong candidate for the causal regulatory element.

Functional Validation of the Causal SNP

To confirm that this SNP was indeed the causal variant, we performed a series of **targeted mutation experiments**:

1. **Mutagenesis of the Risk Allele**:
    - We introduced the risk allele into a **10,000 base pair (bp)** region of the enhancer and observed a significant **gain of function** (increased enhancer activity), suggesting that the risk allele **increases expression** of the downstream target genes.
    - When we introduced the risk allele into a **1,000 bp** region, the gain of function was even more pronounced, indicating that the enhancer activity was localized within a smaller segment but still required a substantial context for full function.
2. **Testing Shorter Regions (100 bp)**:
    - We also tested a **100 bp** region containing the risk allele. Interestingly, this shorter region **did not show any change** in enhancer activity, even when the risk allele was present.
    - **Why was there no effect with the 100 bp region?**
        - The FTO enhancer is a **large, complex regulatory element**, requiring interactions across a broad sequence context. The 100 bp segment was **too short** to capture the full set of transcription factor binding sites and structural elements needed for enhancer function.
        - This finding highlights the importance of testing variants within a **sufficiently large sequence context** to capture the full regulatory potential of the enhancer.

The Causal SNP Recapitulates the Risk Haplotype

By introducing the **single risk SNP** into a longer enhancer segment, we were able to **recapitulate the full gain of function** seen with the entire risk haplotype. This provided strong evidence that the **single nucleotide change** was sufficient to drive the observed changes in gene expression, specifically **de-repressing** the

enhancer activity.

- The **risk allele** led to a **gain of function** by either:
    - **Creating a new binding site** for an activator, increasing enhancer activity.
    - **Disrupting a repressor site**, leading to loss of repression and increased enhancer activation.

In either case, the effect of the variant was to **enhance the expression** of **IRX3** and **IRX5**, the key target genes linked to obesity risk.

Summary of Step 3

In this step, we successfully identified the **causal nucleotide variant** responsible for the regulatory changes at the FTO locus:

- We used a combination of **motif analysis**, **conservation**, and **functional assays** to pinpoint the SNP that disrupted a conserved regulatory motif.
- Experimental validation confirmed that the **risk allele** caused a **gain of function**, increasing enhancer activity and the expression of downstream target genes.
- The effect of the SNP was **context-dependent**, requiring a sufficiently large enhancer segment to observe the change, highlighting the complexity of **regulatory elements** in the genome.

This step completes the identification of the **causal variant**, setting the stage for exploring the **upstream regulators** and the **biological processes** through which this variant influences obesity. By integrating multiple lines of evidence, we provide a comprehensive mechanistic dissection of the FTO locus, serving as a model for fine-mapping and functional characterization of non-coding GWAS hits.

59:59 Step 4 - Establish upstream regulator causality

Having identified the **relevant cell type** (preadipocytes), the **downstream target genes** (**IRX3** and **IRX5**), and the **causal nucleotide variant** (**rs1421085**), the next logical step was to identify the **upstream regulator** that interacts with this variant. This step is crucial for understanding **which transcription factor (TF)** directly mediates the effects of the risk allele, thereby influencing the enhancer activity and gene expression changes.

Identifying Candidate Transcription Factors

The first challenge was determining which transcription factor might be binding to the **AT-rich motif** disrupted by the risk SNP (**rs1421085**). Several candidate transcription factors were considered, including:

- **LHX6**: A known transcription factor involved in developmental processes.
- **NKX6.3**: Another candidate based on its motif recognition properties.
- The **ARID family**: A large group of transcription factors (e.g., **ARID1A**, **ARID1B**, **ARID2**, **ARID3A**, **ARID3B**, **ARID3C**, and so on) that recognize AT-rich DNA motifs.

To narrow down this list, we focused on the **expression patterns** of these transcription factors, particularly in preadipocytes, since we had already established this as the relevant cell type.

Expression Profiling and Initial Findings

We examined the **expression levels** of the ARID family transcription factors in both risk and non-risk individuals. One transcription factor, **ARID5B**, stood out due to its high expression level in preadipocytes across both risk and non-risk genotypes.

Is it problematic that ARID5B is expressed in both risk and non-risk individuals?

- **No, it is not problematic.** The key issue here is **not the expression of the transcription factor** (the *trans* component), but the integrity of the **binding site** (the *cis* component).
- The **trans-acting regulator (ARID5B)** remains expressed regardless of genotype, but its ability to bind to the enhancer is **compromised** when the **cis-regulatory motif** is disrupted by the risk allele.

In other words, the **problem lies in the DNA sequence alteration** (the disrupted motif) rather than the transcription factor itself. This distinction is crucial: the transcription factor (ARID5B) is present, but the risk SNP disrupts its **binding site**, preventing successful repression of the enhancer.

Testing Binding Specificity

To validate ARID5B as the likely **upstream regulator**, we tested its binding specificity using:

1. **Whole adipose tissue** samples.
2. **Isolated adipocyte stem cells** from both lean and obese individuals.

ARID5B exhibited the **strongest binding affinity** to the enhancer region containing the AT-rich motif, confirming it as the prime candidate for regulating this locus.

Establishing Causality Through Epistasis

Identifying ARID5B as the key transcription factor was an important step, but to **prove causality**, we needed to demonstrate that ARID5B directly mediates the effects of the risk SNP. We used a method called **epistasis analysis** to establish whether the SNP and the transcription factor are part of the **same regulatory pathway**.

Epistasis Analysis:

- **Epistasis** involves testing the combined effects of **two genetic disruptions**: the *cis* element (the SNP) and the *trans* element (the transcription factor).
- We performed **cis perturbations** by altering the nucleotide sequence (introducing the risk allele or non-risk allele).
- We performed **trans perturbations** using **small interfering RNA (siRNA)** to knock down the expression of ARID5B.

Results of the Epistasis Experiments

We conducted enhancer activity assays using **luciferase reporters** and measured the expression levels of the **target genes** (**IRX3** and **IRX5**). The results were striking and confirmed the **interdependence** of the SNP and the transcription factor:

1. **Non-risk haplotype with intact ARID5B**:
    - Successful **repression** of the enhancer and downstream gene expression.
    - This serves as the **baseline** state, where the motif is intact, and ARID5B can bind effectively.
2. **Non-risk haplotype with ARID5B knockdown**:
    - **De-repression** of the enhancer and increased target gene expression.
    - This shows that the **presence of ARID5B** is necessary for repression.
3. **Risk haplotype with intact ARID5B**:
    - **De-repression** of the enhancer despite the presence of ARID5B.
    - This indicates that the **disrupted motif** (due to the risk SNP) prevents ARID5B from binding, leading to a loss of repression.
4. **Risk haplotype with ARID5B knockdown**:
    - **No additional effect** beyond the de-repression observed with the risk haplotype alone.
    - This suggests that the disruption of either the **motif (cis)** or the **transcription factor (trans)** alone is sufficient to cause de-repression, and disrupting both does not have an additive effect.

Interpretation of Epistasis Results

The epistasis analysis demonstrated that:

- **Both the motif and the transcription factor** are required for successful enhancer repression.
- The **risk SNP** disrupts the motif, preventing ARID5B from binding effectively, leading to **enhancer activation** (de-repression).
- Knocking down ARID5B in the non-risk context **mimics** the effect of the risk SNP, further confirming that ARID5B is the **causal upstream regulator** mediating the effect of the variant.

Summary of Step 4

This step successfully established the **causal relationship** between the disrupted motif, the risk SNP, and the transcription factor ARID5B:

1. **Identification of the Upstream Regulator**:
    - ARID5B was identified as the key transcription factor binding to the AT-rich motif disrupted by the risk SNP.
2. **Validation of Regulatory Binding**:
    - ARID5B showed strong binding to the enhancer region in preadipocytes, supporting its role as the upstream regulator.

3. **Epistasis Analysis**:
    - The combined cis and trans perturbation experiments confirmed that both the motif and ARID5B are required for enhancer repression.
    - Disrupting either the motif (with the risk SNP) or the regulator (with siRNA knockdown) resulted in **loss of repression** and increased expression of the target genes (**IRX3** and **IRX5**).

Conclusion

With the identification of ARID5B as the upstream regulator, we have now established the **full mechanistic pathway** from the genetic variant (SNP) to the enhancer activity, the transcription factor binding, and the downstream effects on gene expression. This comprehensive dissection provides a **causal link** between the non-coding variant at the FTO locus and the molecular mechanisms driving increased obesity risk.

This step completes the core analysis, setting the stage for exploring the **biological and organismal implications** of these findings, particularly how they contribute to adipocyte differentiation and energy metabolism, ultimately influencing obesity and related metabolic disorders.

1:04:58 Step 5 - Establish cellular phenotypic consequences

Having identified the **relevant cell type** (preadipocytes), the **downstream target genes** (**IRX3** and **IRX5**), the **causal variant** (**rs1421085**), and the **upstream regulator** (**ARID5B**), we now turn to the **cellular phenotypic consequences**. This step aims to bridge the gap between **genetic variation** and **disease manifestation** by elucidating how these molecular changes translate into cellular behavior and ultimately influence obesity risk.

Bridging the Gap Between Genetic Variation and Disease

While we had already established the molecular circuitry—showing that the risk variant disrupts a repressive interaction, leading to increased enhancer activity and elevated expression of **IRX3** and **IRX5**—we still needed to understand the **biological impact** of this altered gene expression. Specifically, we sought to determine:

- **How does increased expression of IRX3 and IRX5 affect preadipocytes?**
- **What are the downstream metabolic consequences that link this molecular disruption to obesity?**

To address these questions, we needed to investigate the **cellular phenotypes** influenced by the disrupted enhancer activity.

Co-Expression Analysis of IRX3 and IRX5

We began by examining the **co-expression patterns** of genes in adipocyte precursor cells (preadipocytes). By profiling gene expression across **20 individuals**, we analyzed how the expression of **IRX3** and **IRX5** correlates with other genes:

- **Positive Correlation with Lipid Metabolism Genes**:
    - When **IRX3** and **IRX5** expression levels were **elevated**, we observed a significant increase in the expression of genes involved in **lipid metabolism**.
    - This suggests a shift towards a phenotype characterized by enhanced **lipid storage**, consistent with the risk allele's association with obesity.
- **Negative Correlation with Mitochondrial Genes**:
    - Conversely, when **IRX3** and **IRX5** expression was low, we saw an increase in the expression of genes involved in **mitochondrial function**.
    - This points to an alternative phenotype where there is a greater emphasis on **mitochondrial activity** and **energy expenditure**.

Functional Consequence: Shift in Adipocyte Phenotype

The co-expression analysis suggested a **functional shift** in adipocyte differentiation driven by the risk allele:

- **Increased Expression of IRX3/IRX5** (Risk Allele):
    - Leads to a **reduction in mitochondrial gene expression** and **mitochondrial activity**.
    - Promotes a **lipid storage phenotype**, characterized by larger white adipocytes that are optimized for storing energy in the form of fat.
- **Decreased Expression of IRX3/IRX5** (Protective Allele):
    - Associated with an increase in **mitochondrial gene expression**.

- Favors a phenotype with smaller, more metabolically active adipocytes that prioritize **energy burning** over storage.

Role of Uncoupling Protein 1 (UCP1)

A key finding in our analysis was the differential expression of **uncoupling protein 1 (UCP1)**, a critical protein involved in **thermogenesis**:

- **UCP1** is known to **depolarize the mitochondrial membrane**, leading to **heat production** instead of ATP synthesis—a process called **non-shivering thermogenesis**.
- We observed that **UCP1 expression** was significantly reduced in adipocytes carrying the **risk allele**, indicating a **decrease in mitochondrial thermogenesis**.
- In contrast, individuals with the **protective allele** showed **higher UCP1 expression**, both under normal conditions and upon **cold induction**, suggesting greater **thermogenic capacity** and increased energy expenditure.

Proposed Model: Beige vs. White Adipocyte Phenotype

Our findings suggest a model where the genetic variant at the FTO locus **modulates adipocyte differentiation**, influencing the balance between **beige adipocytes** (calorie-burning) and **white adipocytes** (calorie-storing):

1. **Protective Allele**:
   - Enhancer repression is intact, leading to **lower expression** of IRX3 and IRX5.
   - Promotes differentiation towards a **beige adipocyte phenotype**, which is **rich in mitochondria** and favors **calorie burning**.
   - Increased expression of **UCP1** enhances thermogenesis and energy expenditure.
2. **Risk Allele**:
   - Disrupted enhancer repression leads to **elevated expression** of IRX3 and IRX5.
   - Drives differentiation towards a **white adipocyte phenotype**, characterized by **larger cells** that are optimized for **lipid storage**.
   - Reduced **UCP1 expression** diminishes mitochondrial thermogenesis, favoring **energy storage** over expenditure.

Evidence of Cellular Phenotypic Change

We validated this model through **functional assays** in preadipocytes and adipocytes:

- We observed that adipocytes carrying the **risk allele** exhibited **fewer mitochondria**, consistent with a shift away from energy expenditure.
- These cells also showed a significant increase in **lipid droplet size**, further supporting the shift towards **enhanced lipid storage**.

The **protective allele**, on the other hand, was associated with **higher mitochondrial content** and **increased thermogenic activity**, as evidenced by elevated **UCP1 expression** and a greater response to **cold induction**.

Summary of Step 5

We have now established a clear link between the genetic variant and its cellular consequences:

1. **Altered Gene Expression**:
   - The risk SNP disrupts the repressive binding of ARID5B, leading to **increased expression of IRX3 and IRX5**.
2. **Shift in Adipocyte Differentiation**:
   - Elevated IRX3/IRX5 expression favors the formation of **white adipocytes**, which store energy.
   - Lower expression (protective allele) promotes the formation of **beige adipocytes**, which burn energy.
3. **Metabolic Consequences**:
   - The risk allele leads to a **decrease in mitochondrial activity** and **thermogenesis**, reducing the body's capacity to burn calories.

- This imbalance shifts the overall energy homeostasis towards **lipid accumulation**, contributing to an increased risk of obesity.

Conclusion

This step successfully elucidates the **cellular mechanism** linking the FTO locus variant to obesity. By disrupting the regulatory balance between energy storage and energy expenditure in adipocytes, the risk allele shifts adipocyte differentiation towards a **calorie-storing phenotype**, contributing to increased fat accumulation and a higher predisposition to obesity. This mechanistic insight provides a direct bridge from **genetic variation** to **disease phenotype**, paving the way for potential therapeutic interventions targeting adipocyte metabolism and differentiation.

The next and final step involves exploring the **organismal implications** of these findings and validating the model in vivo to understand the broader metabolic consequences and potential for therapeutic modulation.

1:07:38 Step 6 - Establish organismal phenotypic consequences

Having established the **tissue**, **target genes**, **causal nucleotide variant**, and **upstream regulator**, and identified the **cellular phenotypic changes**, the final step is to validate these findings at the **organismal level**. We need to demonstrate how these molecular and cellular changes manifest in **whole-body phenotypes**, particularly with respect to **obesity**, and how they ultimately influence **energy homeostasis** and **weight regulation**.

Testing in Mouse Models

To assess the organismal consequences, we turned to **mouse models**, where we could manipulate the key components of the identified circuitry and evaluate their impact on **whole-body metabolism** and **weight regulation**.

Experimental Approach:

1. **Manipulation of ARID5B (Upstream Regulator):**
   - We performed both **knockdown** and **overexpression** of ARID5B in mouse adipocytes.
   - Additionally, we utilized **genome editing** to directly alter the **rs1421085 SNP** (C/T allele) to examine its causal effect.
2. **Manipulation of IRX3 and IRX5 (Target Genes):**
   - We knocked down and overexpressed **IRX3** and **IRX5** in both risk and non-risk backgrounds to determine their effect on **thermogenesis** and **energy expenditure**.

Cellular and Organismal Consequences

1. Cellular Thermogenesis:

- **Knockdown of IRX5** in risk individuals led to a **significant increase** in **oxygen consumption rate**, a proxy for **thermogenesis**.
- **Knockdown of IRX3** also showed a similar increase in thermogenesis.
- Conversely, **overexpression of IRX3** in non-risk individuals led to a **decrease in thermogenesis**, indicating its role in **suppressing energy expenditure**.

These results confirm that the **expression levels** of IRX3 and IRX5 are key modulators of **thermogenic activity**, directly linking the genetic variant to **altered cellular metabolism**.

2. Whole-Body Metabolic Changes in Mice:

- We generated mice carrying a **dominant-negative allele** of IRX3, specifically in adipose tissue using an **adipocyte-specific promoter**.
- The results were striking:
  - These mice showed a **dramatic reduction in fat mass**, with up to **50% less body fat** compared to control mice.
  - Despite their lower fat mass, these mice did not show any significant change in **respiration rate** under normal conditions, indicating that the reduction in fat mass was not due to increased physical activity.

3. Increased Energy Expenditure:

- Detailed metabolic profiling revealed that the **energy expenditure** of these mice was consistently higher:

- Both during the **night**, when mice are most active, and during the **day**, when they are resting.
- These mice effectively exhibited a **higher basal metabolic rate**, leading them to **burn more calories even while sleeping**.

4. High-Fat Diet Resistance:

- When subjected to a **high-fat diet**, normal mice typically gained significant weight.
- However, the **IRX3-dominant negative mice** showed remarkable **resistance to weight gain**:
  - Despite consuming the same high-fat diet, these mice were unable to accumulate additional body fat, highlighting a protective effect against **diet-induced obesity**.

Genome Editing Validation

To provide conclusive evidence of the **causal role of the SNP** (rs1421085), we employed **CRISPR genome editing** to introduce a **single nucleotide change** in the mice:

1. **Editing the C Allele to T (Risk to Protective):**
   - We altered the **risk allele (C)** to the **protective allele (T)** in mice carrying the risk haplotype.
   - The change resulted in a **reversion of the phenotype**:
     - **Thermogenic activity increased**, and the mice exhibited a **leaner body phenotype**.
   - This confirmed that the single nucleotide alteration was sufficient to switch the **molecular, cellular, and organismal phenotypes**, establishing direct **causality**.
2. **Reversion to Risk Allele (T to C):**
   - To rule out any off-target effects of the CRISPR edit, we performed a second genome edit, reverting the T back to the C allele.
   - The **risk phenotype** was restored, demonstrating that the observed effects were specifically due to the **rs1421085 SNP** and not any unintended genomic changes.

Integrated Model of Genetic and Phenotypic Consequences

The results from the mouse model provide a comprehensive link between the **genetic variant**, the **molecular circuitry**, and the **organismal phenotype**:

- **Risk Allele (C)**:
  - Disrupts ARID5B binding, leading to **increased enhancer activity**.
  - Upregulates **IRX3** and **IRX5**, shifting adipocyte differentiation towards a **white adipocyte phenotype**.
  - Reduces thermogenic capacity, leading to **increased fat storage** and a higher risk of obesity.
- **Protective Allele (T)**:
  - Maintains ARID5B binding, keeping the enhancer **repressed**.
  - Suppresses IRX3 and IRX5 expression, favoring differentiation towards **beige adipocytes**.
  - Enhances mitochondrial activity and **thermogenesis**, promoting **calorie burning** and a leaner phenotype.

Implications for Obesity and Therapeutic Interventions

This study serves as a **proof-of-concept** for how **genome-wide association studies (GWAS)** can be systematically dissected to uncover **disease mechanisms**:

1. It highlights the **complex interplay** between genetic variants, regulatory elements, and downstream gene networks in shaping **disease phenotypes**.
2. It suggests potential therapeutic strategies:
   - Targeting **IRX3** and **IRX5** could shift adipocyte metabolism towards a more **thermogenic profile**, counteracting obesity.
   - Modulating the activity of the identified enhancer or its **upstream regulator (ARID5B)** could serve as an **intervention point** for altering adipocyte differentiation.

Conclusion of Step 6

The organismal phenotypic analysis validated the entire proposed mechanism, demonstrating that the single

SNP identified at the **FTO locus** influences not just **gene expression**, but also **cellular metabolism**, and ultimately affects **whole-body energy balance** and **obesity risk**. By employing a combination of **genetic manipulation**, **functional assays**, and **in vivo validation**, we have provided a robust framework for understanding the causal relationship between **genetic variation** and **complex traits** such as obesity.

This comprehensive dissection of the FTO locus exemplifies a **systematic approach** for translating **genetic insights** into **mechanistic understanding** and **therapeutic opportunities**, serving as a model for future studies across a broad range of complex diseases.

1:12:33 Quantitative Trait Loci (QTLs): Bridge Genetics-to-Phenotype with Molecular Traits

The challenge in human genetics lies in understanding how **genetic variants** translate into **molecular and cellular changes**, ultimately leading to complex traits and diseases. A single GWAS locus may help identify associations, but to gain a mechanistic understanding, we must systematically connect **genetic variation** to **intermediate phenotypes** like gene expression and DNA methylation. This process involves identifying **Quantitative Trait Loci (QTLs)**, including **eQTLs** (expression QTLs) and **meQTLs** (methylation QTLs), as intermediate links between genotype and phenotype.

From Genome-Wide Association to Molecular Insights

While **GWAS** links genotype directly to a disease phenotype, it is often a black box, providing limited information on **mechanistic pathways**. To unpack this, we can break the process down into **smaller, analyzable pieces**:

1. **Genotype → Intermediate Phenotype (eQTL/meQTL Analysis):**
    - An **eQTL** study examines the relationship between **genetic variants** and changes in **gene expression**.
    - An **meQTL** study explores the relationship between **genetic variants** and changes in **DNA methylation** levels.
2. **Intermediate Phenotype → Disease:**
    - By understanding how genetic variants affect intermediate molecular traits (e.g., expression, methylation), we can better link these effects to **disease outcomes**.

Understanding the Directionality of Effects

One key concept here is the **unidirectional relationship** from genotype to molecular phenotypes:

- **Genetics affects molecular phenotypes, not vice versa**. For example, a SNP might alter the expression of a gene (eQTL) or the methylation status of a nearby CpG site (meQTL).
- However, the reverse is not true: **disease does not alter the genotype**, nor does it directly affect inherited genetic variation.

This **unidirectional arrow** becomes more complex when we examine the relationship between molecular phenotypes and disease. For example, changes in **DNA methylation** can occur as a consequence of disease, creating a **bidirectional arrow**. This makes it challenging to establish whether the methylation change is a **cause** or a **consequence** of the disease.

Testing Causality: Genetic Component of Methylation

To address the **causality challenge**, we can focus on the **genetic component** of methylation:

1. **Predict Methylation Levels from Genotype:**
    - Using **meQTL analysis**, we can impute or predict methylation levels based on the genotype. This allows us to isolate the **genetically driven** component of methylation changes.
2. **Associate the Genetic Component with Disease:**
    - By correlating the **genetically imputed methylation levels** with disease phenotypes, we create a **unidirectional pathway** that suggests a **causal role** for the methylation changes, as they are driven by genetics and not influenced by the disease state.

Systematic Analysis of QTLs

The advantage of using eQTL and meQTL analyses is that they provide a **genome-wide map** of how genetic variation affects molecular traits. This enables us to:

- Identify **many more loci** associated with disease than using GWAS alone.
- Enhance the **statistical power** by aggregating signals from multiple SNPs that collectively influence a

molecular trait (e.g., gene expression or methylation).

Integrative Framework: Relating Genotype, Molecular Traits, and Disease

To perform a comprehensive analysis, we relate:

1. **Genotype**: The underlying genetic variation across individuals.
2. **Molecular Traits**: Intermediate phenotypes like gene expression (eQTLs) and DNA methylation (meQTLs).
3. **Disease Phenotype**: The clinical outcome or trait of interest.

By integrating these components, we create a **multi-layered framework** that can systematically dissect the genetic basis of disease. For example:

- We can relate **genotype** to **expression** using eQTL analysis.
- We can relate **methylation** to genotype using meQTL analysis.
- We can then associate **expression or methylation** changes with the **disease outcome**, leveraging the **QTL data**.

The Challenge of Confounding Factors

In real-world data, numerous **confounding factors** can obscure the relationships between genotype, molecular traits, and disease. These include:

- **Cell Type Composition**: Different cell types may exhibit different expression or methylation profiles, affecting the observed signals.
- **Batch Effects**: Variations due to differences in experimental conditions or processing (e.g., different plates or batches).
- **Demographic Variables**: Factors like age, sex, or environmental exposures can influence molecular phenotypes and disease outcomes.

To address these, we must carefully **control for confounders**, using statistical techniques like **surrogate variable analysis** or including known covariates (e.g., age, sex, technical variables).

Visualizing QTL Associations

To illustrate the relationship between SNPs and molecular phenotypes, we often use **Manhattan plots**, where:

- The **x-axis** represents the genomic coordinates (SNPs across the genome).
- The **y-axis** shows the **-log10 P-value** of association (e.g., between a SNP and gene expression or methylation level).
- **eQTLs** and **meQTLs** with high P-values (low P-values on the logarithmic scale) are considered significant and often cluster near the **locus of interest**, suggesting a **local regulatory effect**.

For example, a SNP with an **astronomically significant P-value** (e.g., 10−20010^{-200}10−200) strongly predicts methylation at a nearby CpG site. This indicates a **highly predictable relationship**, where knowing the SNP genotype allows us to **accurately infer** the methylation status, even decades later in life.

Local Effects of QTLs

A key finding from QTL studies is that:

- **50% of meQTLs** are found within **20 kilobases** (KB) of the methylation site.
- **90% of meQTLs** are within **75 KB**, indicating a predominantly **local regulatory effect**.

This suggests that genetic variation typically exerts its influence on nearby molecular traits, rather than acting through distant or trans effects.

Summary

The systematic study of **Quantitative Trait Loci (QTLs)**, including **eQTLs** and **meQTLs**, provides a powerful framework for understanding the molecular basis of complex traits and diseases:

- By connecting genotype to intermediate molecular phenotypes, QTL analysis helps bridge the gap between genetic variation and clinical outcomes.
- **eQTL** and **meQTL** studies enable us to pinpoint **causal loci** and understand the **local regulatory effects** of genetic variants.
- Integrating QTL analysis with GWAS enhances our ability to identify **causal mechanisms** and offers

new avenues for therapeutic intervention.

This approach sets the stage for a comprehensive and **systematic dissection** of the molecular pathways linking genetic variation to disease, ultimately facilitating the development of **precision medicine** strategies tailored to individual genetic profiles.

1:18:37 Mediation Analysis + Mendelian Randomization to Establish Causality

Understanding the **causal relationships** between genetic variants, molecular traits, and disease outcomes is one of the fundamental goals of genomics research. While **genome-wide association studies (GWAS)** can identify loci associated with disease, they do not establish **causal mechanisms**. To bridge this gap, we use **mediation analysis** and **Mendelian randomization**. These methods help determine whether a genetic variant influences disease **directly**, or **indirectly** through an intermediate molecular trait (e.g., gene expression, methylation).

The Challenge of Causality

A key issue in interpreting genetic associations is determining whether the effect of a SNP on disease is **direct** or **mediated** through an intermediate phenotype:

1. **Direct Effect**: The SNP directly influences the disease without any intermediate factor.
2. **Mediated Effect**: The SNP influences an intermediate phenotype (e.g., gene expression), which in turn affects the disease outcome.
3. **Reverse Causality**: The observed intermediate phenotype change may actually be a consequence of the disease, not a cause.

To address these, we use **conditional analysis** and **causal inference methods**.

Mediation Analysis: Testing for Indirect Effects

Mediation analysis evaluates whether the effect of a genetic variant (SNP) on a disease outcome is mediated through an **intermediate phenotype** (e.g., gene expression, methylation). The key components are:

1. **SNP → Intermediate Phenotype**: The association between the genetic variant and the intermediate phenotype (e.g., eQTL or meQTL).
2. **SNP → Disease**: The direct association between the genetic variant and the disease outcome (identified through GWAS).
3. **Intermediate Phenotype → Disease**: The association between the intermediate phenotype and the disease outcome.

To establish **causality**, we test whether the effect of the SNP on disease **diminishes** when we account for the intermediate phenotype. If the association weakens or disappears, this suggests that the intermediate phenotype **mediates** the SNP's effect on the disease.

Conditional Analysis

The core idea is to perform a **conditional analysis**:

- **Unconditional Effect**: Measure the total effect of the SNP on the disease outcome.
- **Conditional Effect**: Measure the effect of the SNP on the disease **while controlling for the intermediate phenotype**.

If the conditional effect is substantially reduced compared to the unconditional effect, it suggests that the SNP's impact on disease is **mediated through** the intermediate phenotype.

Distinguishing Mediation from Confounding

In real-world data, observed associations may be influenced by **confounding factors**. A **confounder** is a variable that influences both the intermediate phenotype and the disease, creating a **spurious association**. To distinguish true mediation from confounding, we examine the **residual correlation** after accounting for the intermediate phenotype:

- If the residual association between the SNP and disease is **weak** after conditioning on the intermediate phenotype, this supports **mediation**.
- If the residual association remains **strong**, it suggests either a **direct effect** or the presence of unmeasured confounders.

Mendelian Randomization: Leveraging Genetic Variants for Causal Inference

Mendelian randomization (MR) is a powerful technique that uses genetic variants as **instrumental variables**

to infer causality. It is based on the principle that **genetic variants are randomly assorted at conception**, akin to a natural randomized controlled trial. This allows us to use SNPs as **proxies for exposure** to test for causal relationships between the exposure (e.g., gene expression) and the outcome (e.g., disease).

Key Steps in Mendelian Randomization

1. **Instrumental Variable (IV)**: The SNP serves as an instrument for the exposure. It must satisfy three conditions:
    - The SNP is associated with the exposure (e.g., gene expression).
    - The SNP influences the disease outcome **only through the exposure** (no direct effect).
    - The SNP is not associated with any confounder of the exposure-outcome relationship.
2. **Causal Estimate**: The causal effect of the exposure on the outcome is estimated by **dividing the SNP-disease association** by the **SNP-exposure association**. This method effectively filters out confounding, leveraging the random assignment of alleles.

Example. Consider a scenario where we want to test if **lipid levels** (the exposure) mediate the effect of a SNP on cardiovascular disease (the outcome):

- We first establish that the SNP is associated with **lipid levels** (e.g., LDL cholesterol).
- We verify that the SNP is also associated with **cardiovascular disease**.
- Using Mendelian randomization, we estimate whether the effect of the SNP on cardiovascular disease is **mediated by changes in lipid levels**. If the causal effect remains significant after accounting for lipid levels, this suggests a direct impact of the SNP on the disease.

Interpreting Results: Causal Pathways vs. Confounding

By combining **mediation analysis** and **Mendelian randomization**, we can distinguish between three scenarios:

1. **Mediated Causal Effect**: The SNP affects the disease primarily through an intermediate phenotype (e.g., gene expression, lipid levels).
2. **Independent Causal Effect**: The SNP directly affects the disease without mediation.
3. **Confounded Association**: The observed effect is due to unmeasured confounding variables influencing both the intermediate phenotype and the disease.

This integrated approach provides a robust framework for identifying **causal relationships**, moving beyond mere associations to uncover **mechanistic insights**.

Summary

- **Mediation Analysis** helps determine whether the effect of a SNP on disease is **indirect**, acting through an intermediate phenotype.
- **Mendelian Randomization** uses genetic variants as natural instruments to test for **causal relationships**, minimizing confounding bias.
- Together, these methods provide a powerful toolkit for dissecting the **complex pathways** linking genetic variation to disease, enabling us to identify potential **targets for therapeutic intervention**.

This comprehensive approach is crucial for advancing our understanding of the molecular mechanisms underlying complex traits and diseases, paving the way for **precision medicine** based on causal insights.

1:19:50 Heritability and Systems Genetics

Heritability is a key concept in genetics, capturing the proportion of the **total phenotypic variation** in a population that can be attributed to **genetic differences**. It provides a measure of how much **genetic variation** influences the observed differences in a trait among individuals.

Defining Heritability

Heritability (h^2) is the fraction of the **phenotypic variance** that can be explained by **genetic variance**. Formally, it is expressed as:

h_2=Genetic Variance/Total Phenotypic Variance

- If a trait has **70% heritability**, this means that **70%** of the variability in that trait across individuals can be explained by genetic factors, while the remaining **30%** is due to **environmental factors** and measurement noise.

- Heritability is a **population-level statistic**. It does not imply that 70% of an individual's trait value is determined by genetics but rather that 70% of the variability **across the population** is due to genetic differences.

Estimating Heritability

Heritability can be estimated using different methods, primarily based on **familial relationships** and genetic similarities:

1. **Twin Studies**:
    - **Identical twins (monozygotic twins)** share **100% of their genetic material**. Thus, the correlation in their phenotypic traits is expected to be very high if the trait is highly heritable.
    - **Fraternal twins (dizygotic twins)** share **50% of their genetic material** on average. Comparing the phenotypic correlations between identical and fraternal twins allows us to estimate the **heritability** of a trait.
2. h2=2×(Correlation of Identical Twins−Correlation of Fraternal Twins)h^2 = 2 \times (\text{Correlation of Identical Twins} - \text{Correlation of Fraternal Twins})h2=2×(Correlation of Identical Twins−Correlation of Fraternal Twins)
3. **Family and Pedigree Studies**:
    - By examining **siblings**, **cousins**, and more distant relatives, we can extend this analysis:
        - **Full siblings** share, on average, **50%** of their genetic makeup.
        - **First cousins** share **25%** of their genetic makeup.
        - **Second cousins** share **12.5%** of their genetic makeup.
    - The phenotypic similarity is expected to decrease as the genetic relatedness decreases. This trend can be used to estimate heritability.
4. **Genome-Wide Complex Trait Analysis (GCTA)**:
    - In modern genetics, heritability can also be estimated from **unrelated individuals** using their **genomic data**. This method examines the **genetic similarity** across the entire genome and correlates it with **phenotypic similarity**.
    - The approach uses **single nucleotide polymorphisms (SNPs)** across many individuals to estimate the fraction of phenotypic variance explained by the combined effects of all measured SNPs.

Narrow-Sense vs. Broad-Sense Heritability

- **Narrow-sense heritability (h^2)** considers only the additive genetic variance (the effects of individual alleles summed up).
- **Broad-sense heritability (H^2)** includes not only additive genetic effects but also **dominance** and **gene-gene interactions** (epistasis).

For most complex traits, we focus on **narrow-sense heritability** because it is more straightforward to estimate using population-based data.

Heritability in Context: The Role of Environment

It is crucial to remember that heritability is not a fixed property of a trait; it can vary depending on the **population** and **environment**:

- In a **homogeneous environment**, the phenotypic differences are more likely due to genetic variation, leading to **higher heritability** estimates.
- In a **heterogeneous environment** with many external factors influencing the trait, the **environmental variance** increases, reducing the estimated heritability.

For example, the heritability of **height** is very high (~80-90%) in populations with good nutrition, but it can be lower in populations where malnutrition or other environmental factors play a significant role.

Partitioning Heritability by Genetic Loci

To understand the genetic architecture of complex traits, we can **partition heritability** into components associated with different genetic loci:

- **Polygenic Traits**: Traits influenced by many genetic loci, each contributing a small effect. For example,

height and intelligence are highly **polygenic**, with contributions from thousands of SNPs.

- **Oligogenic Traits**: Traits influenced by a few loci with larger effects.

By conducting **genome-wide association studies (GWAS)** and examining the **heritability explained by individual loci**, we can gain insights into the genetic architecture of the trait. Typically, GWAS studies capture only a fraction of the total heritability, a phenomenon known as the **"missing heritability" problem**.

Systems Genetics: Integrating Multi-Omics Data

Systems genetics aims to bridge the gap between genetic variation and phenotypic traits by integrating multiple layers of biological data, including:

- **Genomic Data**: DNA sequence variations (e.g., SNPs).
- **Transcriptomic Data**: Gene expression profiles (eQTL analysis).
- **Epigenomic Data**: DNA methylation and chromatin accessibility (meQTL analysis).
- **Proteomic Data**: Protein expression levels and interactions.

By using **multi-omics approaches**, we can trace the flow of information from **genetic variants** to changes in gene expression and epigenetic modifications, ultimately influencing the observed phenotype.

Summary

- **Heritability** quantifies the genetic contribution to phenotypic variation within a population.
- Estimation methods include **twin studies**, **family studies**, and modern approaches using **genome-wide data**.
- Heritability varies with the **environment** and **population** context and can be partitioned into **narrow-sense** and **broad-sense** components.
- The concept of **systems genetics** provides a framework for integrating genetic, transcriptomic, and epigenomic data to unravel the complex pathways linking **genetic variation** to **disease phenotypes**.

This comprehensive approach is crucial for advancing our understanding of the **molecular mechanisms underlying complex traits**, paving the way for **precision medicine** and the development of targeted interventions.

1:21:20 Partitioning heritability by functional annotations

The next step in understanding **heritability** is not only to measure how much of the phenotypic variance can be attributed to genetic factors, but to **partition** this heritability based on specific **functional annotations** of the genome. This approach helps us identify **which genomic regions** or **functional elements** contribute most to the genetic basis of complex traits.

Heritability Decomposition: Analyzing Genetic Relatedness

To estimate heritability, we often look at the **genetic relatedness** between pairs of individuals and correlate it with their **phenotypic similarity**:

- **Genetic Relatedness**: For any pair of individuals, we can quantify how much of their genome they share. This measure is typically based on **SNP (single nucleotide polymorphism) similarity**.
- **Phenotypic Correlation**: We then assess how similar these individuals are in terms of the **trait of interest** (e.g., height, schizophrenia risk).

By regressing **phenotypic similarity** against **genetic similarity**, we obtain an estimate of the **total heritability** of the trait. However, this estimate includes contributions from **all parts of the genome**, without distinguishing which regions are most relevant.

Partitioning Heritability by Genomic Regions

Instead of considering the entire genome, we can **partition the heritability** by focusing on **specific subsets of the genome**, such as:

1. **Individual Chromosomes**:
    - We can estimate heritability separately for each chromosome. This allows us to identify chromosomes that might have a **disproportionate contribution** to the trait.
    - For example, if chromosome 21 explains a larger portion of the heritability for a given trait, it may contain regions with strong genetic effects.
2. **Functional Genomic Annotations**:

    - We can refine this further by focusing on specific **functional elements** like **enhancers**, **promoters**, and **coding regions**.
    - For instance, we could estimate heritability using only SNPs located within **enhancers** that are active in certain cell types, such as **lipid-regulating enhancers** or **brain-specific enhancers**.

This approach allows us to ask more nuanced questions about where the genetic signals contributing to a trait are localized.

Partitioning Heritability by Tissue-Specific Enhancers

By focusing on **tissue-specific regulatory elements**, we can gain insights into the **biological pathways** underlying a trait. For example:

- In **schizophrenia**, most of the heritability can be attributed to SNPs located within enhancers active in the **central nervous system (CNS)**.
- In contrast, SNPs located in enhancers active in **skeletal muscle** explain little to none of the heritability for schizophrenia, indicating that the trait is primarily driven by **neurological factors** rather than muscular ones.

This type of analysis helps to **validate the biological relevance** of specific tissues or cell types in relation to a trait. It provides a **systematic framework** for identifying which **biological systems** are involved in the genetic basis of complex diseases.

The Power of Partitioning Heritability

Partitioning heritability has become an **extraordinarily powerful tool** in human genetics because it enables us to:

- **Attribute Genetic Influence**: We can determine which genomic regions or functional annotations contribute most to the trait.
- **Identify Relevant Biological Pathways**: By examining tissue-specific regulatory elements, we can uncover the **tissue context** of genetic associations, helping us pinpoint the **biological processes** that are disrupted in disease.
- **Guide Therapeutic Targeting**: This information can direct researchers to the **most relevant cell types and pathways**, aiding in the development of **precision therapies**.

For example, if we find that the heritability of **type 2 diabetes** is enriched in enhancers active in **liver cells**, it suggests that genetic risk factors are likely affecting **liver metabolism**. Conversely, if we find heritability for **cardiovascular disease** enriched in **endothelial cell enhancers**, this points to the role of **vascular biology** in the genetic risk for the disease.

Summary

- Partitioning heritability by functional annotations helps us break down the **total genetic influence** into contributions from **specific genomic regions** or **functional categories**.
- This approach provides insights into the **biological mechanisms** underlying complex traits and diseases, highlighting the **tissues and pathways** most implicated by genetic variation.
- It is a key method for moving from **genome-wide signals** to a **more detailed understanding** of the underlying genetic architecture, guiding both **basic research** and **clinical applications**.

In conclusion, partitioning heritability allows us to **go beyond simple heritability estimates** and dissect the **genomic architecture** of complex traits, paving the way for a deeper understanding of human genetics and the **biological basis of disease**.

1:23:15 Omnigenic vs. polygenic vs. monogenic inheritance

In the landscape of genetic architecture, we can categorize traits based on the **number and nature of loci** that contribute to their variation. These categories span from **monogenic**, through **polygenic**, to the more recent concept of **omnigenic** inheritance. Understanding these distinctions is critical for appreciating the complexity of genetic influences on various traits.

1. Monogenic Traits

Monogenic traits are characterized by a **single genetic locus** that has a **major effect** on the phenotype. These are often **Mendelian diseases**, where mutations in a single gene are sufficient to cause the trait:

- **Examples**: Cystic fibrosis (caused by mutations in the *CFTR* gene), Huntington's disease, and sickle cell anemia.

- **Mechanism**: The phenotype can be directly attributed to a specific variant, and the inheritance pattern follows **Mendelian principles** (dominant, recessive, etc.).
- **Clinical Relevance**: Monogenic traits are relatively straightforward to study and diagnose because of their clear genetic basis.

While monogenic disorders provide clear examples of genetic causality, they are the exception rather than the rule in complex traits and common diseases.

2. Polygenic Traits

Polygenic traits are influenced by **many genetic variants**, each contributing a **small effect** to the overall phenotype:

- **Examples**: Height, schizophrenia, type 2 diabetes, and obesity.
- **Mechanism**: The trait arises from the cumulative effects of **hundreds to thousands of loci**, each with a modest influence. The risk is distributed across many variants, each with a small additive effect.
- **Genetic Architecture**: Polygenic traits typically show a **normal distribution** in the population, resulting from the combined effect of many variants.
- **Clinical Implications**: The polygenic nature of these traits complicates their study, as no single variant has a decisive impact. Instead, **polygenic risk scores** are used to aggregate the effects of many SNPs to predict an individual's risk.

Polygenic inheritance reflects the complexity of biological systems, where many small perturbations across the genome collectively influence the phenotype.

3. Omnigenic Traits

The concept of **omnigenic inheritance** takes the idea of polygenicity even further. It suggests that for some traits, **nearly every gene** can influence the phenotype, either directly or indirectly:

- **Definition**: An **omnigenic trait** is influenced not only by many loci with small effects but also by the **entire network of genes and biological pathways**. In other words, **nearly every gene** can potentially impact the trait through its connections to core regulatory pathways.
- **Examples**: Traits like **coronary artery disease (CAD)**, **lifespan**, and **healthspan** may exhibit omnigenicity. For these traits, genetic associations are not confined to a few core pathways but are seen across a wide range of biological processes.
- **Mechanism**: Omnigenic traits arise because of the **interconnectedness** of biological systems. While core genes may have a direct and strong influence, **peripheral genes** can also contribute through their indirect effects on core pathways. This reflects the **complex network structure** of gene regulation, where peripheral genes can modulate the activity of key regulatory nodes.
- **Implications for Genetic Studies**: As we increase the power of genome-wide studies (e.g., by increasing sample size), we tend to find more and more loci associated with a trait. This suggests that **many biological processes** might influence the trait, leading us to the idea of **omnigenicity**.

The Shift from Polygenicity to Omnigenicity

The shift from thinking about traits as purely **polygenic** to considering them as **omnigenic** represents an evolution in our understanding of **genetic architecture**:

- In a **polygenic model**, we assume that a **finite set of variants** contribute to the trait, primarily through their additive effects.
- In an **omnigenic model**, we assume that **nearly every gene** has some potential impact on the trait, either through direct effects on core pathways or through **indirect modulation** of those pathways.

This broader perspective acknowledges the **network-like nature** of biological systems, where **distant genetic variants** can still influence the trait through their effects on the broader regulatory network.

Centrality vs. Diffuseness in Genetic Effects

A key distinction in the omnigenic framework is the difference between:

- **Central Effects**: These are the effects of **core genes** that are directly involved in the primary biological pathway underlying the trait. For example, genes involved in **neuronal function** for schizophrenia or genes regulating **lipid metabolism** for coronary artery disease.
- **Diffuse Effects**: These are the effects of **peripheral genes**, which might influence the trait indirectly

through their interactions with core regulatory genes. This is akin to the **butterfly effect**, where small changes in peripheral genes can have broader, less predictable impacts on the phenotype.

The challenge in genetics is not merely to detect associations but to **distinguish central, causally important effects** from diffuse, background effects that contribute indirectly.

Summary

- **Monogenic Traits**: Governed by a single genetic locus; relatively straightforward Mendelian inheritance.
- **Polygenic Traits**: Influenced by many loci, each contributing a small effect; cumulative genetic risk can be estimated through polygenic risk scores.
- **Omnigenic Traits**: Influenced by almost the entire genome; nearly every gene can impact the trait through its connections to core pathways, reflecting the **interconnectedness** of biological systems.

Conclusion

The concept of **omnigenicity** expands our understanding of complex traits, suggesting that **biological processes are deeply interconnected**. As we increase the power of genetic studies and delve deeper into the genome, we may find that nearly every biological process can influence nearly every trait. The challenge is to move beyond detecting associations and to develop methods that help us **dissect the core pathways** and **causal mechanisms** amidst the vast network of interconnected genetic effects.

This realization underscores the need for **systems-level approaches** in genetics, integrating diverse data types (e.g., gene expression, epigenomics, protein interactions) to build a comprehensive understanding of how genetic variation translates into complex phenotypes.

1:25:25 Summary

We covered a wide array of interconnected topics, building a detailed framework for understanding how **genetic variation influences disease phenotypes** through a series of molecular, cellular, and organismal mechanisms. Let's review the main points discussed:

1. Genetic Variation, GWAS, and Polygenic Scores

- We started with a **review of genetic variation** and genome-wide association studies (**GWAS**, pronounced "G-was"), emphasizing how we identify **risk loci** associated with complex traits.
- The concept of **polygenic scores (PGS)** was introduced, showcasing how aggregating the small effects of many common variants can explain a significant portion of individual risk, often on par with family history.
- We discussed how GWAS provides a **global view of genetic associations**, and how we can use **functional enrichment analysis** across diverse tissues and cell types to gain biological insights into the underlying disease mechanisms.

2. From GWAS Locus to Biological Mechanism

- We dived deeper into the analysis of **individual GWAS loci**, using an example to show how we can integrate multiple lines of evidence:
    - **Linkage disequilibrium** for identifying candidate variants.
    - **Epigenomic annotations** to prioritize functional regions.
    - **Chromatin conformation capture** for linking enhancers to target genes.
- This approach allowed us to go from an associated region to identifying specific **causal variants**, the likely **regulatory elements**, and the **target genes** they influence, thus elucidating the **molecular mechanism**.

3. Quantitative Trait Loci (eQTLs and meQTLs) Analysis

- We discussed **expression quantitative trait loci (eQTLs)** and **methylation quantitative trait loci (meQTLs)** as powerful tools to bridge the gap between genotype and phenotype:
    - eQTLs identify genetic variants that influence gene expression.
    - meQTLs identify genetic variants that impact DNA methylation levels.
- These analyses provide **intermediate phenotypes**, helping us understand how genetic variants influence molecular traits, which in turn affect disease risk.
- The importance of distinguishing between **genetic causes** and **disease consequences** was

emphasized, especially when interpreting correlations between molecular traits and disease phenotypes.

4. Heritability and Systems Genetics

- We defined **heritability** as the proportion of phenotypic variance attributable to genetic variance and discussed methods for estimating it using both **related and unrelated individuals**.
- We then extended this to **partitioning heritability** by functional annotations (e.g., enhancers active in specific tissues), allowing us to identify the **biological pathways** most relevant to a given trait.
- This analysis helps us understand not just **how much** of a trait's variance is explained by genetics, but also **where in the genome** (and in which functional contexts) the relevant genetic signals are concentrated.

5. Omnigenic, Polygenic, and Monogenic Models

- We explored the spectrum of genetic inheritance models:
    - **Monogenic traits** are controlled by single genes (e.g., cystic fibrosis).
    - **Polygenic traits** are influenced by many loci with small effects (e.g., height, schizophrenia).
    - **Omnigenic traits** reflect an emerging view where nearly every gene may have a potential influence through its interactions within a complex regulatory network.
- The concept of **omnigenicity** highlights the pervasive interconnectedness of biological processes, suggesting that **nearly every pathway** could potentially affect the trait, albeit with varying degrees of centrality and specificity.

6. Building a Systematic Framework for Genetic Studies

- We concluded with a call to **build systematic methods** for studying genetic effects, emphasizing the integration of **GWAS**, **QTL mapping**, and **functional genomics**.
- The goal is to create a **comprehensive pipeline** that can handle multiple lines of evidence and elucidate the complex pathways from genetic variation to disease phenotypes.