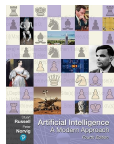# Machine Learning Techniques

Instructor: Dr. GP Saggese - gsaggese@umd.edu
References: - Burkov: *"The Hundred-Page Machine Learning Book"* (2019) - Russell et al.: *"Artificial Intelligence: A Modern Approach"* (4th ed, 2020) - Hastie et al.: *"The Elements of Statistical Learning"* (2nd ed, 2009)

SCIENCE ACADEMY

- ***Machine Learning Paradigms***
  - Major Paradigms
  - Some Examples
- Machine Learning Techniques

- Machine Learning Paradigms
  - *Major Paradigms*
  - Some Examples
- Machine Learning Techniques

Machine Learning Paradigms with Examples (1/3)

- **Supervised Learning**
  - Learn from labeled data to predict labels for new inputs
  - E.g., image classification using ResNet on ImageNet
- **Unsupervised Learning**
  - Discover hidden patterns or structure in unlabeled data
  - E.g., K-means clustering for customer segmentation
- **Reinforcement Learning**
  - Learn through interaction with an environment, receiving rewards/punishments
  - E.g., deep Q-Learning for playing Atari games
- **Self-Supervised Learning**
  - Generate pseudo-labels from unlabeled data to pre-train models
  - E.g., BERT (Masked Language Modeling)
- **Semi-Supervised Learning**
  - Combine small labeled data with large unlabeled data to improve performance
  - E.g., named entity recognition (NER) using annotated sentences with entity tags combined with many raw text documents

SCIENCE
ACADEMY

Machine Learning Paradigms with Examples (2/3)

- **Online Learning**
  - Learn incrementally from a stream of data in real time
  - E.g., online logistic regression for click-through rate prediction
- **Multi-Task Learning**
  - Train simultaneously a model to perform multiple related tasks
  - E.g., learn sentiment analysis and question answering
- **Meta-Learning**
  - "Learning to learn": adapt quickly to new tasks using prior experience
  - E.g., a model can be fine-tuned quickly on a new task using just a few gradient steps
- **Zero-Shot / Few-Shot Learning**
  - Generalize to new tasks with no or few labeled examples
  - E.g., GPT-4 solving tasks with zero-shot prompting
- **Active Learning**
  - The model selects the most informative samples to be labeled by an oracle (e.g., a human)
  - E.g., pick samples where the model is least confident to get more examples

SCIENCE
ACADEMY

Machine Learning Paradigms with Examples (3/3)

- **Federated Learning**
  - Train models across decentralized devices without sharing raw data
  - E.g., fraud detection or credit scoring across banks
- **Evolutionary Learning**
  - Optimize model structures or parameters using evolutionary algorithms inspired by natural selection and genetics
  - Gradient free, global search, variable length inputs
  - E.g., genetic algorithms
- **Curriculum Learning**
  - Train models on easier tasks first, gradually increasing difficulty
  - E.g., training in robotic control simulations
- **Multi-Agent Learning**
  - Multiple agents learn and interact in shared environments, often in game-theoretic settings (e.g., competition, collaboration)
  - E.g., AlphaStar to play StarCraft II

SCIENCE
ACADEMY

- Machine Learning Paradigms
    - Major Paradigms
    - *Some Examples*
- Machine Learning Techniques

Supervised Learning

- Learn a function $f : X \rightarrow Y$ that maps inputs to correct outputs
  - Training examples $(\underline{x}, y)$ with pairs of inputs and correct outputs
  - Requires labeled data for training
  - Measure performance with error on a separate test set
- **Classification**: output is a discrete label, e.g.,
  - `Spam` vs `Not Spam`
  - Digit recognition 0, 1, . . .
  - Sentiment analysis `Pos`, `Neg`, `Neutral`
- **Regression**: output is a continuous value, e.g.,
  - House prices given features like size and location
  - House demand
  - Stock prices
- **Common algorithms**:
  - Linear Regression
  - Decision Trees
 - K-nearest neighbors
  - Neural Networks
  - . . .

SCIENCE
ACADEMY

Unsupervised Learning

- Learn from data **without** labeled outputs
  - Goal: discover patterns or structure in the data
  - No explicit feedback signal
  - Evaluation can be qualitative
- **Main techniques**:
  - **Clustering**: Group similar examples, e.g.,
    - Customer segmentation
    - Grouping news articles by topic without knowing the topics
  - **Dimensionality Reduction**: Reduce number of variables with PCA while preserving structure
    - E.g., visualize high-dimensional data in 2D
  - **Density Estimation**: Estimate probability distribution of data
    - E.g., anomaly detection in server logs
  - **Association Rule Learning**: Discover interesting relations between variables
    - E.g., market basket analysis (e.g., "people who buy X also buy Y")
- **Common algorithms:**
  - K-means
  - PCA
  - Autoencoders

SCIENCE
ACADEMY

Reinforcement Learning

- Learn by **interacting with an environment** to **maximize cumulative reward**
    - Learn policy $\pi(s) \to a$ that maximizes expected reward
    - Trade-off between exploration (trying new actions) and exploitation (using known good actions)
    - Environments provide clear rules and feedback (win/loss/reward)
    - Often involve physical simulation or real-world interaction
- **Core elements:**
    - Agent: Learner and decision maker
    - Environment: Everything the agent interacts with
    - State s
    - Action a
    - Reward r
- **Algorithms:**
    - Q-learning
    - Policy Gradient methods
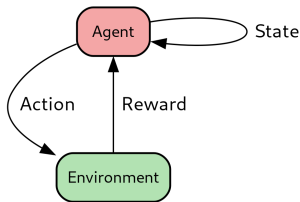
Reinforcement Learning: Examples

- In **game playing**, learn strategies through trial and error
  - E.g., AlphaGo mastering the game of Go
- In **robotics**, learn control policies for movement and manipulation
- In **autonomous driving**, learn safe and efficient driving behaviors
- In **resource management**, optimize allocation of limited resources over time
  - E.g., data center cooling or CPU job scheduling
- In **personalized recommendations**, adapt suggestions based on user interaction
  - E.g., newsfeed ranking adjusting based on user clicks
- In **healthcare**, optimize treatment plans over time

- Machine Learning Paradigms
- ***Machine Learning Techniques***
  - Machine Learning in Practice
  - How to Do Research
  - Pipeline Organization
  - Input Processing
  - Learning Algorithms
  - Performance Metrics
  - Model Selection
  - Aggregation

- Machine Learning Paradigms
- Machine Learning Techniques
  - ***Machine Learning in Practice***
  - How to Do Research
  - Pipeline Organization
  - Input Processing
  - Learning Algorithms
 - Performance Metrics
  - Model Selection
  - Aggregation

SCIENCE
ACADEMY

Machine Learning Flow (1/2)

- **Question**
  - E.g., "How can we predict house prices?"
- **Input data**
  - E.g., historical data of house sales
- **Features**
  - E.g., number of bedrooms, location, square footage
- **Algorithm**
  - E.g., linear regression, decision trees
- **Parameters**
  - E.g., learning rate, number of trees in a random forest
- **Evaluation**
  - E.g., accuracy, precision, recall

SCIENCE
ACADEMY

Machine Learning Flow (2/2)

Question → Input data → Features → Algorithm → Parameters → Evaluation
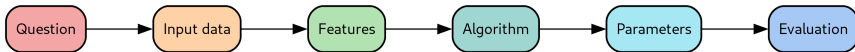
- **Not all phases are equally important!**
  - Question > Data > Features > Algorithm
- Clarity of the **question** impacts project success
- Quality and relevance of **data** are crucial for performance
- Proper **feature** selection simplifies the model and improves accuracy
- **Algorithm** is often less important (contrary to popular belief!)

SCIENCE
ACADEMY

Question

- **Make the question concrete and precise**
  - Define the problem clearly
  - Specify inputs and expected outputs
  - Align question with business or research objectives
  - E.g.,:
    - **Bad**: *"How can we improve sales?"*
    - **Good**: *"What factors most significantly impact sales of product X in region Y during season Z?"*
- Formulating question is **the most important part** of the machine learning problem
  - Misunderstanding leads to:
    - Solving the wrong problem
    - Collecting wrong data
    - . . .
- *"If I were given one hour to save the planet, I would spend 59 minutes defining the problem and one minute resolving it"* (Einstein)

Input Data

- Ensure **data is specific to prediction** goal
    - E.g., use known movie ratings to predict unseen movie ratings from the same population
    - Training set $\approx$ test set
- Relationship between data and prediction goal is **not always direct**
    - E.g., interested in prices but predict supply and demand instead
- Poor-quality data leads to inaccurate predictions
    - *"Garbage in - garbage out"*
- Recognize **when data is insufficient** for valid answers
    - *"Combination of data and desire for an answer does not ensure a reasonable answer can be extracted"* (John Tukey)
- **More data vs better models**
    - Meta-studies show difference between generic and best model is like 5%
    - *"It's not who has the best algorithm that wins. It's who has the most data"* (Google researcher)
    - *"Every time I fire a linguist, the performance of the speech recognizer goes up"* (IBM researcher in speech recognition)

SCIENCE
ACADEMY

Features

- **Features** provide high-level information about inputs
  - E.g., use intensity and symmetry for scanned numbers instead of raw bit maps
- **Characteristics of good features**:
  1. Perform data compression
  2. Retain relevant information
  3. Often created with expert knowledge
- **Common mistakes in feature building**:
  1. Automating feature selection may lead to overfitting
     - Black box predictions can be accurate but stop working anytime
     - E.g., Google Flu's unclear feature-model link
  2. Ignoring data-specific quirks
     - E.g., mislabeling outliers
  3. Unnecessarily discarding information

Models

- Best models are:
  - **Interpretable**
    - Allow users to understand and trust the model's decisions
    - E.g., decision trees are appropriate in medical studies since they produce a "reasoning"
  - **Simple**
    - Easier to implement and maintain
    - Reduces the risk of overfitting
  - **Accurate**
    - Often accuracy is traded off for remaining characteristics
    - E.g., accuracy vs interpretability, accuracy vs speed
  - **Fast**
    - Reduce computational costs to train and test
    - E.g., real-time applications
  - **Scalable**
    - Can handle large datasets efficiently (growing data and user bases)
    - E.g., in the Netflix prize, Netflix didn't end up implementing the best algorithm since it wasn't scalable enough

SCIENCE
ACADEMY

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - ***How to Do Research***
  - Pipeline Organization
  - Input Processing
  - Learning Algorithms
  - Performance Metrics
  - Model Selection
  - Aggregation

SCIENCE
ACADEMY

Occam's Razor

- **Simple is better**
  - *"The simplest model that fits the data is also the most plausible"* (Occam)
  - *"An explanation of the data should be as simple as possible, but not simpler"* (Einstein?)
- Corollary:
  - Trim the model to the bare minimum necessary to explain the data
- What does **simple** mean?
  - A model is simple when it is one of few possible objects
- What does **better** mean?
  - A model is better means better out of sample performance
- **Why it's true**?
  - Less likely to fit a given data by coincidence
  - An event is more significant if it is unlikely (formalized in terms entropy)

What is a "simple model"?

- An object is **simple** when it is one of few possible objects

- **Examples**:
    - Polynomial of order 2 is simpler than order 17
        - More polynomials of order 17 than order 2, though both are infinite sets
    - SVM (Support Vector Machine) characteristics:
        - Separating hyperplane appears wiggly, but defined by few support vectors

- **Measures of complexity**
    - Complexity of a single hypothesis h
        - E.g., polynomial order, MDL (describe hypothesis in bits), Kolmogorov complexity
    - Complexity of a hypothesis set $\mathcal{H}$
        - E.g., VC dimension of the model
    - Complexity of h and $\mathcal{H}$ related by counting:
        - If you need l bits to specify h, then h is one of 2^l elements of set $\mathcal{H}$

SCIENCE
ACADEMY

Model Soundness

- **Model should tell a story**
  - You cannot blindly accept results of modeling
  - Ask: *"What criticisms would you give if the model was presented for the first time?"*
- **Benchmark models**: compute performance if model outputs:
  - Always 0 or 1
    - E.g., long-only model for stock predictions
  - Random results
    - E.g., bootstrap of null hypothesis "no prediction power"
- **Perfect fit can mean nothing**, e.g.,
  - Get 2 data points on a plane
    - Fit data with a linear relationship
    - Perfect fit
  - Means nothing since:
    - Always a line between 2 points
    - Data cannot falsify hypothesis
  - The model (line) is too complex for data set (only 2 points)

SCIENCE
ACADEMY

Sampling Bias (1/2)

- **Sampling bias** occurs when data is not representative of the intended population
    - Biased sampling leads to biased outcomes
    - Model views world through training data
    - Hoeffding's hypothesis: training and testing distributions are the same
- **Causes of Sampling Bias**
    - *Non-random sampling*: Favors certain outcomes or groups
    - *Undercoverage*: Inadequate representation of some population members
    - *Survivorship bias*: Includes only successful subjects
    - *Self-selection bias*: Participants choose to participate, introducing bias
- **Consequences**
    - Systematic errors distort relationships or predictions
    - Models trained on biased samples perform poorly on real-world data
    - Leads to incorrect or unfair decisions

SCIENCE
ACADEMY

Sampling Bias (2/2)

- **Examples**
  - Training facial recognition on lighter-skinned faces reduces accuracy on darker-skinned faces
  - Predicting income using data only from employed individuals omits unemployed people
- **Mitigation**
  - Compare sample statistics with population statistics
  - Use stratified sampling or resampling to balance data
  - Apply re-weighting or bias correction techniques
  - If data points have zero probability ($Pr = 0$), no remedies possible

SCIENCE
ACADEMY

Data Snooping (1/2)

- **Data snooping** occurs when test set information improperly influences model-building

- **Typical Sources**
  - *Train-test contamination*: Test data leaks into training
    - E.g., Using test data during feature engineering, model selection, or tuning
  - *Multiple hypothesis testing*: Trying many models inflates performance

- **Consequences**
  - Overly optimistic evaluations
  - Models perform well in validation but fail on unseen data

- **Why It's Dangerous**
  - Misleading results
  - False confidence in model quality
  - Common pitfall in machine learning workflows

Data Snooping (2/2)

- **Examples**
  - Choosing features based on the full dataset before splitting gives foresight not present in deployment
  - E.g., demean the entire data and then split in train / test
- **Preventive Strategies**
  - Keep training, validation, and test sets separate
  - Use cross-validation properly
  - Fit preprocessing steps only on training data, then apply to test data

SCIENCE
ACADEMY

"Burning the Test Set"

- **Problem**
  - Repeated test set use during development leaks into training
  - Overfitting to test data
  - *"If you torture the data long enough, it will confess whatever you want"* (Coase, 1982)
- **Symptoms**
  - Test accuracy improves; real-world performance stagnates or degrades
  - Model evaluation becomes untrustworthy
- **Solutions**
  - *One-time use principle*: Evaluate on test set once, post-tuning
  - *Use a validation set*: For model development and tuning
  - *Statistical adjustments*:
    - Adjust p-values for multiple testing
    - Use Bonferroni or False Discovery Rate corrections
  - *Theoretical tools*:
    - VC dimension reflects learning capacity, including repeated trials
    - Minimum Description Length (MDL) penalizes complex models and repeated fitting

How to Achieve Out-Of-Sample Fit

- **Goal**: Choose an hypothesis $g \in \mathcal{H}$ approximates the unknown target function f
    - What does $g \approx f$ mean?

$$g \approx f \iff E_{out}(g) \approx 0$$

- **Solution**:
    - Achieve:
        - Good in-sample performance $E_{in}(g) \approx 0$
        - Good generalization $E_{out}(g) \approx E_{in}(g)$
    - Together $\implies$ good out-of-sample performance $E_{out}(g) \approx 0$

SCIENCE
ACADEMY

What If Out-Of-Sample Fit Is Poor?

- The model performs well in sample ($E_{in} \approx 0$) but poorly out of sample ($E_{out} \gg E_{in}$)

- **What does it mean?**
  - In-sample performance is optimistic
  - Model is overfitted and fails to generalize

- **What do you do?**
  - Run diagnostics before long-term projects
  - Gain insight on what works/doesn't to improve performance
    - E.g., bias-variance curves and learning curves

- **How to fix?**
  1. Training data
     - Get more training data $\iff$ fixes high variance
  2. Features
     - Remove features $\iff$ fixes high variance
     - Add features $\iff$ fixes high bias
     - Add derived features (e.g., polynomial features) $\iff$ fixes high bias
  3. Regularization
     - Decrease regularization λ $\iff$ fixes high bias
     - Increase regularization λ $\iff$ fixes high variance

Why Using a Lot of Data?

- **Several studies** show that:
  - Different algorithms/models have remarkably similar performance
  - Increasing training set improves performance
- **Consequence**
  - High capacity model + massive training set = good performance
- **Why**
  - Using a high capacity model with many parameters (e.g., neural network)

$$E_{in} \approx 0$$

    due to low bias (and high variance)
  - A massive data set helps avoid overfitting

$$E_{out} \approx E_{in}$$

  - These two conditions together

$$E_{out} \approx E_{in} \approx 0 \implies E_{out} \approx 0$$

SCIENCE
ACADEMY

What to Do When You Have Lots of Data?

- You have $m = 100M$ examples in data set: great!

- **Cons**
    - Scalability issue (slow, lots of compute)
    - Require work on infrastructure

- **First step**:
    - Plot learning curves (in-sample and out-of-sample performance) for increasing amount of data $m = 1k, 10k, 100k, 1M, ...$

- **Next step**:
    - If model has large bias
        - Training and validation performance are similar at $1M$
        - Next step: use more complex model rather than training on $100M$ instances
    - If model has large variance
        - Next step: use all $100M$ instances to train the model
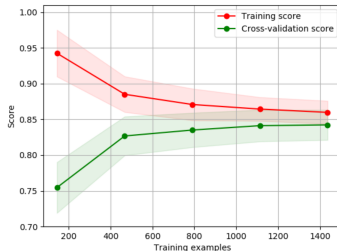
Why We Do Things?

- **Always, always, always:**
  - Understand the purpose of the task
    - Ask: *"Why are we doing something?"*
  - Clarify goals and outcomes of the task
    - Ask: *"What do we hope to determine by performing the task?"*
  - Think about actions with the bigger picture in mind
  - Avoid going through the motions
  - Prioritize tasks by importance and impact
- E.g., when conducting a customer survey:
  - Q: *"Why is the feedback being collected?"*
    - A: To improve product features and customer service
  - Q: *"What is the desired outcome?"*
    - A: To identify areas for improvement and innovation
- E.g., before starting a marketing campaign:
  - Q: *"Why is this campaign run?"*
    - A: To increase brand awareness or drive sales
  - Q: *"What are the specific goals?"*
    - A: Set target number of new leads or click-through rates

SCIENCE
ACADEMY

Summary of the Results, Next Steps

- **Always have a summary of results**
  - High-level map of discoveries
    - E.g., "smoothing model coefficients helps"
  - Highlight major findings
  - Interpret results
    - E.g., *"Increase in sales likely due to new marketing strategy."*
  - Conclusions
    - Summarize data suggestions or confirmations
    - E.g., *"Hypothesis that user engagement increases retention is supported"*
- **Always have a reference to detailed results**
  - Provide quick insights before details
- **Always have next steps**
  - What do you expect to happen?
    - Like thinking n moves ahead in chess
  - E.g., *"Next, conduct detailed analysis on demographics contributing most to sales growth"*
    - Outline potential experiments or analyses to validate findings further

SCIENCE
ACADEMY

Example: Spam Filter Classification

- You use $N = 5$ words in an email to distinguish spam / non-spam emails using logistic regression
  - Words are `buy`, `now`, `deal`, `discount`, `<your name>`
- **Goal**: improve classifier performance
  1. **Collect more data**
     - Honeypot project: fake email account to collect spam
  2. **Use better features**
     - Email routing info (spammers use unusual accounts, mask emails as legitimate)
  3. **Use better features from message body**
  4. **Detect intentional misspellings**
     - Spammers use misspelled words (e.g., `w4tch` for `watch`)
     - Use stemming software

SCIENCE
ACADEMY

Right and Wrong Approach to Research

- It is not clear how to prioritize the different possible tasks

- **Bad**

  1. Use gut feeling to choose a task
  2. Complete task
  3. Re-evaluate performance

- **Good**

  1. Build a simple algorithm
     - Within 1 day
  2. Set up performance evaluation
     - Single number and bounds to improve
     - Use cross-validation
  3. Set up diagnostic tools
     - Compute learning and bias-variance curves
     - Understand issues before fixing (avoid premature optimization)
  4. Review misclassified emails manually
     - Identify features to improve performance
     - E.g., what are types of misclassified emails?

- Sometimes you just need to try approaches to see if they work
  - E.g., use stemming software to equate words
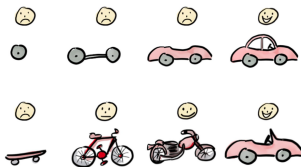
Incremental vs Iterative

- **Incremental Development**
  - Each increment adds functional components
  - Require upfront planning to divide features meaningfully
  - Integration of increments can be complex
- **Iterative Development**
  - Each increment delivers usable system
  - Refine and improve product through repeated cycles
  - Get feedback
  - Uncover and adjust for unknown requirements
- **Incremental ≫ Iterative**

_Incremental

Iterative

Incremental vs Iterative

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - How to Do Research
  - ***Pipeline Organization***
  - Input Processing
  - Learning Algorithms
  - Performance Metrics
  - Model Selection
  - Aggregation

How Are Machine Learning Systems Organized?

- Machine learning systems organized in a **pipeline**
    1. **Break down** problem into sub-problems
    2. **Solve** problems one at a time
    3. **Combine** solutions to solve initial problem
- Performance *p* of a pipeline:

$$p_{system} = \sum_i p_i \cdot \alpha_i$$

where:
  - p_i: Performance of each stage
  - α_i: Importance of each stage

SCIENCE
ACADEMY

Example of Photo OCR System

- **Goal**: build systems to read text in a picture
  - OCR = "Optical Character Recognition"
- **Stages of ML pipeline for OCR:**
  - Text detection: find text areas in the picture
  - Character segmentation: split text into letter boxes
    - E.g., h e l l o
  - Character classification: classify characters individually
  - Spelling correction: fix text errors using context
    - E.g., hell0 corrected to hello
- **Sliding window approach**
  - **Problem**: Unknown text location and size
  - Text detection
    - Train classifier to recognize letters vs non-letters
    - Scan image in two directions with different sizes for text
    - Evaluate classifier cheaper than training
    - Create text likelihood map (e.g., heatmap) using classifier probabilities
    - Enclose text areas in boxes
    - Discard boxes not fitting aspect ratio (valid text width > height)
  - Character segmentation
    - Use sliding window classifiers to find "breaks" between characters

SCIENCE
ACADEMY

Getting More Data

- The ideal recipe for ML is: "low-bias algorithm + train on massive amount of data"

- Use learning curves to make sure we are taking advantage of more data

- The **problem** is getting large amount of data

- Always ask yourself: *"how much work is to get 10x more data than we currently have?"*

- Often it is not that difficult:
    1. Artificial data
        - E.g., synthesize or amplify data set
    2. Collect and label by hand
        - E.g., crowd sourcing like Amazon Mechanical Turk

OCR Pipeline: Example of Artificial Data Synthesis

- How can we increase data set size?
    1. Synthesize data set
        - Use font libraries to generate large training sets
        - Paste characters against random backgrounds
        - Apply scaling, distortion, adding noise, etc
    2. Amplify a data set
        - Add examples by warping/distorting existing examples
- Transformations and noise should be specific to the application domain
    - E.g., Gaussian noise is not always appropriate

SCIENCE
ACADEMY

Ceiling Analysis for ML Pipeline

- The most valuable resource is time
  - You work on optimization for months
  - Find out that the optimization doesn't make much difference

- **Problem**: On which part of the pipeline should time/resource be spent?

- **Solution**: Ceiling analysis to Analyze pipeline performance
  - Single number for system performance
    - E.g., accuracy for OCR
  - For each component:
    - Mock component with "oracle" (always give correct output)
    - Leave others untouched
    - Compute pipeline performance
  - Identify critical components by estimating performance improvement with 10% component improvement
  - Measure, don't guess!

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - How to Do Research
  - Pipeline Organization
  - ***Input Processing***
  - Learning Algorithms
  - Performance Metrics
  - Model Selection
  - Aggregation

SCIENCE
ACADEMY

Data Processing Transformations

- **Purpose of data processing**
  - Prepare raw data for effective machine learning
  - Improve model performance and generalization

- **Data cleanup**
  - Apply filters or smoothing to remove irrelevant variations

- **Handling missing data**

- **Types of transformations**
  - Normalization and standardization
  - Encoding categorical data
  - Feature construction
  - Dimensionality reduction
  - Discretization

- **Data augmentation**
  - Increase dataset size using transformations (common in vision)

SCIENCE
ACADEMY

Data Cleaning

- **Purpose of data cleaning**
    - Ensure data quality for accurate model training
    - Detect and correct errors or inconsistencies in the dataset
- **Typical steps in data cleaning**
    - *Remove duplicates*: Identical records are eliminated
    - *Correct data entry errors*: Fix misspellings or misformatted entries
    - *Standardize data*:
        - Convert dates formatted as both MM/DD/YYYY and DD-MM-YYYY into consistent format
        - String normalization (e.g., lowercase conversion)
        - Type conversion (e.g., strings to integers)
        - Dealing with unexpected characters or encodings
- **Relevance to ML models**
    - Poor data quality leads to biased or incorrect predictions
    - Clean data reduces variance and improves generalization

SCIENCE
ACADEMY

Handling Outliers and Missing Data

- **Outliers**
  - *Definition*: Data points significantly different from others
  - *Causes*: Measurement errors, variability in the data
  - *Detection*: Box plots, Z-scores, or interquartile method
  - *Treatment*: Removal, capping, or transformation (e.g., log scale)
- **Missing data**
  - *Detection*: Count of null values or incomplete entries
- **Remediation**
  - *Deletion*: Remove rows or columns with too many missing values
  - *Imputation*:
    - Mean/median/mode substitution
    - K-nearest neighbors (KNN)
    - Regression or model-based approaches
    - E.g., for a missing temperature reading, impute using the mean of the day's surrounding values

SCIENCE
ACADEMY

Normalization and Standardization

- **Goal**: Adjust feature scales for better convergence and learning
  - *Normalize*: Rescale to $[0, 1]$
  - *Standardize*: Zero mean, unit variance
- **Why it helps**
  - Equal feature contribution in distance-based models
  - Faster convergence in gradient-based algorithms
  - Enables regularization
  - Easier feature interpretation
- **Common methods**
  - Min-Max normalization: $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Z-score standardization: $x' = \frac{x - \mu}{\sigma}$

Encoding Categorical Data

- **Goal**: Convert non-numeric categories into numeric representations

- **Label encoding**
  - Assigns an integer to each category
    - E.g., red, green, blue → 1, 2, 3
  - Can mislead models if order is not meaningful

- **One-hot encoding**
  - Creates binary vector per category
    - E.g., red, green, blue → [1,0,0], [0,1,0], [0,0,1]
  - Avoids ordinal assumption
  - Increases dimensionality

Feature Construction

- **Goal**: Derive more informative features from raw inputs

- **Methods**
  - Combining variables (e.g., area = height $\times$ width)
  - Extracting parts (e.g., year from a date)
  - Logical features
    - E.g., transform `2023-04-15` $\rightarrow$ (`Saturday, is_weekend = True`)

- **Why it helps**
  - Encodes domain knowledge
  - Improves model expressiveness and performance

Dimensionality Reduction

- **Goal**: Reduce number of features while preserving key information

- **Why it helps**
    - Reduce overfitting
    - Reduce data redundancy
    - Improve model speed
    - Allow visualization

- **Common techniques**
    - PCA: linear combinations that maximize variance
    - LDA: projects data to maximize class separability

- **Example**
    - Reduce 1024x640 image pixels to 10 principal components
    - Quantize color images into gray scale

SCIENCE
ACADEMY

Discretization

- **Goal**: Convert continuous values into categorical bins

- **Why it helps**
  - Simplifies models or enables categorical algorithms
  - Helps detect threshold effects in data

- **Techniques**
  - Equal-width binning
  - Quantile binning

- **Example**
  - Discretize age
    - `Child`: $[0, 13)$
    - `Teen`: $[13, 20)$
    - `Adult`: $[20, 65)$
    - `Senior`: $[65, \infty)$
 - Age $32 \rightarrow$ `Adult`

Noise Removal

- **Goal**: Remove irrelevant or corrupt data variations

- **Why it helps**
  - Improves signal clarity and model robustness
    - E.g., clean noisy speech by removing high-frequency noise

- **Methods**
  - Smoothing (e.g., moving average)
  - Filtering (e.g., low-pass filter in audio)

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - How to Do Research
  - Pipeline Organization
  - Input Processing
  - *Learning Algorithms*
 - Performance Metrics
  - Model Selection
  - Aggregation

SCIENCE
ACADEMY

The Problem of Minimizing a Function

- **Goal**: minimize scalar function $J(\underline{w})$ of n-variables $\underline{w}$
  - E.g., in-sample error $E_{in}(\underline{w})$
- **Solutions**:
  1. Analytical solution
     - Impose the gradient of $J(\underline{w})$ to equal 0
     - Find a closed-form solution for $\underline{w}^*$
  2. Numerical solution:
     - Use an iterative method to update $\underline{w}$ to reach the minimum value of $J(\underline{w})$
     - E.g., gradient descent
     - It works even if there is an analytical solution

Weights, $\mathbf{w}$
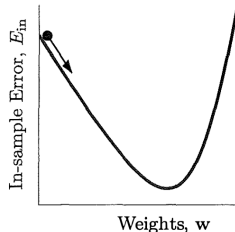
Gradient Descent: Intuition

- **Problem**:
  - You are on a hilly surface and we want to walk down to the bottom of the hill
- **Solution**:
  - At each point:
    - You look around
    - You move a step in the direction where the surface is steepest
  - You keep doing until we reach the bottom
- **Gradient descent**
  - Is a general technique for minimizing twice-differentiable functions
  - In general, converge to a local minimum
  - If $J(\underline{w})$ is convex, converge to the global minimum
    - E.g., logistic regression and linear models
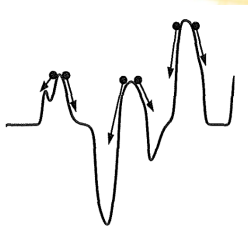
Gradient descent with fixed learning rate (1/3)

- Consider the contour plot of a function

- **Start** from a point $\underline{w}(0)$ (random, the origin, . . .)

- **At each step**

  - Move a fixed amount η in weight space (fixed learning rate):

  $$\underline{w}(t+1) = \underline{w}(t) + \eta\hat{\underline{v}}$$

  where $\hat{\underline{v}}$ is a unit vector

  - Pick $\hat{\underline{v}}$ to move to a value of $E_{in}(\underline{w})$ as negative as possible

- The **change for** E_{in} is:

  $$\begin{aligned}
  \Delta E_{in} &= E_{in}(\underline{w}(t+1)) - E_{in}(\underline{w}(t)) \\
  &= E_{in}(\underline{w}(t) + \eta\hat{\underline{v}}) - E_{in}(\underline{w}(t)) \quad \text{(replacing the expression of } \underline{w}(t+1)) \\
  &\approx \eta\nabla E_{in}(\underline{w}(t))^T\hat{\underline{v}} + O(\eta^2) \quad \text{(using Taylor expansion)}
  \end{aligned}$$

- Gradient descent keeps only $O(\eta)$ the term and ignores the rest

- Conjugate gradient considers up to $O(\eta^2)$ and ignores higher infinitesimals

SCIENCE
ACADEMY

Gradient Descent with Fixed Learning Rate (2/3)

- You have:
$$\Delta E_{in} \approx \eta \nabla E_{in}(\underline{w}(t))^T \hat{\underline{v}}$$

- The **minimal value of the scalar product** ΔE_{in}:
  - Happens when $\hat{\underline{v}} = -\frac{\nabla E_{in}(\underline{w}(t))}{\|\nabla E_{in}(\underline{w}(t))\|}$
  - Is $-\eta \|\nabla E_{in}(\underline{w}(t))\|$

- The **change in weights** $\Delta \underline{w}$ is:
$$\Delta \underline{w} = \underline{w}(t+1) - \underline{w}(t)$$
$$= \eta \hat{\underline{v}}$$
$$= -\eta \frac{\nabla E_{in}(\underline{w}(t))}{\|\nabla E_{in}(\underline{w}(t))\|}$$

- Called **"gradient descent"** since it descends along the gradient of the function to optimize

Gradient Descent with Fixed Learning Rate (3/3)

- Each j component of the weight $\underline{\boldsymbol{w}}$ is updated with the partial derivative with respect to that coordinate:

$$\underline{\boldsymbol{w}}(t+1) = \underline{\boldsymbol{w}}(t) - \eta \frac{\nabla E_{in}(\underline{\boldsymbol{w}}(t))}{\|\nabla E_{in}(\underline{\boldsymbol{w}}(t))\|}$$

$$w_j(t+1) = w_j(t) - \eta \frac{1}{\|\nabla E_{in}(\underline{\boldsymbol{w}}(t))\|} \frac{\partial E_{in}(\underline{\boldsymbol{w}})}{\partial w_j}$$

  - The update of all components should be **simultaneous**, i.e., computed at once
- A step of the optimization when we update the solution (weights) is called **epoch**

Gradient Descent: Stopping Criteria

- **In theory**, stop when $\Delta E_{in} = \underline{\mathbf{0}}$
  - Numerically, this might not occur
- **In practice**, stop when:
  - Variation of E_{in} is smaller than threshold $\Delta E_{in} < \theta$
  - Reached a certain number of iterations
- **Monitoring gradient descent**
  - In theory, only compute derivatives of function $J(\underline{\boldsymbol{w}})$ to optimize
  - In practice, monitor progress by recomputing cost function $J(\underline{\boldsymbol{w}})$ periodically to ensure it decreases

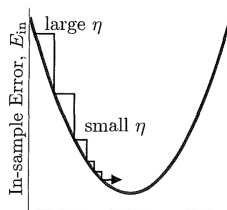# Gradient Descent with Fixed Learning Rate

- Consider a 1D convex function
  - If η **is too small**:
    - Linear approximation of E_{in} is effective
    - Many steps needed to converge to minimum
  - If η **is too large**:
    - Linear approximation fails (higher terms affect values)
    - It "bounces around"
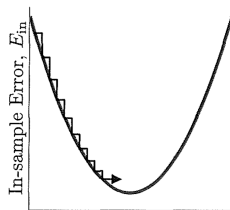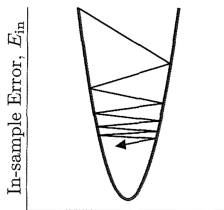
η too small η too large   variable η just right

- **Idea**: vary learning rate η during gradient descent
  - Smaller learning rates may find a better minimum
  - Reduce η as a function of iterations
  - **Cons**: introduces an additional parameter to tune

Gradient Descent with Variable Learning Rate

- In **gradient descent with fixed learning rate** (constant change in weight space), use:

$$\Delta \underline{\boldsymbol{w}} = -\eta \frac{\nabla J}{\|\nabla J\|}$$

- **To converge quickly:**
    - Move fast (large change) in weight space (large η) when surface is steep (large gradient)
    - Move slow (small change) in weight space (small η) near minimum to avoid bouncing (small gradient)
    - Ideally, η should increase with slope: $\eta \propto \|\nabla J\|$

- **Gradient descent with variable learning rate**:

$$\Delta \underline{\boldsymbol{w}} = -\eta \nabla J$$

SCIENCE
ACADEMY

Feature Scaling in Gradient Descent

- **Problem**: Different gradient components have different errors due to numerical approximation, causing bouncing
  - Unscaled features lead to slow, unstable convergence
  - E.g., feature ranging from 1 to 1000 vs. 0.01 to 1 causes inefficient updates
- **Solution**: Converge faster by scaling features to the same range
  - Min-max scaling
  - Standardization (transforms feature values to mean 0, standard deviation 1)

Issues with Batch Gradient Descent

- Consider squared error with n samples:

$$E_{in}(\underline{w}) \triangleq \frac{1}{n} \sum_{i=1}^{n} (h_{\underline{w}}(\underline{x}_i) - y_i)^2$$

- **Batch Gradient Descent (BGD)** updates each weight component:

$$\underline{w}(t + 1) = \underline{w}(t) - \eta \frac{\nabla E_{in}}{\|\nabla E_{in}\|}$$

- Coordinate update for squared error:

$$w_j(t + 1) = w_j(t) - \eta \frac{2}{n} \sum_{i=1}^{n} (h_{\underline{w}}(\underline{x}_i) - y_i) \frac{\partial h_{\underline{w}}(\underline{x}_i)}{\partial w_j}$$

- **Cons**: gradient descent with many training examples (e.g., $N = 10^6$)
  - Is computationally expensive, requiring gradient evaluation from all examples for one update
  - Requires storing all data in memory

Stochastic Gradient Descent

- Idea of **Stochastic Gradient Descent (SGD)**: update the weights only for one training example picked at random

- **Algorithm**
    - Pick one $(\underline{x}_n, y_n)$ at a time from the available examples
    - Compute $\nabla e(h(\underline{x}_n), y_n)$ to update the weights: $\Delta \underline{w} = -\eta \nabla e$
    - Update the weight considering only one random example:

$$w_j(t+1) = w_j(t) - \eta \frac{2}{n}(h_{\underline{w}}(\underline{x}_t) - y_t)\frac{\partial h_{\underline{w}}(\underline{x}_t)}{\partial w_j}$$

- **Cons**: in Stochastic Gradient Descent
    - Path in weight space is random
    - Oscillates around local minimum

- **Why SGD works**
    - ∇e is a function of a random var $\underline{x}_n$
    - The average direction of SGD is the same direction as batch version

$$\mathbb{E}[\nabla e] = \frac{1}{N}\sum \nabla e(h(\underline{x}_n), y_n) = \nabla \frac{1}{N}\sum e(h(\underline{x}_n), y_n) = \nabla E_{in}$$

Mini-Batch Gradient Descent

- **Mini-Batch Gradient Descent** brings together characteristics of both Batch and Stochastic Gradient Descent
    - Use b examples to make an update to the current weight
    - b represents the batch size
    - A common choice is $b = 32$ or $b = 64$
- Mini-batch GD offers a balance between SGD noisiness and full-batch approaches, using small, random data samples for updates

On-Line Learning and Gradient Descent

- In many applications, **continuous stream** of training examples → requires updating the model
  - In real-time systems, adapt model with new data points without full retraining
  - Handle variation in underlying process dynamics
- Stochastic GD and mini-batch GD **suitable for online learning**
  - Update model one example at a time
  - Discard examples for "compressed" model representation
    - Useful for large data streams where storing every data point is impractical
    - E.g., in stock market prediction
    - E.g., in training a language model on live chat data, discard older conversations after updates


SCIENCE
ACADEMY
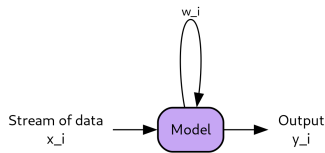
SGD vs BGD vs Mini-Batch

- To update the weights:
  - **BGD (batch gradient descent)** uses all the training examples
  - **SGD (stochastic gradient descent)** uses a single random training example
  - **Mini-batch GD** uses a random subset of training examples

Aspect	Batch Gradient Descent	Stochastic Gradient Descent
Computation	Uses all examples	One example at a time
Memory	Requires all examples in memory	Require less memory
Randomization	More likely to terminate in flat regions	Avoid local minima due to randomness
Regularization	No implicit regularization	Oscillations act as regularization
Parallelization	Can be parallelized	Less parallel-friendly
Online Learning	Not suitable	Suitable for online learning

SCIENCE
ACADEMY

Map-Reduce for Batch Gradient Descent

- Batch GD (and many learning algorithms) can be expressed in map-reduce
- **In map-reduce**
  - **Map step**: use k machines to parallelize summation
  - **Reduce step**: send k partial sums to a single node to accumulate result
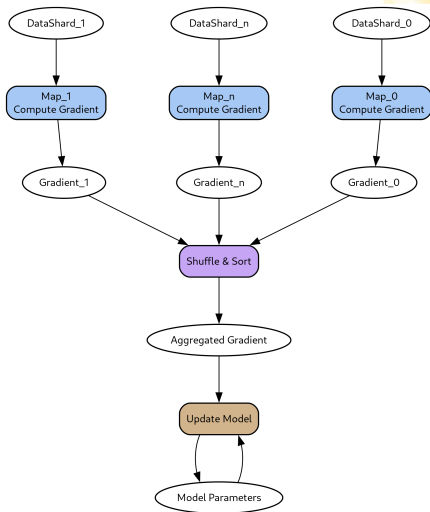
Coordinate Descend

- **Idea**: minimize $J(x_0, ..., x_n)$ by optimizing along one direction x_i at a time
    - Instead of computing all derivatives
- **Algorithm**
    - Pick a random starting point $\underline{w}(0)$
    - Pick a random order for the coordinates $\{x_i\}$
    - Find the minimum along the current coordinate (1D optimization problem)
    - Move to the next coordinate x_{i+1}
    - The sequence of $\underline{w}(t)$ is decreasing
    - A minimum is found if there is no improvement after one cycle of scanning all coordinates
    - The minimum is local

Gradient Descent vs Pseudo-Inverse

- For linear models, you can use pseudo-inverse or gradient descent to find optimal $\underline{\boldsymbol{w}}^*$

- **Gradient descent**
    - Choose learning rate η
    - Requires many iterations
    - Monitor stopping criteria, oscillations
    - Effective for many features P

- **Pseudo-inverse**
    - No parameter selection
    - Converges in one iteration
    - Computes a $P \times P$ matrix $(\underline{\underline{\boldsymbol{X}}}^T \underline{\underline{\boldsymbol{X}}})^{-1}$
        - Complexity of matrix inversion $O(P^3)$
        - For $P \approx 10,000$, prefer gradient descent

SCIENCE
ACADEMY

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - How to Do Research
  - Pipeline Organization
  - Input Processing
  - Learning Algorithms
  - *Performance Metrics*
  - Model Selection
  - Aggregation

How to Make Progress in ML Research

- In ML there are **many possible directions for research**, e.g.,

  - Different data preprocessing methods
  - Different features
  - Different models
  - Different training algorithms
  - Different evaluation techniques

- **What to do?**

- **Right approach**

  - Evaluate models systematically using a single number
    - Implement metrics (e.g., accuracy, F1 score) for insight
    - Use cross-validation for model validation
  - Statistical tests to ensure differences are not random
    - Utilize hypothesis testing for genuine improvements
    - Conduct A/B testing in real-world

SCIENCE
ACADEMY

How to Measure Classifier's Performance?

- **Success/hit/win rate (or error/miss rate)**
  - Measures correct predictions proportion
  - Important for overall accuracy
  - E.g., 80 correct out of 100 = 80% success rate
- **Log probability/cross-entropy error**
  - Evaluates classification with probabilities 0 to 1
  - Lower cross-entropy loss = better performance
- **Precision/recall/F-score**
  - Useful for imbalanced data
  - Precision: correctly predicted positives / total predicted positives
    - E.g., 0.75 precision = 75% true positives
  - Recall: correctly predicted positives / actual positives
    - E.g., 0.60 recall = 60% actual positives identified
  - F-score: weighted harmonic mean of precision and recall
- **Utility function**
  - Customizes evaluation to prioritize errors and success
    - E.g., true/false positives/negatives
  - E.g., in medical diagnosis, prioritize minimizing false negatives

SCIENCE
ACADEMY

Training vs Test Set

- Performance on train set E_{in} is an **optimistic estimate** of E_{out}
  - A model can have:
    - 0% error rate on training data (e.g., memorizing responses for training set)
    - 50% error rate on test set (e.g., answering randomly)
- To **properly evaluate model performance**:
  - Use test set not involved in training
  - Training and test sets should be representative samples
  - E.g., credit risk problem
    - Cannot use data from a Florida bank branch to assess a model built with New York data
    - Populations have different characteristics

Lots of Data Scenario vs Scarce Data Scenario

- **Lots of data scenario**
  - Ideal to have lots of data (ideally infinite)
  - Learn on lots of data
    - $\rightarrow$ Fit all degrees of freedom of a complex model
  - Predict on lots of data
    - $\rightarrow$ Assess precise out-of-sample performance
- **Scarce data scenario**
  - Often data is scarce
    - E.g., facial recognition datasets with limited annotated data
  - Cannot use all data as a training set
  - Hold out data to estimate performance metrics and bounds
    - Split data 70-30 or 80-20 in train and test sets
    - Use cross-validation to maximize data usage
  - Other approaches:
    - Augment data artificially, like in image processing
    - Use transfer learning with pre-trained models on related tasks

Splitting Data Into Training, Validation, Test Sets

- Training, validation, and test sets **must be**:
  - Distinct
  - Representative of the problem
    - E.g., each class in all sets represented according to original data
  - Sized based on data and problem needs
- Ensure sets have the **same distribution**:
  - Stratified sampling
    - E.g., each class label proportionally represented in each set
  - Shuffle and sample
    - Achieves randomization, maintains distribution
  - Sample and check statistics of variables
    - E.g., mean, std dev, PDF
    - Compare statistics to ensure each set mirrors broader dataset

SCIENCE
ACADEMY

Rule of Thumbs for Data Set Splits

- If n is **large** $\rightarrow$ use a 60-20-20 split
  - Training: 60%
  - Validation: 20%
  - Test: 20%
- If n is **medium** $\rightarrow$ use a 60-40 split
  - Training: 60%
  - Test: 40%
  - Not possible to learn hyperparameters, so no validation set
- If n is **small** $\rightarrow$ use cross-validation and report "small data size"
  - Use K-fold cross-validation
  - Be cautious of the increased chance of high accuracy by chance
  - Is machine learning even suitable for this problem?

SCIENCE
ACADEMY

Can You Use Test Set for Training?

- Once the model is selected and validated, reuse all data (including the test set) for deployment
  - Ensures the model benefits from all information
- More data is generally better, though returns diminish after a certain volume
  - Initially, increasing data size significantly improves performance
  - Eventually, adding more data results in smaller accuracy gains and may not justify increased computational cost
  - Use learning curves

SCIENCE
ACADEMY

In-Sample vs Out-Of-Sample Error Expressions

- **Goal of ML**: find a function h that approximates the unknown function f, $h \approx f$ over the space of inputs $\underline{x} \in \mathcal{X}$ ("script X")

- Usually the **error is point-wise**:

$$e(h(\underline{x}_i), f(\underline{x}_i))$$

  - E.g.,
    - Squared error: $e(\underline{x}) = (h(\underline{x}) - f(\underline{x}))^2$
    - 0-1 binary error: $e(\underline{x}) = I[h(\underline{x}) == f(\underline{x})]$
    - Log probability: $e(\underline{x}) = -\log(\Pr(h(\underline{x}) == f(\underline{x})))$

- **In-sample error** computed using all points in the training set:

$$E_{in}(h) = \frac{1}{N} \sum_{i=1}^{N} e(h(\underline{x}_i), f(\underline{x}_i))$$

- **Out-of-sample error** computed on the entire space of inputs $\mathcal{X}$:

$$E_{out}(h) = \mathbb{E}_{\underline{x} \in \mathcal{X}}[e(h(\underline{x}), f(\underline{x}))]$$

SCIENCE
ACADEMY

Mean Squared Error (MSE)

- MSE is the **average squared difference of error**:

$$\text{MSE} \triangleq \frac{1}{N} \sum_{i=1}^{N} (h(\underline{x}_i) - f(\underline{x}_i))^2$$

  - Measures estimator quality, quantifying difference between estimated and actual values
  - E.g., in house price prediction, determines how close predicted prices are to actual prices
- **Pros:**
  - Related to Gaussian error
  - Optimization friendly
- **Cons:**
  - Doesn't share unit of measure with output
    - Distorts error interpretation
  - Sensitive to outliers
    - Single large error can disproportionately affect MSE
    - Use median absolute deviation (MAD), median of squared error for robustness against outliers

Root Mean Squared Error (RMSE)

- RMSE is the **standard deviation of Mean Squared Error (MSE)**:

$$\text{RMSE} \triangleq \sqrt{\text{MSE}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (h(\underline{\boldsymbol{x}}_i) - f(\underline{\boldsymbol{x}}_i))^2}$$

- **Pros:**
    - Same units as output, allowing intuition of magnitude compared to mean
    - Facilitates comparison between data sets or models since metric is normalized to output's scale

- **Cons:**
    - Sensitive to outliers, which can excessively affect metric
    - May not be suitable for outliers or skewed distributions

SCIENCE
ACADEMY

Median-Based Metrics

- Use metrics based on median (i.e., the 0.5 quantile of absolute error):

- **Median absolute deviation**:

$$\text{MAD} \triangleq \text{median}_i(|h(\underline{\boldsymbol{x}}_i) - f(\underline{\boldsymbol{x}}_i)|)$$

- **Median squared error**:

$$\triangleq \text{median}_i(|h(\underline{\boldsymbol{x}}_i) - f(\underline{\boldsymbol{x}}_i)|^2)$$

- **Pros:**
    - Robust to outliers

- **Cons:**
    - Difficult to evaluate

SCIENCE
ACADEMY

How to Choose an Error Measure?

- Error measure depends on **application** and **"customer"** should define acceptable error level for their use case, e.g.,
  - Medical applications have low error tolerance
  - Recommendation systems have higher tolerance
- Otherwise, pick error measure that is
  - **Plausible**:
    - E.g., squared error for Gaussian noise
  - **Friendly**:
    - E.g., measures allowing closed-form solutions simplify calculations
    - Convex optimization-friendly measures ensure algorithms find global minimum easily

Error Measures: Fingerprint Verification Example

- In **fingerprint verification**:
  - Recognizing a valid fingerprint → no error
  - Otherwise → false positive or false negative
- **Error weight depends on the application**

Application	False Positive	False Negative
Supermarket	✅ Minor issue One extra discount	⚠️ Costly Annoyed customer, delays
CIA Building	💀 Critical Security breach	✅ Acceptable Triggers further checks

- For the same problem in two set-ups, the error measure is the opposite!

Error Metrics for Skewed Classes

- When **classes are skewed** (i.e., one class is very rare)
  - Accuracy can be misleading
  - Use metrics like confusion matrix, precision, and recall
- **Example:**
  - Train a classifier to distinguish tumors as:
    - $y = 1$: malignant
    - $y = 0$: benign
  - Classifier's error rate is 1% (i.e., guess correctly 99% of the time)
    - 1% error rate seems good!
  - But only 0.5% of patients have cancer
    - A trivial classifier that always outputs $y = 0$ has a 0.5% error rate!
    - Now a 1% error rate does not look good anymore

Confusion Matrix

- Assume
  - Actual/predicted class $\in \{0, 1\}$
  - Typically $y = 1$ encodes the rare class to predict
- You have 4 possible cases:
  - $act = 1$, $pred = 1 \rightarrow$ true positive (TP)
  - $act = 0$, $pred = 0 \rightarrow$ true negative (TN)
  - $act = 1$, $pred = 0 \rightarrow$ false negative (FN)
    - Model outputs $pred = 0$, but it is wrong
  - $act = 0$, $pred = 1 \rightarrow$ false positive (FP)
    - Model outputs $pred = 1$, but it is wrong

	pred $= 1$	pred $= 0$
act $= 1$	TP	FN
act $= 0$	FP	TN

- You can aggregate confusion matrix in **precision** and **recall**

SCIENCE
ACADEMY

Precision vs Recall: Definition

- Assume that $y = 1$ encodes the rare event

- **Precision** measures how often there is a true positive *given that pred = 1*

$$\text{precision} \triangleq \Pr(\text{TP}|\text{pred} == 1) = \frac{|\text{pred} == 1 \wedge \text{act} == 1|}{|\text{pred} == 1|} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall** measures how often there is a true positive *given that act = 1*

$$\text{recall} \triangleq \Pr(\text{TP}|\text{act} == 1) = \frac{\text{TP}}{|\text{act} == 1|} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- Both are conditional probability measuring the fraction of TP under different circumstances:

  - (Pre)cision: pred $= 1$
  - Rec(a)ll: act $= 1$

- Precision/recall are widely used in information retrieval

  - E.g., a search engine:
    - Returns 30 pages; only 20 are relevant $\rightarrow$ precision $= \frac{20}{30} = \frac{2}{3}$
    - Fails to return other 40 relevant pages $\rightarrow$ recall $= \frac{20}{(40+20)} = \frac{20}{60} = \frac{1}{3}$

SCIENCE ACADEMY

Precision / Recall as Quality / Quantity

- **"Increasing precision"** means when you predict 1, you are more likely to be right
  - A higher precision indicates fewer false positives
    - E.g., in a spam email detection system, "precision is 90%" means 90% of the emails marked as spam are actually spam
  - Measures "quality" of prediction
- **"Increasing recall"** means you predict more instances when the outcome is 1
  - A higher recall means fewer false negatives
    - E.g., in a spam email detection system, "recall is 80%" indicates 80% of all actual spam emails were correctly identified as spam
  - Measures "quantity" of prediction (coverage)

SCIENCE
ACADEMY

Precision / Recall for Trivial Classifiers

- A **classifier that outputs always most common class** has:

$$\text{precision} \triangleq \Pr(\text{TP}|\text{pred} == 1) = 0 \text{(since TP} = 0)$$
$$\text{recall} \triangleq \Pr(\text{TP}|\text{act} == 1) = 0 \text{ (since TP} = 0)$$

- A **classifier that outputs always rare class** has:

$$\text{recall} = 1 \qquad\qquad\qquad\qquad \text{(since FN} = 0)$$
$$\text{precision} \triangleq \Pr(\text{TP}|\text{pred} == 1) \qquad\qquad \text{(by definition)}$$
$$= \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad\qquad (TP + FP = n \text{ because}$$
$$= \frac{\#(y = 1)}{n} \qquad\qquad \text{classifier always emits 1)}$$
$$= \Pr(\text{pos}) \approx 0 \qquad \text{(the positive class is very rare)}$$

- A trivial classifier has precision or recall close to 0

Trading Off Precision and Recall

- In theory, you want to increase **both precision and recall**

- In practice, you can modify the threshold of a probabilistic classifier to **trade off precision and recall**

- E.g., use logistic regression to predict cancer:
    - With a threshold $\theta = 0.5$, the classifier has:
        - Precision $= \frac{TP}{|\text{pred} == 1|}$
        - Recall $= \frac{TP}{|\text{act} == 1|}$
    - Increase threshold θ
        - Output 1 only if more confident
        - I.e., increase precision
    - Decrease threshold θ
        - Output 1 more often
        - Decrease the chances of missing a possible case of cancer
        - I.e., increase recall

Precision-Recall: Pros / Cons

- **Pros:**
  - Give insight on the behavior of a classifier (e.g., confusion matrix)
  - Avoid mistaking a trivial classifier for a good classifier
- **Cons:**
  - You have two different numbers, thus it is difficult to compare classifiers to each other
  - Solutions: F-score, AUC

Precision-Recall Curves

- Aka ROC curves

- Show precision vs recall trade-off for a classifier

  - Plot the curve on a precision-recall plane (y = precision, $1 - x$ = recall)
  - E.g., changing threshold of logistic regression

- Precision-recall plot can have **different shapes**, e.g.,

  - Diagonal (pure luck)
  - Convex up (better than luck)
  - Convex down (worse than luck)

- A curve **higher** than another means a better classifier, since for the same recall you get higher precision

  - Best classifier (precision = recall = 1) is in the top-right corner

Area Under the Curve

- AUC is the **area under the precision-recall curve**
  - Provides a robust metric by integrating over all thresholds
  - Higher AUC indicates better class differentiation
  - AUC = 0.5 suggests no discriminative power, like random guessing
  - AUC closer to 1.0 indicates high performance
- **Pros:**
  - Single number summarizing classifier behavior, useful for comparing models
  - Does not require selecting a threshold for performance calculation
  - Handles imbalanced datasets effectively
- E.g., consider a classifier for medical diagnosis
  - AUC helps understand model's ability to distinguish between patients with and without a disease

F-Score

- The F-score is defined as the **harmonic mean of precision and recall**:

$$\text{F-score} \triangleq \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2\frac{P \cdot R}{P + R}$$

- **Interpretation:**
    - Trivial classifiers
        - $P = 0$ or $R = 0 \implies$ F-score $= 0$
    - Perfect classifiers
        - $P = R = 1 \implies$ F-score $= 1$
    - For F-score to be large, both P and R must be high

- Why not just averaging P, R?

    - A classifier that always outputs 1 has:
        - $R = 1$ and $P = 0$
        - $\frac{P+R}{2} = \frac{1}{2}$
        - We prefer a low value (ideally 0)

SCIENCE
ACADEMY

- Machine Learning Paradigms
- Machine Learning Techniques
  - Machine Learning in Practice
  - How to Do Research
  - Pipeline Organization
  - Input Processing
  - Learning Algorithms
  - Performance Metrics
  - *Model Selection*
  - Aggregation

The Problem of Model Selection

- Model selection:
  - Chooses the best model from candidates based on performance
  - Is needed when multiple hypotheses explain data
- Across models certain parameters are fixed
- Others need selection, e.g.,
  - Set of features
    - E.g., select a subset from 100 variables
  - Learning algorithms
    - E.g., decide how to train a neural network
  - Model types
    - E.g., linear regression vs. SVM
  - Model complexity
    - E.g., polynomials of degree $d < 10$
  - Values of regularization parameter
    - E.g., try values like 0.01, 0.1, 1.0
- Evaluate metrics (e.g., model accuracy, precision, recall) with cross-validation

SCIENCE
ACADEMY

Model Selection Process

1. Split data into $D_{train}, D_{val}, D_{test}$
   - Commonly: 60% training, 20% validation, 20% test
   - Like splitting 80% training between two learning phases

2. Given N hypotheses, learn on D_{train} to get $g_1, ..., g_N$

3. Evaluate hypotheses on D_{val} estimating errors $E_{val}^{(1)}, ..., E_{val}^{(N)}$

4. Pick model g_m with minimum $E_{val}^{(m)}$

5. Use test set D_{test} to estimate fair performance of model g_m, i.e., $E_{val} \approx E_{out}$

6. Retrain model with entire $D = D_{train} \cup D_{val} \cup D_{test}$ to get final g_m^*

Model Selection as Learning

- *"Picking the model with smallest E_{val}"* is a **form of learning**:
  - Hypothesis set: $\{g_1, ..., g_N\}$
  - Training set: D_{val}
  - Pick the best model g_m
- After model selection, you need to **assess the true performance** on D_{test}
  - Experimentally

  $$E_{val}(g_m) < E_{out}(g_m)$$

    - I.e., $E_{val}(g_m)$ is a (optimistically) biased estimate of $E_{out}(g_m)$
  - Theoretically:
    - The penalty for model complexity with a finite set of hypotheses is

    $$E_{out}(g_m) \leq E_{val}(g_m) + O(\sqrt{\log(N/K)})$$

    - Use VC dimension for an infinite number of hypotheses (e.g., choice of λ for regularization)

SCIENCE
ACADEMY

- Machine Learning Paradigms
- Machine Learning Techniques
    - Machine Learning in Practice
    - How to Do Research
    - Pipeline Organization
    - Input Processing
    - Learning Algorithms
    - Performance Metrics
    - Model Selection
    - ***Aggregation***

Ensemble Learning: Intuition

- **Idea**
  - A group of weak learners can form a strong learner
  - Combine multiple models to improve prediction accuracy
- **Combine outputs of models** X_i to build a model X^* better than any X_i
  - Utilize diversity in model predictions to improve accuracy
  - Each model contributes its unique perspective, reducing overfitting
  - E.g., like a panel of voting experts
- **Example**: in computer vision detecting a face is difficult task (circa 2010)
  - Look for different features:
    - Are there eyes?
    - Is there a nose?
    - Are eyes and nose in the correct position?
  - Each feature is weak by itself, but together they become reliable

Ensemble Learning: Different Techniques

- **Bagging** (bootstrap + aggregation)
  - Reduces variance by averaging predictions from different models
  - E.g., decision trees + bagging = random forest
    - Creates multiple versions of a decision tree
    - Each tree is trained on a random sample of data
    - Averages predictions to improve accuracy
- **Boosting**
  - Reduces bias by focusing on errors made by previous models
  - Sequentially adds models, each correcting its predecessor
  - E.g., adaBoost increases weights of incorrectly classified data points to learn the next model
- **Stacking**
  - Uses a meta-model to combine separate models using weights
  - E.g., a stacking ensemble
    - Uses logistic regression as a meta-model
    - Combines predictions of other models (e.g., decision trees, support vector machines, neural networks)

Ensemble Learning: Relation with Statistics

- **Bagging**
  - Improves performance by adding randomized variants (mimicking multiple training sets)
  - Reduce variance without affecting bias
- **Boosting**
  - Use another model to learn the residuals, i.e., difference between predicted and true values
  - Related to "forward stagewise additive models"
- **Stacking**
  - If we have 3 independent classifiers, each with $\Pr(\text{correct}) = 0.7$

$$\Pr(\text{majority correct}) = \Pr(\text{at least 2 classifiers correct})$$
$$= \binom{3}{2} 0.7^2 0.3 + 0.7^3$$
$$= 3 \times 0.7^2 \times 0.3 + 0.7^3$$
$$\approx 0.78 > 0.7$$

SCIENCE
ACADEMY

Ensemble Learning: Pros and Cons

- **Pros**
  - Hypothesis set $\mathcal{H}$ is increased by combining hypotheses from different models
- **Cons**
  - More computationally intensive to train and evaluate
  - Loss of interpretability
  - Risk of overfitting (model complexity is increased)
  - Ensemble learning contradicts Occam's razor, which advocates simplicity

When Ensemble Learning Works

- Combining multiple models with ensemble learning works when models:
    - Are very different from each other
    - Treat a reasonable percentage of the data correctly
        - E.g., you cannot do much if all classifiers have 50% accuracy
    - Complement each other: they are specialists in a part of the domain where the others don't perform well

How to Combine Outputs in Ensemble Learning

- **Regression**
  - Weighted average of prediction
  - E.g., by accuracy of each model or by a prior
- **Classification**
  - Weighted vote of predicted classes
  - It needs an odd number of models to break ties
- **Probabilistic classification**
  - Weighted average of class probabilities
- **Learn a meta-learner** (i.e., stacking) to combine multiple models

Bagging

- Bagging stands for "Bootstrap AGGregation"

- **Learning procedure**
  - Several training datasets are extracted randomly by sampling with replacement from the original dataset (i.e., bootstrap)
  - Learn multiple models, one for each training set
  - Combine outputs using various methods
  - Result is a better model than a single model

- **Why bagging works?**
  - From the bias-variance decomposition view, combining multiple models:
    - Reduces the variance component
    - Without compromising the bias (bagged models are typically unbiased)
  - Bagging mimics extracting more training sets (though not independent) from the unknown distribution

SCIENCE
ACADEMY

Bagging and Instability in Learning Algorithms

- Bagging works best with highly different models, especially non-linear models
  - Introduce randomization in the learning algorithm intentionally
- **Decision Trees**
  - Disable pruning
  - Break ties randomly when selecting the best attribute to split
  - E.g., bagging trees results in random forests
- **Multilayer Perceptrons**
  - Use different initial weights in backpropagation to reach different local minima
- **Nearest Neighbor Classifier**
  - Use a random subset of features
  - Resampling the training set has limited impact, as it is equivalent to changing example weights

SCIENCE
ACADEMY

Boosting

- Boosting builds models that complement each other
  - Typically use homogeneous models, i.e., parametrized models from $\mathcal{H}$
- Strong classifiers can be built from weak classifiers
  - E.g., decision stumps = decision trees with one level
- **Why boosting works?**
  - Boosting implements forward stagewise additive modeling
  - Use another model to learn residuals (difference between predicted and true values)
- Boosting does not work for linear regression:
  - Combination of linear models is still a linear model
  - OLS finds optimal weights in one step
  - Combining linear regressions from different attributes is equivalent to a single multiple linear regression

SCIENCE
ACADEMY

Adaboost.M1

- Widely used for classification
- Assume examples can be weighted in the cost function used to learn
    - Otherwise use resampling
- **Learning procedure**
    - Start with equal weights for all examples
    - Iterate:
        - Learn a classifier based on current weights for examples
        - Weight the answer of each model by overall score (e.g., accuracy) or probability
        - Evaluate the ensemble
        - Adjust weights for examples classified correctly/incorrectly

Stacking

- Stacking is about learning how to combine models (not necessarily of the same type)

- The problem is that with voting / averaging we don't know which model to trust

- Instead of voting or weighting we can use a meta-learner (level 1) to learn how to pick / mix models (level 0)

- **Learning procedure**
  - Learn "level 0" models
  - Learn "level 1" model using hold-out data from learning of level 0 models (like in model selection)
    - Build training data with predicted values from level 0 models
    - Then learn level 1
    - Use a simple model for level 1 (e.g., linear models or trees) to avoid overfitting
    - Use probabilities from level 0, so level 1 can assess the confidence of each model

SCIENCE
ACADEMY

Boosting vs Bagging vs Stacking

Aspect	Bagging	Boosting	Stacking
Combines	Models of the same type	Models of the same type	Models of different types
Learning	Models trained independently	Iterative training	Models trained independently
Predicting	Uses uniform or data-driven weights	Uses learned weights from training	Uses learned weights or confidence
Main Objective	Reduce variance	Reduce bias	Improve generalization through diversity
Base Learners	Often strong learners	Often weak learners	Any model type (heterogeneous ensemble)
Sensitivity to Noise	Low	High	Medium
Parallelizable	Yes	No (sequential dependency)	Partially (base models parallelized)
Meta-model	Not used	Not used	Required
Examples	Random Forest	AdaBoost, Gradient Boosting	Stacked Generalization, Blending