

LB2 project

Introduction

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project general objectives

- Address a **real-world problem** in Bioinformatics, following-up from LB1 project in the process of **protein functional annotation**
- Learn how to **describe and visualize data** using a real-world dataset
- Learn how to design and implement a complete, end-to-end **machine-learning workflow**
- Understand and use **machine-learning techniques**
- Practice on **Python programming** on concrete examples

Project topic

Perform a comparison between different approaches for **predicting the presence of the secretory signal peptides in proteins**

A subproblem of **protein function and subcellular localization prediction**

Proteins directed toward the **cell secretory pathway** (ER-Golgi-Membrane-Extracellular) are endowed with a **signal sequence** in the N-terminal region

The **signal sequence is cleaved** after the protein reaches its final destination

In silico recognition of the presence of the signal peptide is a key step for the characterization of protein function and subcellular localization

Project work plan

We will implement and compare two different approaches for signal peptide detection

Approaches considered are:

- A simple statistical method based on motif discovery and recognition originally introduced by Gunnar von Heijne in 1986
- A more advanced machine-learning approach based on **Support Vector Machines (SVM)**
- **Optional: you are free of extending/proposing different solutions based on more advanced approaches e.g. (deep) neural networks**

Project work plan

You have to:

- Retrieve a dataset from UniProtKB
- Prepare the dataset for cross-validation and benchmarking
- Perform and visualize statistics on both datasets
- Prepare your data for training and prediction -> feature extraction
- Implement the von Heijne algorithm
- Implement the SVM-based approach (using sklearn)
- Perform experiments (in cross-validation and blind test) and discuss results
- (Optional) Implement and test different solutions
- Write the manuscript

Lecture organization

Laboratory activity is fundamental in this course

We will try develop the project together, step-by-step, during lectures

Lectures will be structured as to firstly **provide basic concepts** for each project step, then **followed by practical activities**

The implementation of a specific project step can span throughout many days of practical activities

I'll be always available during practical activities to discuss any issue you may have during project development

Raising issues

During lectures:

- Please, feel free to interrupt me every time you need clarifications or repetitions

After lectures:

- Email: castrense.savojardo2@unibo.it
- MS Teams: catch me on Teams when available

Considerations on the project development

- We will try to complete the entire project during lectures
- Some steps of the project, e.g. data collection involve the interaction with external resources like UniProtKB
 - I tried to execute in advance all the steps of the project, but unexpected issues may happen during the execution of the exercise
 - **Finding together solutions to these issues is part of the exercise**
- We will discuss all the concepts that are needed to finalize the project
 - Everybody can reach the goal, provided all lectures are attended
 - Completing the basic project is sufficient to reach the maximum score
 - You are free to propose new solution and extend the project if you like

Exams

- Submission of a project report in the form of a scientific paper
 - At least one week before the exam date
 - You can be asked to re-submit the report if incomplete
- Oral exam:
 - Discussion of the project
 - Questions on other theoretical topics covered during the course

Lecture content

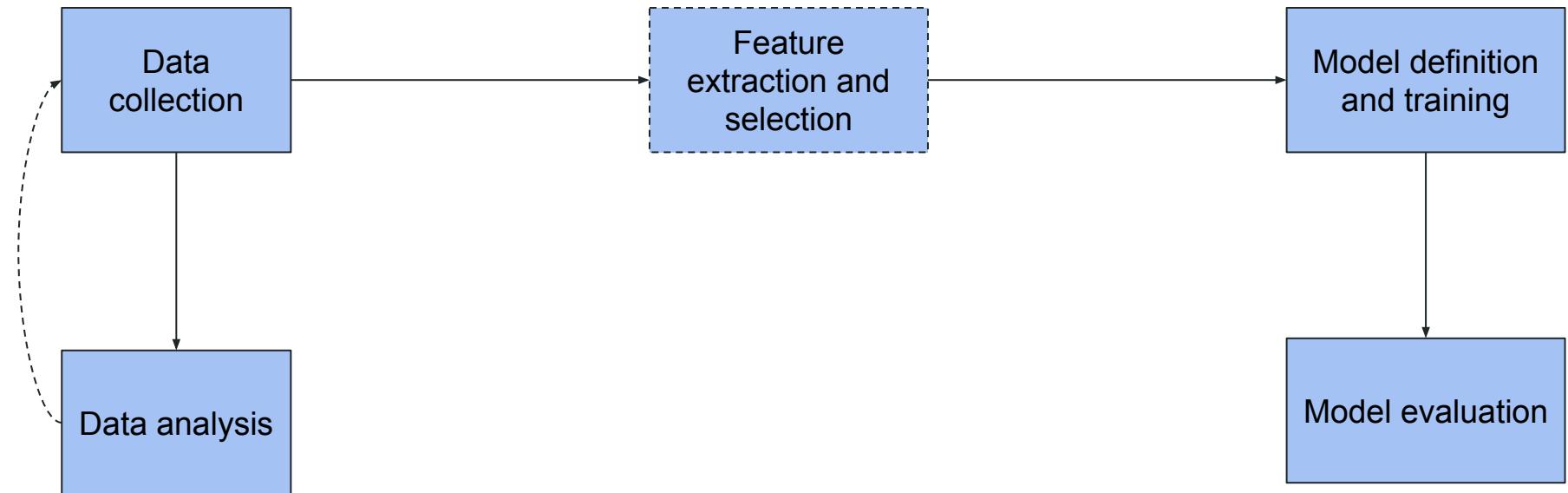
The following aspects, strictly connected with the project workflow, will be covered:

- Data collection
- Data analysis, description and visualization
- Data pre-processing for input preparation
- Methods definition and implementation
- Performance evaluation procedures
- How to write the final report

Other topics (to be given in the last part of the course):

- Introduction to deep learning and applications

Machine-learning workflow



Data collection and preparation

- The first step of any machine-learning workflow
- Data are collected from domain-specific databases (e.g. UniProtKB for protein sequences or PDB for protein structures) according to different criteria
 - **Quality** (e.g. overall quality of the entries, experimental evidence for the feature of interest)
 - **Redundancy**: collected data (e.g. protein sequences) must be as diverse as possible to properly cover the domain of interest
- **Secondary data and metadata**: additional information useful for contextualizing the primary data
- Perform necessary steps to:
 - Reduce data redundancy
 - Splitting data for cross-validation and blind test
- Data collected are usually available in raw formats e.g. TSV, FASTA or CSV files
- Put the raw data in a proper format to be then processed by ML/algorithmic tools
 - Choose the proper format which facilitate all the subsequent steps

Data analysis, description and visualization

- Explore features of the collected datasets e.g. protein compositions, lengths, taxonomy etc
- Perform statistics on primary data and metadata
- Report statistics using tables or visualize them by means of plots
- Features to be explored depends on the problem at hand
- Overall, useful for proving the adequacy of the datasets for the project goal
- **Statistical analysis can reveal biases in the data, suggesting to revise the data collection process**

Feature extraction and selection

- **Feature extraction:** extract and encode all the features of interest that will be subsequently used for learning ML models
- **Feature selection:** selection of the most informative subset of features:
 - Can be an independent step performed before model optimization
 - Can be part of the method optimization process

Model definition, implementation and training

- Choose a bunch of algorithms to test
- For classification:
 - Support Vector Machines
 - Neural Networks
 - Random Forests
 - Naive Bayes
 - Logistic Regression
- For regression:
 - Support Vector Regression
 - Neural Networks
 - Linear Regression
 - Ensemble Methods
- Implement the required code for training and testing
- Run cross-validation to select the best model and optimize model architecture and hyperparameters

Model evaluation

- Cross-validation already provides some estimate of the model performance
- Additional evaluation done using hold-out independent data (blind test or benchmarking set)
- Define and compute scoring measures appropriate for the task domain:
 - Accuracy
 - Precision and Recall
 - **F1-score**
 - **Matthews Correlation Coefficient**
 - Area Under the ROC Curve (AUC)
 - Other, task-specific measures
- Compare with simple baseline approaches
- Include a comparative benchmark with other methods at the state-of-the-art

LB2 project

VM and environment setup

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project development

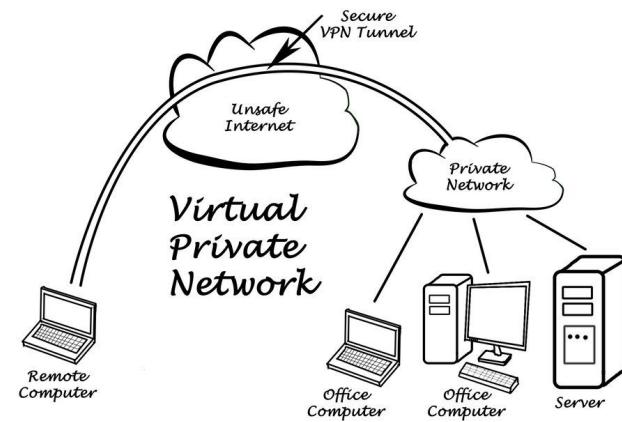
- We will provide each of you with a **Linux Virtual Machine (VM)** which will be used to develop the project
- Each VM is already **equipped with all the required software** for the project development
- You will be instructed on **how to access your personal VM** and how to use it

Virtual machines

- 4 cores
- 16 GB of RAM
- 100 GB of disk
- Software required for the project in a conda environment
- Personal:
 - You can't share with others
 - You are responsible for it

How it works

- Virtual machines are hidden from the public
- Accessible only via Virtual Private Network (VPN)
 - A private, encrypted communication channel
- Login allowed only using SSH keys
 - No password, but key files
- Conda environment
 - Provides the software used for the project



What do you need

- We will provide the following:
- OpenVPN certificate
 - Filename like `LSB_<your_name>_lsb-gw.ovpn`
 - e.g. `LSB_SavojardoCastrense_lsb-gw.ovpn`
- Private SSH key
 - Filename like `umXX_id_rsa` e.g. `um50_id_rsa`
- **These files are present inside the zip archive you received via Filesender**
- **Please, uncompress the zip archive in your laptop home directory**

The ZIP archive content

SurnameName/

SurnameName/LSB_SurnameName_lsb-gw.ovpn

SurnameName/mXX/umXX/

SurnameName/mXX/umXX/umXX_id_rsa

SurnameName/mXX/umXX/umXX_id_rsa.pub

Installing OpenVPN

- Linux (Debian/Ubuntu either native or Vbox)

```
sudo apt-get install openvpn
```

- Mac OSx

The easiest way is to install Tunnelblick: <https://tunnelblick.net>

Otherwise, installing OpenVPN through the Homebrew package manager:

```
brew install openvpn
```

- Windows users

Download and install the OpenVPN client Community version from:

<https://openvpn.net/community-downloads/>

Step 1: OpenVPN (Linux)

- Connects to the private network
- You must be superuser to run it
- Command:

```
sudo openvpn --config LSB_<yourname>_lsb-gw.ovpn
```

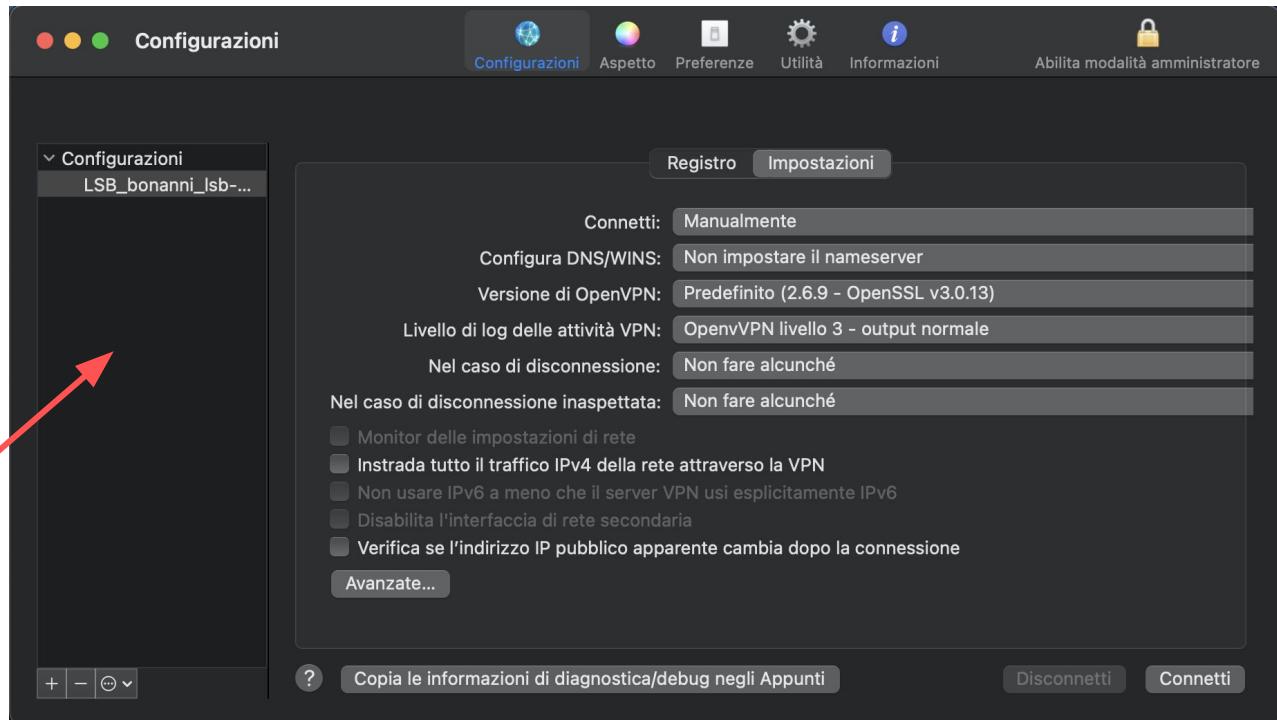
- “**Initialization Sequence Completed**” means OK
- As far as this terminal is up you can access the private network
 - Close the terminal/connection only when you are done

Step 1: example terminal output (Linux and OSX)

```
Fri Jul  3 15:31:09 2020 DEPRECATED OPTION: --max-routes option ignored. The number of routes is unlimited as of OpenVPN 2.4. This
option will be removed in a future version, please remove it from your configuration.
Fri Jul  3 15:31:09 2020 OpenVPN 2.4.9 x86_64-apple-darwin18.7.0 [SSL (OpenSSL)] [LZO] [LZ4] [PKCS11] [MH/RECVDA] [AEAD] built on
Apr 17 2020
Fri Jul  3 15:31:09 2020 library versions: OpenSSL 1.1.1g  21 Apr 2020, LZO 2.10
Fri Jul  3 15:31:09 2020 Outgoing Control Channel Authentication: Using 160 bit message hash 'SHA1' for HMAC authentication
Fri Jul  3 15:31:09 2020 Incoming Control Channel Authentication: Using 160 bit message hash 'SHA1' for HMAC authentication
Fri Jul  3 15:31:09 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]137.204.193.15:443
Fri Jul  3 15:31:09 2020 Attempting to establish TCP connection with [AF_INET]137.204.193.15:443 [nonblock]
Fri Jul  3 15:31:10 2020 TCP connection established with [AF_INET]137.204.193.15:443
Fri Jul  3 15:31:10 2020 TCP_CLIENT link local: (not bound)
Fri Jul  3 15:31:10 2020 TCP_CLIENT link remote: [AF_INET]137.204.193.15:443
Fri Jul  3 15:31:10 2020 VERIFY OK: depth=1, O=58b40ee4d8e99843199ce5a1, CN=58b40ee4d8e99843199ce5a7
Fri Jul  3 15:31:10 2020 VERIFY KU OK
Fri Jul  3 15:31:10 2020 Validating certificate extended key usage
Fri Jul  3 15:31:10 2020 NOTE: --mute triggered...
Fri Jul  3 15:31:10 2020 4 variation(s) on previous 3 message(s) suppressed by --mute
Fri Jul  3 15:31:10 2020 [58b40ee5d8e99843199ce5b5] Peer Connection Initiated with [AF_INET]137.204.193.15:443
Fri Jul  3 15:31:12 2020 Outgoing Data Channel: Cipher 'AES-128-CBC' initialized with 128 bit key
Fri Jul  3 15:31:12 2020 Outgoing Data Channel: Using 160 bit message hash 'SHA1' for HMAC authentication
Fri Jul  3 15:31:12 2020 Incoming Data Channel: Cipher 'AES-128-CBC' initialized with 128 bit key
Fri Jul  3 15:31:12 2020 NOTE: --mute triggered...
Fri Jul  3 15:31:12 2020 1 variation(s) on previous 3 message(s) suppressed by --mute
Fri Jul  3 15:31:12 2020 Opening utun (connect(AF_SYS_CONTROL)): Resource busy (errno=16)
Fri Jul  3 15:31:12 2020 Opened utun device utun1
Fri Jul  3 15:31:12 2020 /sbin/ifconfig utun1 delete
ifconfig: ioctl (SIOCDIFADDR): Can't assign requested address
Fri Jul  3 15:31:12 2020 NOTE: Tried to delete pre-existing tun/tap instance -- No Problem if failure
Fri Jul  3 15:31:12 2020 /sbin/ifconfig utun1 192.168.233.4 192.168.233.4 netmask 255.255.255.0 mtu 1500 up
add net 192.168.233.0: gateway 192.168.233.4
add net 172.16.201.0: gateway 192.168.233.1
Fri Jul  3 15:31:12 2020 [REDACTED] -- do not store the passwords in memory -- use the auth-nocache option to prevent this
Fri Jul  3 15:31:12 2020 Initialization Sequence Completed
```

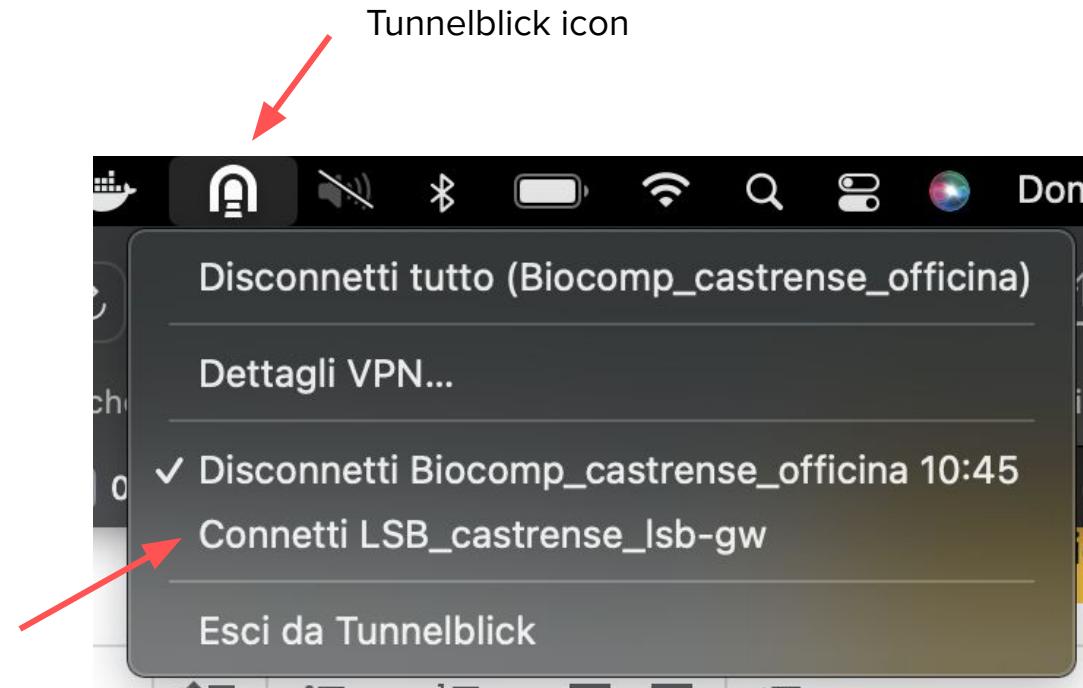
This means you are connected

Tunnelblick (OSx)



Tunnelblick (OSx)

Connect to
specific VPNs



Step 1: OpenVPN (Windows)

Start the OpenVPN client program (from the link on Desktop). An icon should appear in the status bar (right-left, close to the clock)

Copy the file **LSB_<yourname>_lsb-gw.ovpn** into:

C:\Programs\OpenVPN\config

Right click the OpenVPN icon on the status bar and do “Connect”

A window should appear showing you the connection status

Step 2: shell on the server (Linux and OSX)

- Open a new terminal or tab (leaving the one with VPN connection open)
- Connect to your machine
- Username is: **umXX**
- Host is: **mXX.lsb.biocomp.unibo.it**
- Command:

```
ssh -i keys/umXX_id_rsa umXX@mXX.lsb.biocomp.unibo.it
```

Step 2: shell on the server (Windows)

Download the MobaXterm portable version from:

https://download.mobatek.net/2032020060430358/MobaXterm_Portable_v20.3.zip

Unzip the archive and run the program

Configure the connection to the server:

Remote host: **mXX.1sb.biocomp.unibo.it**

Specify username: **umXX**

In **Advanced SSH settings** check **Use private key** then browse and select the private key file: **umXX_id_rsa**

Step 3: activate the environment

- Before running any program you need to activate the conda environment
- Run the following commands (sequentially):

```
source /opt/conda/bin/activate
```

```
conda activate tools
```

- That activate the environment with all the required software and libraries
- To exit from the environment:

```
conda deactivate
```

Main packages available

- biopython
- scikit-learn (machine learning library)
- numpy
- scipy
- keras (deep learning)
- pytorch (deep learning)
- matplotlib (plotting)
- seaborn (plotting)
- pandas (data manipulation)
- blast

Copy files from/to VM

- Execute both commands from your local computer (laptop)
 - You can't reach your laptop from inside the VM
- Copy a local file to the VM:

```
scp -i <key_file> <source path> umXX@mXX.lsb.biocomp.unibo.it:<destination path>
```

- Copy a file from the VM to a local folder:

```
scp -i <key_file> umXX@mXX.lsb.biocomp.unibo.it:<source path> <destination path>
```

Running programs in background

- Some steps of the project are computational intensive and require a long time to complete
 - To execute these steps you need a way to keep the terminal open until they complete
- Problem: you need to maintain the connection up and running until completion
 - This because, in the event that the terminal is closed, the terminal together with all child processes will be terminated
- Solution: detach processes from the controlling terminal (i.e. run in background)
 - You “submit” the script (Python or BASH) that executes the computation and then you can disconnect from the VM
 - Later, you can reconnect to the VM to check if the script terminated or not.
- Many tools are available to run scripts in background:
 - at, nohup, screen etc

GNU Screen

- Formally, a **terminal multiplexer**
 - a. You can start a screen **session** and then open any number of windows (**virtual terminals**) inside that session
- Processes running in Screen will continue to run even if you get disconnected
- The typical workflow for using screen requires the following steps:
 - a. Create a new screen session with at least one virtual terminal
 - b. Execute commands (possibly long-running tasks) within the virtual terminal
 - c. Detach (i.e. disconnect from session without closing it) from the screen session
 - d. Reattach to session and continue working on it
- After creation (step a) you will iterate through steps b-c until you are done.

Screen commands

- To create a new screen session, simply type screen in your console:

screen

- This will:
 - open a screen session
 - create a new window
 - start a shell in that window.
- The virtual screen terminal works exactly as the normal one
 - you can move through the filesystem, open files with editors, execute commands etc

Screen commands

- Detach from the current session:

Ctrl+a d (i.e. press **Ctrl** and **a** together, then **d**)

- Get the list of detached screen sessions:

screen -ls

- Reattach to the previously created session (the only one):

screen -x

- Reattach to a specific session:

screen -r <session_id>

Screen commands

- Create a new window within the current session:

Ctrl+a c

- List all windows within the current session:

Ctrl+a "

- Get help (display a list of commands):

Ctrl-a ?

LB2 project

Introduction to signal peptide prediction

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project topic

Perform a comparison between different approaches for **predicting the presence of secretory signal peptides in proteins**

A subproblem of **protein function and subcellular localization prediction**

Proteins directed toward the **cell secretory pathway**

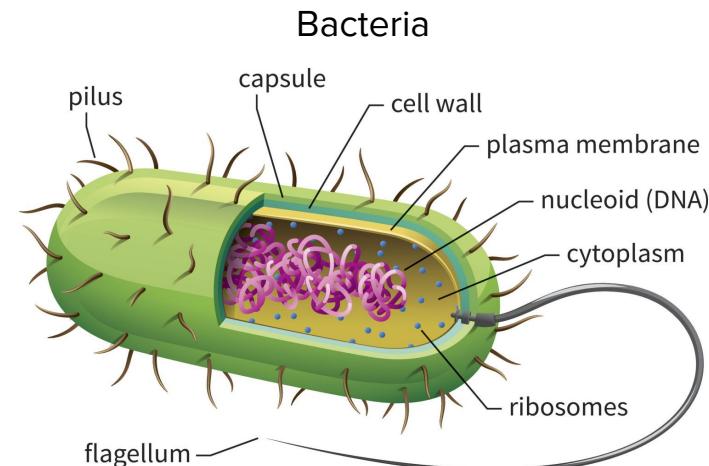
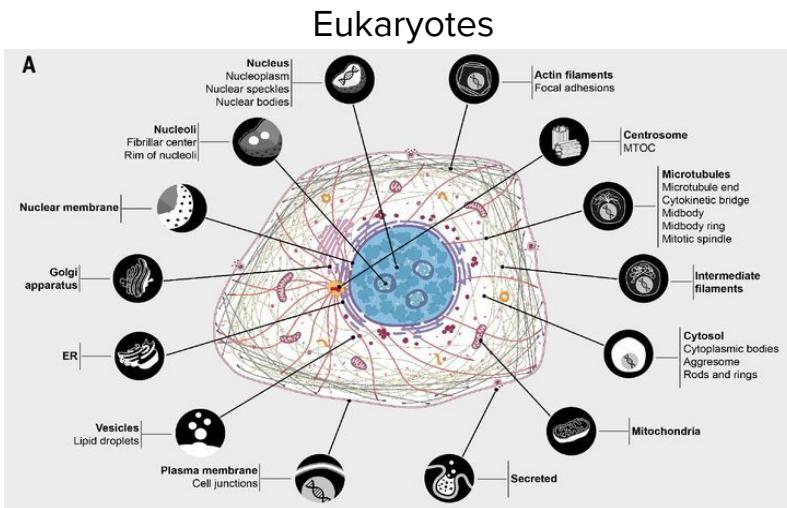
(ER-Golgi-Membrane-Extracellular) are endowed with a **signal sequence** in the N-terminal region

The **signal sequence is cleaved** after the protein reaches its final destination

In silico recognition of the presence of the signal peptide is a key step for the characterization of protein function and subcellular localization

Protein subcellular localization

- At the cellular level, proteins function at specific subcellular locations or organelles
- Locations provide a specific chemical environment and interaction partners necessary to fulfil the protein's function
- Altered protein localization can lead to cellular dysfunction and disease

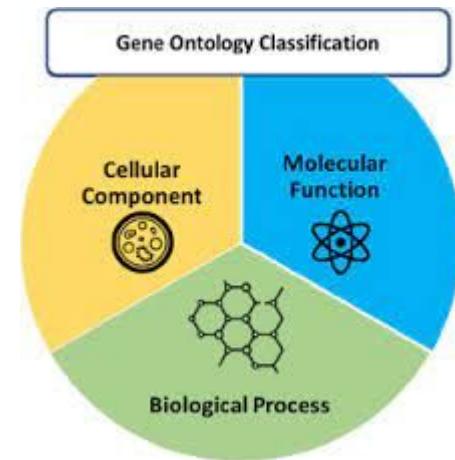


Adapted from Thul et al., Science 356, eaad3321 (2017)

Copyright © Jackom/Getty Images

Protein function and localization

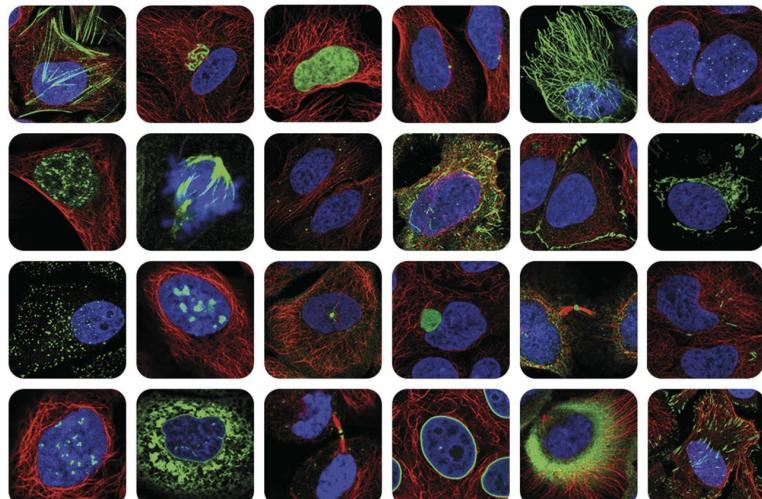
- Together with **Biological Process** and **Molecular Function**, **Cellular Component** is one of the three aspects describing protein function in the **Gene Ontology (GO)**
- Knowledge of protein localization allows the identification of potential protein interactors (participating in the same biological process)
- Applications also in **drug discovery**: plasma membrane or cell surface exposed proteins are good candidates as drug targets
- Bacterial exposed proteins are also interesting for **vaccine development**



Experimental methods

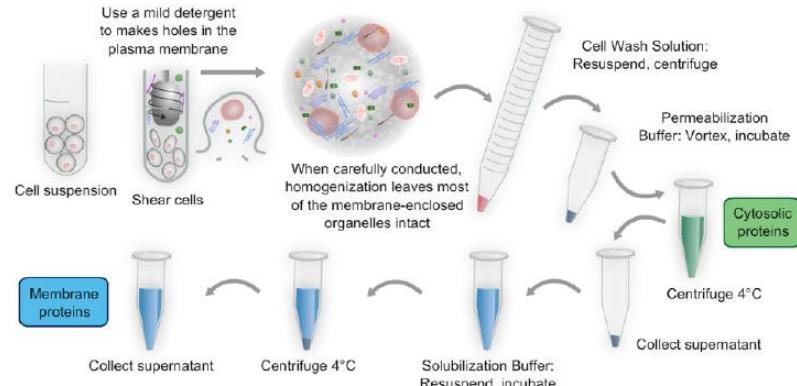
Subcellular fractionation

Suited for membrane-bound organelles



Subcellular Fractionation

- Optimization of centrifugation and detergent-buffer formulations: cleaner and easier separation of subcellular structures
- Hydrophobic membrane proteins can be solubilized and separated from hydrophilic proteins
- Isolate intact nuclei, mitochondria, and other organelles



Immunofluorescence images

Depends on antibody availability

RESEARCH

RESEARCH ARTICLE SUMMARY

PROTEOMICS

A subcellular map of the human proteome

Peter J. Thul,¹ Lovisa Åkesson,¹ Mikaela Wiklund,² Diana Mahdedian,¹ Alka Patel,¹ Gert Käll,¹ Hans Ekström,¹ Daniel Karlsson,¹ Anna-Lena Karlsson,¹ Birgitta Liss,¹ Maria Åberg,¹ Åsa Bläckström,¹ Pär Daniellsson,¹ Linn Fagerberg,¹ Jenny Fall,¹ Laurent Gatto,¹ Christian Giannai,¹ Sophia Höber,¹ Martin Hjelmåsén,¹ Freddie Johansson,¹ Susanne Lee,¹ Cecilia Lindskog,¹ Jan Mutter,¹ Claire M. Mufvey,¹ Peter Nilsson,¹ Per Olofsson,¹ John Olofsson,¹ Kristina Schermer,¹ Magnus Söderström,¹ Åsa Stålhammar,¹ Evelina Svartberg,¹ Marie Sköog,¹ Charlotte Ståhl,¹ Drevin P. Sallustio,¹ Åsa Engd,¹ Casper Wimmes,¹ Cheng Zhang,¹ Martin Zváral,¹ Adil Marzinel,¹ Fredrik Pontén,¹ Kalle von Feilitzen,¹ Kathryn S. Lilley,¹ Mathias Uhlen,¹ Emma Lundberg¹

Applied to 12,003 human proteins



Human Protein Atlas
<https://www.proteinatlas.org/>

SL annotation in UniProtKB



Statistics on the Subcellular Location feature taken from UniProt release 2022_03

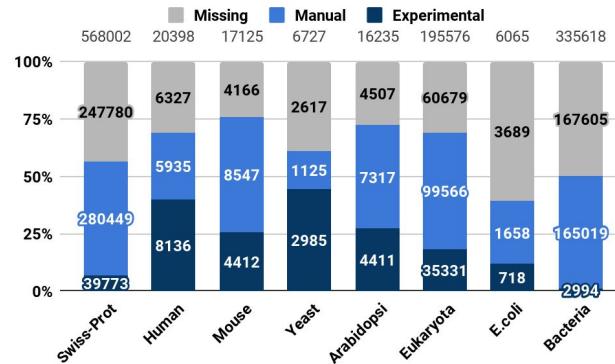
Swiss-Prot (568,002)



Manually annotated and reviewed.

Records with information extracted
from literature and curator-evaluated
computational analysis.

SL annotation in Swiss-Prot

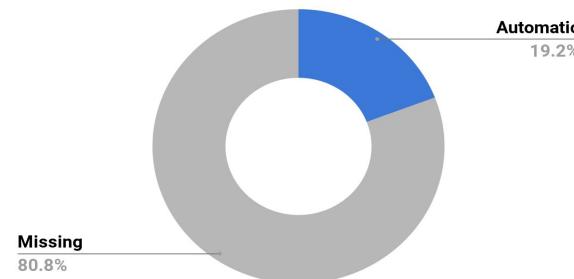


TrEMBL (226,771,949)

Automatically annotated and not
reviewed.

Records that await full manual
annotation.

SL annotations in TrEMBL



Mechanisms of protein sorting

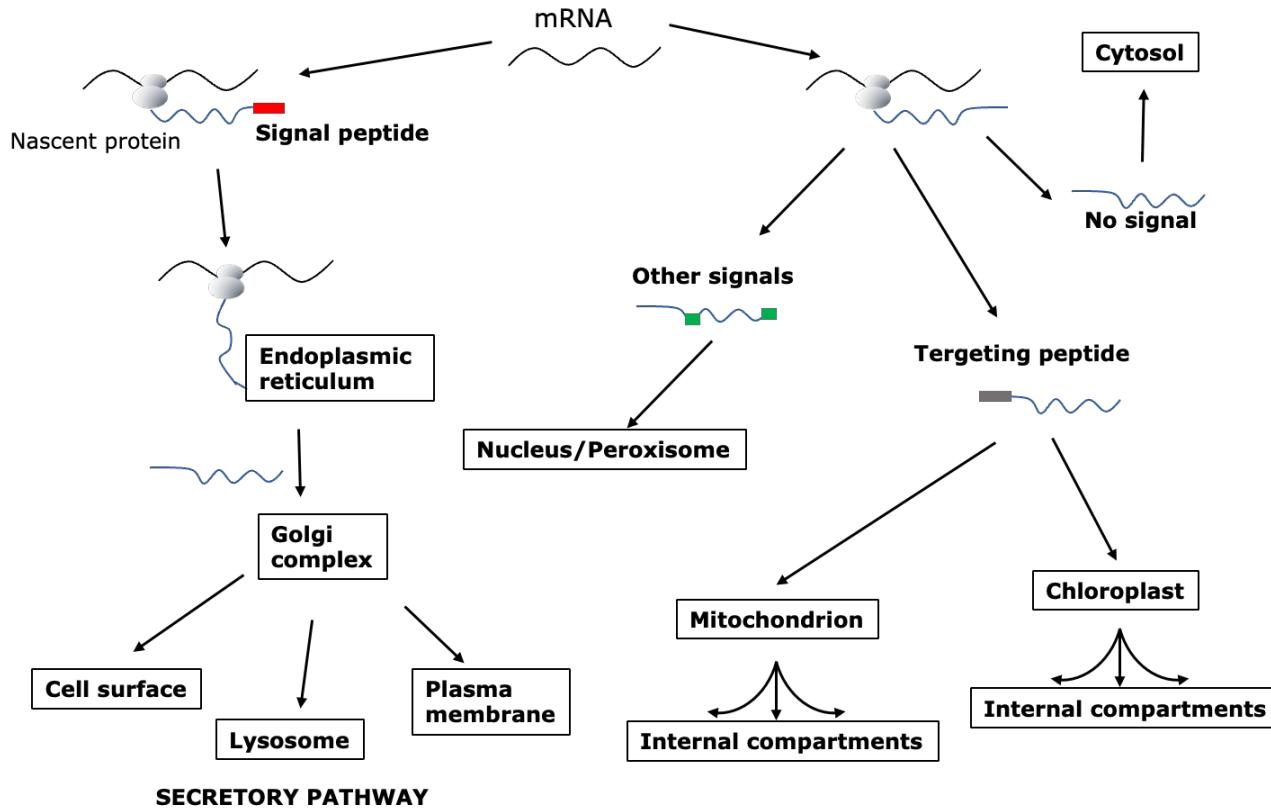
Mainly mediated by protein-protein and protein-membrane interactions

Well-characterized mechanisms based on molecular recognition of sorting/targeting signals:

- Signals contiguous in sequence
 - Signal peptides (secretory pathway)
 - Organelle-targeting peptides (mitochondria, chloroplasts, peroxisomes)
 - GPI-anchors (plasma membrane)
 - TM regions (plasma membrane, organelle membranes)
- Motifs non-contiguous in sequence
 - Nuclear localization signals
- Other signals/mechanisms
 - Non-canonical secretion, exosomes

Protein compositions reflect the different physico-chemical features of compartments they are located in

Schematic overview of protein sorting



In-silico prediction of protein SL

Available methods can be divided in two main classes:

Homology-based approaches

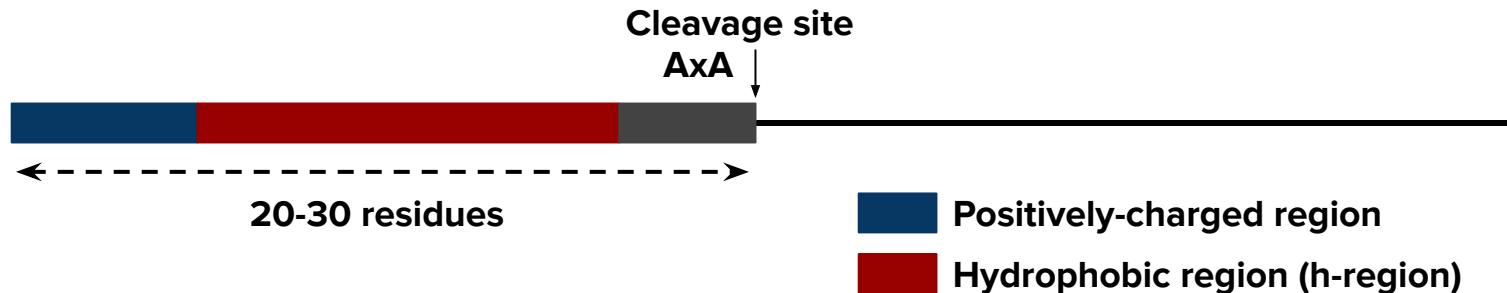
- Transfer SCL from closely-related sequences
- Can be very accurate but require a good template

Algorithmic and Machine-learning approaches

- Methods detecting specific sorting signals
- Multi-category SL predictors exploiting protein global features

Signal Peptide (SP) prediction

- Recognized to direct the protein toward the **secretory pathway**
- Modular “structure” comprising three separate **regions with different properties**
- **Hydrophobic core** similar to a transmembrane helix
- Weakly conserved **sequence motifs** can be observed **at cleavage sites**



Two distinct prediction tasks:

1. **Discrimination/detection:** Detect the presence of the SP sequence
2. **Labelling:** Identification of the precise position of the cleavage site along the sequence

SP prediction: challenges

SP show hydrophobic profiles that are very similar to those of N-terminal transmembrane alpha helices (TMAH)

A major challenge for prediction methods is to properly distinguish true SP from N-terminal TMAH

Moreover, the correct localization of the cleavage site is still problematic

Methods are able to detect the presence of the signal, but fail in detecting the correct localization of the site

Classes of methods for SP prediction

- Weight matrix approaches
- Feature-based approaches
- Artificial Neural Networks (including recent deep-learning methods)
- Hidden Markov Models and other probabilistic approaches
- Support Vector Machines
- Homology-based approaches

History of SP prediction: first method

- The first method was introduced by Gunnar von Heijne in 1983
- Based on a reduced-alphabet weight matrix combined with a rule for narrowing the search region
 - Only seven different weights at each position, corresponding to groups of AAs with similar properties
- The weights of the matrix were estimated manually rather than using an automatic procedure to estimate from data
- The matrix score aimed at recognizing the location of the SP cleavage site
 - Cleavage site assigned to position with the highest score
- The scores were computed for positions 12-20 counted from the beginning of the h-region
 - Defined as the first quadruplet of AAs with at least three hydrophobic residues
- In the first benchmark the procedure correctly predicted 92% of sites on data used to estimate it
- On a later benchmark the test performance was only 64%
 - Overfitting is also possible for simple methods such as weight matrices

An updated weight-matrix approach

- A updated version of the first method introduced by von Heijne in 1986
- A “real” weight matrix approach using log-odds scores and estimation from data
- Introduction of “pseudocounts” in weights calculation for mitigating the “sampling error” issue: the distribution is estimated from a limited number of examples
- The first 40/50 positions are scanned for finding the cleavage-site
- **This method will be implemented in this project**
 - We will describe it in more details later

Feature-based approaches

Basic concepts:

- Define a set of feature descriptors that are potentially conducive to the recognition of the signal peptide
- Use a classifier (e.g. SVM or Random Forest) to discriminate SP from non-SP proteins

During this project we will implement a feature-based approach using different sequence feature descriptors and SVMs

The first feature-based approach

- Published in 1985 by McGeoch
- Tested different sequence-derived features to find a combination discriminating between SPs and other sequences
- No cleavage-site prediction was implemented
- Two features identified:
 - Length of charged region: defined as starting at the first charged residue after the first 11 positions and ending at the next uncharged residue
 - Maximal hydrophobicity in a window of 8-AA (using Kyte-Doolittle scale): calculated 18 positions downstream the from the start of uncharged region
- Non-linear discriminative function, separating positive (SP) from negative (non-SP) examples estimated manually

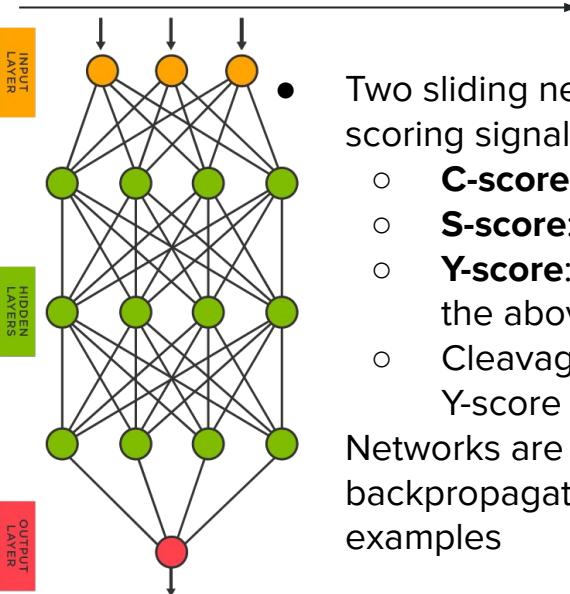
Neural networks for SP prediction

- The most successful framework for predicting SPs and their cleavage site
- Early approaches used “shallow networks” in two flavours
 - Using a fixed-size portion of the N-terminus (e.g. the first 20 residues) as input to the NN
 - Using a sliding network predicting both presence of the SP and cleavage-site position
 - **SignalP1-4**: developed at DTU in 1997, 1999, 2004 and 2011, use a sliding NN
 - **SPEPLip**: the first NN-based method introduced by our group in 2003
- Novel approaches have progressively evolved toward the use of more complex and deeper network architectures
 - **DeepSig**: the first method using deep learning, described by our group in 2018
 - **SignalP5-6**: the most recent versions of SignalP adopting deep architectures

Early NN-based approaches: SignalP4

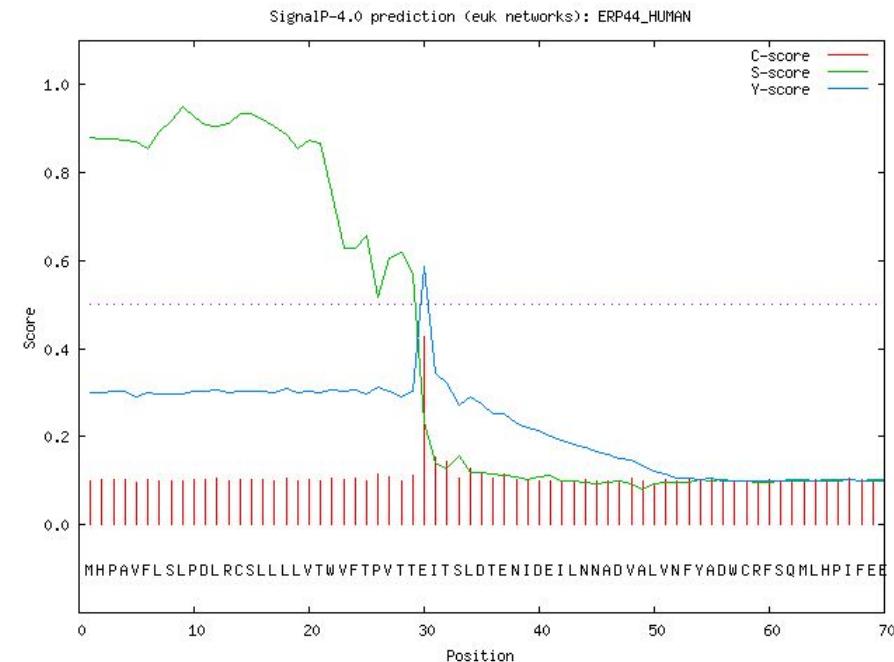
Input sequence (encoded using standard one-hot encoding)

MHPAVFLSLPDLRCSLLLLVTWVFTPVTTEITSLDTENIDEILNNADVALVNFYADWCRCFSQMLHPIFE

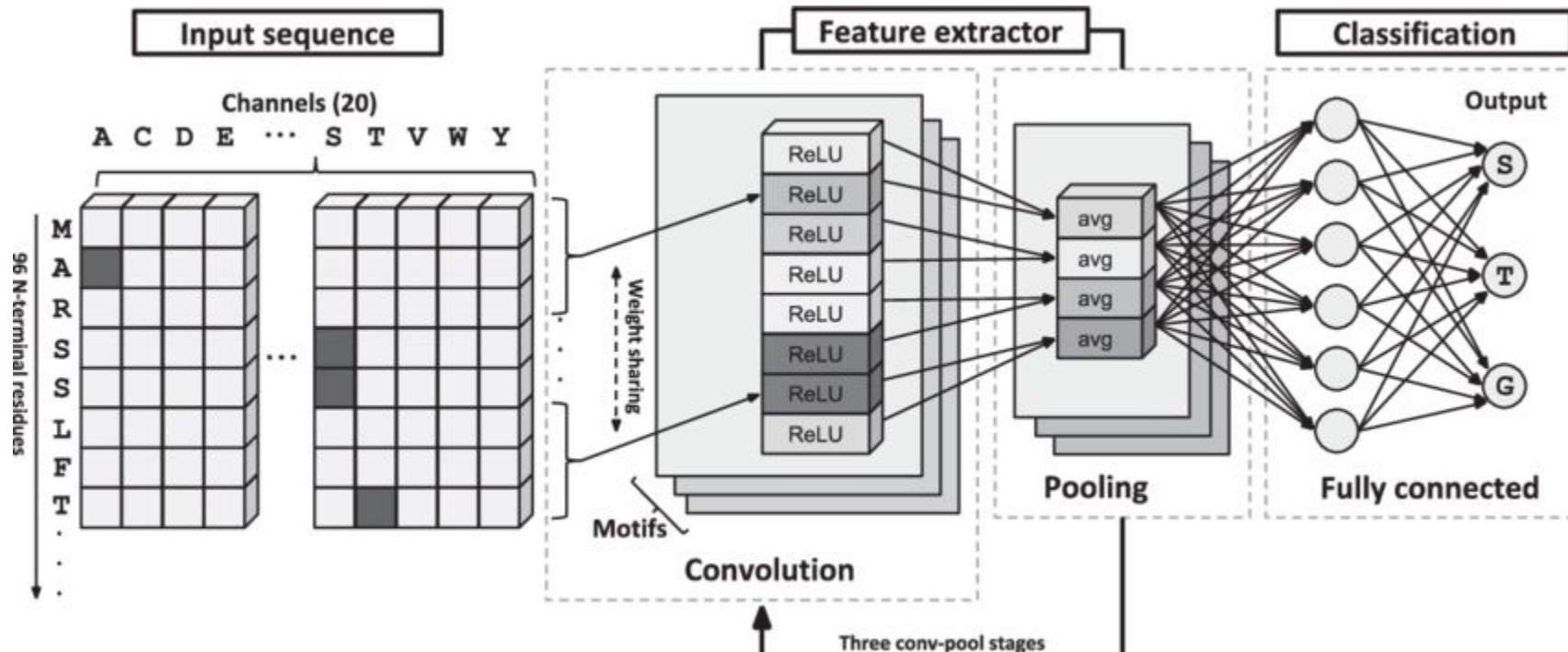


- Two sliding networks producing scoring signals along the sequence:
 - C-score:** cleavage-site score
 - S-score:** SP score
 - Y-score:** geometric average of the above scores
 - Cleavage site predicted where Y-score is maximal

Networks are trained using standard backpropagation on large set of examples

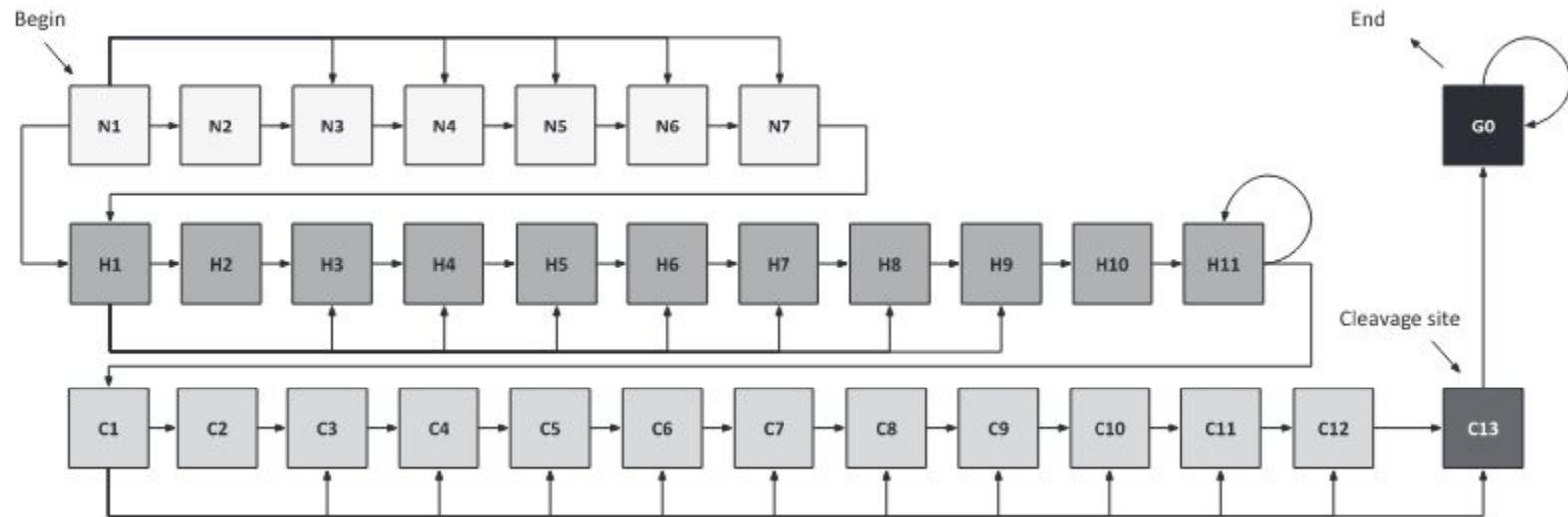


Deep-learning based approaches: DeepSig

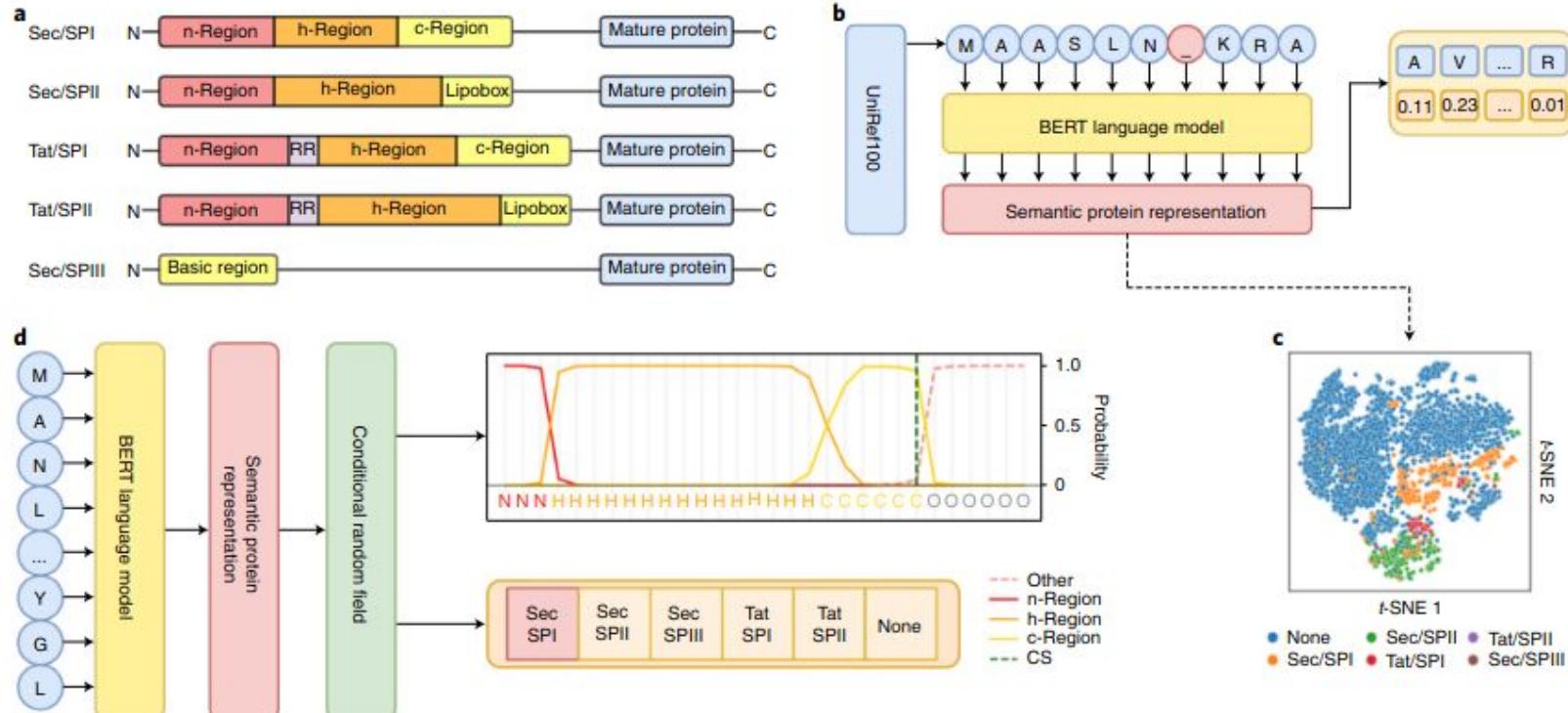


DeepSig: cleavage-site prediction

DeepSig adopts a Conditional Random Field (similar to an HMM) to assign position of the cleavage site



SignalP6



Teufel, F., Almagro Armenteros, J. J., Johansen, A. R., Gislason, M. H., Pihl, S. I., Tsirigos, K. D., Winther, O., Brunak, S., von Heijne, G., & Nielsen, H. (2022). **SignalP 6.0 predicts all five types of signal peptides using protein language models.** *Nature biotechnology*, 40(7), 1023–1025.

Support Vector Machines and SP

- Less popular than NNs for SP detection and labelling
- A few approaches use a special type of kernel function called **string kernel**
 - A kernel evaluating the similarity between protein sequences (or fragments) on the basis of the composition in term of short k-mers
- Other methods adopts standard one-hot encoding and N-terminal global AA composition to discriminate SP from non-SP

Vert JP. **Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings.** Pac Symp Biocomput. 2002:649-60.

Cai YD, Lin SL, Chou KC. **Support vector machines for prediction of protein signal sequences and their cleavage sites.** Peptides. 2003 Jan;24(1):159-61.

Wang M, Yang J, Chou KC. **Using string kernel to predict signal peptide cleavage site based on subsite coupling model.** Amino Acids. 2005 Jun;28(4):395-402.

Nugent T, Jones DT. **Transmembrane protein topology prediction using support vector machines.** BMC Bioinformatics. 2009 May 26;10:159.

Summary

- Early SP detection methods adopted weight matrices or simple feature extraction
- Not all methods predicts cleavage sites
- Neural networks are the most successful approaches for SP prediction
- Deep learning is nowadays the gold standard
- SVMs have been used in combination with string kernels
- HMMs/CRFs are used to refine predictions according to the well-known structure of the SP and identify cleavage sites

References

1. Almagro Armenteros,J.J. et al. (2019) SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat Biotechnol*, **37**, 420–423.
2. Bendtsen,J.D. et al. (2005) Prediction of twin-arginine signal peptides. *BMC Bioinformatics*, **6**, 167.
3. Berks,B.C. (1996) **A common export pathway for proteins binding complex redox cofactors?** *Molecular Microbiology*, **22**, 393–404.
4. Cai,Y.-D. et al. (2003) Support vector machines for prediction of protein signal sequences and their cleavage sites. *Peptides*, **24**, 159–161.
5. Chou,K.-C. (2001a) Prediction of signal peptides using scaled window. *Peptides*, **22**, 1973–1979.
6. Chou,K.-C. (2001b) Using subsite coupling to predict signal peptides. *Protein Engineering, Design and Selection*, **14**, 75–79.
7. Chou,K.-C. and Shen,H.-B. (2007) Signal-CF: A subsite-coupled and window-fusing approach for predicting signal peptides. *Biochemical and Biophysical Research Communications*, **357**, 633–640.
8. Cristobal,S. (1999) Competition between Sec- and TAT-dependent protein translocation in Escherichia coli. *The EMBO Journal*, **18**, 2982–2990.
9. Dyrlov Bendtsen,J. et al. (2004) Improved Prediction of Signal Peptides: SignalP 3.0. *Journal of Molecular Biology*, **340**, 783–795.
10. Fariselli,P. et al. (2003) SPEPlip: the detection of signal peptide and lipoprotein cleavage sites. *Bioinformatics*, **19**, 2498–2499.
11. Heijne,G. (1983) Patterns of Amino Acids near Signal-Sequence Cleavage Sites. *Eur J Biochem*, **133**, 17–21.
12. Heijne,G. (1986) The distribution of positively charged residues in bacterial inner membrane proteins correlates with the trans-membrane topology. *EMBO J*, **5**, 3021–3027.

References

12. Hiller,K. et al. (2004) PrediSi: prediction of signal peptides and their cleavage positions. *Nucleic Acids Research*, **32**, W375–W379.
13. Juncker,A.S. et al. (2003) Prediction of lipoprotein signal peptides in Gram-negative bacteria. *Protein Sci.*, **12**, 1652–1662.
14. Nielsen,H. et al. (1997) Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering Design and Selection*, **10**, 1–6.
15. Nielsen,H. and Krogh,A. (1998) Prediction of signal peptides and signal anchors by a hidden Markov model. *Proc Int Conf Intell Syst Mol Biol*, **6**, 122–130.
16. Nugent,T. and Jones,D.T. (2009) Transmembrane protein topology prediction using support vector machines. *BMC Bioinformatics*, **10**, 159.
17. Petersen,T.N. et al. (2011) SignalP 4.0: discriminating signal peptides from transmembrane regions. *Nat Methods*, **8**, 785–786.
18. Savojardo,C. et al. (2018) DeepSig: deep learning improves signal peptide detection in proteins. *Bioinformatics*, **34**, 1690–1696.
19. Schneider,G. and Wrede,P. (1993) Development of artificial neural filters for pattern recognition in protein sequences. *J Mol Evol*, **36**, 586–595.
20. Schneider,G. and Wrede,P. (1994) The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site. *Biophysical Journal*, **66**, 335–344.

References

21. Teufel,F. et al. (2022) SignalP 6.0 predicts all five types of signal peptides using protein language models. *Nat Biotechnol*, **40**, 1023–1025.
22. Thul,P.J. et al. (2017) A subcellular map of the human proteome. *Science*, **356**, eaal3321.
23. Vert,J.-P. (2001) SUPPORT VECTOR MACHINE PREDICTION OF SIGNAL PEPTIDE CLEAVAGE SITE USING A NEW CLASS OF KERNELS FOR STRINGS. In, *Biocomputing 2002*. WORLD SCIENTIFIC, Kauai, Hawaii, USA, pp. 649–660.
24. Wang,M. et al. (2005) Using string kernel to predict signal peptide cleavage site based on subsite coupling model. *Amino Acids*, **28**, 395–402.

LB2 Project

Data collection

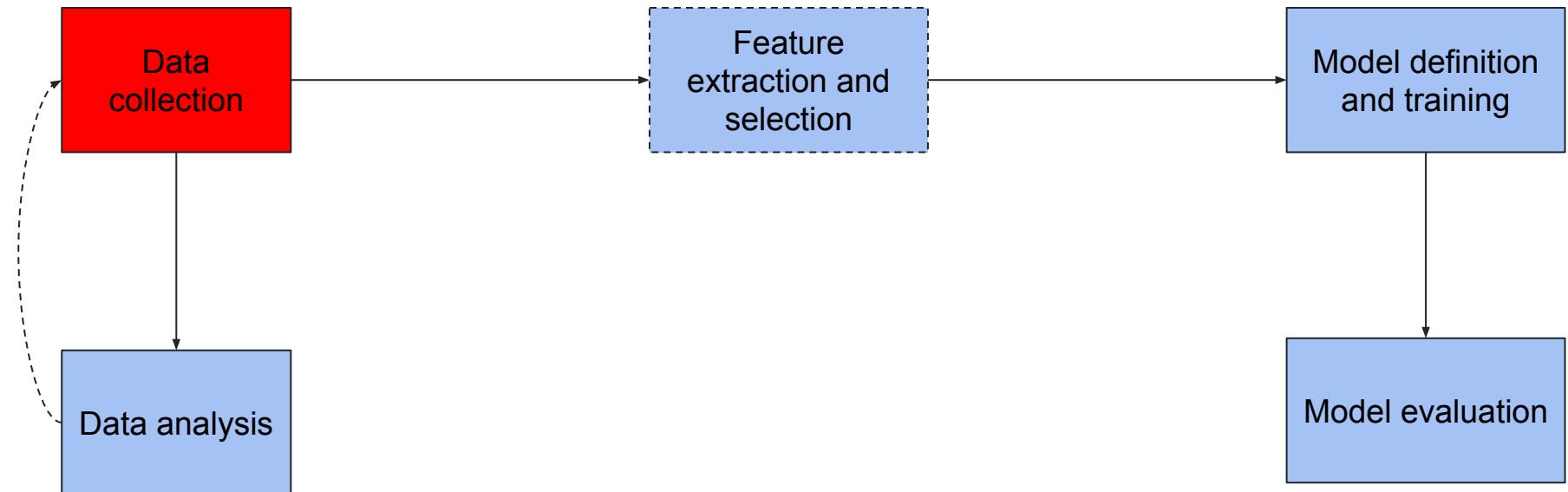
Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project setting

You have to:

- **Retrieve a dataset from UniProtKB**
- Prepare the dataset for cross-validation and benchmarking
- Perform and visualize statistics on both datasets
- Prepare your data for training and prediction -> feature extraction
- Implement the von Heijne algorithm
- Implement the SVM-based approach (using sklearn)
- Perform experiments (in cross-validation and blind test) and discuss results
- (Optional) Implement and test different solutions
- Write the manuscript

Machine-learning workflow



Data collection

- The first step of any machine-learning workflow
- Data are collected from domain-specific databases (e.g. UniProtKB for protein sequences) according to different criteria
 - **Quality** (e.g. experimental evidence for the feature of interest)
 - **Redundancy**: collected data (e.g. protein sequences) must be as diverse as possible to properly cover the domain of interest
 - **Secondary data and metadata**: additional information useful for contextualizing the primary data
- A preliminary processing is performed at this step e.g. for generating training/testing datasets and cross-validation splits

How to retrieve data from UniProt

- Using UniProt website advanced search
 - Easy to use directly on the UniProt website
 - Filter data according to different criteria
 - Results can be downloaded in different formats, including TSV, JSON, FASTA, XML
 - Complex filtering steps need to be applied after downloading data
- Using the UniProt REST Application Programming Interface (API)
 - Data are accessed, filtered, formatted and downloaded programmatically
 - The entire collection process can be entirely handled in a single Python script/notebook
 - Improve reproducibility of the process
 - Allows performing complex filtering steps that are not directly supported using the website
 - Allows more control in the output format of the data

APIs

From Wikipedia:

*An **application programming interface (API)** is a way for two or more computer programs or components to communicate with each other. It is a type of software interface, offering a service to other pieces of software*

***Web APIs** are a service accessed from client devices (mobile phones, laptops, etc.) to a Web server using the Hypertext Transfer Protocol (HTTP). Client devices send a request in the form of an HTTP request, and are met with a response message usually in JavaScript Object Notation (JSON) or Extensible Markup Language (XML) format.*

Developers typically use Web APIs to query a server for a specific set of data from that server.

UniProt APIs

UniProt provides several application programming interfaces (APIs) to query and access its data programmatically:

UniProt website REST API

What: RESTful URLs that can be bookmarked, linked and used in programs for all entries, queries and tools available through this website. Data is available in all formats provided on the website, e.g. text, XML, RDF, FASTA, GFF, tab-separated for UniProtKB protein data.

Why: Access data and tools from the UniProt website with any programming language.

Documentation: <https://www.uniprot.org/help/api>

Proteins REST API

What: Extended REST API with a service providing [genomic coordinates](#) of UniProtKB sequences, and other services providing annotations imported and mapped from Large Scale data Sources (LSS), such as 1000Genomes, ExAC, PeptideAtlas, and HPA via the [variation](#), [proteomics](#), and [antigen](#) services.

Why: Access data sets mapped to UniProt and integrated through a single service.

Documentation: <https://www.ebi.ac.uk/proteins/api/doc>

UniProt SPARQL API

What: [SPARQL](#) API for all UniProt data, stored in Resource Description Framework (RDF) format ([Help](#)). An SQL-like graph query language that allows to perform complex queries across all UniProt data, as well as across other resources that provide a SPARQL endpoint, such as Ensembl or Wikidata.

Why: Access data from UniProt, and other resources, using a low-cost alternative to importing the data into e.g. a relational database and building a local data warehouse.

Documentation: <https://sparql.uniprot.org>

UniProt Java API

What: A Java library that provides a stable remote API for programmatically accessing UniProt data.

Why: Access data and tools from UniProt using Java.

Documentation: <https://www.ebi.ac.uk/uniprot/japi/>

UniProt website REST API

Offer API endpoints to retrieve:

- UniProtKB individual entries in different formats
- Multiple UniProtKB entries via queries

All the UniprotProtKB database is accessible, including:

- UniProt (SwissProt and TrEMBL)
- UniRef
- UniParc
- Taxonomies
- Subcellular locations
- etc...

Programmatic access - Retrieving individual entries

The web address for an entry consists of a data set name (e.g. uniprot, uniref, uniparc, taxonomy,...) and the entry's unique identifier, e.g.:

```
https://www.uniprot.org/uniprotkb/P12345
```

By default, a web page is returned. Depending on the data set, other formats may also be available (click on "Formats" on the entry's web page). Here are some examples:

```
https://rest.uniprot.org/uniprotkb/P12345.txt
```

```
https://rest.uniprot.org/uniprotkb/P12345.xml
```

```
https://rest.uniprot.org/uniprotkb/P12345.rdf
```

```
https://rest.uniprot.org/uniprotkb/P12345.fasta
```

```
https://rest.uniprot.org/uniprotkb/P12345.gff
```

```
https://rest.uniprot.org/uniref/UniRef90_P99999.xml
```

```
https://rest.uniprot.org/uniref/UniRef90_P99999.rdf
```

```
https://rest.uniprot.org/uniref/UniRef90_P99999.fasta
```

```
https://rest.uniprot.org/uniref/UniRef90_P99999.tsv
```

```
https://rest.uniprot.org/uniparc/UPI000000001F.xml
```

```
https://rest.uniprot.org/uniparc/UPI000000001F.rdf
```

```
https://rest.uniprot.org/uniparc/UPI000000001F.fasta
```

```
https://rest.uniprot.org/uniparc/UPI000000001F.tsv
```

Example usage (Bash):

```
curl https://rest.uniprot.org/uniprotkb/P12345.json > P12345.json
```

Return Status Codes

The following **status codes** ↗ may be returned:

| Code | Description |
|------|--|
| 200 | The request was processed successfully. |
| 400 | Bad request. There is a problem with your input. |
| 404 | Not found. The resource you requested doesn't exist. |
| 410 | Gone. The resource you requested was removed. |
| 500 | Internal server error. Most likely a temporary problem, but if the problem persists please contact us. |
| 503 | Service not available. The server is being updated, try again later. |

Programmatic access - Retrieving entries via queries

Using specific API endpoints, it is possible to run complex queries on the databases to retrieve one or more entries satisfying specific search parameters

Two endpoints can be used:

- **stream**: return results all together as a continuous stream of data. Limited to 10M entries, can lead to memory issues
- **search**: return results in batches. More efficient than stream and the only way to retrieve very large sets of data

In both cases, the basic URL for a query has the following syntax:

The main API URL

`https://rest.uniprot.org/uniprotkb/search?query=(reviewed:true)%20AND%20(organism_id:9606)`

The API endpoint, either search or stream

The UniProt dataset name you are actually interrogating, e.g. uniprotkb, uniref ect...

The query with search parameters

Other supported parameters

| Parameter | Values | Description |
|----------------|--|---|
| query | <i>string</i> | See query syntax and query fields for UniProtKB . An empty query string will retrieve all entries in a data set. Tip: Refine your search by clicking Advanced in the search bar. |
| format | See section, "What formats are available?" | See section, "What formats are available?" |
| fields | comma-separated list of column names | Columns to retrieve in the results. Applies to <code>tsv</code> , <code>xslx</code> and <code>json</code> formats only. (For UniProtKB you can also read the full list of UniProtKB column names). |
| includeIsoform | <code>true</code> or <code>false</code> | Whether or not to include isoforms in the search results. <i>Note:</i> Only applies to UniProtKB searches. |
| compressed | <code>true</code> or <code>false</code> | Return results gzipped. Note that if the client supports HTTP compression, results may be compressed transparently even if this parameter is not set to <code>true</code> . |
| size | <i>integer</i> | Maximum number of results to retrieve. <i>Note:</i> Only takes effect on searches. |
| cursor | <i>string</i> | Specifies the cursor position in the entire result set, from which returned results will begin. Cursors are used to allow paging through results. Typically used together with the <code>size</code> parameter. |

Change output format

Query output format can be changed. Default is JSON

`https://rest.uniprot.org/uniprotkb/search?query=(organism_id:9606) &format=tsv`

List of all formats

| Description | Accept Header Media Type | Format Parameter |
|--|-----------------------------|------------------|
| JavaScript Object Notation (JSON) | application/json | json |
| Extensible Markup Language (XML) | application/xml | xml |
| Text file representation | text/plain; format=flatfile | txt |
| List of one or more IDs | text/plain; format=list | list |
| Tab-Separated-Values | text/plain; format=tsv | tsv |
| FASTA: a text-based format representing nucleotide / peptide sequences | text/plain; format=fasta | fasta |
| Genomic Feature Format (GFF) | text/plain; format=gff | gff |
| Open Biomedical Ontologies (OBO) | text/plain; format=obo | obo |
| Resource Description Framework (RDF) | application/rdf+xml | rdf |
| Excel | application/vnd.ms-excel | xlsx |

Results in TSV format



Only include specific fields in the output

Can be only done when format is json, tsv or xslx

`https://rest.uniprot.org/uniprotkb/search?query=\(organism_id:9606\)&fields=accession,cc_function`

The list of all available fields can be found here:

`https://www.uniprot.org/help/return_fields`



Only return
accession and
function in the
output

Use pagination to retrieve large datasets in batches

- Implemented using the size and cursor parameters
- Suppose you want to retrieve all human proteins (204411) in batches of 500 entries
- First search call retrieve the first 500 entries:

```
https://rest.uniprot.org/uniprotkb/search?query=(organism_id:9606)&size=500 [This will also return a HTTP Header]
```

- Next calls will be done specifying a cursor field to get the subsequent batches:

```
https://rest.uniprot.org/uniprotkb/search?query=(organism_id:9606)&cursor=c9bacmxsqhkqgdwarf0s7k7ocgdxh2ipmxgs4&size=500
```

```
https://rest.uniprot.org/uniprotkb/search?query=(organism_id:9606)&cursor=28m7xk8oeej15mhh124urq40qqp62dwsnzwslro&size=500
```

...

- The next cursor to be specified (actually, the entire URL) is obtained from the HTTP HEADER associated to the previous API call

Python generators

Paginated search makes use of Python generator functions

See examples here:

<https://colab.research.google.com/drive/1cR3AQt46uR8QvoFOS2PMMyLhxXBJDCrq?usp=sharing>

Running API calls in Python

Steps:

1. Got the UniProt website and perform the required Advanced search
2. Click Download in the toolbar
3. Click Generate URL for API
4. Under the header API URL using the searching endpoint click Copy to get the URL which will be used in the example

See the following Colab:

[https://colab.research.google.com/drive/1RG_BQVA1kUmhmSfgoOf3IZ3XZ_A4jvK7
?usp=sharing](https://colab.research.google.com/drive/1RG_BQVA1kUmhmSfgoOf3IZ3XZ_A4jvK7?usp=sharing)

A more complex example (in Python)

<https://colab.research.google.com/drive/1BqDHnCEqzaxpHwYAVyPzOcZYIYGHTyDk?usp=sharing>

Search for all human protein with the following criteria:

- Reviewed proteins
- Protein is not a fragment
- Protein has a coiled-coil region in the first 100 residues

Store the data in TSV format with the following four fields:

- Protein accession
- Protein length
- Start position of the first coiled-coil segment
- End position of the first coiled-coil segment

What we need to retrieve for our project

- Our goal is to retrieve datasets for training/testing methods for predicting the presence of signal peptides in proteins
- **We'll focus only on Eukaryotes, neglecting bacterial and archaea proteins**

To this aim, we need:

- A positive dataset comprising eukaryotic sequences endowed with experimentally determined SP sequences
- A negative dataset comprising eukaryotic sequences without a SP

Signal Peptides in UniProt

- The presence of the SP is reported under the “**PTM-Processing/Molecule Processing/Signal peptide**” feature on each UniProt entry
- The SP annotation is usually accompanied with an evidence code from Evidence and Conclusion Ontology (ECO)
 - Evidence EC0:0000269 is used for experimental evidence supported by one or more scientific publications
 - Other codes used in UniProt can be found on the dedicated page:
<https://www.uniprot.org/help/evidences>
- Besides SP presence, also SP cleavage site is reported
 - When it is clear that a protein is cleaved, but direct protein sequencing has not been used to determined the precise cleavage position, UniProt use a question mark instead of a position
 - In some rare cases, signal sequences are not cleaved. This is indicated by the note 'Not cleaved' in the feature description.
- Example protein: <https://www.uniprot.org/uniprotkb/O00300/entry>

Signal Peptides in UniProt: JSON format

```
entryType: "UniProtKB reviewed (Swiss-Prot)"  
primaryAccession: "O00300"  
secondaryAccessions: [...]  
uniProtKBId: "TR11B_HUMAN"  
entryAudit: {...}  
annotationScore: 5  
organism: {...}  
proteinExistence: "1: Evidence at protein level"  
proteinDescription: {...}  
genes: [...]  
comments: [...]  
features:  
  0:  
    type: "Signal"  
    location:  
      start:  
        value: 1  
        modifier: "EXACT"  
      end:  
        value: 21  
        modifier: "EXACT"  
    description: ""  
    evidences:  
      0:  
        evidenceCode: "ECO:0000269"  
        source: "PubMed"  
        id: "15340161"  
      1:  
        evidenceCode: "ECO:0000269"  
        source: "PubMed"  
        id: "9571159"
```

<https://rest.uniprot.org/uniprotkb/O00300.json>

Reported under the “features” field

Start position is always 1

End position, can be also “?” when cleavage site is not known

Annotation evidence code (ECO) is reported here. Many evidences can be present

Not cleaved SP

```
entryType:          "UniProtKB reviewed (Swiss-Prot)"  
primaryAccession:   "O60721"  
secondaryAccessions: [...]  
uniProteinId:      "NCKX1_HUMAN"  
entryAudit:         {...}  
annotationScore:   5  
organism:          {...}  
proteinExistence:  "1: Evidence at protein level"  
proteinDescription: {...}  
genes:              [...]  
comments:           [...]  
features:  
  ▶ 0:             {...}  
  ▶ 1:  
    type:           "Signal"  
    ▶ location:  
      ▶ start:  
        value:       1  
        modifier:    "EXACT"  
      ▶ end:  
        value:       null  
        modifier:    "UNKNOWN"  
        description: "Not cleaved"  
    ▶ evidences:  
      ▶ 0:  
        evidenceCode: "ECO:0000269"  
        source:       "PubMed"  
        id:          "10608890"
```

<https://rest.uniprot.org/uniprotkb/O60721.json>

End position is “null” and description reports “Not cleaved”

How to select negative data

- Negative data must include proteins that do not have signal peptides
- Filter out proteins having SP annotated with ANY LEVEL OF EVIDENCE
- Check protein Subcellular Location:
 - Include only proteins annotated with experimental evidence into cellular compartments not related with the SP: **cytosol, nucleus, mitochondrion, plastid, peroxisome, cell membrane** (some membrane proteins may have SP)

Common selection criteria

For both positive and negative data:

- Protein evidence
- Protein length
- Protein annotation status (reviewed)
- Protein superkingdom (we will focus only on eukaryotic data)

For positive data:

- Signal peptide evidence (experimental)
- Knowledge of SP-cleavage site
- SP length > 13

For negative data:

- Absence of SP sequence (at any evidence level)
- Experimental evidence for non SP-related compartments

Output format

Both the positive and the negative dataset need to be stored in two different formats:

- A TSV file reporting relevant information about the proteins included in the dataset
- A FASTA file reporting the protein sequences

Information included in the TSV files is different for positive and negative data

Output TSV: positive data

TSV for the positive dataset must include:

1. The protein UniProt accession
2. The organism name
3. The Eukaryotic kingdom (Metazoa, Fungi, Plants, Other)
4. The protein length
5. The position of the signal peptide cleavage site

Output TSV: negative data

TSV for the positive dataset must include:

1. The protein UniProt accession
2. The organism name
3. The Eukaryotic kingdom (Metazoa, Fungi, Plants, Other)
4. The protein length
5. Whether the protein has a transmembrane helix starting in the first 90 residues (true or false)

Information about transmembrane helix presence is important to analyse prediction results during method evaluation, since SP are physico-chemically similar to transmembrane helices.

We expect a high false positive rate on transmembrane proteins

Practical session I (part A)

Collect the preliminary dataset from UniProt:

- Retrieve a preliminary dataset of positive data (i.e. sequences endowed with SP)
 - Select only eukaryotic proteins
 - Filter-out sequences shorter than 40 residues
 - Filter-out unreviewed proteins
 - Select on protein with experimental SP evidence
 - Filter out proteins with SP shorter than 14 residues
- Retrieve a preliminary dataset of negative data
 - Select only eukaryotic proteins
 - Filter-out sequences shorter than 40 residues
 - Filter-out sequences having SP (any evidence)
 - Select only proteins experimentally verified to be localized into: cytosol, nucleus, mitochondrion, plastid, peroxisome, cell membrane
- Store all data in TSV and FASTA formats
 - Format TSVs accordingly

LB2 Project

Data pre-processing

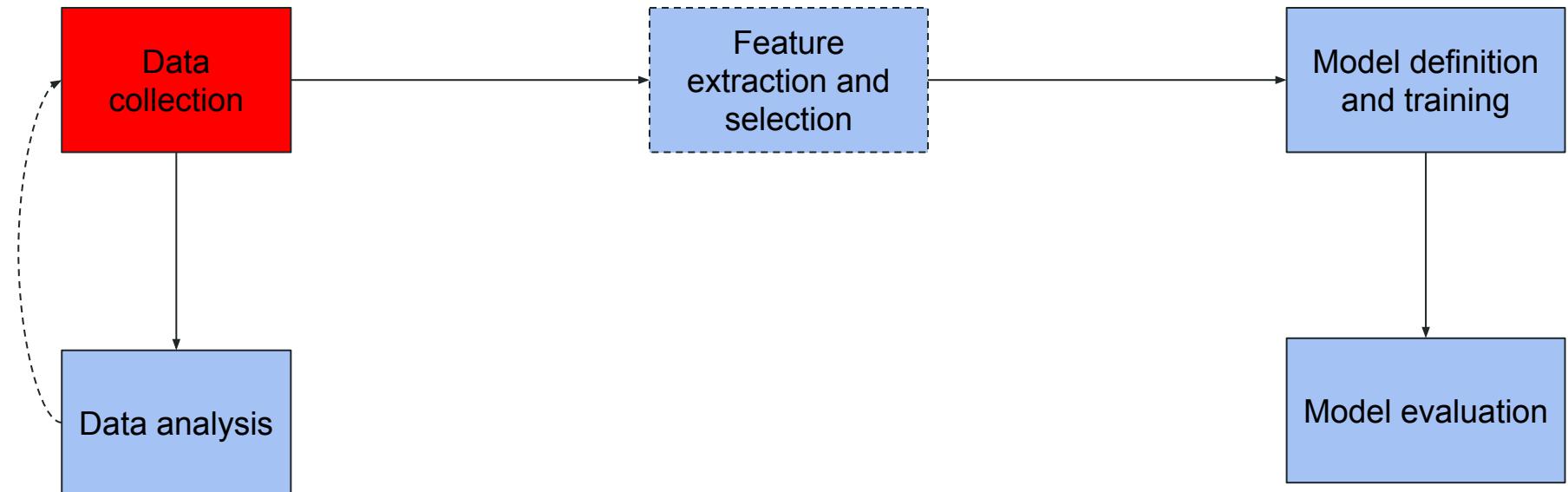
Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project setting

You have to:

- Retrieve a dataset from UniProtKB
- **Prepare the dataset for cross-validation and benchmarking**
- Perform and visualize statistics on both datasets
- Prepare your data for training and prediction -> feature extraction
- Implement the von Heijne algorithm
- Implement the SVM-based approach (using sklearn)
- Perform experiments (in cross-validation and blind test) and discuss results
- (Optional) Implement and test different solutions
- Write the manuscript

Machine-learning workflow



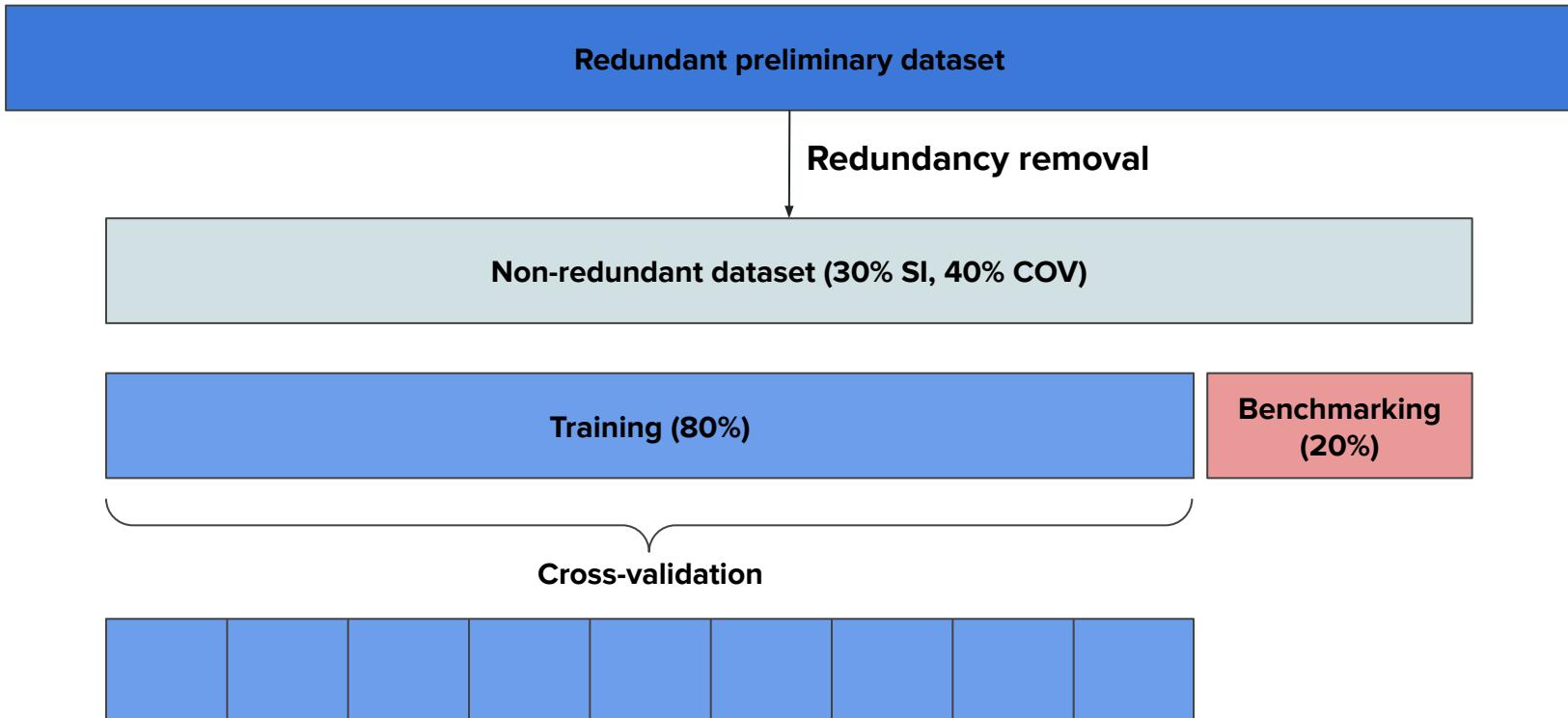
Building training and benchmarking datasets

- Once preliminary data have been collected, we need to split the dataset into two separate subsets:
 - **Training set:** used to train the methods, optimize model hyperparameters and perform cross-validation experiments
 - **Benchmarking set:** used to test the generalization performance of the different models
- Before splitting data, we need to obtain a non-redundant dataset
 - **Maximum pairwise sequence similarity is 30% with alignment coverage 40%**
 - Split can be then safely performed randomly
- Redundancy reduction can be performed using a clustering program e.g. MMSeqs2
- We can then select one representative sequence per cluster

Benchmarking set: motivation

- Also known as **holdout dataset**
- Cross-validation is not sufficient to estimate unbiased generalization performance
 - Model hyper-parameters are still optimized on the training set through cross-validation and grid-search
 - This may lead to some overfitting on training data
 - Using a holdout dataset we can better ensure the **never-seen-before condition**
- The model tested on the benchmarking set is the one going to production
 - Individual models generated during cross-validation runs are trained on slightly different subsets of the complete training dataset
- Generation of a holdout dataset is not trivial and require to meet specific criteria to avoid biased results
 - Particularly in the protein Bioinformatics field, sequence similarity issues must be carefully taken into consideration (for both cross-validation split and selection of a proper holdout set)

Dataset generation and data splits



Protein sequence clustering

Cluster sequences contained in the input FASTA file according to two combined criteria:

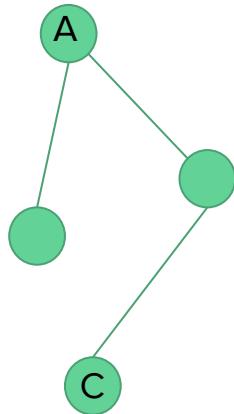
- Sequence identity
- Alignment coverage
 - This can be computed on both sequences or only on the shorter one

We will implement **single-linkage clustering**: begins with pairwise matches and then places a sequence in a cluster if the sequence matches at least one sequence already in the cluster

- Matches are valid if they meet the above criteria

Single-linkage clustering

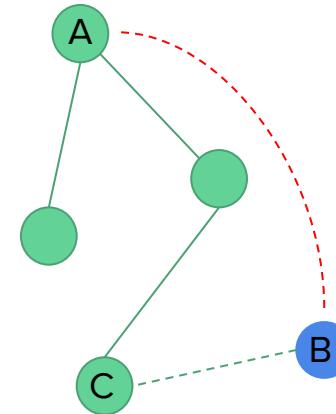
Cluster X



New node



New cluster X



Produces long thin clusters in which nearby elements of the same cluster have small distances, but elements at opposite ends of a cluster may be much farther from each other than two elements of other clusters i.e., **C is the closest node to B, so B is placed in Cluster X but A and B may be more distant than B and elements of another cluster**

Advantages: distantly-related homologs are placed in the same cluster

MMseqs2: an alternative to blustclust

<https://github.com/soedinglab/MMseqs2>

Fastest method available, it can be used to cluster large databases of sequences

Supports different clustering modes:

- **Greedy set cover**: tries to cover the database by as few clusters as possible. Almost all pairs of sequences in a cluster meet the clustering criteria, leading to more “connected” clusters
- **Connected components clustering** i.e., single-linkage clustering

MMseqs2: command line

```
mmseqs easy-cluster input.fa cluster-results tmp --min-seq-id 0.3 \
      -c 0.4 --cov-mode 0 --cluster-mode 1
```

Options are:

- Input fasta file: **input.fa**
- Output clusters prefix: **cluster-results**
- Temporary folder: **tmp**, created in the same directory
- Percentage sequence identity threshold: **--min-seq-id 0.3 (30% sequence identity)**
- Length coverage threshold: **-c 0.4 (40% coverage)**
- Coverage computation mode: **--cov-mode 0 (coverage of query and target)**
- Clustering mode: **--cluster-mode 1 (Connected component, i.e. single linkage)**

Many other options are available

MMSeqs2 output

Two main output files:

- **cluster-results_rep_seq.fasta**: a FASTA file containing all the representative sequences, one for each found cluster
- **cluster-results_cluster.tsv**: a TSV containing two columns:
 - Column 1: reports the ID of each sequence in the input file
 - Column 2: reports the ID of the representative sequence identifying the cluster the sequence in column 1 has been assigned to

Practical session I (part B)

Reduce data redundancy

- Cluster positive and negative sequences separately using MMSeqs or BLASTClust
 - Cluster sequences at 30% sequence identity and 40% of coverage
- Select representative sequences from each cluster
- Go back to the original TSV files and retain only representative sequences
 - Generate new TSVs comprising only selected proteins
- Separate training from benchmarking data
 - Randomly assign 80% of positive and negative sequences to training set
 - Assign the remaining 20% of positive and negative sequence to the benchmarking set
- Build 5-fold cross-validation subsets
 - Randomly split the training set into 5 different subsets, preserving the overall positive/negative ratio on each subset
 - Store information about the cross-validation subset each protein belongs to

LB2 Project

Data analysis

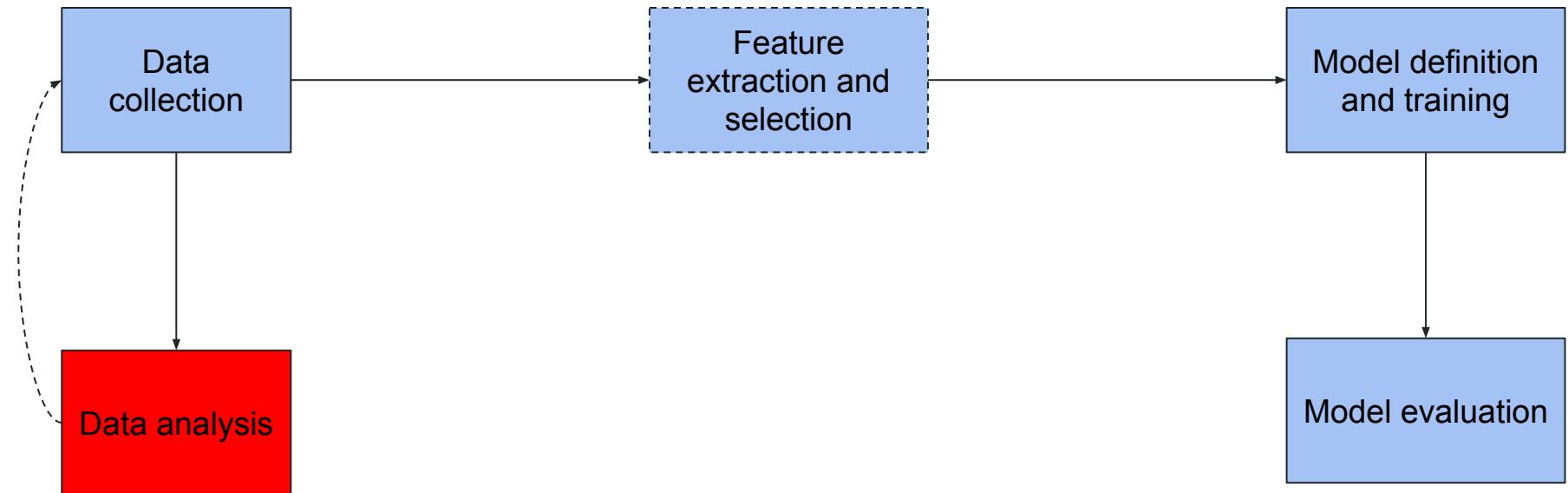
Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Project setting

You have to:

- Retrieve training and benchmarking datasets from UniProtKB
- Prepare the datasets for cross-validation
- **Perform and visualize statistics on both datasets**
- Prepare your data for training and prediction -> feature extraction
- Implement the von Heijne algorithm
- Implement the SVM-based approach (using sklearn)
- Perform experiments (in cross-validation and blind test) and discuss results
- (Optional) Implement and test different solutions
- Write the manuscript

Machine-learning workflow



Data description

- The second important step of your project is the **statistical analysis** of the datasets
 - Highlight distributions of different aspects of data (compositions, taxonomy, SP length)
 - Can be useful to identify biases in the data that can affect to proper development of ML-based approaches
 - Important to **prove the effectiveness of the procedure to produce a good dataset**
- Results of these analyses are presented using different kinds of **graphs**

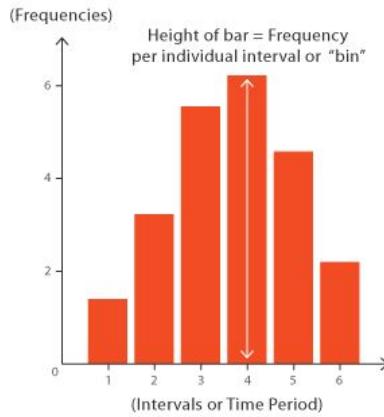
Statistical analysis of the datasets

- Computing basic statistics on our training/benchmark datasets
- Assessing the adequacy of the dataset for our purposes
- Demonstrate that the datasets are a proper snapshot of the real protein space

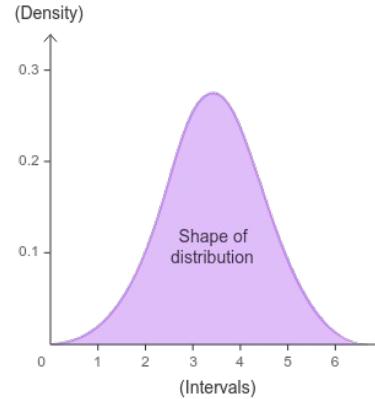
Describe the datasets producing the following plots:

- The distribution of SP lengths
- The distribution of protein lengths (comparing positive and negative)
- Comparative amino-acid composition of SPs against some background distribution e.g. amino acid composition of SwissProt available at
<https://web.expasy.org/docs/relnotes/relstat.html>
- Taxonomic classification (at kingdom and species levels)
- Sequence logos of SP cleavage sites

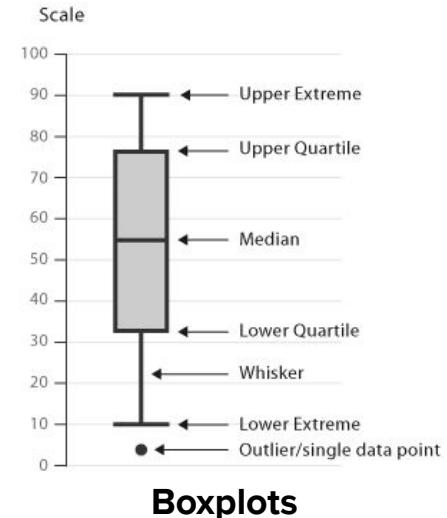
How to display data: distributions



Histograms



Density plots

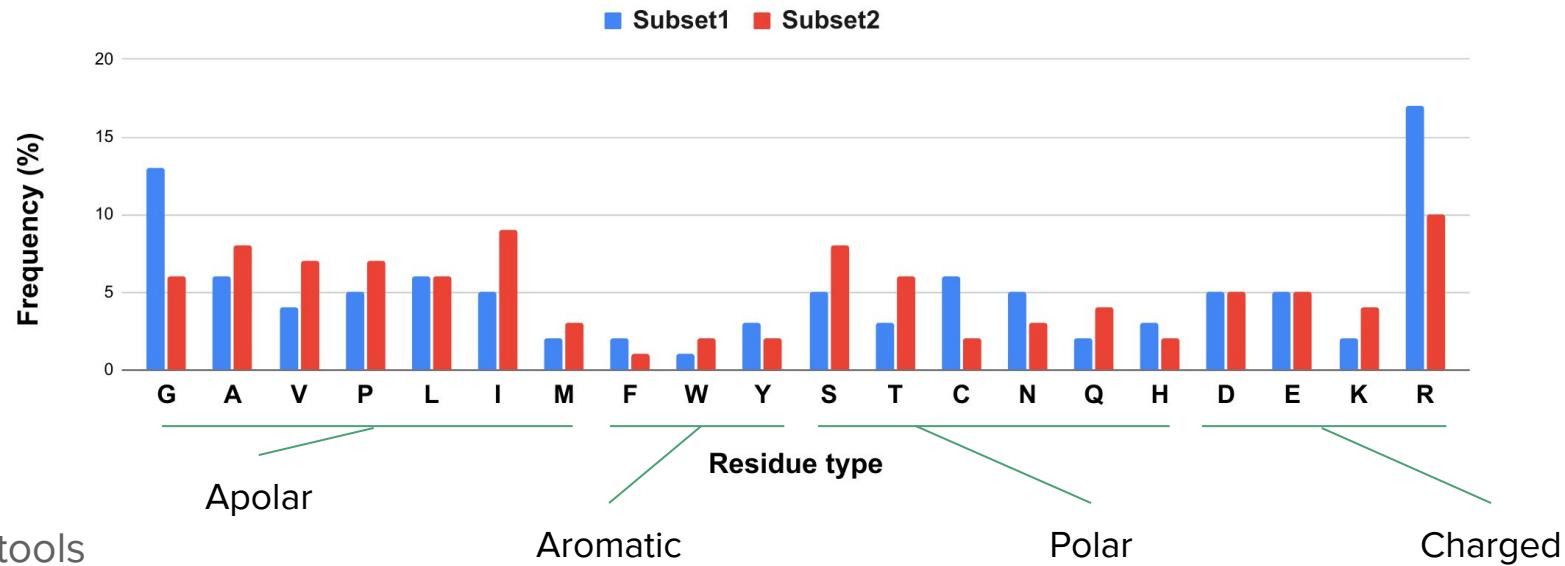


Boxplots

Useful tools

- Matplotlib: <https://matplotlib.org/>
- Python Seaborn: <https://seaborn.pydata.org/tutorial/distributions.html>
- Google Sheets
- Excel

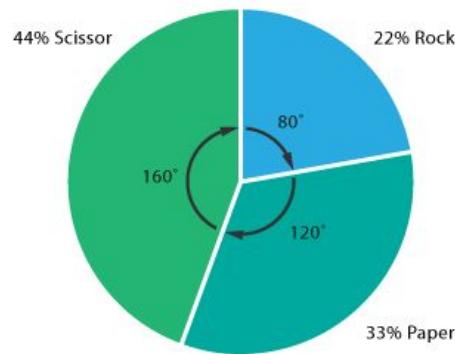
How to display data: residue compositions



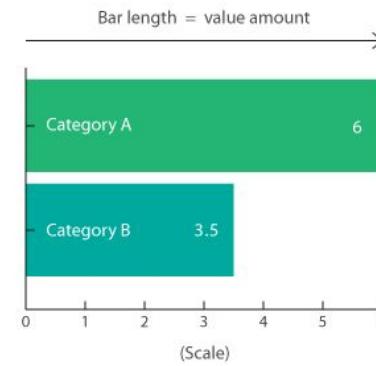
Useful tools

- Google Sheets
- Excel

How to display data: categorical data



Pie charts



Barplots

Useful tools

- **Google Sheets**
- **Excel**
- **Pandas:** https://pandas.pydata.org/docs/user_guide/visualization.html#visualization

Sequence logos



- A graphical representation of sequence conservation (of amino acids/nucleotides)
- Created from a collection of **aligned sequences**
- The logo consists of a stack of letters for each aligned position
 - The height of a letter is proportional to its frequency in the aligned position
 - The total height of the letters represents the **information content** in that position (in bits, the higher it is the lower is the entropy)
- Can be produced using one of the many tools available on the web e.g., WebLogo:
<https://weblogo.berkeley.edu/logo.cgi>
- You must provide the set of alignment sequence you want to display

Sequence logos of SP cleavage sites

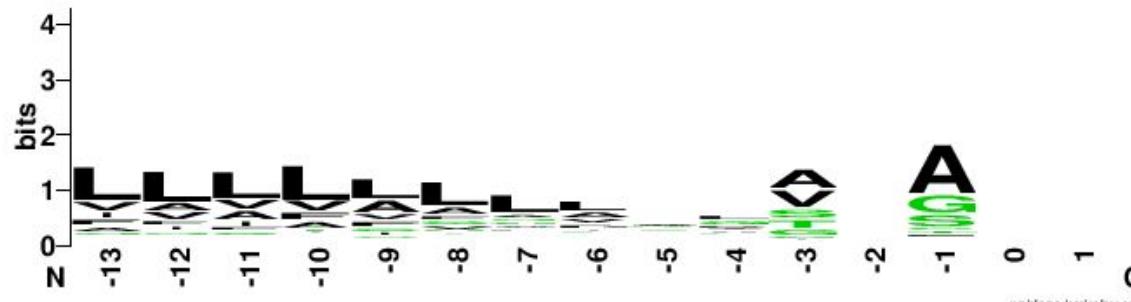
Sequences with SP (in red)

MRFLAAT**FLLLALSTAQA**EPVQFKDCGSVDGVIKEVNSPCPTQPCQLS...
MAGKEV**I**FIMALFIAVESSP**I**FSFDDLVCPSVTSLRVNVECKNECSTKKDC...
MRVF**L**LAICL**S**LTVALAAETGKYTPFQYNR**V**STVSPFVYKPGRYVADPGR...
MASSSAK**I**LLPLS**I**LL**T**LL**SLSQS**TNPNFILT**L**VNNCPYTIWPAIQPNA...
MNYLV**M**ISL**A**LL**L**MIG**V**ESVRDGY**I**IVYPHNCVYHCIPSCDGLCKENGATS...
MNNS**I**IL**I**IFVAIL**I**IFPNEFS**K**PTR**A**FSNNNFVYT**D**GTHF**A**LN**G**KS**L**YIN...
...

Aligned cleavage-sites regions (-13,+2)

TF**LLLALSTAQA**EP
F**IMALFIAVESSP**IF
FLA**I**CL**S**LTVAL**A**AE
LSLLFT**LLSLSQS**TN
ISLA**LLMIGVES**VR
IIFPNEFS**K**PTR**A**FS
...

Logo



Practical session II

Describe the training and benchmarking datasets independently, producing the following plots:

- The distribution of protein lengths comparing positive and negative sequences
 - **Use either histograms, density or boxplot.**
- The distribution of SP lengths
 - **Use either histograms, density or boxplot**
- Comparative amino-acid composition of SPs against some background distribution e.g. amino acid composition of SwissProt available at <https://web.expasy.org/docs/relnotes/relstat.html>
 - **Display AA compositions of SP sequences and SwissProt using a combined barplot as shown before**
- Taxonomic classification (at kingdom and species levels)
 - **Use either a pie chart or a barplot**
- Sequence logos of SP cleavage sites
 - **Extract the cleavage-site motifs [-13,+2] and use WebLogo to produce the sequence logo**

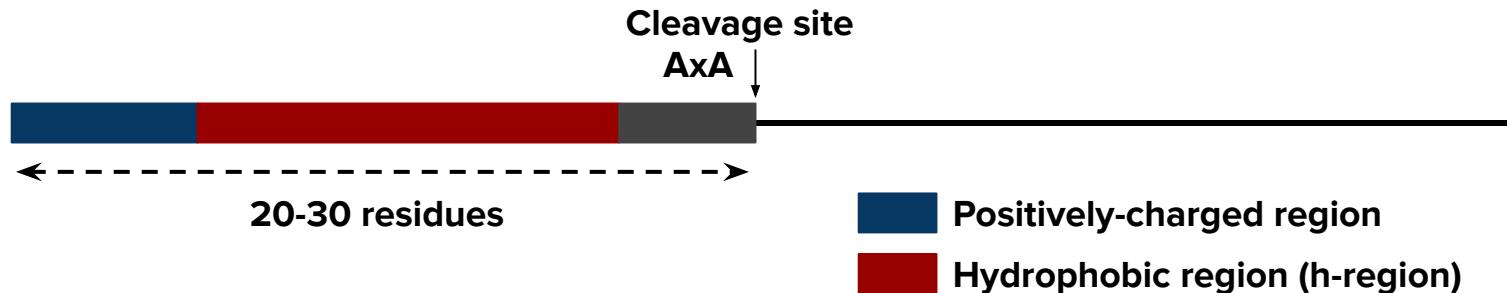
LB2 Project

The vonHeijne method for SP detection

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Signal Peptide (SP) prediction

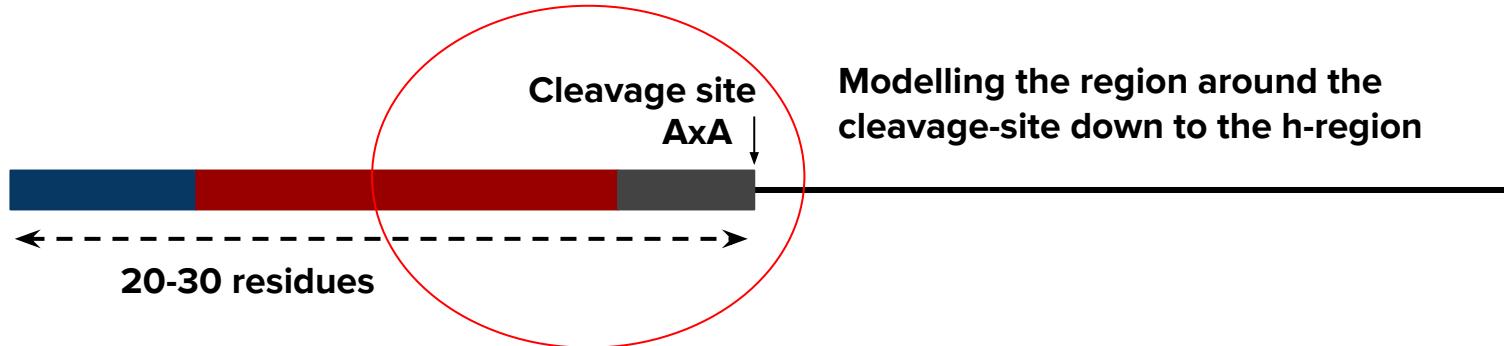
- Recognized to direct the protein toward the **secretory pathway**
- Modular “structure” comprising three separate **regions with different properties**
- **Hydrophobic core** similar to a transmembrane helix
- Weakly conserved **sequence motifs** can be observed **at cleavage sites**



Two distinct prediction tasks:

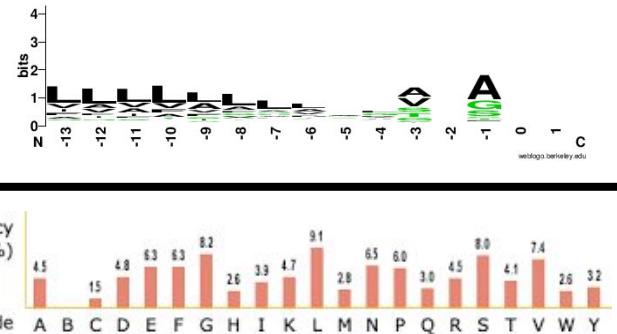
1. **Discrimination/detection:** Detect the presence of the SP sequence
2. **Labelling:** Identification of the precise position of the cleavage site along the sequence

The vonHeijne method: basic principles



Modelling the region around the cleavage-site down to the h-region

Cleavage-site context composition (-13,2)



Background AA distribution (e.g. SwissProt)

Position-Specific Weight Matrix

| | | | | | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A | 0 | 0 | 0 | -4 | 1 | 1 | -4 | -4 | -4 | 1 | -4 | -4 | -4 | | | | | | |
| R | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| N | 1 | -3 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| D | 1 | 2 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 | -4 | 2 | -4 | | | | | |
| C | -1 | -1 | -1 | -1 | 3 | -1 | -1 | -1 | -1 | 3 | 3 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Q | 1 | 3 | 1 | 2 | -3 | -3 | 1 | -3 | -3 | -3 | 3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| E | 1 | 2 | 1 | 1 | -4 | 1 | -4 | -4 | -4 | -4 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| G | 0 | -4 | -4 | -4 | -4 | -4 | 0 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 2 | -4 |
| H | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 3 | 2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 |
| I | -4 | 1 | -4 | 2 | -4 | 1 | -4 | -4 | -4 | -4 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| L | -4 | 4 | 2 | -4 | 1 | 3 | -4 | -4 | 1 | -4 | 0 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| K | 1 | -4 | -4 | 1 | -4 | 1 | 1 | 1 | -4 | -4 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| M | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 4 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 |
| F | -3 | -3 | -3 | -3 | 2 | -3 | -3 | 2 | 2 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| P | -3 | -3 | -3 | -3 | 2 | -3 | 2 | -3 | -3 | 3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| S | 1 | -4 | -4 | -4 | -4 | 1 | -4 | 2 | -4 | 1 | -4 | -4 | 1 | -4 | -3 | -4 | -4 | -4 | -4 |
| T | 2 | 1 | -3 | 2 | -3 | 1 | 1 | -3 | 1 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| W | -1 | -1 | 4 | -1 | -1 | -1 | -1 | -1 | 3 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Y | -3 | -3 | -3 | -3 | 2 | -3 | -3 | -3 | 3 | 2 | -3 | -3 | 2 | -3 | -3 | -3 | -3 | -3 | -3 |
| V | -4 | -4 | 0 | 0 | -4 | 0 | -4 | 0 | -4 | 1 | -4 | 0 | -4 | 0 | -4 | 0 | -4 | 0 | -4 |

Scores presence of residue type i at position j

Position-Specific Weight Matrices (PSWM)

- A way of representing **patterns or motifs** in biological sequences (DNA/RNA or proteins)
- The number of rows is equal to the number of different characters in the alphabet (20 for proteins, 4 for nucleotide sequences)
- The number of columns is equal to the length of the motif
- The starting point is a **set of aligned/stacked sequence fragments** of length L

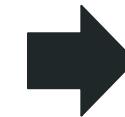
← L →
TFLLLALSTAAQAEP
FIMALFIAVESSPI F
FLAICLSLTVALAAE
LSLLFTLLSQST N
ISLALLLMIGIVESV R
IIFPNEFSKPTRA F S

Aligned/stacked sequence fragments



| | | L | |
|---|--|---|--|
| A | 10 10 10 00 20 -13 00 20 00 00 00 22 00 00 00 00 | | |
| R | 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 | | |
| N | 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 | | |
| D | 10 30 10 00 00 00 00 00 00 00 11 00 20 00 00 00 | | |
| C | 00 00 00 00 10 00 00 00 00 10 11 00 00 00 00 00 | | |
| Q | 10 00 10 20 00 00 10 00 00 00 22 00 00 00 00 00 | | |
| E | 10 30 10 10 00 13 00 00 00 00 11 00 00 00 00 00 | | |
| G | 10 00 00 00 00 00 00 10 00 00 00 30 00 00 00 00 | | |
| H | 00 10 00 00 00 00 00 10 20 00 00 00 00 00 00 10 | | |
| I | 00 10 00 20 00 00 10 00 10 00 00 13 00 10 00 00 | | |
| L | 00 00 30 00 20 63 00 00 20 00 11 00 00 00 10 00 | | |
| K | 10 00 00 10 00 00 10 10 10 00 00 00 13 00 00 00 | | |
| M | 00 00 00 00 00 00 00 00 00 22 00 00 00 00 00 00 | | |
| F | 00 00 00 20 00 30 00 20 20 11 00 00 00 00 00 50 | | |
| P | 00 00 00 20 00 30 20 00 00 33 00 65 00 00 00 00 | | |
| S | 10 00 00 00 13 00 20 00 10 00 11 00 40 00 00 00 | | |
| T | 20 10 00 20 00 00 10 10 00 10 00 11 00 00 00 00 | | |
| W | 00 20 00 00 00 00 00 10 00 20 10 00 00 00 00 00 | | |
| Y | 00 00 00 10 00 00 10 00 20 10 00 00 13 00 20 00 | | |
| V | 00 00 10 10 00 00 10 00 10 00 11 11 00 10 00 00 | | |

Position Specific Probability Matrix (PSPM): frequency of each residue type at each position



| | | L | |
|---|---|---|--|
| A | 0 0 0 -4 1 1 -4 1 -4 -4 -4 1 -4 -4 -4 -4 | | |
| R | -3 -3 -3 1 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 | | |
| N | 1 -3 -3 1 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 | | |
| D | 1 2 1 -4 -4 -4 -4 -4 -4 -4 -4 -4 1 4 2 -4 | | |
| C | -1 -1 -1 -1 3 -1 -1 -1 -1 3 3 -1 -1 -1 -1 | | |
| Q | 1 -3 1 2 -3 -3 -3 1 -3 -3 -3 -3 -3 -3 -3 | | |
| E | 1 2 1 4 -4 1 -4 -4 -4 -4 -4 -4 1 -4 -4 -4 | | |
| G | 0 -4 -4 -4 -4 -4 0 -4 -4 -4 -4 -4 2 4 | | |
| H | -2 2 -2 -2 -2 -2 -2 -2 2 3 -2 -2 -2 -2 -2 | | |
| I | -4 1 -4 2 -4 1 -4 -4 1 -4 -4 -4 1 -4 -4 1 | | |
| L | -4 2 4 1 3 -4 -4 1 -4 0 -4 -4 -4 -4 0 | | |
| K | 1 -4 -4 1 -4 -4 1 1 1 -4 -4 -4 1 -4 -4 | | |
| M | -2 -2 -2 -2 -2 -2 -2 -2 -2 2 -4 -2 -2 -2 -2 | | |
| F | -3 -3 -3 -3 2 -3 3 -3 2 2 1 -3 -3 -3 4 | | |
| P | -3 -3 -3 -2 -3 3 2 -3 -3 -3 4 -3 -3 | | |
| S | 1 -4 -4 -4 -4 1 -4 -2 -4 1 -4 4 -3 4 | | |
| T | 2 1 -4 2 -3 -3 1 1 -3 1 -3 1 -3 -3 -3 | | |
| W | -1 -1 4 -1 -1 -1 -1 3 -1 -1 -1 -1 -1 -1 -1 | | |
| Y | -3 -3 -3 -2 -3 -3 -3 3 2 -3 -3 2 -3 3 | | |
| V | -4 0 0 -4 -4 0 -4 0 -4 1 1 -4 0 -4 0 -4 | | |

Position-Specific Weight Matrix (PSWM): log-odds between frequencies in the PSPM and some background model (e.g., the SwissProt composition)

PSWM: formal definition

Given a set S of N aligned sequences of length L , the PSPM M is computed as follows:

$$M_{k,j} = \frac{1}{N} \sum_{i=1}^N I(s_{i,j} = k)$$

Where:

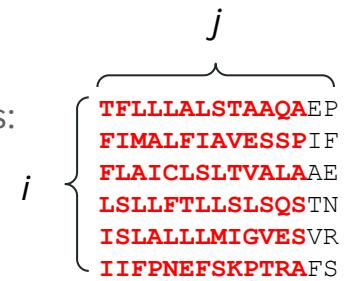
- $s_{i,j}$ is the observed residue of aligned sequence i at position j
- k is the residue corresponding to the k -th row in the matrix
- $I(s_{i,j} = k)$ is an indicator function (1 if the condition is met, 0 otherwise)

From the PSPM M , the PSWM W is computed as follows:

$$W_{k,j} = \log \frac{M_{k,j}}{b_k}$$

Where:

- b_k is the frequency of residue type k in the **background model**



| | j | | | | | | | | | | | | | | | | | | | |
|-----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| i | | | | | | | | | | | | | | | | | | | | |
| A | 0 | 0 | 0 | -4 | 1 | 1 | -4 | 1 | -4 | -4 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| R | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| N | 1 | -3 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| D | 1 | 2 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C | -1 | -1 | -1 | -1 | 3 | -1 | -1 | -1 | -1 | -1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Q | 1 | -3 | 1 | 2 | -3 | -3 | 1 | -3 | -3 | -3 | 3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| E | 1 | 2 | 1 | 1 | -4 | 1 | 4 | -4 | -4 | -4 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| G | 0 | -4 | -4 | -4 | -4 | 0 | -4 | -4 | -4 | -4 | 2 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| H | -2 | 2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 |
| I | -4 | 1 | -4 | 2 | -4 | -4 | 1 | -4 | 1 | -4 | -4 | -4 | 1 | -4 | 1 | -4 | 1 | -4 | 1 | -4 |
| L | -4 | -4 | 2 | -4 | 1 | 3 | -4 | 1 | -4 | 0 | -4 | -4 | -4 | 0 | -4 | -4 | -4 | -4 | -4 | -4 |
| K | 1 | -4 | -4 | 1 | -4 | -4 | 1 | 1 | 1 | -4 | -4 | -4 | 1 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |
| M | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 |
| F | -3 | -3 | -3 | 2 | -3 | 2 | -3 | 2 | 2 | 1 | -3 | -3 | -3 | 4 | -3 | -3 | -3 | -3 | -3 | -3 |
| P | -3 | -3 | -3 | 2 | -3 | 3 | 2 | -3 | 3 | -3 | -3 | -3 | 3 | -3 | 4 | -3 | -3 | -3 | -3 | -3 |
| S | 1 | -4 | -4 | -4 | 1 | 4 | -4 | 1 | -4 | 1 | -4 | 1 | 4 | 3 | -4 | -3 | -3 | -3 | -3 | -3 |
| T | 2 | 1 | -3 | 2 | -3 | 1 | 1 | -3 | 1 | -3 | 1 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| W | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Y | -3 | -3 | -3 | 2 | -3 | -3 | 3 | 2 | -3 | -3 | 2 | -3 | -3 | 2 | -3 | -3 | 2 | -3 | -3 | -3 |
| V | -4 | -4 | 0 | 0 | -4 | 4 | 0 | -4 | 0 | -4 | 1 | -4 | 0 | -4 | -4 | -4 | -4 | -4 | -4 | -4 |

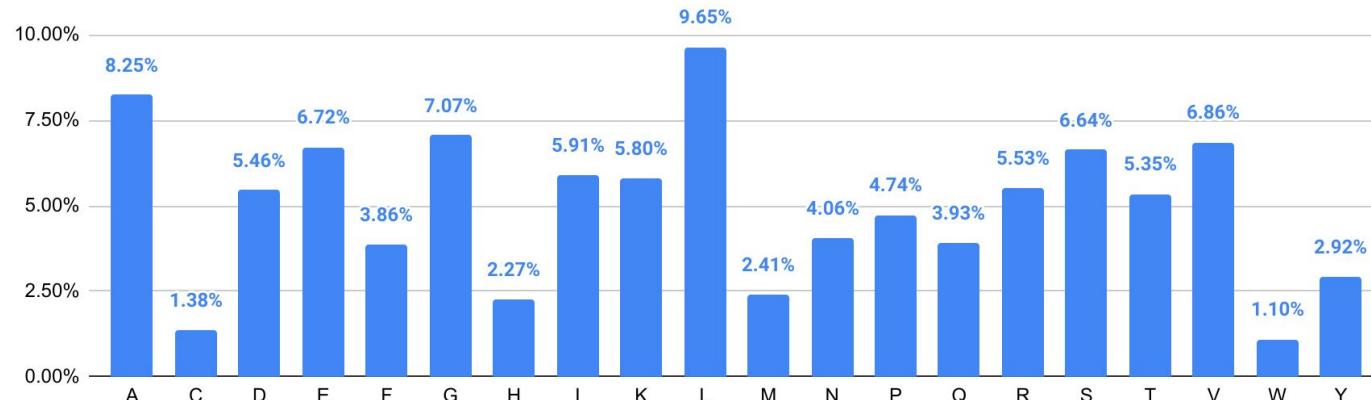
PSWM: meaning of the values

Values $W_{k,j}$ can be:

- Positive, when $M_{k,j} > b_k$: the probability of residue k at position j in the motif differs from the background and it is higher (more likely to be an important/functional site than random)
- $M_{k,j} = b_k$: the two probabilities are the same (more likely to be a random site than a functional one).
- Negative, when $b_k > M_{k,j}$: the probability of residue k at position j in the motif differs from the background and it is lower (residue k is underrepresented in that position)

PSWM: background model

- The simplest background model is the **uniform distribution** i.e. assuming all the residues as equally frequent with probability $1/20=0.05$
- In general, any background distribution can be adopted e.g., the overall AA composition computed on the SwissProt database (<https://web.expasy.org/docs/relnotes/relstat.html>)



PSWM: pseudocounts

- When dealing with datasets of finite size some of the counts may be zero i.e., not all residues are observed in some positions along the motifs
- In order to avoid zero probabilities in the PSPM and hence the impossibility of computing the log-odds, **pseudocounts** are added during computation of PSPM
- In the most simple setting, the count matrix is initialized assuming each residue is observed at least once in all positions
- Formally, the formula for computing the PSPM M becomes:

$$M_{k,j} = \frac{1}{N+20} (1 + \sum_{i=1}^N I(s_{i,j} = k))$$

PSWM: scoring motifs on new sequences

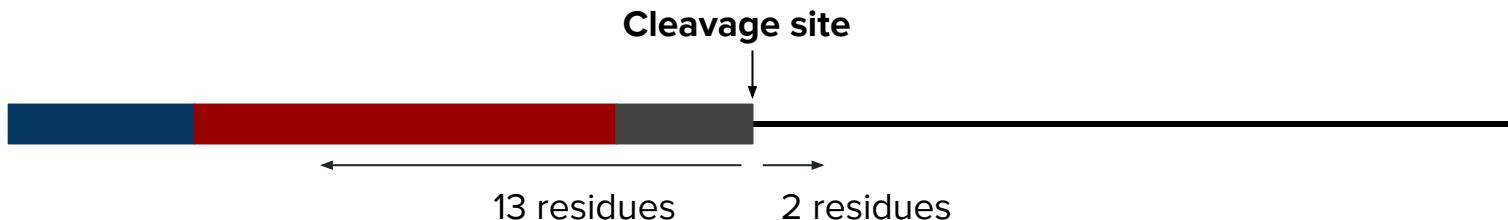
- Given any piece of sequence $X=(x_1, \dots, x_L)$ of length L , one can compute the (log-likelihood) score of X given the PSWM W as:

$$score(X|W) = \sum_{i=1}^L W x_i, i$$

- The values of the scores can be used to scan a protein sequence in order to find regions where the likelihood of occurrence of the motif represented by W is high

The vonHeijne weight matrix

- Using a dataset of proteins endowed with SPs, estimates a PSWM from the region surrounding the cleavage-sites
- The cleavage-site context covers 15 residues from residue -13 (downstream) to residue +2 (upstream) w.r.t. the position of the site



- Background model used is SwissProt AA composition
- Pseudocounts of 1 are added to all positions

Scoring the presence of SPs in new sequences

- Extract the first 90 N-terminal positions from the complete protein sequence
- Scan positions from 1 to 76 (=90-15+1) with the PSWM (extracting all subsequence of 15 residues), obtaining a score for each position
- Compute the global score for SP detection as the maximum positional score along the sequence: the higher the global score the higher is the likelihood that the protein has a SP

>sp | P01229 | LSHB_HUMAN

MEMLQGLLLLLLSMGAWASREPLRPWCHPINAILAVEKEGCPVCITVN
MEMLQGLLLLLLSM

EMLQGLLLLLLSMG
MLQGLLLLLLSMG
...

Subsequences

| | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A | 0 | 0 | 0 | 4 | 1 | 1 | 4 | 1 | -4 | -4 | 1 | -4 | -4 | 4 |
| R | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | 1 | 1 | -3 | -3 | -3 | -3 |
| N | 1 | -3 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| D | 1 | 2 | 1 | 4 | -4 | -4 | -4 | -4 | -4 | -4 | 1 | -4 | 2 | -4 |
| C | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | 3 | 3 | -1 | -1 | 2 | -1 |
| Q | 1 | -3 | 1 | 2 | -3 | -3 | 1 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| E | 1 | 2 | 1 | 1 | -4 | 1 | 4 | -4 | -4 | -4 | -4 | 1 | -4 | -4 |
| G | 0 | -4 | -4 | -4 | -4 | -4 | 0 | -4 | -4 | -4 | -4 | 4 | -2 | -4 |
| H | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 3 | 2 | -2 | -2 | -2 | 2 |
| I | -4 | 1 | -4 | 2 | -4 | 4 | 1 | -4 | -4 | -4 | 1 | -4 | 1 | -1 |
| L | -4 | -4 | 2 | -4 | 1 | 3 | -4 | -4 | 1 | -4 | 0 | -4 | -4 | -4 |
| K | 1 | -4 | -4 | 1 | -4 | 1 | 1 | -4 | -4 | -4 | 1 | -4 | -4 | -4 |
| M | -2 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | -2 | -2 | -2 | -2 |
| F | -3 | -3 | -3 | 2 | -3 | 3 | -3 | 2 | 2 | 1 | -3 | -3 | -3 | 4 |
| P | -3 | -3 | -3 | 2 | -3 | 3 | 2 | -3 | -3 | -3 | 4 | -3 | -3 | -3 |
| S | 1 | -4 | -4 | -4 | 1 | -4 | 2 | -4 | 1 | -4 | 1 | -4 | 3 | -4 |
| T | 2 | 1 | 3 | 2 | -3 | -3 | 1 | 1 | -3 | 1 | -3 | 1 | -3 | -3 |
| W | -1 | -1 | 4 | -1 | -1 | -1 | -1 | -1 | 3 | 1 | -1 | -1 | -1 | -1 |
| Y | -3 | -3 | -3 | 2 | -3 | 3 | -3 | 3 | 2 | 3 | 3 | 2 | -3 | 3 |
| V | -4 | -4 | 0 | -4 | -4 | 0 | -4 | 0 | 4 | 1 | 1 | -4 | 0 | -4 |



PSWM

50aa

| Positional scores | | | | | | | | | |
|-------------------|----|----|----|---|----|----|-----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | 35 | 36 |
| -10 | -4 | -8 | -3 | 6 | 11 | 13 | ... | -1 | -4 |

Highest score = 13

The method at work: training

Cleavage site

STAAQAE

AVESSP T E

ТРИАДА

ESTIVADA
E SISGOS

LSTSQS TI

MIGVES VÉ

SKPTRAFS

Training set:

- N=6 sequences
 - [-6,+2] cleavage-site context

The method at work: training

Matrix initialization: using **pseudocounts** -> initialize all the cells of the matrix with **one** (i.e., each residue is observed by default once)

Cleavage site

**STAAQAEI
AVESSPIH
LTVALAAM
LSLSQSTM
MIGVESVH
SKPTRAFS**

- N=6 sequences
- [-6,+2] cleavage-site context

The method at work: training

First training cleavage-site context

STAAQAEP

**STAAQAEP
AVESSPIF
LTVALAEE
LSLSQS TN
MIGVES VR
SKPTRAFS**

- Training set:
 - N=6 sequences
 - [-6,+2] cleavage-site context

The method at work: training

First training cleavage-site context

STAAQAEP



Update (add 1) all the counts accordingly

Cleavage site

STAAQAEP
AVESSPIF
LTVVALAEE
LSLSQSNTN
MIGVESVR
SKPTRAFS

Training set:
• N=6 sequences
• [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|----------|---|---|---|---|----------|----------|---|---|---|---|---|---|---|---|----------|----------|---|---|---|----|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | -6 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -5 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -4 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -3 |
| 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -2 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 |

The method at work: training

Next training cleavage-site context

AVESSPIF

Update (add 1) all the counts accordingly

Cleavage site

STAQAEP

AVESSPIF

LTVAAAE

LSLSQSTN

MIGVESVR

SKPTRAFS

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|----------|---|---|---|---|---|----------|---|----------|---|---|---|----------|---|----------|----------|---|---|---|-------------|--|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 -5 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -4 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -3 | |
| 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -2 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | -1 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | |

The method at work: training

Next training cleavage-site context

LTVALAAE



Update (add 1) all the counts accordingly

Cleavage site

STAAQAE^P
AVESSP^IF
LTVALAAE
LSLSQS^TN
MIGVES^VR
SKPTRAFS

- Training set:
- N=6 sequences
 - [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|----------|---|---|---|---|---|----------|---|---|---|----------|---|---|---|---|---|----------|---|---|----------|--|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | -5 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -4 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -3 | |
| 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -2 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -1 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | |

The method at work: training

Next training cleavage-site context

LSLSQS^{TN}



Update (add 1) all the counts accordingly

Cleavage site

STAAQ**A**EP

AVESSP**I**F

LTVALA**AE**

LSLSQS^{TN}

MIGVESVR

SKPTRAFS

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|---|---|----------|---|---|----------|---|---|---|---|----------|---|---|---|---|----------|----------|---|---|-----------|-----------|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 2 | -5 |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | -4 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | -3 |
| 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | -2 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | -1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 2 |

The method at work: training

Next training cleavage-site context

MIGVESVR

Update (add 1) all the counts accordingly

Cleavage site

STA**AQAE**P

AVESSP**IF**

LTV**ALAAE**

LSLSQS**TN**

MIGVESVR

SKP**TRAFS**

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|---|----------|---|---|---|---|----------|----------|---|----------|---|---|----------|---|---|----------|---|---|----------|----------|--|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | -5 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -4 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 2 | -3 | |
| 1 | 1 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -2 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | -1 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | |
| 1 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 2 | |

The method at work: training

Next training cleavage-site context

SKPTRAFS

Update (add 1) all the counts accordingly

Cleavage site

STAQAEP

AVESSPIF

LTVVALAAE

LSLSQSTN

MIGVESVR

SKPTRAFS

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|----------|----------|---|---|---|---|---|---|---|---|---|----------|---|----------|----------|----------|----------|---|---|----|--|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 3 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | -5 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | -4 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | -3 | |
| 1 | 2 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -2 | |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | -1 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | |
| 1 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | |

The method at work: training

Compute the PSPM

Divide all the counts by $N+20=26$

Cleavage site

STAQA_{EP}

AVESSP_{IF}

LTVVALAA_E

LSLSQS_{TN}

MIGVES_{VR}

SKPTR_{AFS}

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|--|
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 2 | 1 | 1 | 3 | 1 | 1 | 1 | -6 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | -5 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | -4 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | -3 | |
| 1 | 2 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | -2 | |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | -1 | |
| 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | |
| 1 | 2 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | |

The method at work: training

Compute the PSPM

Divide all the counts by N+20=26

Cleavage site

STAAQAEP
AVESSPI F
LTVVALAA E
LSLSQST N
MIGVESV R
SKPTRA F S

Training set:
 • N=6 sequences
 • [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------|
| .08 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | -6 |
| .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .08 | -5 |
| .08 | .04 | .04 | .04 | .04 | .04 | .08 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .08 | -4 |
| 0.12 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | -3 |
| .04 | .08 | .04 | .04 | .04 | 0.12 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | -2 |
| 0.15 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .04 | -1 |
| .08 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | .04 | .04 | .08 | 1 |
| .04 | .08 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .08 | .08 | .04 | .04 | .04 | .04 | 2 |

The method at work: training

Cleavage site

STAAAQ**A**EP

AVESSP**I**F

LTVALA**A**E

LSLSQS**T**N

MIGVES**V**R

SKPTRA**F**S

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

Compute the PSWM

A:0.08, R:0.06, N:0.04, D:0.06,
 C:0.01, Q:0.04, E:0.07, G:0.07,
 H:0.02, I:0.06, L:0.10, K:0.06,
 M:0.02, F:0.04, P:0.05, S:0.07,
 T:0.05, W:0.01, Y:0.03, V:0.07

Divide all the counts by the corresponding residue frequency in the SwissProt background distribution

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----------|-----------|
| .08 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | -6 | |
| .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | -5 | |
| .08 | .04 | .04 | .04 | .04 | .04 | .08 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .08 | -4 | |
| .12 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .12 | .08 | .04 | .04 | .08 | -3 |
| .04 | .08 | .04 | .04 | .04 | .12 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | -2 | |
| .15 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .12 | .04 | .04 | .04 | -1 | |
| .08 | .04 | .04 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .08 | .04 | .08 | 1 | |
| .04 | .08 | .08 | .04 | .04 | .04 | .08 | .04 | .04 | .04 | .04 | .04 | .04 | .08 | .08 | .08 | .04 | .04 | .04 | 2 | |

The method at work: training

Compute the PSWM

A:0.08, R:0.06, N:0.04, D:0.06,
 C:0.01, Q:0.04, E:0.07, G:0.07,
 H:0.02, I:0.06, L:0.10, K:0.06,
 M:0.02, F:0.04, P:0.05, S:0.07,
 T:0.05, W:0.01, Y:0.03, V:0.07

Divide all the counts by the corresponding residue frequency in the SwissProt background distribution

Cleavage site

STAAQA_{EP}

AVESSP_{IF}

LTVVALA_{AE}

LSLSQSQ_{TN}

MIGVES_{VR}

SKPTRA_{FS}

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|-----|-----|---|-----|---|---|-----|-----|---|-----|-----|-----|---|---|-----|-----|-----|---|-----|-----|----|
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 1.2 | 0.7 | 4 | 1 | 0.8 | 1.7 | 0.8 | 4 | 1.3 | 0.6 | -6 |
| 0.5 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 1.4 | 0.4 | 1.4 | 2 | 1 | 0.8 | 1.1 | 2.4 | 4 | 1.3 | 1.2 | -5 |
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 1.2 | 1.2 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 1.6 | 0.6 | 0.8 | 4 | 1.3 | 1.2 | -4 |
| 1.9 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 0.8 | 1.7 | 1.6 | 4 | 1.3 | 1.2 | -3 |
| 0.5 | 1.3 | 1 | 0.7 | 4 | 3 | 1.2 | 0.6 | 2 | 0.7 | 0.8 | 0.7 | 2 | 1 | 0.8 | 1.1 | 0.8 | 4 | 1.3 | 0.6 | -2 |
| 1.9 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 1.6 | 1.7 | 0.8 | 4 | 1.3 | 0.6 | -1 |
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 1.2 | 0.6 | 2 | 1.4 | 0.4 | 0.7 | 2 | 2 | 0.8 | 0.6 | 1.6 | 4 | 1.3 | 1.2 | 1 |
| 0.5 | 1.3 | 2 | 0.7 | 4 | 1 | 1.2 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 2 | 1.6 | 1.1 | 0.8 | 4 | 1.3 | 0.6 | 2 |

The method at work: training

Compute the PSWM

Compute the logarithms

Cleavage site

STAQAEP
AVESSPIF
LTVVALAEE
LSLSQSNTN
MIGVESVR
SKPTRAFS

- Training set:
- N=6 sequences
 - [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|-----|-----|---|-----|---|---|-----|-----|---|-----|-----|-----|---|---|-----|-----|-----|---|-----|-----|-----------|
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 1.2 | 0.7 | 4 | 1 | 0.8 | 1.7 | 0.8 | 4 | 1.3 | 0.6 | -6 |
| 0.5 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 1.4 | 0.4 | 1.4 | 2 | 1 | 0.8 | 1.1 | 2.4 | 4 | 1.3 | 1.2 | -5 |
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 1.2 | 1.2 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 1.6 | 0.6 | 0.8 | 4 | 1.3 | 1.2 | -4 |
| 1.9 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 0.8 | 1.7 | 1.6 | 4 | 1.3 | 1.2 | -3 |
| 0.5 | 1.3 | 1 | 0.7 | 4 | 3 | 1.2 | 0.6 | 2 | 0.7 | 0.8 | 0.7 | 2 | 1 | 0.8 | 1.1 | 0.8 | 4 | 1.3 | 0.6 | -2 |
| 1.9 | 0.7 | 1 | 0.7 | 4 | 1 | 0.6 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 1 | 1.6 | 1.7 | 0.8 | 4 | 1.3 | 0.6 | -1 |
| 1 | 0.7 | 1 | 0.7 | 4 | 1 | 1.2 | 0.6 | 2 | 1.4 | 0.4 | 0.7 | 2 | 2 | 0.8 | 0.6 | 1.6 | 4 | 1.3 | 1.2 | 1 |
| 0.5 | 1.3 | 2 | 0.7 | 4 | 1 | 1.2 | 0.6 | 2 | 0.7 | 0.4 | 0.7 | 2 | 2 | 1.6 | 1.1 | 0.8 | 4 | 1.3 | 0.6 | 2 |

The method at work: training

Compute the PSWM

Compute the logarithms (either natural or base 2)

Cleavage site

STAAQAEP
AVESSPI F
LTVVALAA E
LSLSQST N
MIGVESV R
SKPTRA F S

Training set:
 • N=6 sequences
 • [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|------|-----|------|-----|-----|------|------|-----|------|------|------|-----|-----|------|------|------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: training

Compute the PSWM

Even in presence of a small dataset, some signals are evident

Cleavage site

STAAAQ**A**EP
AVESSP**I**F
LTVAL**A**AE
LSLSQS**T**N
MIGVES**V**R
SKPTRA**F**S

Training set:
 • N=6 sequences
 • [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------------|------|------------|------|-----|------------|------|------|-----|------|------|------|-----|------------|------------|------------|------------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: training

Compute the PSWM

Other, apparent, signals are instead due
to the small number of samples (and
pseudocounts)

Cleavage site

STAQAQAEP
AVESSPI**F**
LTVVALAAE
LSLSQSTN
MIGVESVR
SKPTRAFS

Training set:

- N=6 sequences
- [-6,+2] cleavage-site context

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|------|-----|------|-----|-----|------|------|-----|------|------|------|-----|-----|------|------|------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: prediction

Input sequence

MRFLAAATFLLLALSTAAQAEPVQF

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|------|-----|------|-----|-----|------|------|-----|------|------|------|-----|-----|------|------|------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: prediction

Testing sequence

MRFLAATFLLLALSTAAQAEPVQF

First 8-residues subsequence

X=**MRFLAATF**

$$score_{(X|W)} = \sum_{i=1}^L Wx_i, i = 2.0 - 0.5 + 0 - 1.3 - 1 + 0.9 + 0.7 + 1 = \mathbf{1.8}$$

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|-------------|-----|------|-----|-----|------|------|-----|------|-------------|------|------------|------------|------|------|------------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: prediction

Testing sequence

M**RFLAATFL**LLALSTAAQAEPVQF

Next 8-residues subsequence

X=**RFLAATFL**

$$score_{(X|W)} = \sum_{i=1}^L Wx_i, i = -0.5 + 0 - 1.3 + 0.9 - 1 - 0.3 + 1 - 1.3 = \mathbf{-2.5}$$

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|------|-----|------|-----|-----|------|------|-----|------|-------------|------|-----|------------|------|------|-------------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: prediction

Testing sequence

MR**FLAATFLL**LALSTAAQAEPVQF

Next 8-residues subsequence

X=**FLAATFLL**

$$score_{(X|W)} = \sum_{i=1}^L Wx_i, i = 0 - 1.3 + 0 + 0.9 - 0.3 + 0 - 1.3 - 1.3 = \mathbf{-3.3}$$

| A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V | |
|------|------|-----|------|-----|-----|------|------|-----|------|------|------|-----|-----|------|------|------|-----|-----|------|-----------|
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | 0.3 | -0.5 | 2.0 | 0.0 | -0.3 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -6 |
| -1.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | 0.5 | -1.3 | 0.5 | 1.0 | 0.0 | -0.3 | 0.1 | 1.3 | 2.0 | 0.4 | 0.3 | -5 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | 0.3 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | -0.7 | -0.3 | 2.0 | 0.4 | 0.3 | -4 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.8 | 0.7 | 2.0 | 0.4 | 0.3 | -3 |
| -1.0 | 0.4 | 0.0 | -0.5 | 2.0 | 1.6 | 0.3 | -0.7 | 1.0 | -0.5 | -0.3 | -0.5 | 1.0 | 0.0 | -0.3 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | -2 |
| 0.9 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | -0.7 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 0.0 | 0.7 | 0.8 | -0.3 | 2.0 | 0.4 | -0.7 | -1 |
| 0.0 | -0.5 | 0.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | 0.5 | -1.3 | -0.5 | 1.0 | 1.0 | -0.3 | -0.7 | 0.7 | 2.0 | 0.4 | 0.3 | 1 |
| -1.0 | 0.4 | 1.0 | -0.5 | 2.0 | 0.0 | 0.3 | -0.7 | 1.0 | -0.5 | -1.3 | -0.5 | 1.0 | 1.0 | 0.7 | 0.1 | -0.3 | 2.0 | 0.4 | -0.7 | 2 |

The method at work: prediction

| Input sequence | Scores |
|---------------------------------|-------------|
| MRFLAATFLLLALSTAAQAEPVQF | |
| MRFLAATF | +1.8 |
| RFLAATFL | -2.5 |
| FLAATFLL | -3.3 |
| LAATFLLL | -3.9 |
| AATFLLLA | -5.2 |
| ATFLLLAL | -2.9 |
| TFLLLALS | -3.5 |
| FLLLALST | -7.2 |
| LLLALSTA | -1.2 |
| LLALSTAA | -3.5 |
| LALSTAAQ | -0.6 |
| ALSTAAQA | -2.4 |
| LSTAAQAE | +0.3 |
| STAAQAEP | +6.5 |
| TAAQAEPV | -4.0 |
| AAQAEPVQ | +1.2 |
| AQAEPVQF | -0.7 |

Highest score, assigned to the whole sequence: 6.5

How to transform numerical scores into classifications

- Each sequence in a testing dataset will have a numerical score assigned
 - The higher the score the higher is the likelihood the sequence has a signal peptide
- We need to define a **threshold** for the score values such that:
 - All sequences with a score \geq threshold are predicted as having SP (positive class)
 - All sequences with a score $<$ threshold are predicted as negatives

Two options:

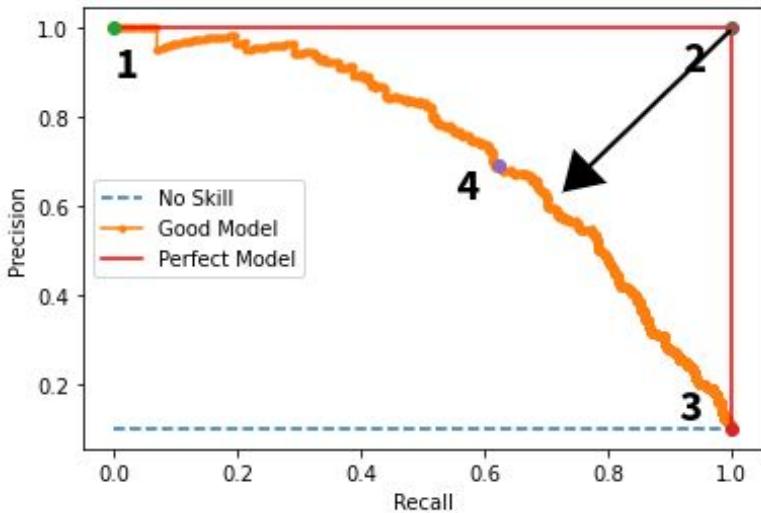
- Choosing a fixed threshold e.g. th=0 (too many false positives, high sensitivity) or th=5 (better to avoid false positives predictions, lower sensitivity)
 - In any case, the best fixed value should be obtained in cross-validation and then fixed for predicting the benchmark set
- Optimizing the value of the threshold in cross-validation for each training run:
 - At each training run, the value of the threshold is estimated using a **validation set** and fixed for predicting the testing set
 - The optimal value for each cross-validation run is obtained as the one maximizing some performance metric on validation data e.g. using precision-recall values at varying prediction thresholds
 - **Never use testing data to select the threshold**

Cross-validated threshold estimation



Precision-recall curve

Compute and display precision and recall metrics obtained varying the prediction threshold
(considering all possible values from predicted data)



$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{Recall} = \frac{TP}{TP+FN}$$

1. Threshold set to the maximum value (precision=1, recall=0)
2. Perfect model (theoretical, precision=1, recall=1)
3. Threshold set to the minimum value (precision=0, recall=1)
4. Good model, balancing precision and recall

Nearly optimal threshold can be identified computing F1-score at varying thresholds

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

How to get best threshold: sklearn

```
# import sklearn precision_recall_curve function
from sklearn.metrics import precision_recall_curve
# binary representation of the true (observed) class for each validation example: 0=NO_SP, 1=SP
y_validation = [1,1,0,0,0,1,1,1...]
# SP detection scores assigned to each validation example by means of weight matrix scanning
y_validation_scores = [2.5,8,-3,-3.3,-7,8,1,9.32...]
# precision-recall curve values
# precision: contains precision scores at varying thresholds
# recall: contains recall scores at varying thresholds
# thresholds: the thresholds values
precision, recall, thresholds = precision_recall_curve(y_validation, y_validation_scores)
# compute f-scores at varying thresholds
fscore = (2 * precision * recall) / (precision + recall)
# get the index of the maximum value of the f-score
index = numpy.argmax(fscore)
# retrieve the threshold value corresponding to the max f-score computed above
optimal_threshold = thresholds[index]
# SP detection scores assigned to each testing example by means of weight matrix scanning
y_test_scores = [1.2,-6,3.33,4,-2,10,-1,7.3...]
# classify examples in the testing set
y_pred_test = [int(t_s >= optimal_threshold) for t_s in y_test_scores]
```

Final remarks and hints

Since you have to run cross-validation, you need to train/test multiple vH models

Develop two different programs:

- **vH-train.py** : train a model from an user-defined training set and store all the weight matrix into an output file
- **vHr-predict.py** : reads the weight matrix from an input model file and predicts SPs for all the sequence in a user-defined testing set, storing the results (predictions) to an output file

Training/testing datasets should be passed as arguments

- In this way you can run the same programs on different training/testing sets

Useful modules: **numpy, argparse**

Practical session III

- Implement the code of the vonHeijne method for SP detection
 - Cleavage-site context: **(-13,+2)**
 - Pseudocounts: **1**
 - Background model: **SwissProt**
 - Threshold selection: **either fixed or cross-validated (using PR curve)**
- Run cross-validation

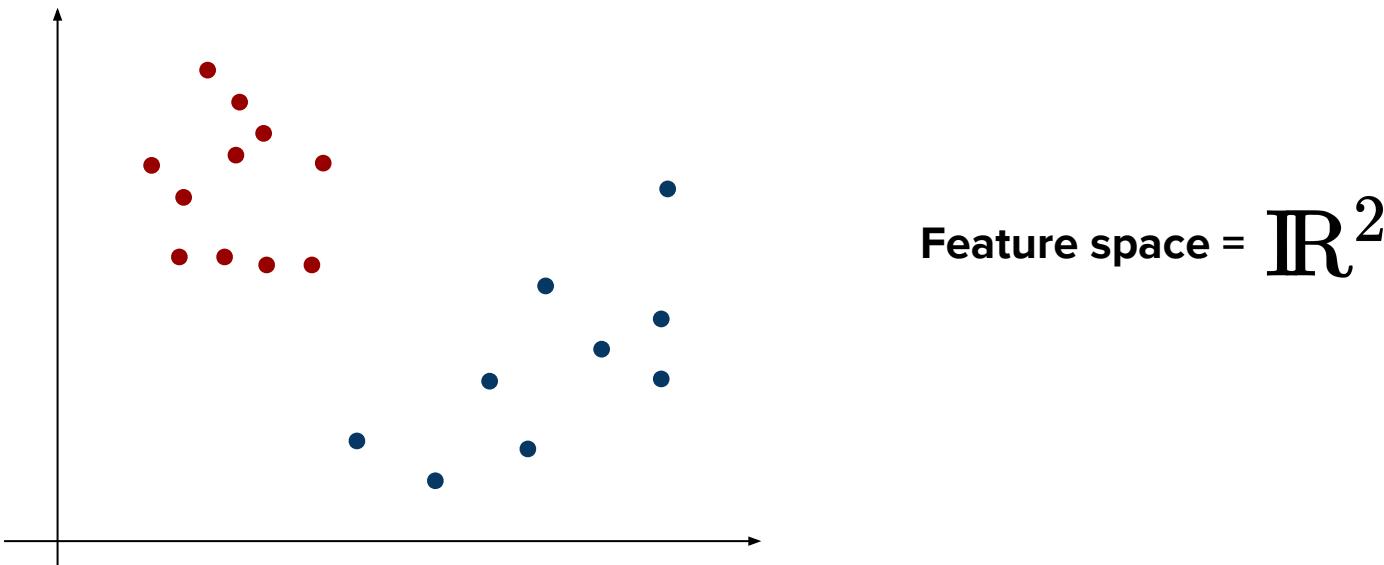
LB2 Project

Support Vector Machine for SP detection

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

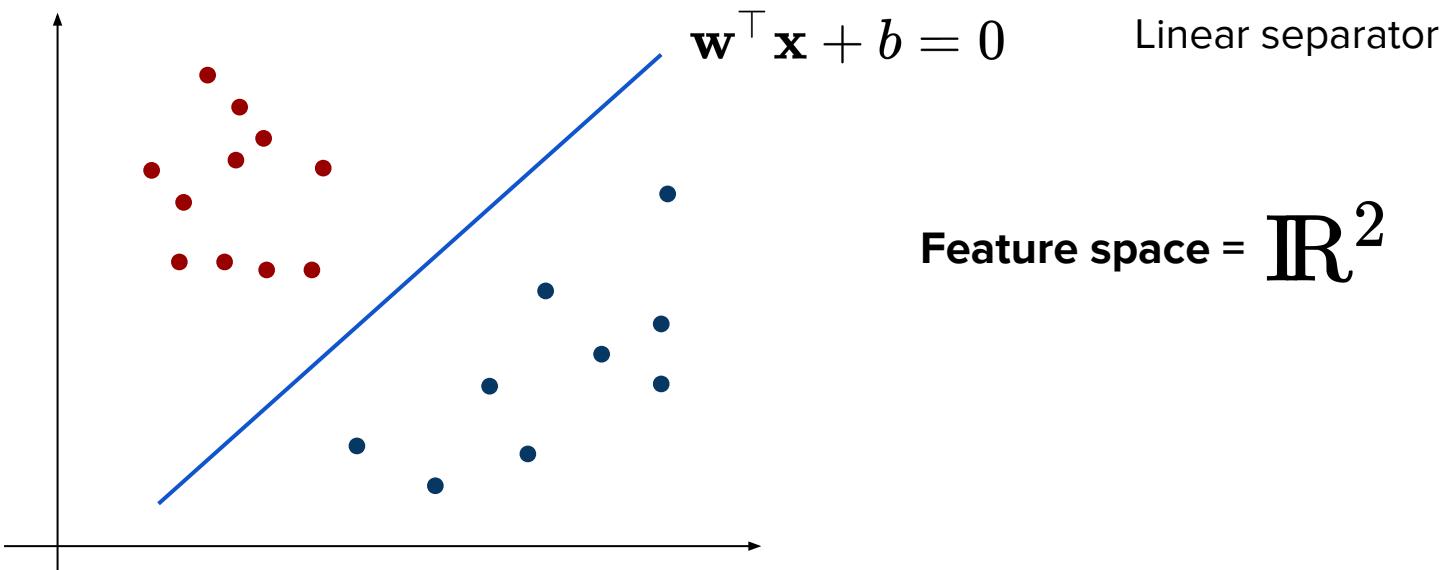
Linear classifiers

Binary classification can be viewed as the task of separating classes in some feature space:



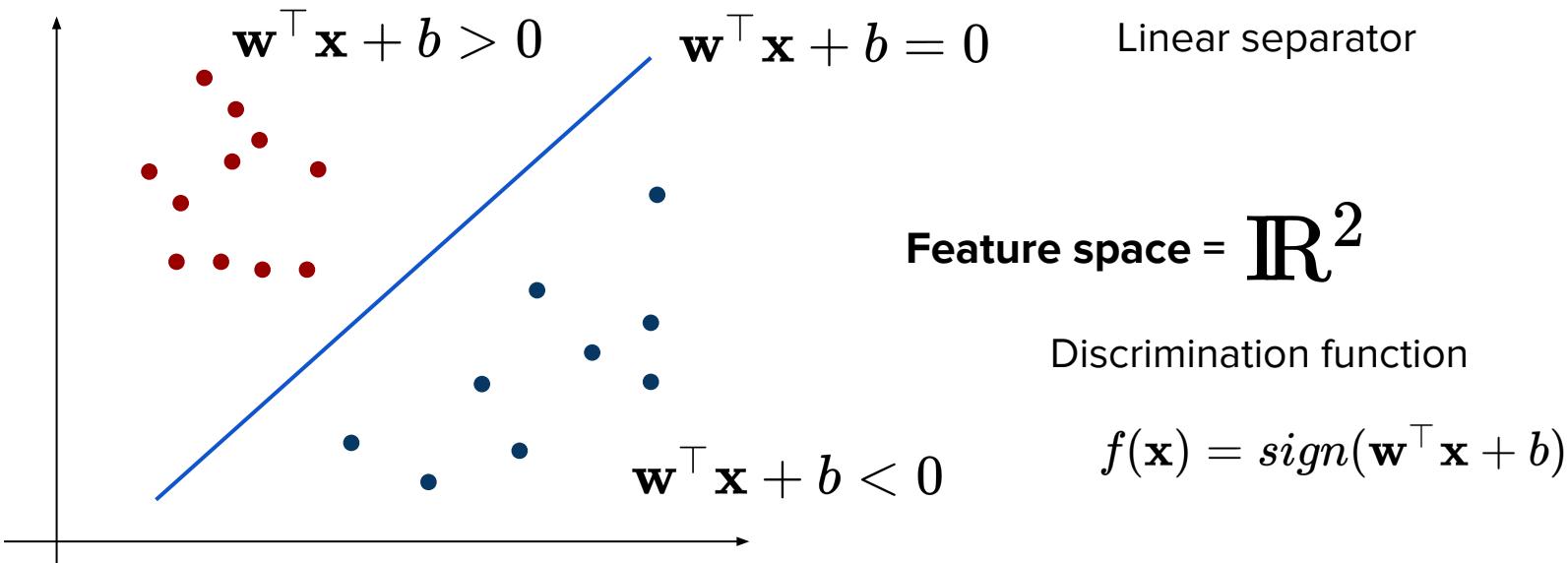
Linear classifiers

Binary classification can be viewed as the task of separating classes in some feature space:



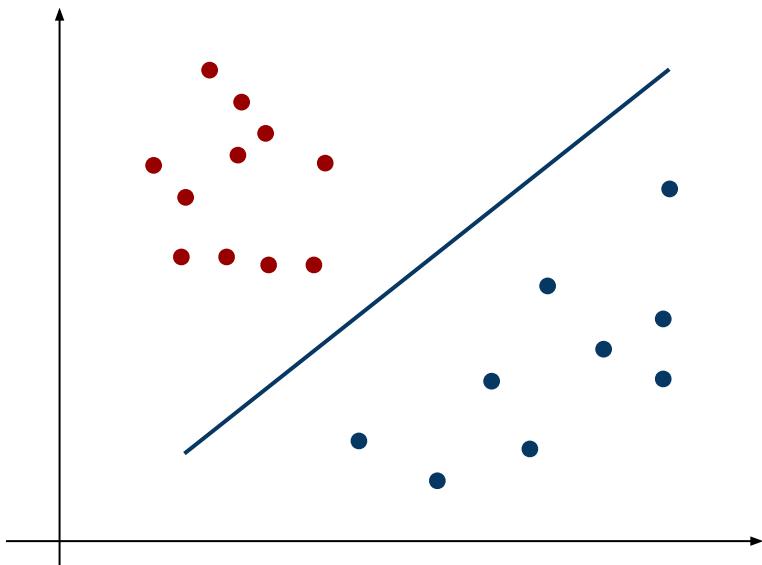
Linear classifiers

Binary classification can be viewed as the task of separating classes in some feature space:



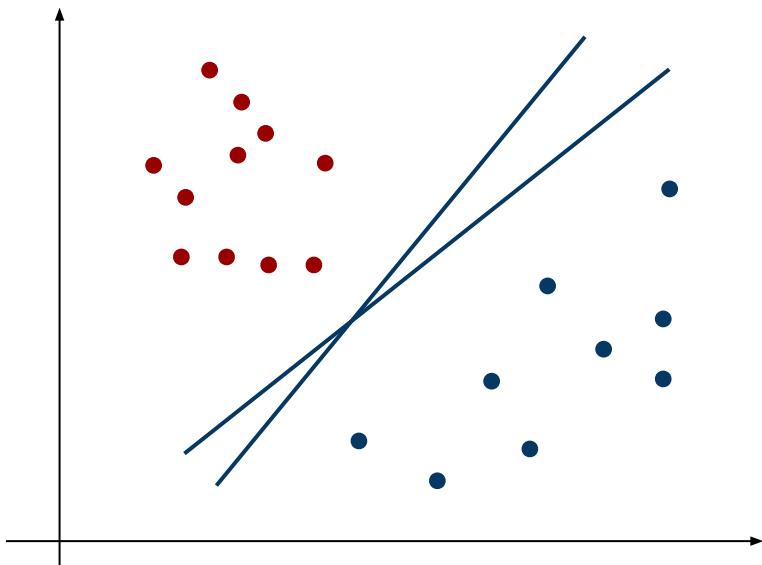
Linear separators

How many linear separators can be defined and which of them is optimal?



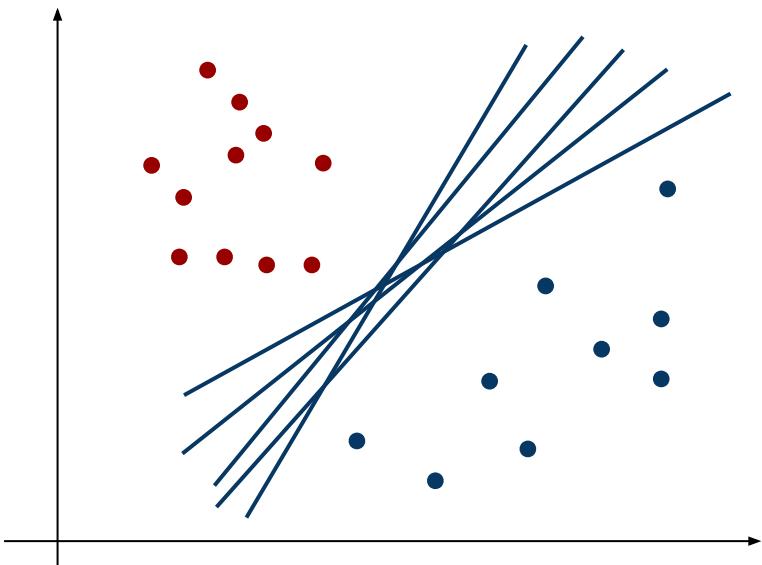
Linear separators

How many linear separators can be defined and which of them is optimal?



Linear separators

How many linear separators can be defined and which of them is optimal?

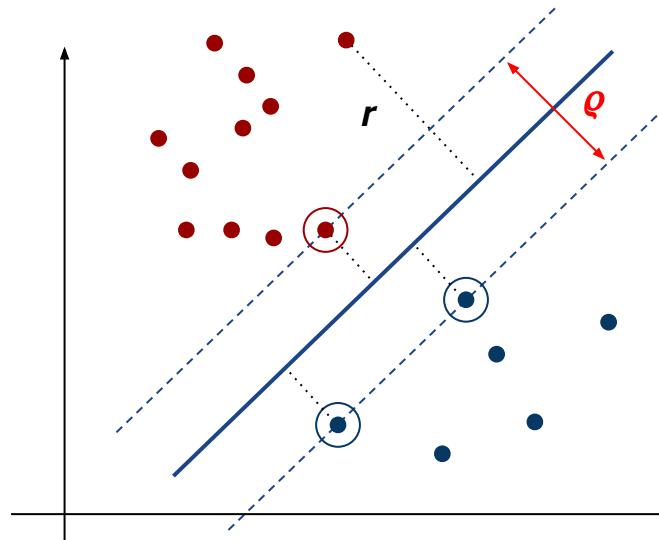


Classification margin

Distance from example \mathbf{x}_i to the separator is: $r = \frac{\mathbf{w}^\top \mathbf{x}_i + b}{\|\mathbf{w}\|}$

Closest points (belonging to the two classes) to the hyperplane are **support vectors**

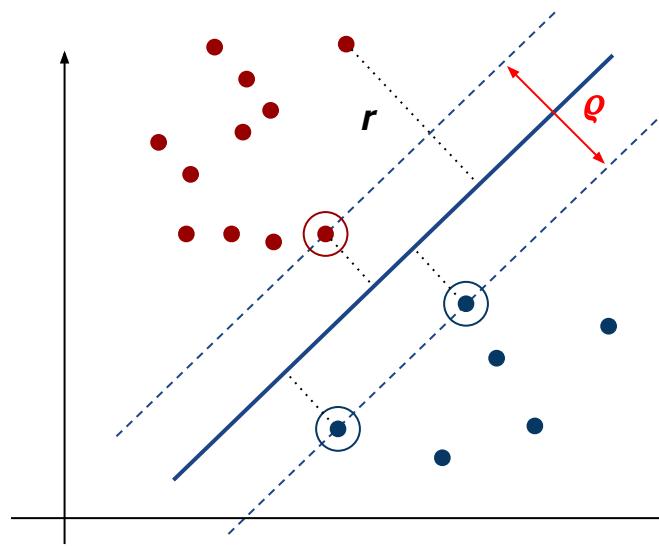
Margin ρ of the separator is the distance between support vectors belonging to the two classes



Maximum-margin classifiers

Optimality criterion: chose the hyperplane which **maximizes the margin**

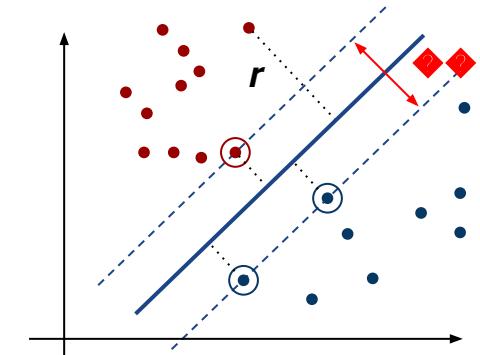
This implies that **only support vectors matter**, other training examples are ignorable



Linear SVM mathematically

Let D be a training set comprising n points in the d -dimensional space:

$$\begin{aligned}\mathcal{D} &= \{(\mathbf{x}_i, y_i) | i = 1, \dots, n\} \\ \mathbf{x}_i &\in \mathbb{R}^d, y_i \in \{-1, 1\}\end{aligned}$$



Suppose training points are separated by a hyperplane, then for each training example we have:

$$\begin{array}{lcl} \mathbf{w}^\top \mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^\top \mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq \rho/2$$

For every support \mathbf{x}_s the above inequality is an equality i.e.: $y_i(\mathbf{w}^\top \mathbf{x}_s + b) = \rho/2$

After rescaling \mathbf{w} and b by $\rho/2$, we obtain that the distance between a support vector and the hyperplane is:

$$r = \frac{y_i(\mathbf{w}^\top \mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$



Margin ρ can be expressed through (rescaled) \mathbf{w} and b

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Optimization Problem (OP)

We can formulate a **quadratic optimization problem**:

$$\begin{aligned} & \text{maximize} && \frac{2}{\|\mathbf{w}\|} \\ & \text{s.t.} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

Which is equivalent to the following

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{s.t.} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

Solving the OP

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{s.t.} && y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \forall i \end{aligned}$$

Need to optimize a **quadratic** function subject to **linear constraints**

The solution involves constructing a **dual problem** where a **Lagrange multiplier α_i** is associated with every inequality constraint in the primal (original) problem:

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad \text{over } \alpha \\ & \text{s.t.} && \alpha_i \geq 0 \quad \forall i \\ & && \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

The OP solution

Given a solution $(\alpha_1, \dots, \alpha_n)$ of the dual problem, where $\alpha_i > 0$ only for support vectors \mathbf{x}_s , the solution of the primal problem is:

$$\begin{aligned}\mathbf{w} &= \sum_s \alpha_s y_s \mathbf{x}_s \\ b &= y_k - \sum_s \alpha_s y_s \langle \mathbf{x}_s, \mathbf{x}_k \rangle \quad \text{for any } k \text{ s.t. } \alpha_k > 0\end{aligned}$$

The classification function is:

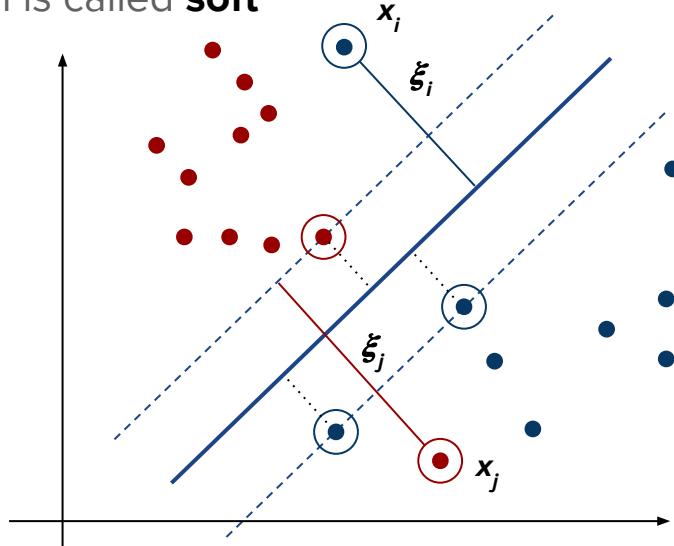
$$f(\mathbf{x}) = \sum_s \alpha_s y_s \langle \mathbf{x}_s, \mathbf{x} \rangle + b$$

Notice that it relies on **inner products** between the test point \mathbf{x} and support vectors \mathbf{x}_s - we will return to this later

Soft margin classification

What if the training set is not **linearly separable**?

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples. The resulting separation margin is called **soft**



Soft margin formulation

The hard margin formulation can be modified to incorporate slack variables:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{over } \mathbf{w} \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \xi_i \geq 0, \forall i \end{aligned}$$

Hyper-parameter C can be viewed as a way to control overfitting: it “trades off” the relative importance of margin maximization and fitting the training data (i.e. minimizing errors on training points)

High C value → narrow margin, low training error

Small C value → wide margin, high training error

Soft margin solution

Dual problem is identical to the separable case (except for the upper-bound on α_i):

$$\text{maximize}_{\alpha} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad \text{over } \alpha$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad \forall i$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

Solution of the primal problem is:

Discrimination function is:

$$\mathbf{w} = \sum_s \alpha_s y_s \mathbf{x}_s$$

$$b = y_k(1 - \xi_k) - \sum_s \alpha_s y_s \langle \mathbf{x}_s, \mathbf{x}_k \rangle \quad k \text{ s.t. } \alpha_k > 0$$

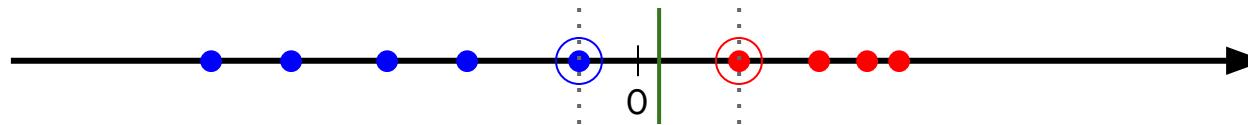
$$f(\mathbf{x}) = \sum_s \alpha_s y_s \langle \mathbf{x}_s, \mathbf{x} \rangle + b$$

Linear SVM summary

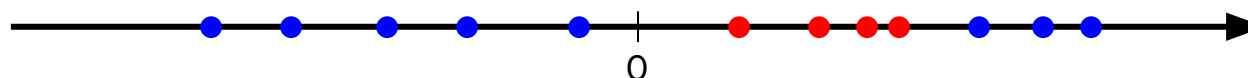
- The classifier is a **separating hyperplane** in the feature space
 - A point in \mathbb{R} , a line in \mathbb{R}^2 , a plane in \mathbb{R}^3 , a hyperplane in \mathbb{R}^d
- Most “important” training points are **support vectors**: they define the **maximum margin hyperplane**
- **Quadratic optimization** algorithms can identify which points are support vectors with **non-zero Lagrangian multipliers**
- Both in the dual formulation of the problem and in its solution training points appear only inside **inner products**

Non-linear SVMs

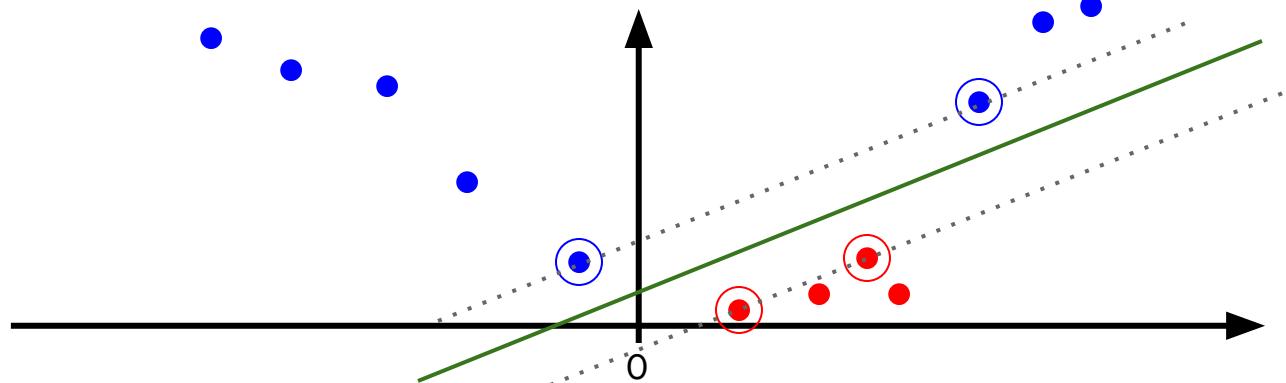
Datasets that are linearly separable (or with some noise) work out great:



But what are we going to do if the dataset is just too hard?

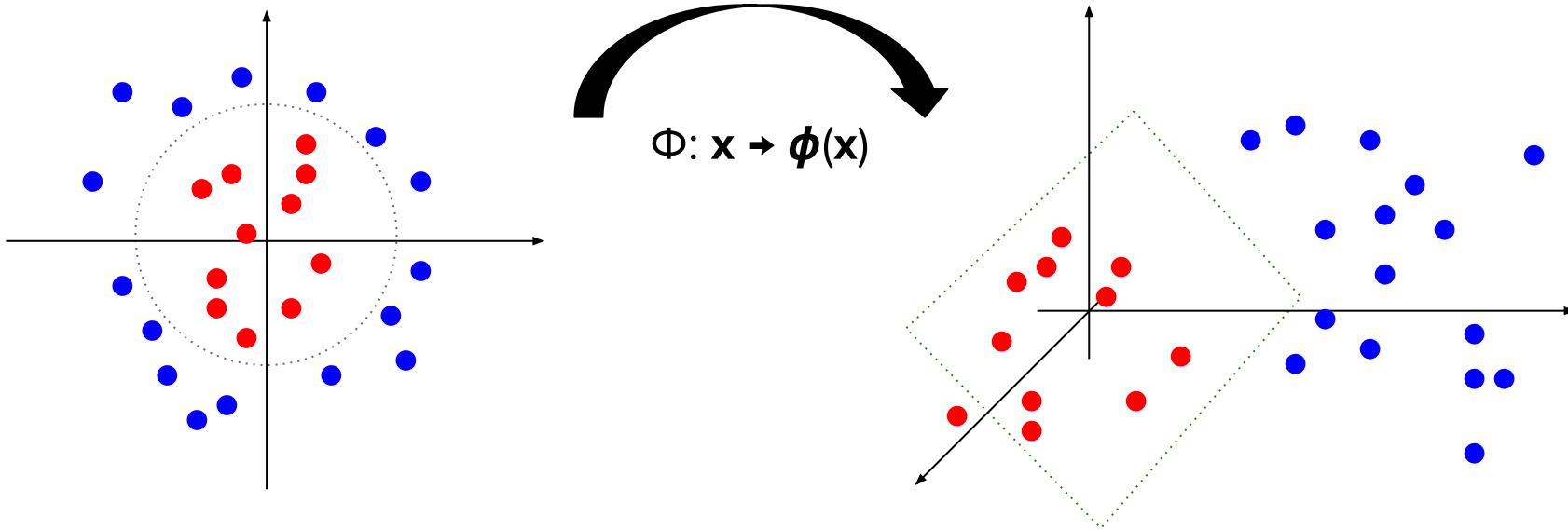


Mapping data to a **higher-dimensional space**:



Non-linear SVMs - Feature spaces

General idea: the original feature space can always be mapped to some **higher-dimensional feature space** where the training set is separable:



The “kernel trick”

The linear classifier relies on inner product between vectors

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

If every datapoint is mapped into high-dimensional space via some transformation Φ : $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

A kernel function is a function that is equivalent to an inner product in some feature space.

Thus, a kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\Phi(\mathbf{x})$ explicitly).

Examples of kernel functions

Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

- Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\Phi(\mathbf{x}) = \mathbf{x}$

Polynomial of power p: $K(\mathbf{x}_i, \mathbf{x}_j) = (a + b\langle \mathbf{x}_i, \mathbf{x}_j \rangle)^p$

- Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\Phi(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

- Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\Phi(\mathbf{x})$ is infinite-dimensional

Non-linear SVMs OP

Dual problem formulation:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{over } \alpha \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad \forall i \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

The solution is:

$$f(\mathbf{x}) = \sum_s \alpha_s y_s K(\mathbf{x}_s, \mathbf{x}) + b$$

Optimization techniques for finding α_i remain the same!

The scikit-learn library

<http://scikit-learn.org>

Machine-Learning library in Python based on numpy, scipy and matplotlib

Many different ML models implemented

- Supervised models for classification and regression (SVMs, SVRs, Random Forests, Logistic Regression, LASSO)
- Unsupervised models (k-Means, spectral clustering, bi-clustering)
- Dimensionality reduction (e.g. PCA, Non-negative Matrix Factorization)
- Feature selection algorithms

Model selection procedures (grid search, cross-validation, metrics)

Data pre-processing (normalization , feature extraction, transformations)

SVMs with scikit-learn

<http://scikit-learn.org/stable/modules/svm.html>

Different implementations for different problems:

- SVC : Support Vector Classification
- NuSVC : A different mathematical formulation of SVC
- SVR : Support Vector Regression

All based on LIBSVM

We will focus on multi-class SVC

Definition of a SVC model

The SVC constructor can be used to define a SVC model:

```
class sklearn.svm.SVC(C=1.0,  
                      kernel='rbf',  
                      degree=3,  
                      gamma='scale',  
                      coef0=0.0,  
                      probability=False)
```

SVC constructor parameters

| Name | Type | Default | Description |
|-------------|---------|---------|---|
| C | float | 1.0 | Penalty parameter C of the error term. |
| kernel | string | 'rbf' | Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid'. |
| degree | int | 3 | Degree of the polynomial kernel function ('poly'). Ignored by all other kernels. |
| gamma | float | 'scale' | Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. If gamma="scale", then it uses $1 / (n_features * X.var())$ as value of gamma. |
| coef0 | float | 0.0 | Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'. |
| probability | boolean | False | Whether to enable probability estimates. This must be enabled prior to calling fit, and will slow down that method. |

SVC class methods

fit(X, y)

Fit the SVM model according to the given training data.

Parameters: **X** : {array-like, sparse matrix}, shape (n_samples, n_features)
Training vectors, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples,)
Target class values

Returns: **self** : object
Returns self.

predict(X)

Perform classification on samples in X.

Parameters: **X** : array-like, shape (n_samples, n_features)

Returns: **y_pred** : array-like, shape (n_samples, n_classes)
Class labels for samples in X.

SVC class methods

decision_function(X)

Distance of the samples X to the separating hyperplane.

Parameters:

x : array-like, shape (n_samples, n_features)

Returns:

x' : array-like, shape (n_samples, n_classes * (n_classes-1) / 2)

Returns the decision function of the sample for each class in the model.
if decision_function_shape='ovr', the shape is (n_samples, n_classes)

predict_proba(X)

Compute probabilities of possible outcomes for samples in X.

The model need to have probability information computed at training time: fit with attribute probability set to True.

Parameters:

x : array-like, shape (n_samples, n_features)

Returns:

T : array-like, shape (n_samples, n_classes)

Returns the probability of the sample for each class in the model.

SVC class attributes

| Name | Type | Description |
|-------------------------------|--|--|
| <code>support_</code> | array-like, shape = [n_SV] | Indices of support vectors. |
| <code>support_vectors_</code> | array-like, shape = [n_SV, n_features] | Support vectors. |
| <code>n_support_</code> | array-like, dtype=int32, shape = [n_class] | Number of support vectors for each class. |
| <code>dual_coef_</code> | array, shape = [n_class-1, n_SV] | Coefficients of the support vector in the decision function. For multiclass, coefficient for all 1-vs-1 classifiers. |
| <code>intercept_</code> | array, shape = [n_class * (n_class-1) / 2] | Constants in decision function. |

WARNING: attributes have defined values only after training

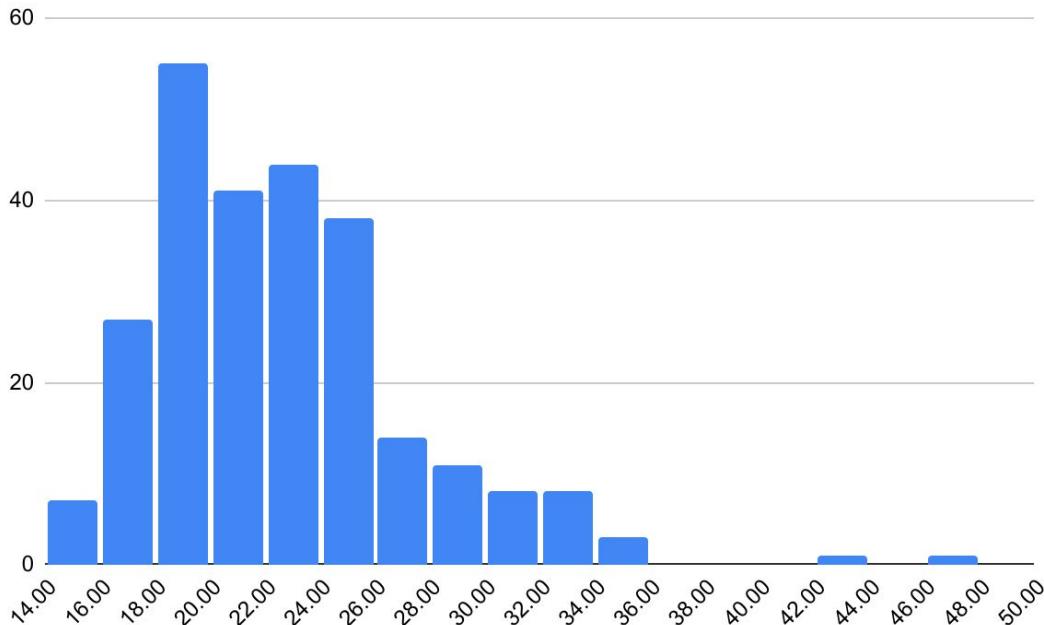
Example

[https://colab.research.google.com/drive/1tVosfu0OaAkAbJj_9wne5usAZwixEkLP?u
sp=sharing](https://colab.research.google.com/drive/1tVosfu0OaAkAbJj_9wne5usAZwixEkLP?usp=sharing)

Designing an SVM for SP detection

- Each sequence (either positive or negative) need to be represented as a point in some D-dimensional space
 - Extract features to represent each protein
- Which features can be discriminative for SP detection?
 - We can try to exploit properties we analyzed when performed statistics on the datasets
- **SPs follow a distinctive length distribution which can be exploited**
- **SP compositions are significantly different from background distributions**

SP length distribution

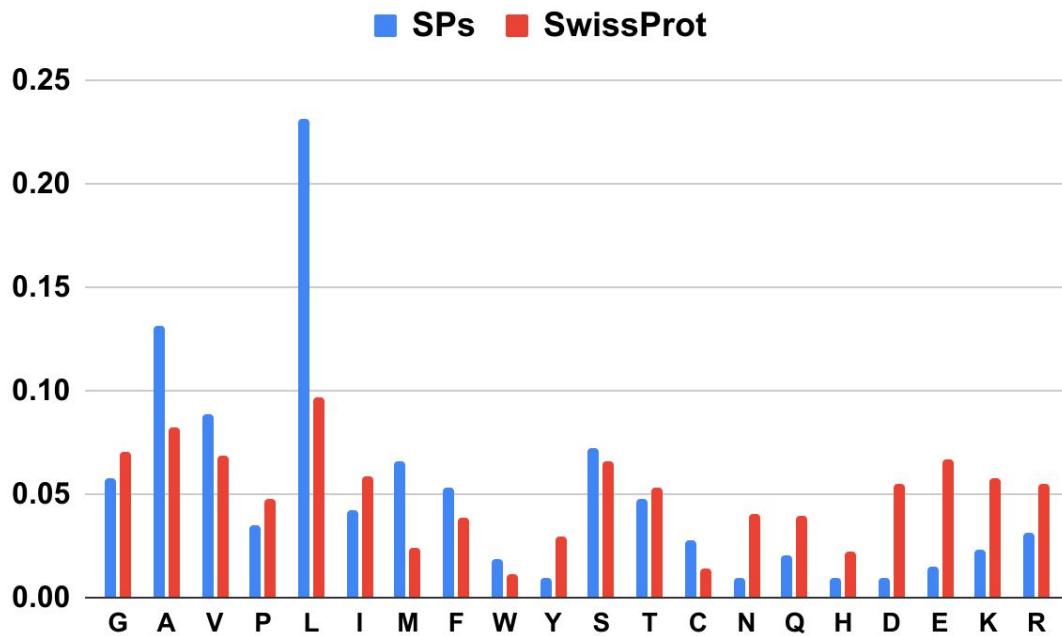


- Median length of SPs in the training set is 21
- Mean length of SPs in the training set is about 22 ± 5



We expect most of the SPs having a length between 19 to 24

SP sequence composition



- SP compositions differ from the background SwissProt distribution
- Non-polar residues are abundant in SP
- Polar and charged residues are less abundant



Composition of the N-terminal region can be used to distinguish SPs from non-SPs

Feature extraction: baseline feature

- Combining the two simple analysis above we can encode each protein by the composition of its N-terminus
- Each protein is encoded with a 20-dimensional vector corresponding to the composition of the first K=22 residues in the protein

MRFLAATFLLLALSTAAQAEPVQFKDCGSVDGVIKEVNVS PCPTQPCQLS

{

K=22



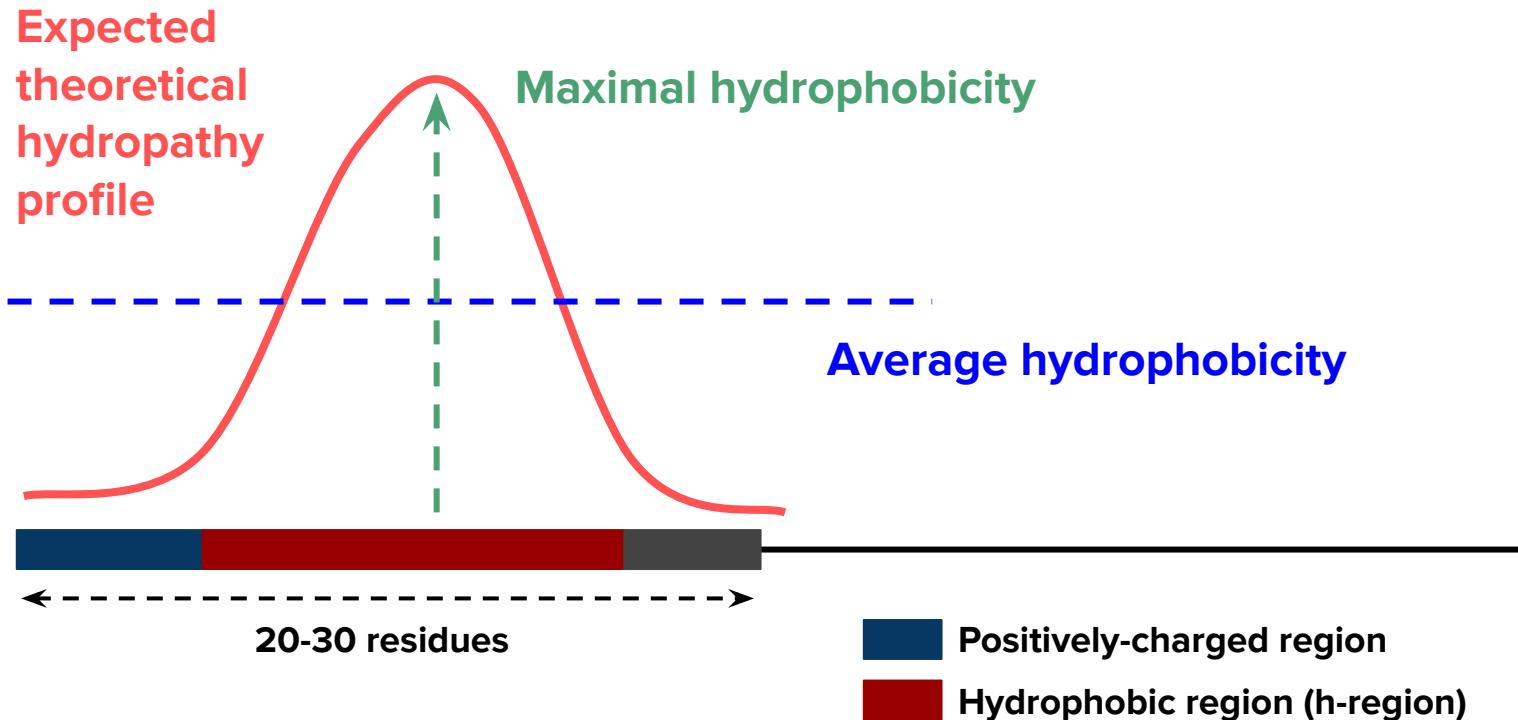
| | | | | | | | | | | | | | | | | | | | |
|---|-----|-----|-----|-----|---|-----|-----|---|---|-----|-----|---|---|-----|---|---|-----|---|-----|
| 0 | .25 | .04 | .04 | .21 | 0 | .04 | .13 | 0 | 0 | .04 | .08 | 0 | 0 | .08 | 0 | 0 | .04 | 0 | .04 |
| G | A | V | P | L | I | M | F | W | Y | S | T | C | N | Q | H | D | E | K | R |

Other features can be defined

- Hydrophobicity
- Charge
- Propensity to form secondary structures
- Propensity to form transmembrane helix
- Scales (e.g. hydropathy scale) can be retrieved from ProtScale or AAIndex
 - <https://web.expasy.org/protscale/>
 - <https://www.genome.jp/aaindex/>
- Additional features can be added (concatenated) to the baseline compositional feature, augmenting the feature vector of each sequence
- Different features can be added one by one to understand their contribution (feature selection)

Hydrophobicity features

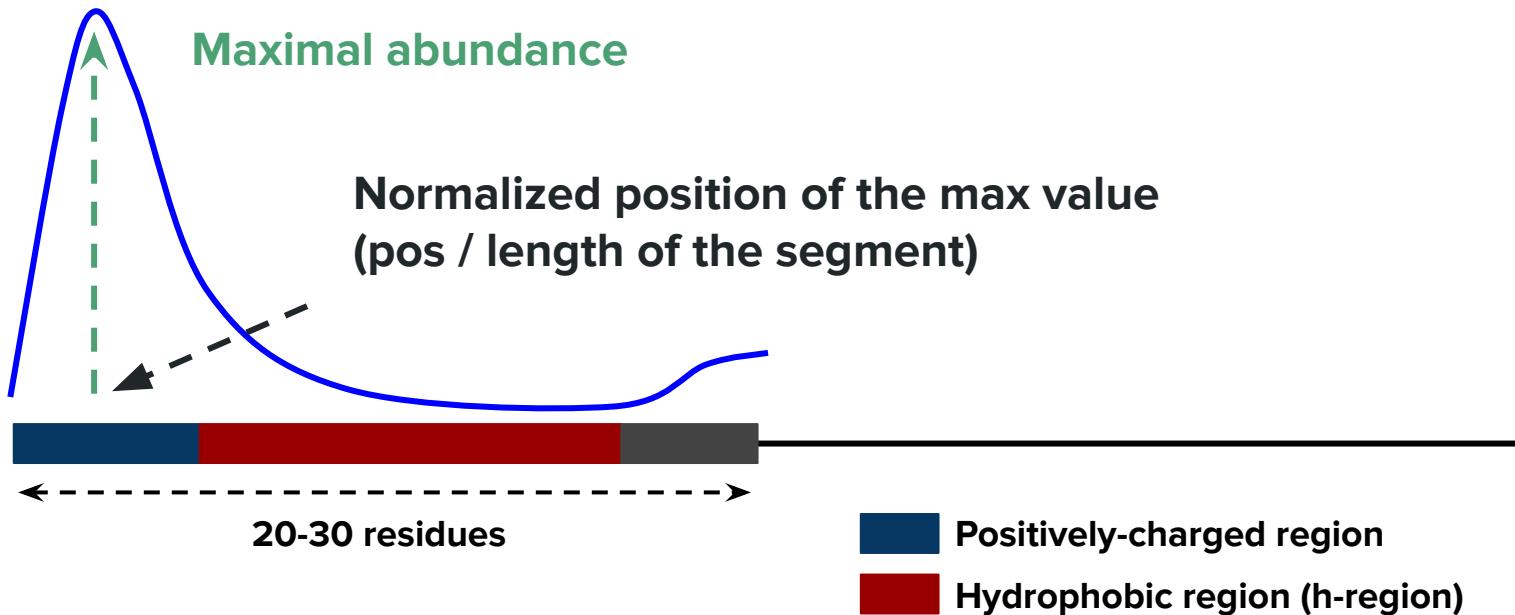
Kyte & Doolittle scale
Window size = 5



Charge features

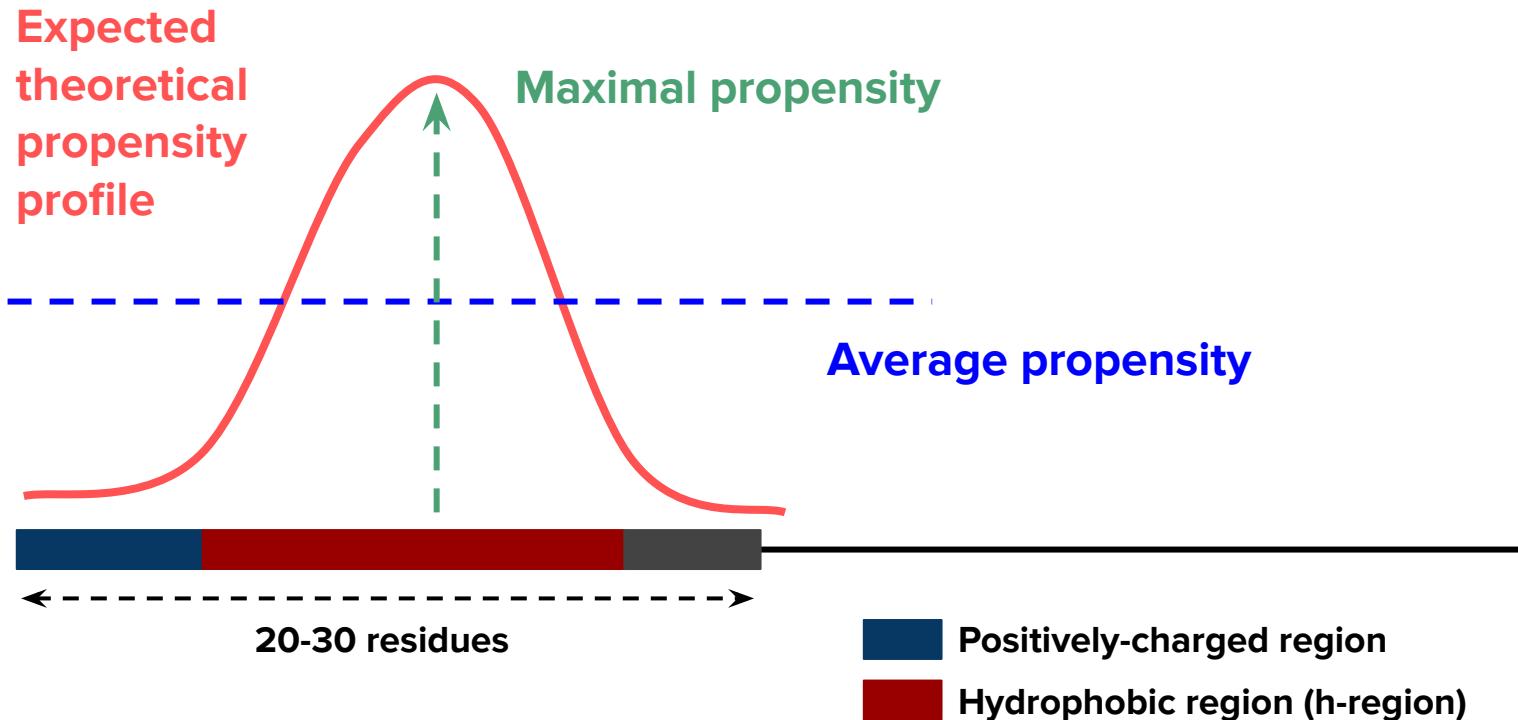
Window size = 3

Expected abundance of
positively-charged residues (K,R)



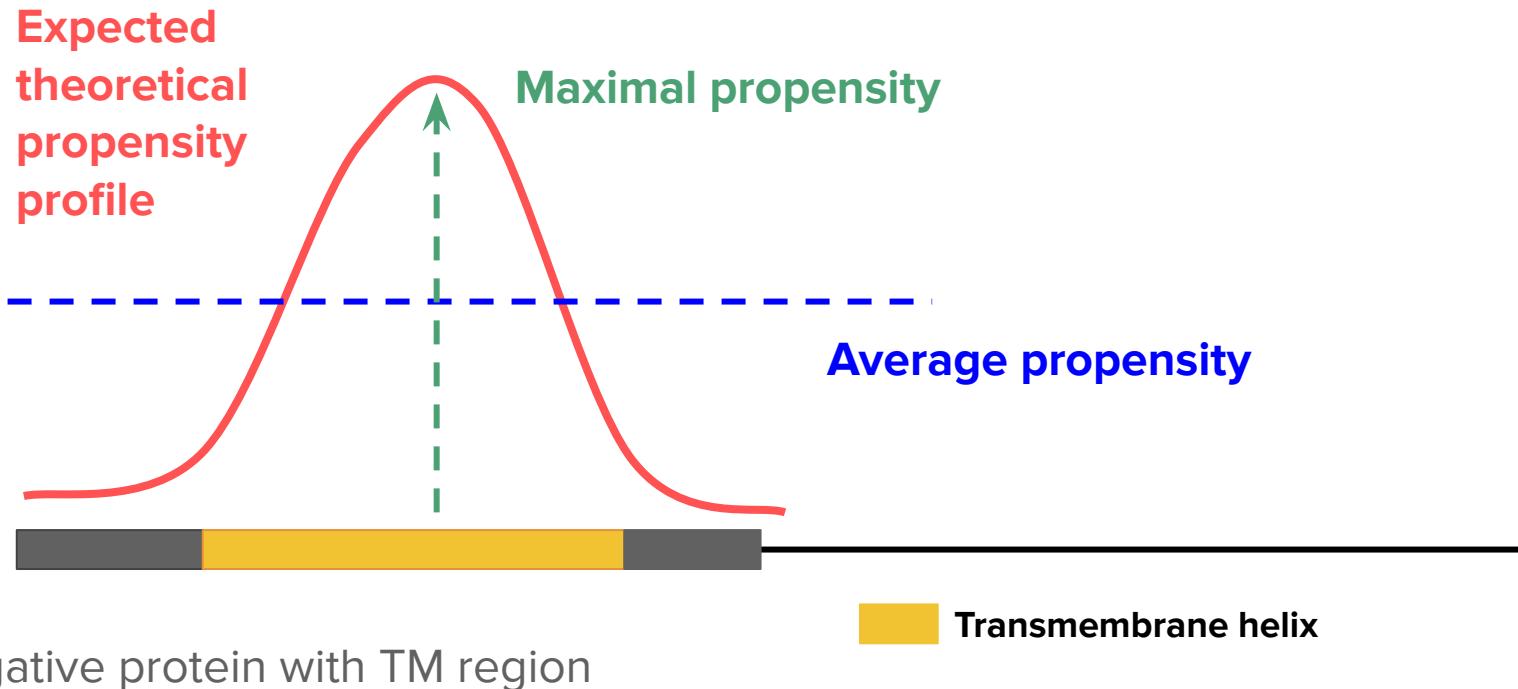
Alpha-helix propensity

Chou&Fasman scale
Window size = 7



Transmembrane propensity

Zhao&London scale
Window size = 7

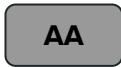


How to compute the features

- For each protein (either positive or negative), analyze a fixed N-terminal portion, e.g. the first 40 residues
- Compute hydropathy, charge, alpha-helix and transmembrane propensity profiles
 - Profiles can be computed using ProteinAnalysis from Biopython
 - <https://colab.research.google.com/drive/1fOSYCyAF9Xn0ZIKnQkYXH2a16K1B55jb?usp=sharing>
- Compute the feature values the corresponding profile:
 - Average and maximal HP (2 values)
 - Maximal charge and normalized position of the peak value (2 values)
 - Average and maximal alpha-helix propensity (2 values)
 - Average and maximal transmembrane tendency (2 values)

Test input combinations

Baseline only



Two features



Three features



All features



An SVM model for SP detection

- Sequences are points in the N-dimensional space corresponding to:
 - AA compositions of the first 22 residues
 - Additional features, if any
- Use non-linear SVMs with RBF kernel
 - C parameter to need to be optimized
 - RBF kernel parameter gamma need to be optimized
- For the two hyperparameters (C and gamma) run a grid-search with cross-validation

Grid search of SVM hyper-parameters

The two SVM hyperparameters C and γ (for the RBF kernel) need to be optimized

This is done through a so called **grid search** procedure:

- A set of possible values for each param is defined
- All possible combinations are tested for each cv run
- The combination achieving the highest performance on the validation set is then selected as the optimal one for that specific cv run, and fixed for testing
- The final value of each parameter is the most frequently selected during cross-validation

A minimal grid search

We will run a grid search considering 4 values for each SVM param

- $C = \{1, 2, 4, 8\}$
- $\gamma = \{0.5, 1, 2, \text{"scale"}\}$

Gamma set to “scale” in SKlearn means setting the value to:

`1 / (n_features * var(X))`

- During each cross-validation run, each value in the grid is obtained after with SVM models trained using the corresponding combination of C and γ and tested on the validation set
- The best combination is selected as the one providing the highest MCC on validation set

$$(K', C', \gamma') = \operatorname{argmax}_{K,C,\gamma} \overline{MCC}_{K,C,\gamma}$$

see: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

Example

| Training | Validation | Testing | Best C (validation) | Best γ (validation) |
|----------|------------|---------|---------------------|----------------------------|
| 2,3,4 | 1 | 0 | 1.0 | 0.5 |
| 0,3,4 | 2 | 1 | 4.0 | 0.5 |
| 0,1,4 | 3 | 2 | 2.0 | 0.1 |
| 0,1,2 | 4 | 3 | 2.0 | “scale” |
| 1,2,3 | 0 | 4 | 2.0 | 0.5 |

- Final selected hyperparameters: $C=2.0$, $\gamma=0.5$
- In case of tie: select the lowest C and γ values (would lead to a more general model)

Practical session IV

Implement the SVM-based approach using sklearn

- AA composition feature computed over the first 22 residues
- Other features computed over the first 40 residues
- Test different combinations of input features
- Kernel function: RBF kernel
- Hyperparameter optimization using cross-validation and grid-search
 - C: [1, 2, 4, 8]
 - gamma: [0.5, 1, 2, “scale”]

LB2 Project

Evaluation of classifiers

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Introduction

Classification problem: classify a set of observations into two or more categories

Examples:

- Spam filtering
- Diagnosis of patients
- Protein variants classification (benign or disease)
- Protein secondary structure prediction

Binary classification: only 2 classes or categories

Multi-class classification: 3 or more categories

Multi-class, multi-label: 3 or more classes, 1 or more labels per example

SP prediction is a binary classification problem

Evaluating binary classifiers (quantitatively)

Binary confusion matrix

| | | predicted class | |
|----------------|----------|-------------------------|-------------------------|
| | | positive | negative |
| observed class | positive | true positives (TP) | false negatives (FN) |
| | negative | false positives (FP) | true negatives (TN) |

Main binary scoring indexes:

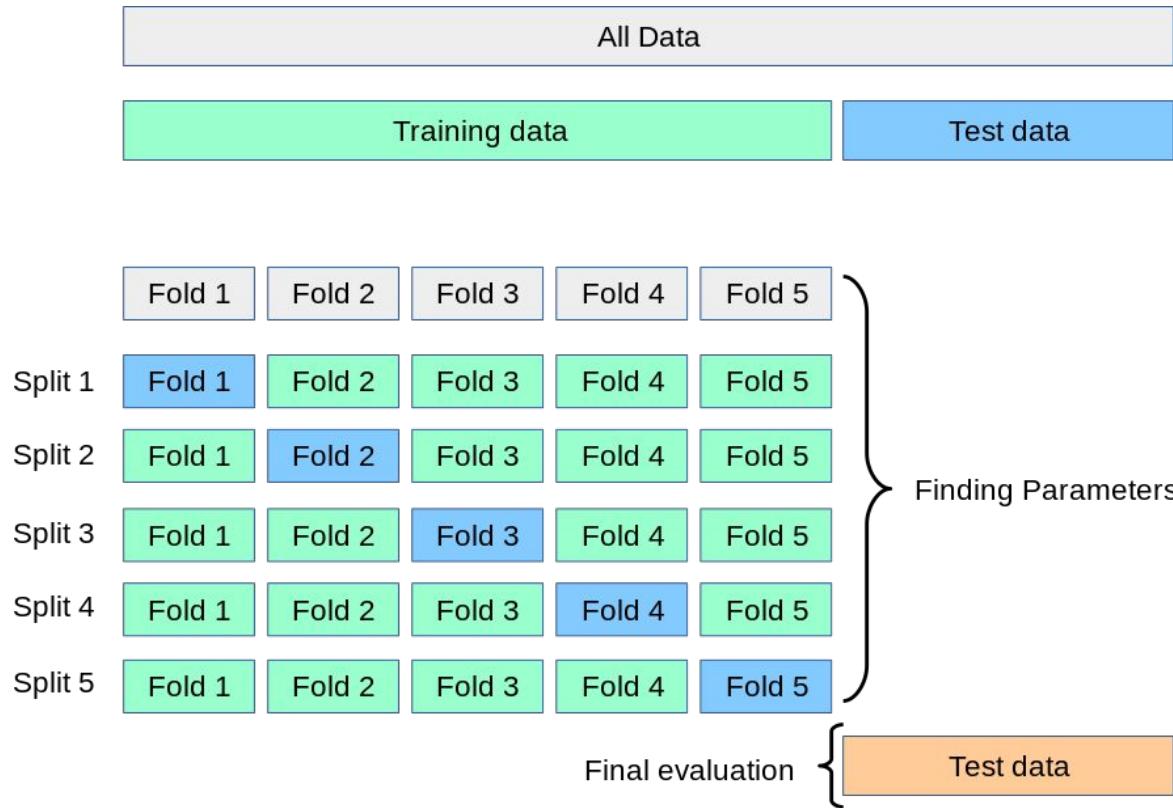
$$ACC = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Sen = \frac{TP}{TP+FN}$$

$$PPV = \frac{TP}{TP+FP}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$$

Evaluation procedure



Cross-validation

After each CV run you compute performance scores on the current testing set

Once the entire cross-validation procedure is completed the final value for each score is computed as the average value over the five CV runs

Average values must be accompanied by **standard errors**, defined as:

$$SE = \frac{\sigma}{\sqrt{n}}$$

where:

- σ is the **standard deviation**
- n is the number of samples, in your case 5

The standard error around the mean is reported using the \pm symbol e.g. **MCC = 0.54±0.02**

LB2 Project

Analysis of results

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Qualitative evaluation of results

Analyze prediction results in details to understand why each model makes mistakes in the discrimination of SPs from non SPs

Using the benchmarking data analyse

- False positive (FP) predictions: non-SP sequences predicted as SPs
- False negative (FN) predictions: SPs predicted as non-SPs

In general:

- Are the errors (both FP and FN) equally distributed among taxa/species?

False positive analysis

- Many errors are due to the presence of an hydrophobic transmembrane helix at the protein N-terminus
- To understand the impact of this situation:
 - Isolate FP predictions in the benchmark dataset
 - Using metadata extracted from UniProtKB, compute the fraction of FP predictions that have a transmembrane helix in the first 90 residues

False negative analysis

- For the von-Heijne method, FNs may be due to a different composition of the cleavage-site context than expected
 - For all FN in the benchmark, we can compute the corresponding LOGO and compare it with the expected one
- For the SVM method
 - FN may be due to a different composition of the N-terminal region (with the selected K) than expected
 - Compare compositions of positive examples from training sets up to K residues, benchmarking true positives up to K and benchmarking false negatives up to K
 - FN may be due to the presence of a longer/shorter SP than expected
 - Compare length distributions of positive training examples, benchmarking true positives and benchmarking false negatives
 - Are other features different?
 - Compare distributions of all features adopted.

Introduction to deep-learning methods

Multi-layer perceptrons

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Outline

- **Introduction**
- **Multi-layer perceptrons (MLPs)**
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Graph neural networks (GNNs)
- Attention mechanisms
- Transformers
- Protein language models
- Applications
- Conclusions

Deep learning at a glance

What is deep learning?

- Deep Learning models are neural networks comprising multiple layers of hidden units
- The multiple layers are used to automatically extract and select features of the input in order to compute complex input-output functions

What's really new in deep learning?

- Multi-layer perceptrons have been around for a long time
- The main novelty is our ability to adopt advanced training techniques to efficiently train such networks

Deep learning as representation learning

More formally, **deep learning** refers to machine-learning models adopting **several computational layers to represent data**

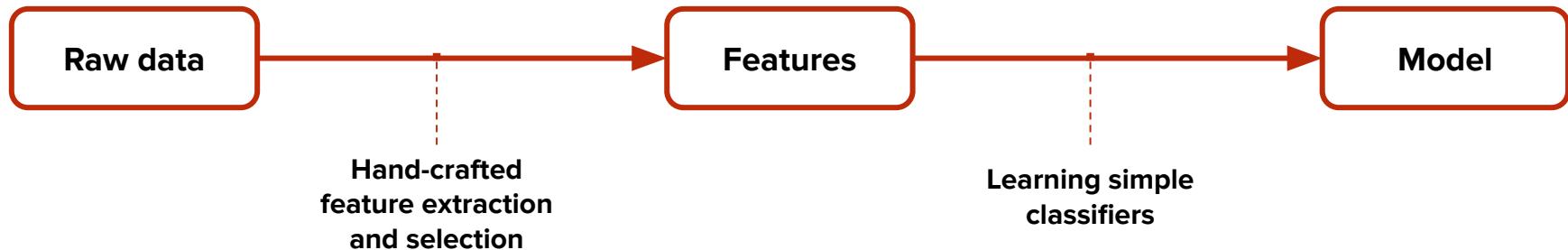
Each layer represents data at a different (incremental) **layer of abstraction**

The model architecture is **hierarchical**: the representation at **a given layer is built on top** of (i.e. abstracting from) representations at **previous layers**

Typically, models are **artificial neural networks** comprising **multiple non-linear hidden layers neurons**

Shallow learning

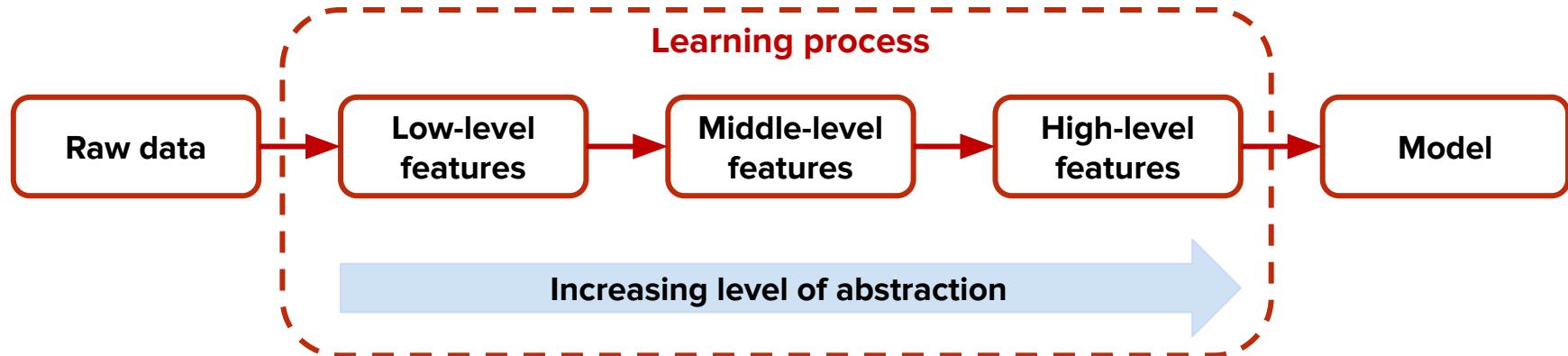
The usual workflow of shallow learning (i.e. any conventional ML)



From raw data we first manually extract features that are then used to train a simple classifier
This approach has the following limitations:

- Extracting **hand-crafted/engineered features** often requires strong domain knowledge and it is in general a time-consuming process
- The features are usually highly dependent on the specific application and cannot be easily “exported” to other applications

Deep learning



In deep learning the **feature extraction and selection** process is **part of the learning procedure**

The data representations are the ones that **best explain data** with respect to the specific detection or classification task

For classification task **higher-level features amplify aspects** of the input data that are important for **discrimination** and **suppress irrelevant features**

Examples of increasing levels of abstraction

Object recognition in images:

Pixel → edge → texton → motif → part → object

Natural Language Processing:

Character → word → word group → clause → sentence → story

Genomics:

Nucleotide → Short Motif → Long motif → TSS/Exon/Intron → Gene

Each level of representation builds on the less abstract representation in the previous layer

Why deep is better than shallow?

- The **Universal Approximation Theorem** states that any continuous function can be approximated (up to precision ε) by a neural network with a single hidden layer and **enough hidden units**
- So the question is: why adopting deep networks if single-layer are already so great?
- We can borrow some ideas from theoretical Computer Science, in particular from theory about **boolean circuit complexity**
- There are functions that will require a **huge number of hidden units** if we force the circuit to be **shallow**
- The same functions could be approximated with a **smaller number of units** if we allow the circuit to be **deeper**
- The famous example is the **parity function**, which is the generalization of the **XOR** problem to d inputs

Why deep is better than shallow?

The parity function is defined as follows:

$$\text{parity}(x) = \begin{cases} 1 & \text{if the number of 1s in } x \text{ is odd} \\ 0 & \text{if the number of 1s in } x \text{ is even} \end{cases}$$

It is easy to define a boolean circuit of depth $O(\log_2 D)$ with $O(D)$ -many gates for computing the parity function.

Each gate is an **XOR**, arranged in a complete binary tree

Why deep is better than shallow?

This shows that if you are allowed to be deep, you can construct a circuit that computes parity using a number of hidden units that is linear in the dimensionality of the input

Can you do the same with shallow circuits? The answer is no.

It is a result of circuit complexity that parity requires exponentially many gates to compute if a constant depth is forced.

The formal theorem is:

Theorem (Parity Function Complexity)

Any circuit of depth $K < \log_2 D$ that computes the parity function of D input bits must contain $O(2^D)$ gates.

Why deep is better than shallow?

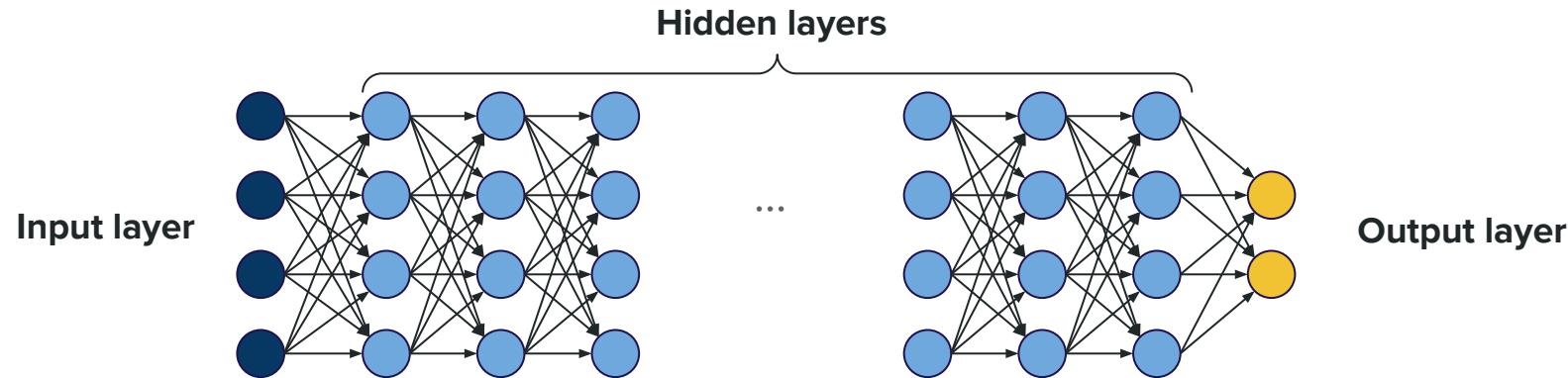
This is a very famous result because it shows that constant-depth circuits are less powerful than deep circuits

Although a neural network is not exactly the same as a circuit, it is generally believed that the same result holds for neural networks

In general, this gives a strong indication that depth might be an important consideration in neural networks

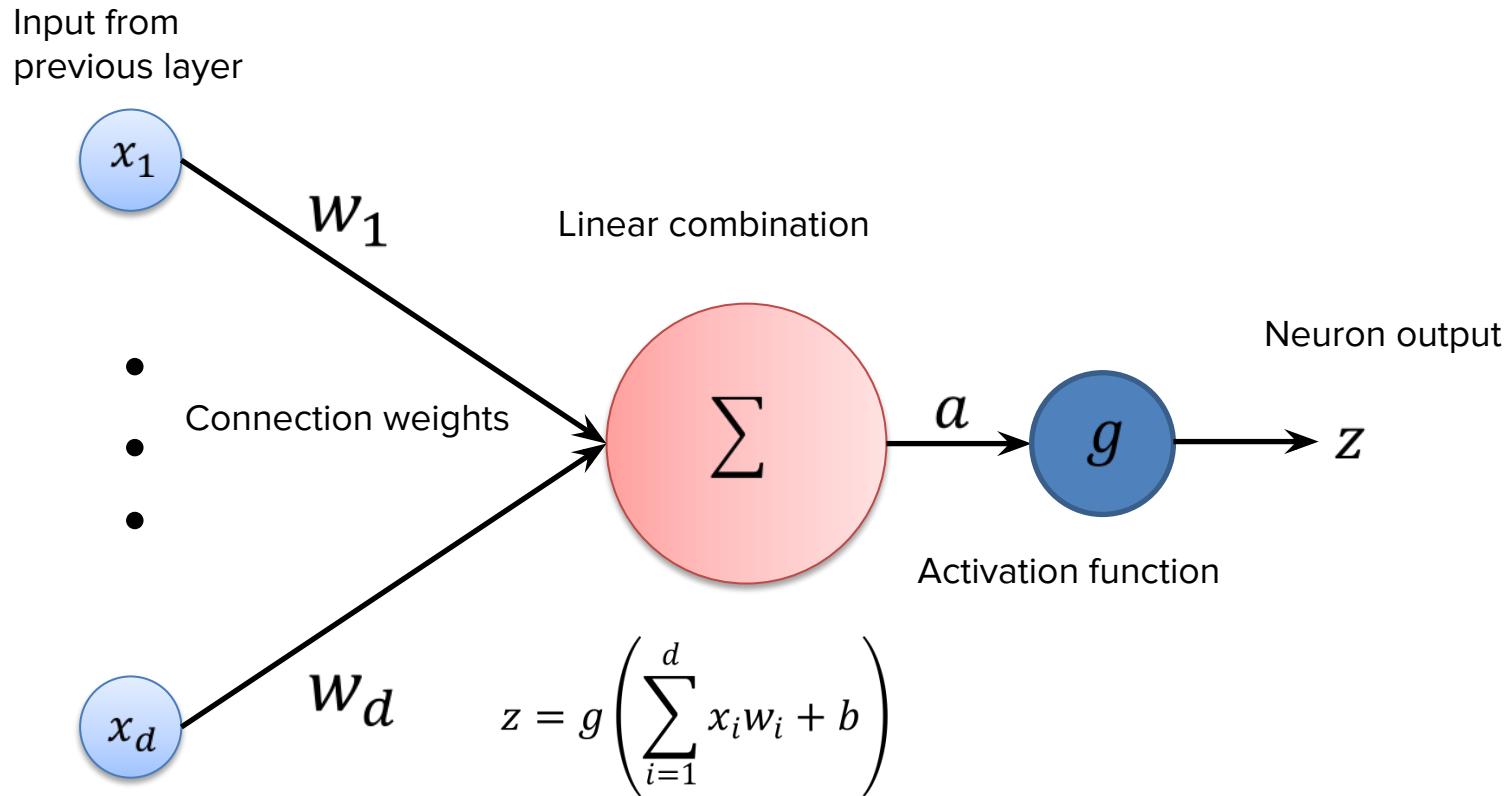
Multi-layer Perceptron (MLP)

The more general form of deep learning model



- Neurons at each layer are **fully connected** to neurons in the subsequent layer
- Information flows from input to output
- The **input layer** is fed with raw input data
- The **output layer** provides the required classification
- **Hidden layers** compute a feature representation of input data

Artificial neuron model



Training MLPs

In principle, MLPs can be trained by standard gradient descent and back-propagation

- No differences with shallow network training

In practice, standard techniques typically lead to poor results

Ultra-deep MLPs are trained using a mixture of the following techniques:

- **Stochastic Gradient Descent** (SGD) and variants (**RMSProp**, **ADAM**)
- **Unsupervised pre-training** to initialize the network weights
- **Rectified linear** activation functions
- Advanced regularization using **dropout** to control overfitting
- **Batch normalization:** speed-up training and reduce overfitting

Stochastic Gradient Descent (SGD)

In conventional gradient descent algorithm, weights w of the network are updated after processing n examples in the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$

$$w(t+1) = w(t) - \eta \frac{1}{n} \sum_{i=1}^n E(w(t), \mathbf{x}_i, y_i)$$

Very expensive and slow in the presence of large training sets

Stochastic Gradient Descent:

- Picks a random subset $\mathcal{M} \subset \mathcal{D}$ of $m \ll n$ (even $m=1$) training examples (a **minibatch**)
- Update network weights w after visiting the examples in the minibatch:

$$w(t+1) = w(t) - \eta \frac{1}{m} \sum_{i \in \mathcal{M}} E(w(t), \mathbf{x}_i, y_i)$$

Faster than standard gradient descent

Theoretical results highlighted good approximation properties with less strong dependence on the weight initialization

RMSProp

- One limitation of plain SGD is the choice of a global learning rate η for all the model weights

Root Mean Square Propagation: unpublished algorithm in the category of **adaptive learning rates methods**

- Maintain a **moving average of the squares of the gradient** for each weights during training
- Divide the learning rate by the square root of this average before weight update
- This ensures that the actual **learning rate is adapted** for each weight in the model
- Reduce oscillations in directions with steep gradients
- Faster movement in flat region of the error function landscape

Adaptive Moment Estimation (ADAM)

Related to RMSProp, but with some difference:

- Use both running average of first- and second-order moment of the gradient
- Update steps are estimated directly from running averages (not used to rescale the learning rate)
- ADAM adopt a bias correction procedure to avoid large update steps due to the initialization of the algorithm hyperparameters

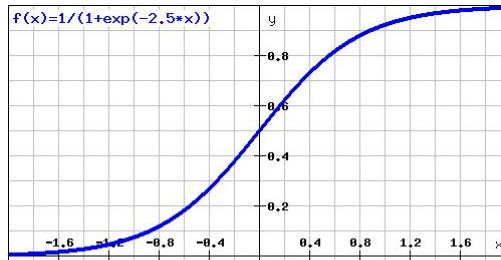
Training MLPs: pre-training

- In standard training procedures, all the weights of the network are randomly initialized
- Problem: using random init of MLPs we increase the probability of getting stuck into local non-optimal solutions
- Possible solution: run a pre-training of the network using only input data without targets (unsupervised training) in order to find a better starting point for training
- The pre-training is performed using a iterative approach training one layer at a time
- Finally, fine-tuning is performed using the pre-trained network on both input and targets (supervised training), adjusting the weights with respect to specific task at hand

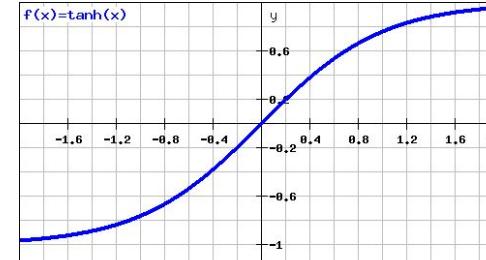
Vanishing (or exploding) gradients

- Activation functions such as tanh or logistic squash their inputs into small output ranges ([0,1] or [-1,1])
- The same happen to gradients which can easily become very small as long as they are back-propagated (through the chain rule) from output layer to front layers
- These vanishing gradients at front layers of the network sensibly slow down the network training
- Exploding gradient is the opposite situation and happens when the derivative of the activation function is expected to be much greater than 1 (e.g. with exponential activation function)

$$f(x) = \frac{1}{1 + e^{-x}}$$



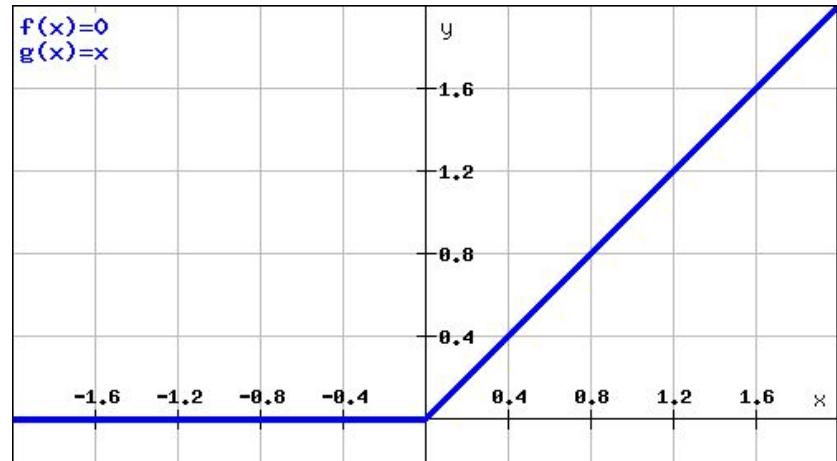
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



ReLU activation functions

Rectifier Linear Unit (ReLU)

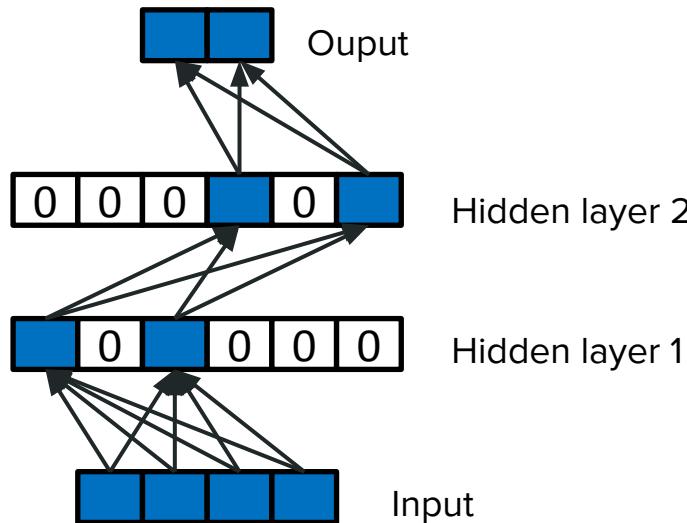
- The function value $f(x)$ is zero when $x \leq 0$
- When $x > 0$, $f(x) = x$ i.e. the function is linear
- Derivative is constant, either 0 or 1,
- Neither vanishing nor exploding gradients



$$f(x) = \max(0, x)$$

ReLU promote sparse networks

ReLU activation functions promote networks that are **sparse, i.e. many neuron outputs are zero**



- Any input produce a different sparse pattern of activation of the network neurons
- The computation is actually linear in the subset of non-zero neurons

Advantages of sparsity

- Sparsity better than density, provided it is carefully controlled
- **Dense networks:** small variations in the input affect most of the activations. **Sparse networks:** the set of non-zero features is less sensitive to small input variations
- Sparse representation are in general more likely to be linearly separable, simply because the information is embedded in a higher-dimensional space
- ReLU activations are cheaper to compute than logistic or tanh: no exponential need to be computed, gradients are constant
- One potential problem is the possible hard saturation (all activations are zero at a given layer) which may block gradient back-propagation
 - This situation rarely happens in real-world problems
 - In many cases it can be solved by augmenting the number of hidden units

Preventing overfitting

MLPs can learn very complex input-output relationships by means of their multi-layer representation of data

However, many of these complicated relationships will be the result of sampling noise, especially when we have limited training data

This situation is known as **overfitting** and it affects any machine-learning model

Many methods exist to prevent overfitting

- **Early stopping** of training as soon as performance on validation data starts to get worse
- **Regularization** to promote sparsity or in general to prevent network weights from assuming large values

Dropout to prevent overfitting

In the ideal case, assuming unlimited computation capability, the best way to completely avoid overfitting would be averaging the predictions over all possible settings of the parameters

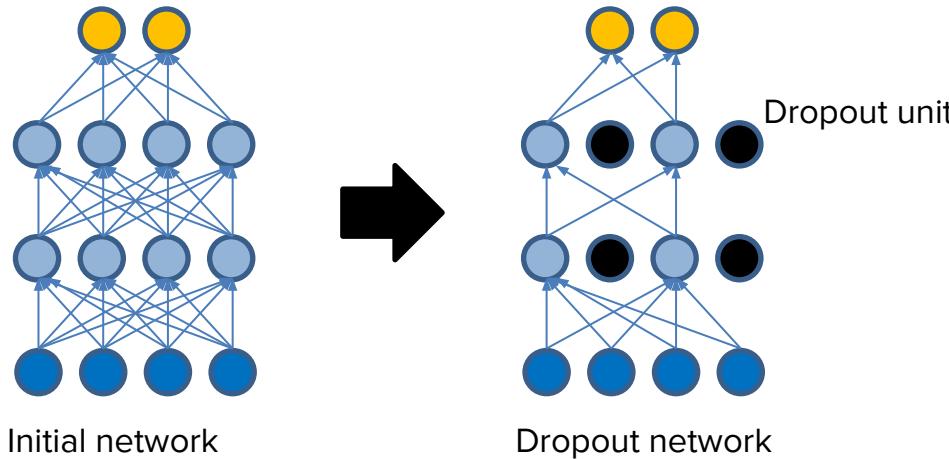
This solution is clearly impractical for large models which can assume a huge number of different architectures each with several millions of parameters

Dropout: a technique to efficiently approximate model averaging of an exponential number of different network architectures

How dropout works

Dropout means “dropping out” units in a neural network, namely temporarily removing them from the network along with all their incoming and outgoing connections

The choice of which unit to dropout is random. Commonly, each neuron is randomly dropped out with a probability of 0.5



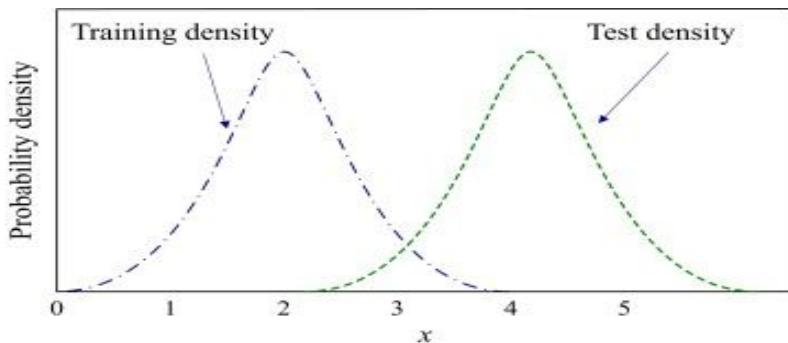
Why dropout works

- Applying **dropout means sampling** a simplified network architecture starting from the initial one
- The **simplified network** is the one consisting of **all units and connections that survived dropout**
- If the **initial network has n different units** the number of possible different **simplified networks contained in it is 2^n**
- **Dropout** is applied during training **before each weight update**
- Training the initial network with dropout can be seen as **training the entire collection of all 2^n network** (of course only a few of them will be actually selected and trained)
- One disadvantage is that **dropout increases training time** by 2-3 times because of the random sampling over the network architecture
- **Trade-off** between **training time** and amount of **overfitting**

Covariance shift

Covariance shift: the change in the distribution of input variables of a learning system e.g. between training and testing data

It is the result of having finite set of training and testing data i.e. sampling noise



Example:

Task: recognize cats in images

Train data: B/W pictures of cats

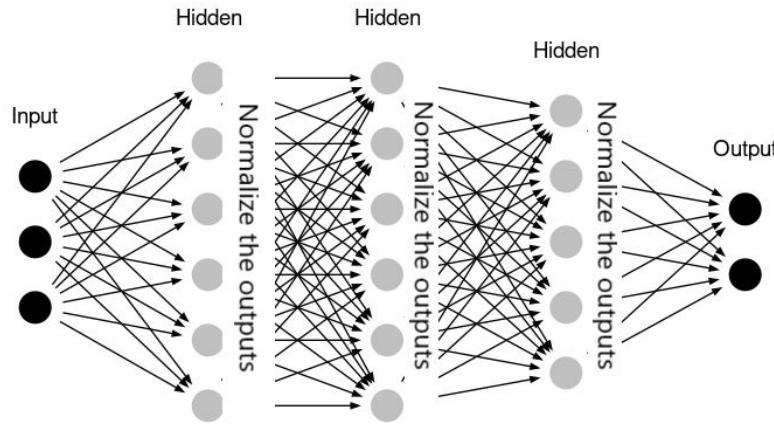
Test data: Colored pictures of cats

Network training converges much faster if all inputs are linearly transformed to have zero mean and unit variances

Batch normalization

Internal covariance shift: the change of the distributions of internal nodes in a neural network during training → increases training time because the learning have to readjust weights to compensate for the change in the distributions

Batch normalization normalizes outputs of hidden layers by subtracting the mean and dividing by the standard deviation computed over the **batch**



Advantages of batch normalization

- Reduce or **eliminate internal covariance shift**
- Allow the adoption of **activation functions different than ReLU** e.g. sigmoid because it prevents training to get stuck in the saturating regimes of nonlinearities leading to vanishing or exploding gradients
- The adoption of **higher learning rates** → reduces training time
- Training is more resilient to parameter scale
- **Reduce overfitting**: the output of a layer for a given network input varies across training because it is influenced by other training examples in the mini-batch → the network output is non-deterministic for a given input example, producing a regularization effect
- For the reason above, **BN reduces the need of dropout** (but better to have both)

Summary of MLPs

- MLPs are powerful models to learn complicated relationships between inputs and outputs
- Training MLPs is hard because of the greater number of layers
- Advanced training methods help us in training deep MLP
- Stochastic Gradient Descent allows reducing training time while obtaining good solutions
- Unsupervised pre-training is used to initialize network weights
- Rectifier linear units solve vanishing gradients and promote sparsity of the solution, leading to more manageable networks
- Dropout allows reducing overfitting by sampling different network architectures during training
- Batch norm reduces internal covariance shift and speed up training

Examples

Prediction of helix residues

- https://colab.research.google.com/drive/1nSf9yLtHp42B-9B0qM99yN2TY_t5xxA?usp=sharing

SCOP class prediction

- <https://colab.research.google.com/drive/1mYq4htkp96TxeXf-pUq2bZKAdK4DqmsT?usp=sharing>

References

1. LeCun *et al.* (2015) Deep Learning, *Nature*, 521: 436-444. <https://doi.org/10.1038/nature14539>
2. Bengio *et al.* (2013) Representation learning: a review and new perspectives, *IEEE Trans. Pattern Analysis and Machine Intell.*, 35: 1798-1828
<https://doi.org/10.1109/tpami.2013.50>
3. Bottou *et al.* (2007) The Tradeoffs of Large Scale Learning, In *Proc. of NIPS* 20, 161-168. <https://dl.acm.org/doi/10.5555/2981562.2981583>
4. Kingma & Ba (2007) Adam: A Method for Stochastic Optimization , arXiv 1412.6980. <https://doi.org/10.48550/arXiv.1412.6980>
5. Bengio *et al.* (2006) Greedy layer-wise training of deep networks, In *Proc. of NIPS* 19, 153-160. <https://dl.acm.org/doi/10.5555/2976456.2976476>
6. Glorot *et al.* (2011) Deep sparse rectifier linear units, In *Proc. of 14th ICML*, 315-323. <https://proceedings.mlr.press/v15/glorot11a.html>
7. Srivastava *et al.* (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *JMLR*, 15: 1929-58.
<https://dl.acm.org/doi/10.5555/2627435.2670313>
8. Ioffe & Szegedy (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariance Shift, arxiv, 1502.03167v3
<https://doi.org/10.48550/arXiv.1502.03167>

Introduction to deep-learning methods

Convolutional Neural Networks

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Outline

- Introduction
- Multi-layer perceptrons (MLPs)
- **Convolutional neural networks (CNNs)**
- Recurrent neural networks (RNNs)
- Graph neural networks (GNNs)
- Attention mechanisms
- Transformers
- Protein language models
- Applications
- Conclusions

Image processing

Classification



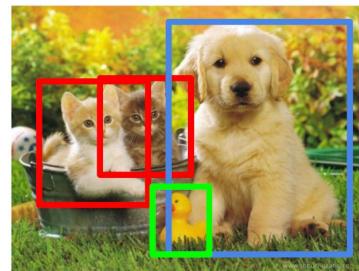
CAT

Classification
+ Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance
Segmentation



CAT, DOG, DUCK

Each image is a grid of pixels, each of which is encoded with three numbers representing pixel intensities in the Red-Green-Blue (RGB) channels

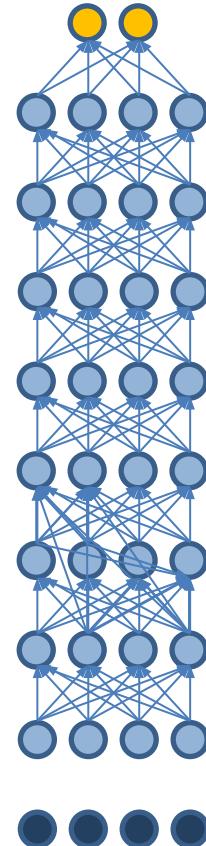
MLPs for high-dimensional data

The MLP architecture described so far connects each neuron in a given layer with ALL neurons in the previous (incoming) and the subsequent (outgoing) layer, i.e. the network has a **fully-connected architecture**

What if the input is a high-dimensional object, e.g. a bi-dimensional RGB image of $64 \times 64 \times 3 = 12288$ pixels?

Number of training parameters with N_0 inputs and N_1, \dots, N_H hidden neurons on each layer is:

- #params = $(N_0+1)*N_1 + (N_1+1)*N_2 + \dots + (N_{H-1}+1)*N_H + (N_H+1)*2$
- **The number of parameters depends on the input size and can be very large**



Detecting objects in images

Object features are **translation invariants**, i.e. an object can be placed anywhere in the input image

- The detection of the object should not depend on the position of the image where the object is present

Low-level features are **local**: adjacent groups of inputs form distinctive local patterns that need to be detected

- Small edges delimiting an object are defined looking at few neighbouring pixels

Higher-level image parts are obtained by combining lower-level patterns

These parts are **coarse**: the relative position/orientation of a low-level local patterns can vary somewhat, hence their combination should take care of these variations

Convolutional Neural Networks (ConvNets)

A special architecture of MLP designed to cope with image detection specificity

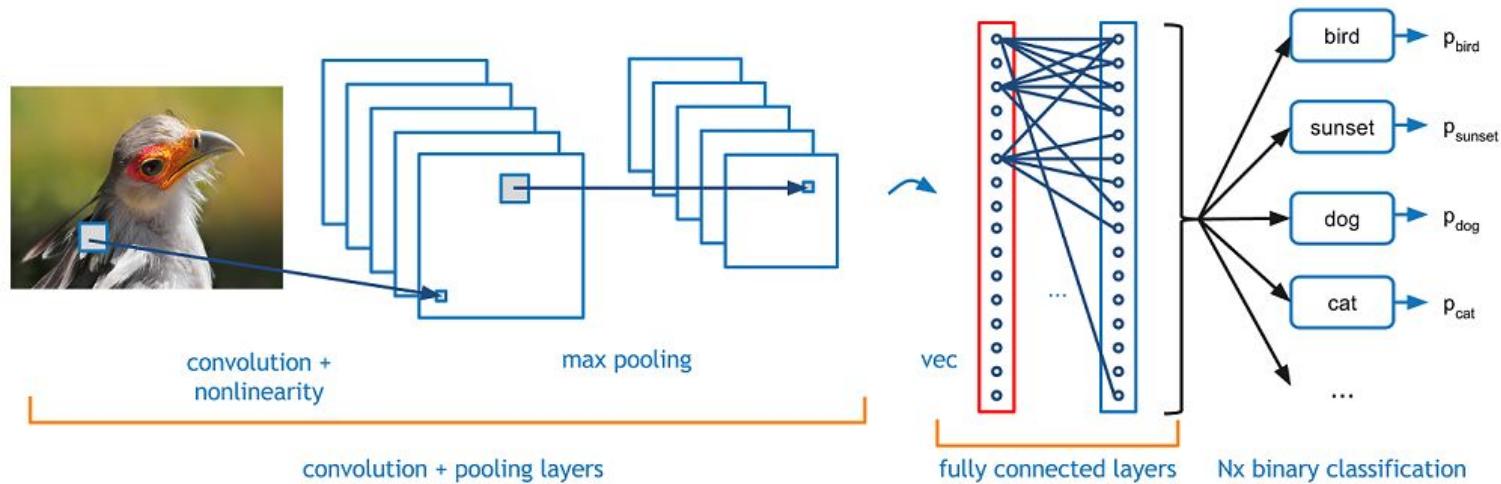
Not limited to image processing but also used for other types of data such as sequences

Translation invariant: ConvNets exploit this property by replicating the same neurons over different parts of the image

Local-features: Each neuron is connected to **local patches** of the previous layer

Coarse-grained high-level features: ConvNets adopt **pooling layers** to average over local patches

Convolutional Neural Networks (ConvNets)



The first part of the network comprises two kinds of hidden layers:

- **Convolution Layers** performing local detection of patterns
- **Pooling Layers** computing average/max over regions of convolution layer outputs

The final output is computed using a fully-connected network

Biological inspiration in ConvNets

The architecture of a ConvNets is inspired by the functioning of the visual cortex in mammals

According to the Hubel and Weisel model, the visual cortex contains neurons that individually responds to small regions of the visual field known as **receptive fields**

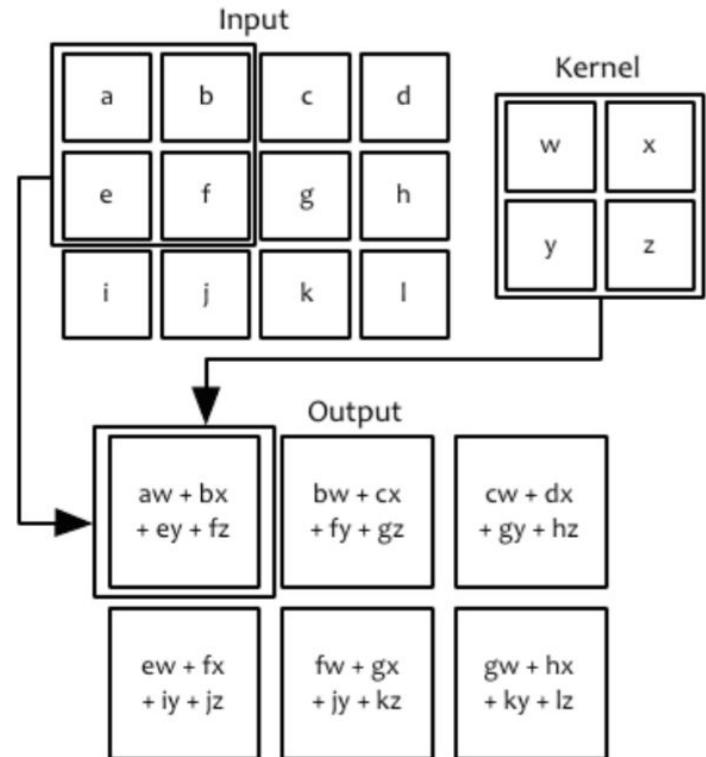
Two basic visual cell types in the brain:

- **Simple cells** whose output is maximized by edges having particular orientation in the receptive field (acting as convolution filters)
- **Complex cells** which responds to larger receptive fields and receive inputs from many simple cells (acting as pooling neurons)

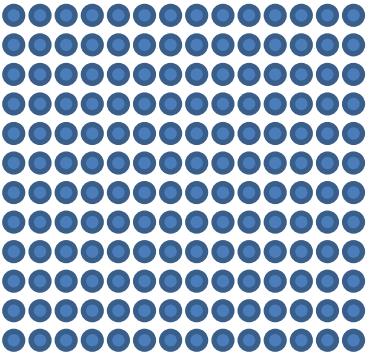
Convolution operator

- Input: an image (2D array) x
- Convolution filter/operator (2D array of learnable parameters): w
- Convolution operation in 2D domains:

$$s[i, j] = (x * w)[i, j] = \sum_{m=-M}^M \sum_{n=-N}^N x[i + m, j + n]w[m, n]$$

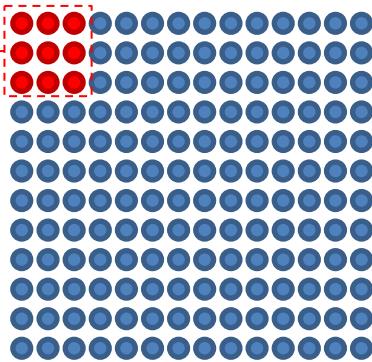


Convolution layer

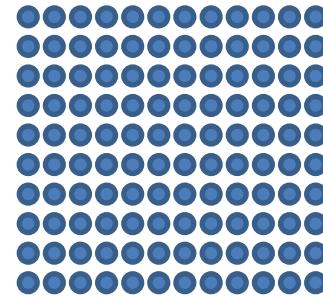


Input image

Convolution layer



Input image

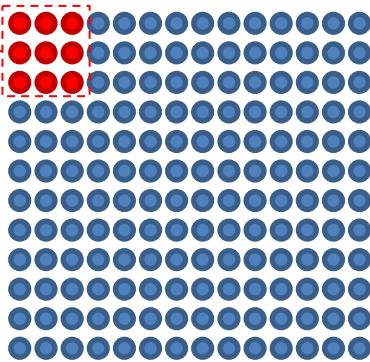


Convolutional layer

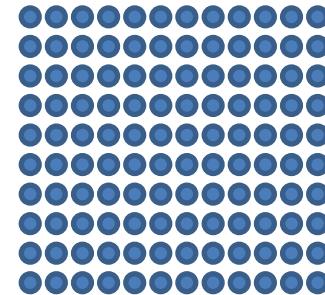
| | | |
|-----|-----|-----|
| 0.5 | 0.2 | 2.1 |
| 0.6 | 1.2 | 2.3 |
| 3.4 | 1.9 | 2.1 |

Local image patch

Convolution layer



Input image



Convolutional layer

| | | |
|-----|-----|-----|
| 0.5 | 0.2 | 2.1 |
| 0.6 | 1.2 | 2.3 |
| 3.4 | 1.9 | 2.1 |

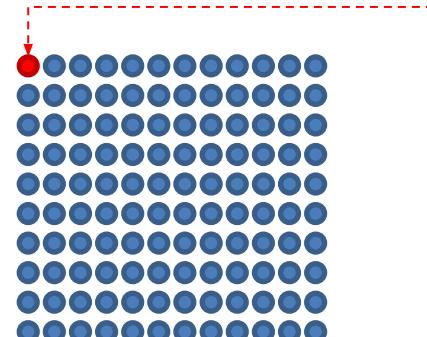
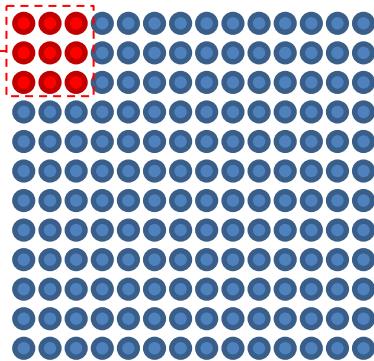
Local image patch



| | | |
|-----|-----|-----|
| 0.1 | 0.2 | 0.5 |
| 0.2 | 0.9 | 0.1 |
| 0.4 | 0.7 | 0.8 |

Local filter or detector (learned during training)

Convolution layer



| | | |
|-----|-----|-----|
| 0.5 | 0.2 | 2.1 |
| 0.6 | 1.2 | 2.3 |
| 3.4 | 1.9 | 2.1 |

X

| | | |
|-----|-----|-----|
| 0.1 | 0.2 | 0.5 |
| 0.2 | 0.9 | 0.1 |
| 0.4 | 0.7 | 0.8 |

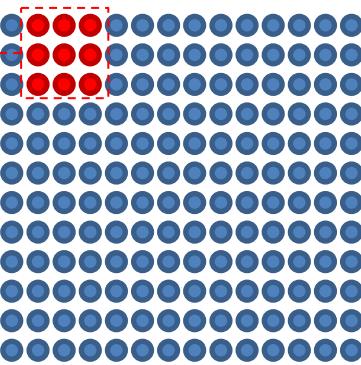
=

6.94

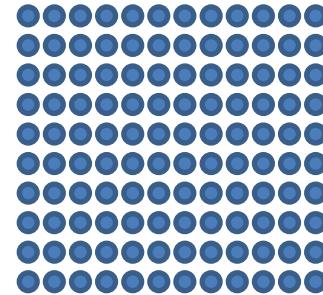
Local image patch

Local filter or detector (learned during training)

Convolution layer



Input image

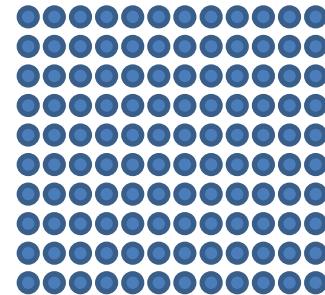
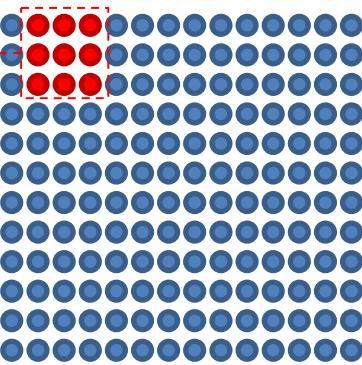


Convolutional layer

| | | |
|-----|-----|-----|
| 0.2 | 2.1 | 1.4 |
| 1.2 | 2.3 | 2.6 |
| 1.9 | 2.1 | 0.7 |

→ Next local image patch

Convolution layer



| | | |
|-----|-----|-----|
| 0.2 | 2.1 | 1.4 |
| 1.2 | 2.3 | 2.6 |
| 1.9 | 2.1 | 0.7 |

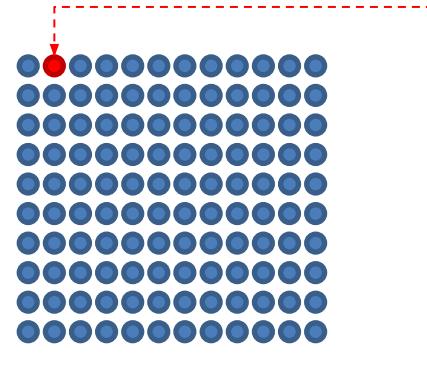
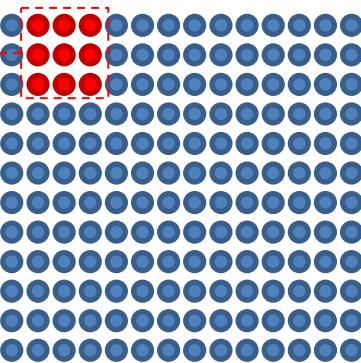
X

| | | |
|-----|-----|-----|
| 0.1 | 0.2 | 0.5 |
| 0.2 | 0.9 | 0.1 |
| 0.4 | 0.7 | 0.8 |

→ Next local image patch

Same local filter or detector

Convolution layer



| | | |
|-----|-----|-----|
| 0.2 | 2.1 | 1.4 |
| 1.2 | 2.3 | 2.6 |
| 1.9 | 2.1 | 0.7 |

X

| | | |
|-----|-----|-----|
| 0.1 | 0.2 | 0.5 |
| 0.2 | 0.9 | 0.1 |
| 0.4 | 0.7 | 0.8 |

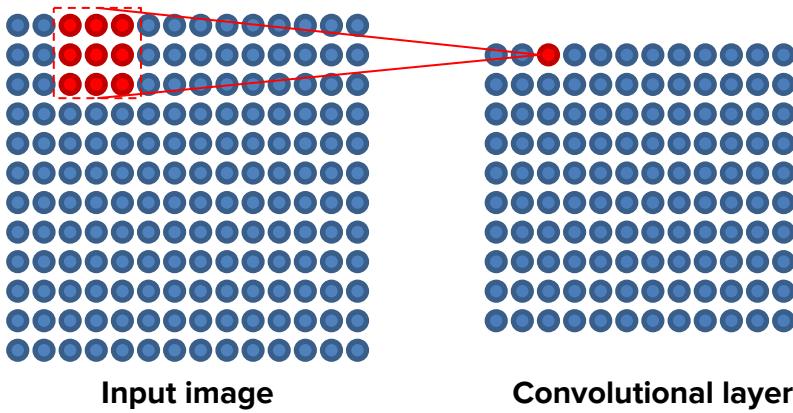
=

6.49

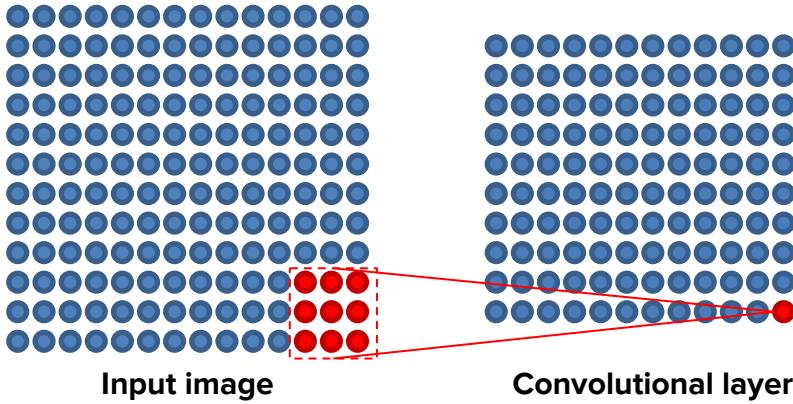
→ Next local image patch

Same local filter or detector

Convolution layer

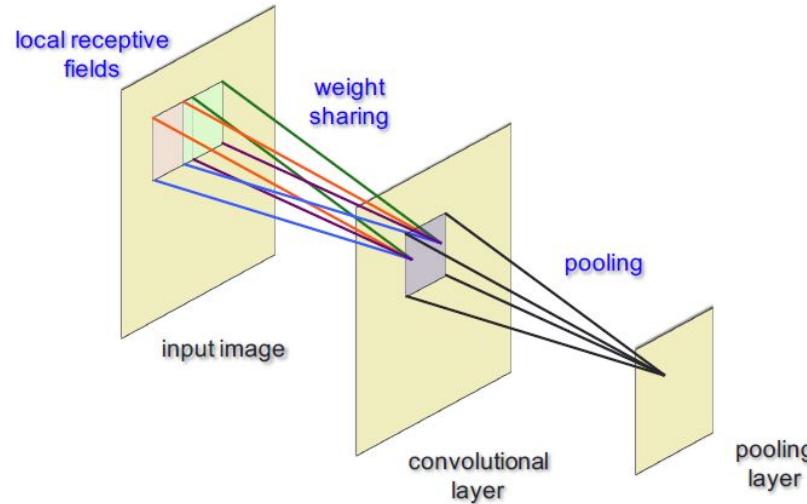


Convolution layer

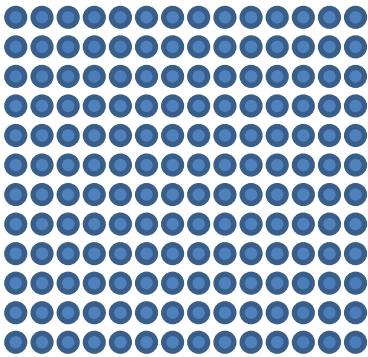


Pooling

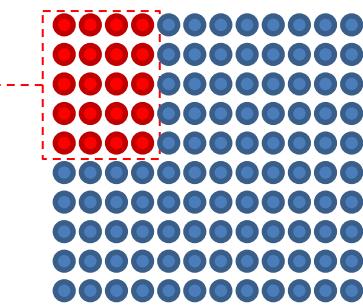
- Pooling: aggregation of neighboring neurons that were previously convolved
- Common pooling operations:
 - **Max pooling**: reports the maximum output within a rectangular neighborhood.
 - **Average pooling**: reports the average output of a rectangular neighborhood.



Convolution layer



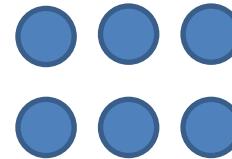
Input image



Convolutional layer

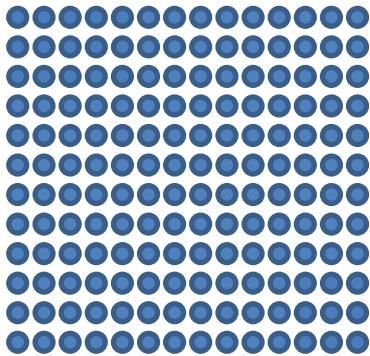
| | | | |
|------|------|------|------|
| 6.94 | 6.49 | 6.5 | 6.78 |
| 7.2 | 7.4 | 7.1 | 6.9 |
| 7.1 | 7.5 | 7.0 | 6.8 |
| 7.5 | 7.3 | 7.32 | 6.9 |
| 6.89 | 6.9 | 6.21 | 6.52 |

Local patch

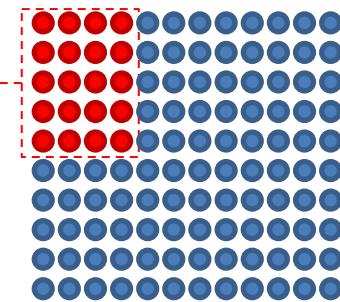


Pooling layer

Convolution layer



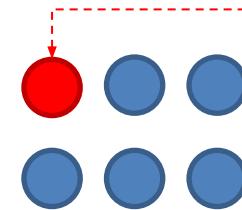
Input image



Convolutional layer

| | | | |
|------|------|------|------|
| 6.94 | 6.49 | 6.5 | 6.78 |
| 7.2 | 7.4 | 7.1 | 6.9 |
| 7.1 | 7.5 | 7.0 | 6.8 |
| 7.5 | 7.3 | 7.32 | 6.9 |
| 6.89 | 6.9 | 6.21 | 6.52 |

Local patch

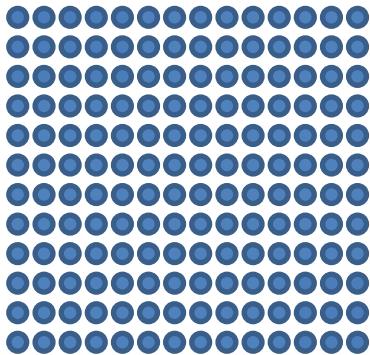


Pooling layer

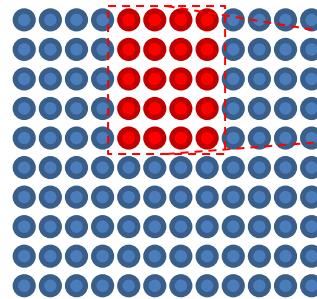
7.5

Computing average
or max value

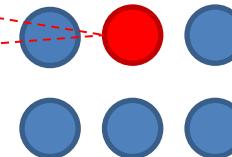
Convolution layer



Input image

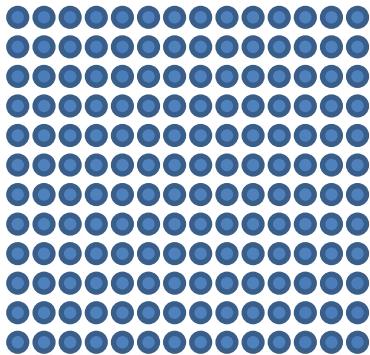


Convolutional layer

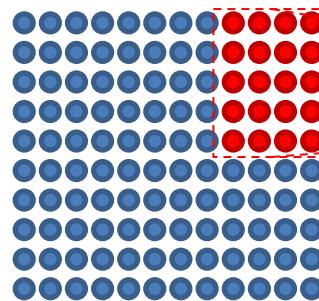


Pooling layer

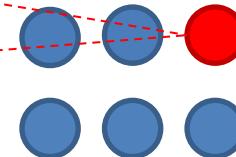
Convolution layer



Input image

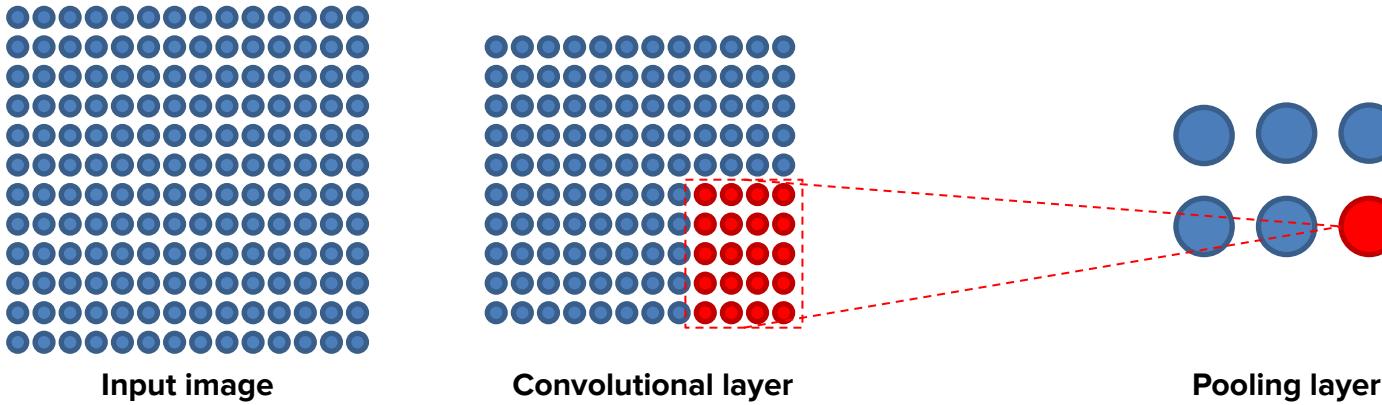


Convolutional layer

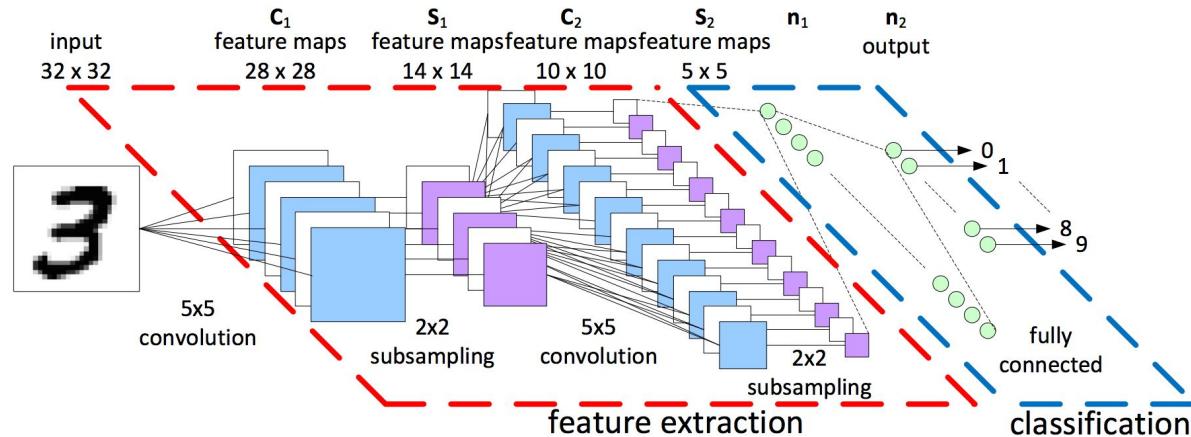


Pooling layer

Convolution layer



Deep Multi-Layer ConvNet



- Each convolution layer applies several local filters in parallel to detect different patterns in the output of the preceding layer
- Convolution/Pooling are typically interleaved
- The final mapping to high-level concepts or classes is always performed using a standard fully-connected network
- During learning, the weights of the filters as well as the weights in the fully-connected layer are estimated
- The number of parameters depends on the number of layers, the size and number of filters and the number of neurons in the final fully-connected layer. **No dependency on the input size.**

Deep ConvNet vs MLPs

A Deep ConvNet has much less trainable parameters than a (corresponding) MLP

Example: image size $32 \times 32 = 1024$ input pixels

ConvNet with one output node with the following architecture:

- Conv[50 filters of shape 3x3] + Pool[2x2]: output matrix of dimension of $15 \times 15 \times 50 = 11250$ total neurons
- Conv[25 filters 2x2] + Pool[2x2]: output dimension of $7 \times 7 \times 25 = 1125$ total neurons
- Fully-connected layer of 10 neurons to 1 output

MLP with 3 layers:

- 1024 input units
- One fully-connected layer with 11250 neurons (approximating the first conv-pool)
- One fully-connected layer with 1225 neurons (approximating the second conv-pool)
- One fully-connected layer with 10 neurons to 1 output

#params of the ConvNet is: **18696**

#params of the MLP is: **25607271**

The number of parameters in the ConvNet does not depends on the primary input size!

Training Deep ConvNet

The same strategies described for MLPs are used to train Deep ConvNets:

- SGD
- Pre-training (not strictly required in the context of ConvNets)
- ReLU
- Dropout
- BatchNorm

Typically training a Deep ConvNet requires less computational time than training a MLP

Other issues are:

- **Padding** to proper handling of variable-size images
- **Optimization of the filter sizes** in different layers
- Choosing the proper pooling function: **average/max/min/sum?**

ConvNet for other types of data

ConvNets are very powerful in image processing where the input is bi-dimensional

However, the same ConvNet architecture can be applied to many different input domains

1D ConvNets for sequences or time-series

- Input data as well as convolution filters are unidimensional
- Convolution filters are essentially a sliding windows **detecting sequence motifs** along the input sequential data

3D ConvNets for videos (2 spatial dimensions + time) or molecules (3 spatial dimensions)

Multi-dimensional ConvNet are also possible even if they can be challenging to train

1D ConvNets for sequence data

The input is now unidimensional, e.g. a nucleotide or protein sequence of length L

Each position of the sequence is represented with a vector of d channels, e.g.:

- Nucleotides with zero-one vectors of 4 channels encoding the presence of one specific nucleotide
- Proteins encoded with the usual 20-dimensional vectors

Filters can be seen as sequence motif detectors of a given length

Front conv layers extract short sequence motifs from the input sequences

Subsequent conv-pool layers progressively select and combine motifs into higher-order features

Example of prediction tasks in computational biology: prediction of binding sites, presence/absence of targeting and/or signal peptides in proteins, protein subcellular localization, prediction of effect of mutations from sequence/structure

1D ConvNet: example

Input sequence

TGGCAGATGACT

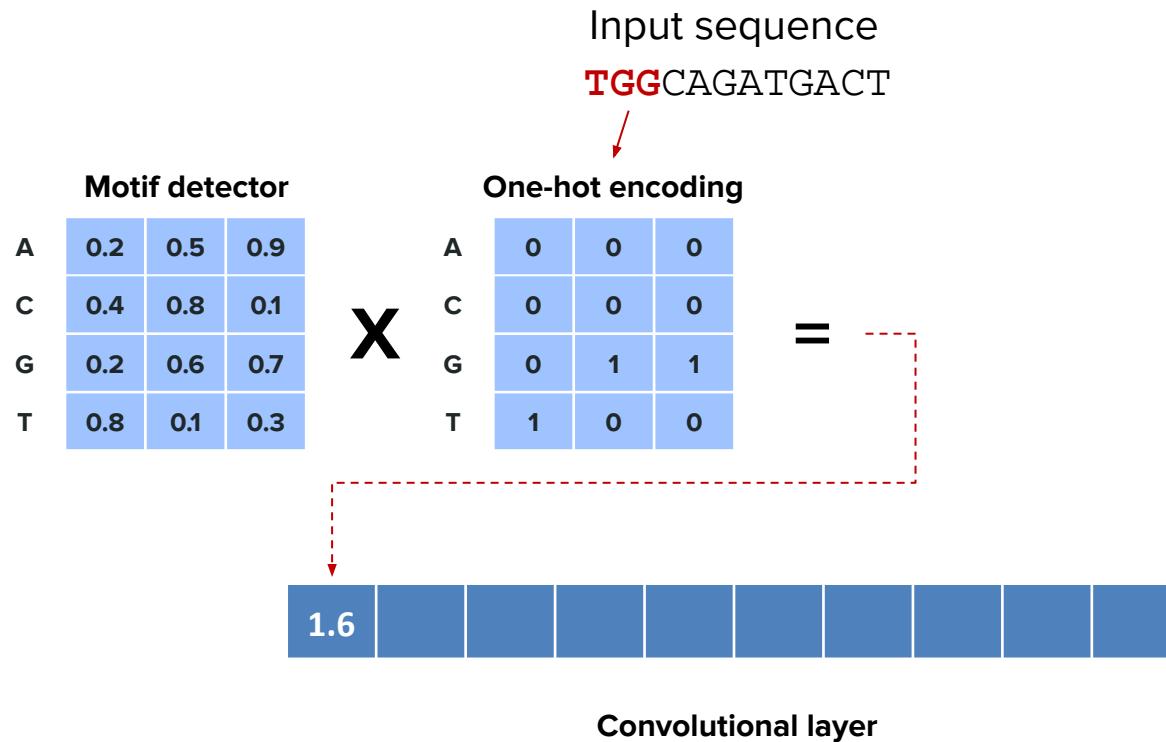
1D ConvNet: example

Input sequence
TGGCAGATGACT

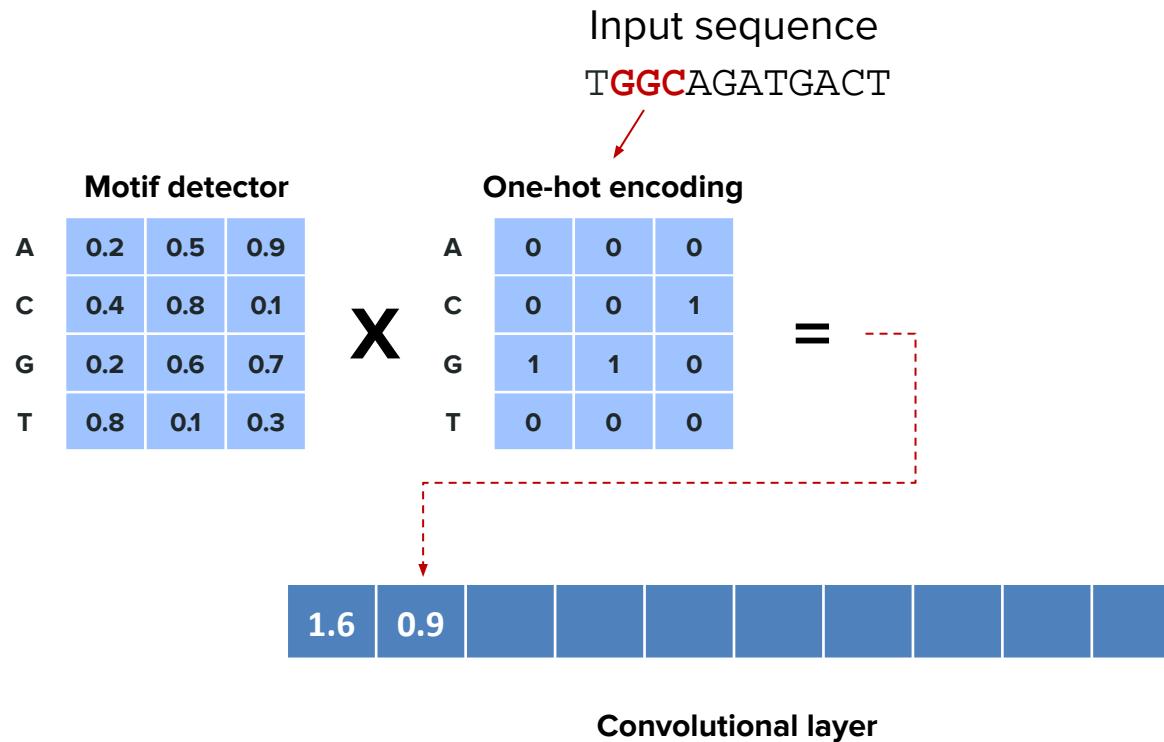
Motif detector

| | | | |
|---|-----|-----|-----|
| A | 0.2 | 0.5 | 0.9 |
| C | 0.4 | 0.8 | 0.1 |
| G | 0.2 | 0.6 | 0.7 |
| T | 0.8 | 0.1 | 0.3 |

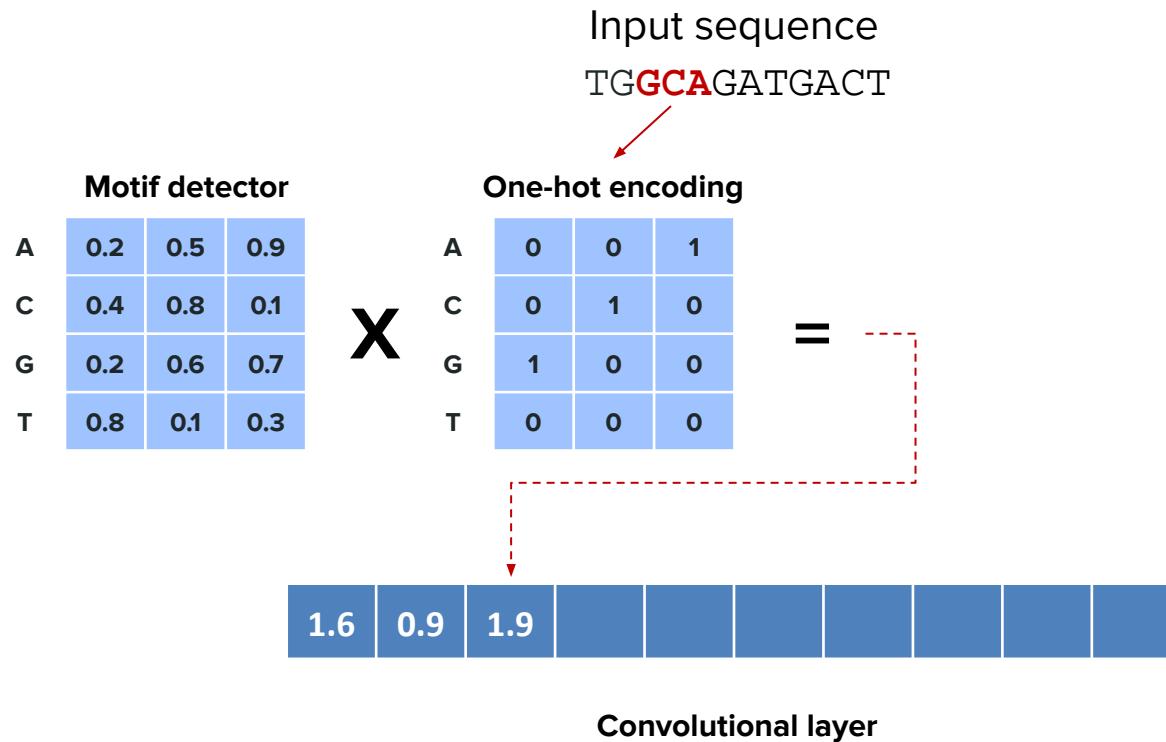
1D ConvNet: example



1D ConvNet: example



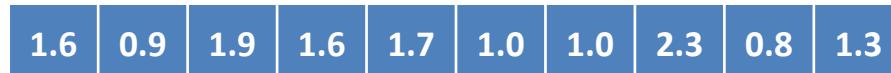
1D ConvNet: example



1D ConvNet: example

Score the presence
the the motif along
the input sequence

Input sequence
TGGCAGATGACT



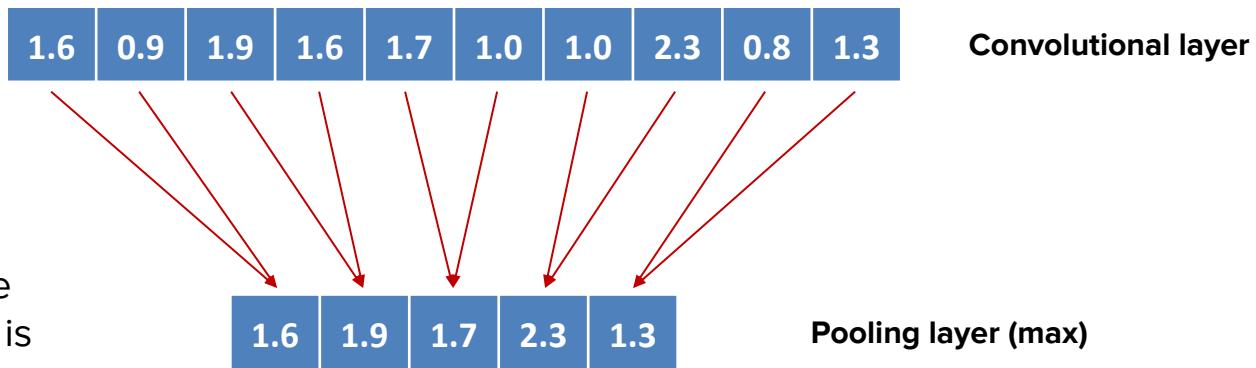
Convolutional layer

1D ConvNet: example

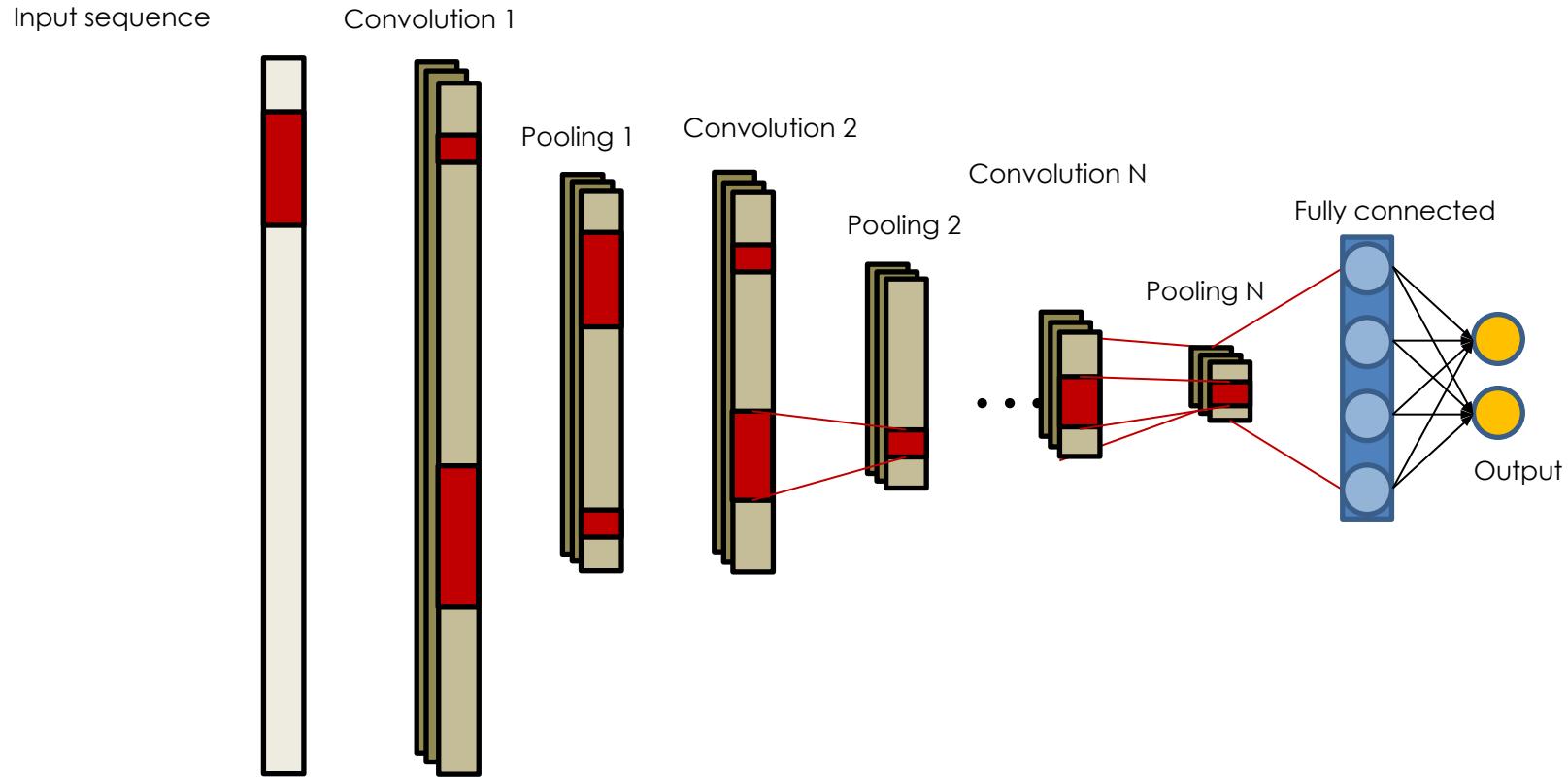
Score the presence
the the motif along
the input sequence

Score positions where
the score of the motif is
maximal

Input sequence
TGGCAGATGACT

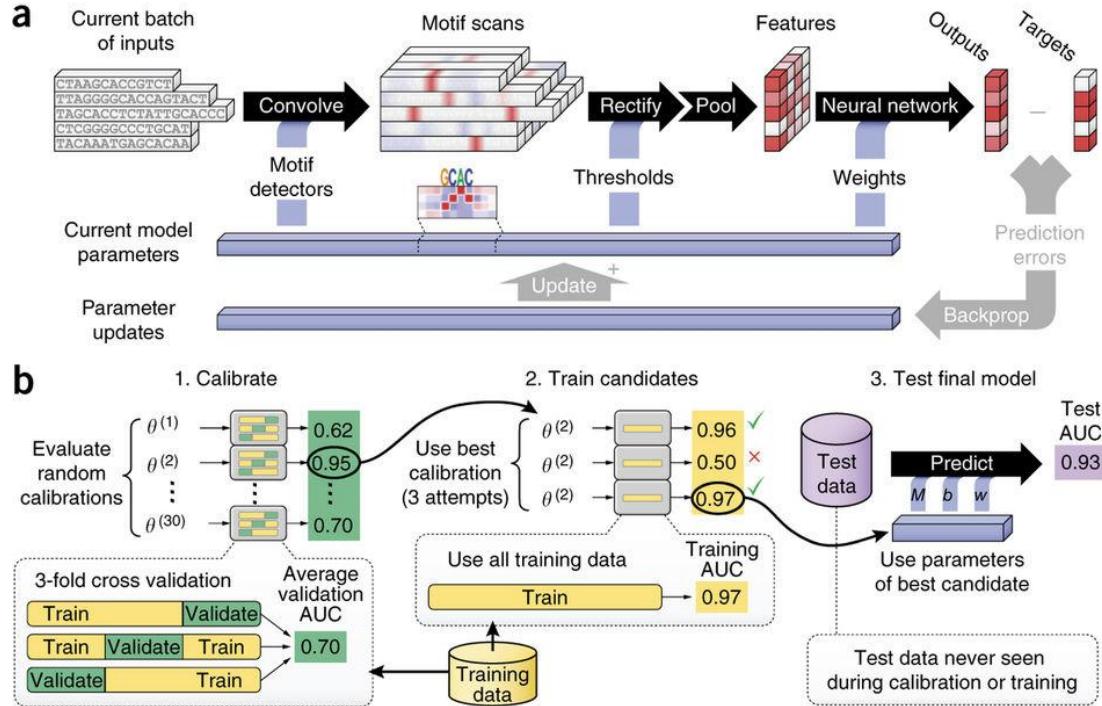


1D ConvNet: general architecture



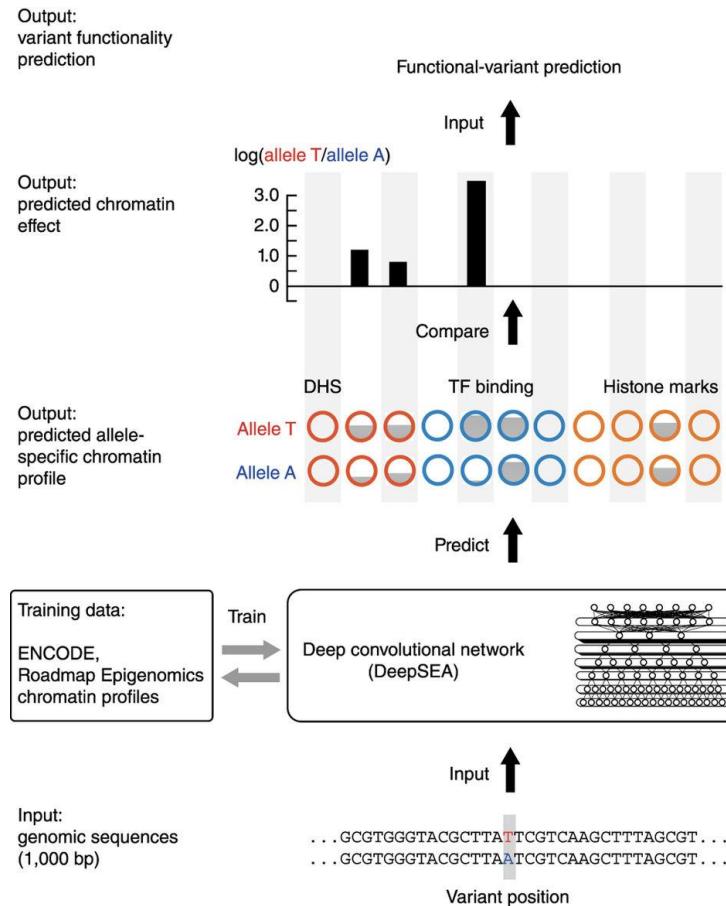
DeepBind

1D convolution network applied to genomic sequences to predict sequence specificities of DNA- and RNA-binding proteins



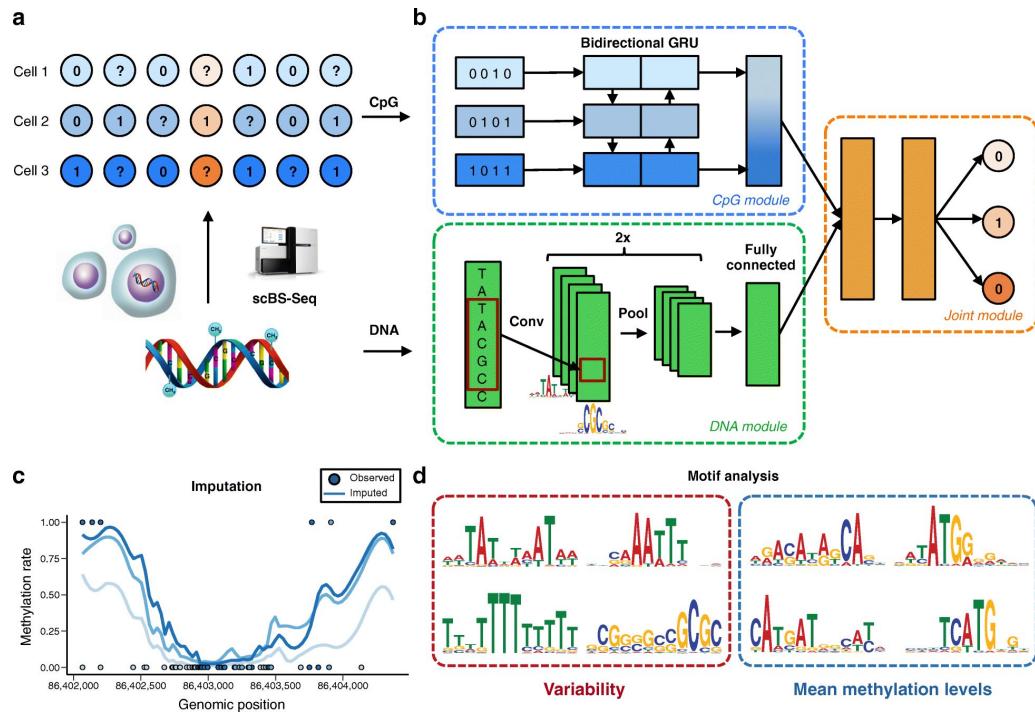
DeepSEA

Prediction of the functional effect of non-coding variants using a deep ConvNet



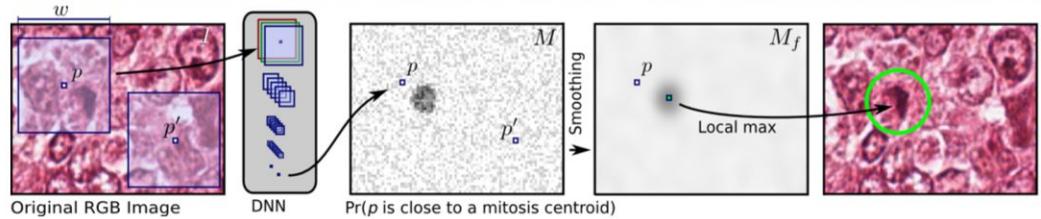
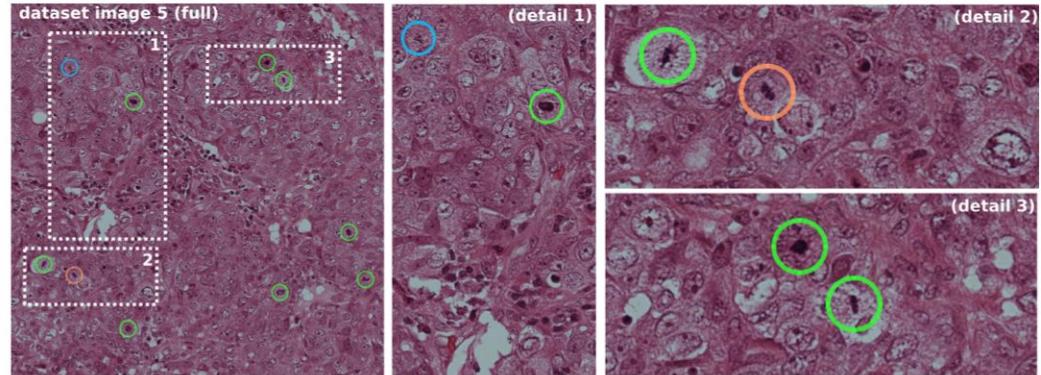
DeepCpG

Prediction of methylation states using a system comprising a deep ConvNet and a Recurrent Network



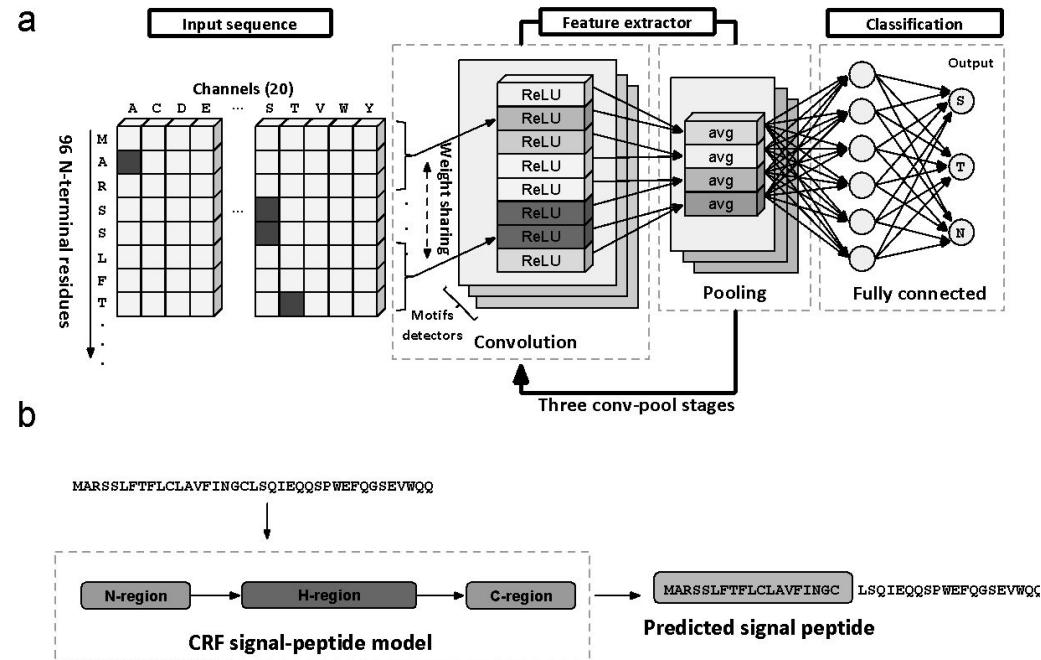
Mitosis detection in breast cancer images

Bio-image analysis: prediction of mitosis in breast cancer histology images using ConvNet



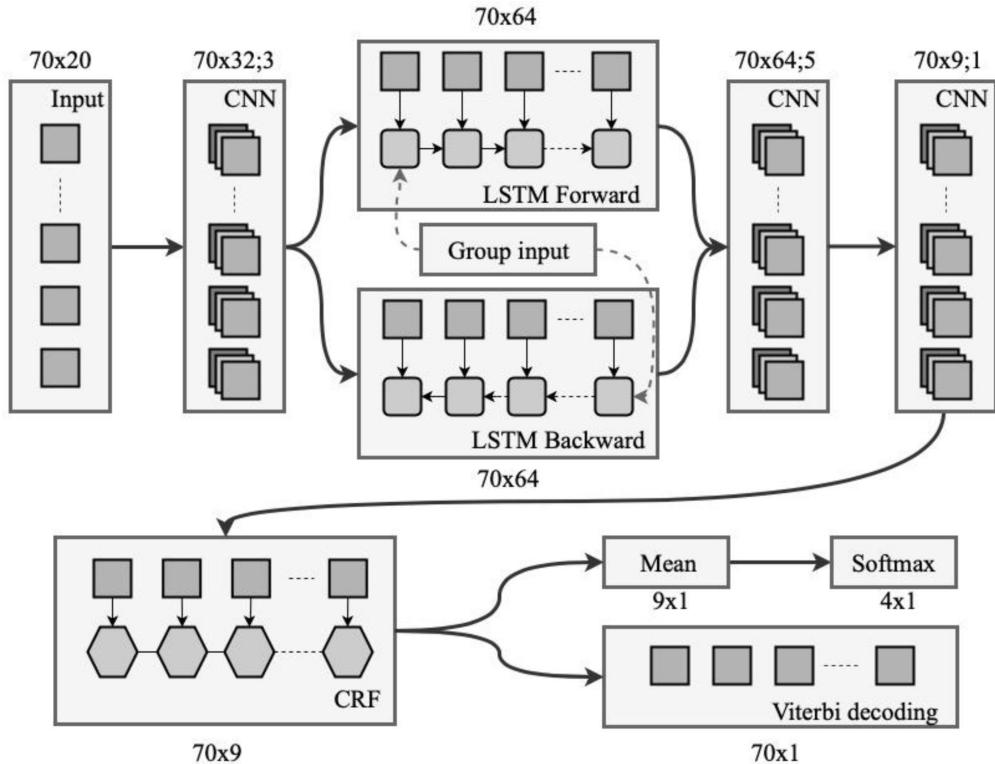
DeepSig

Prediction of signal peptides in proteins



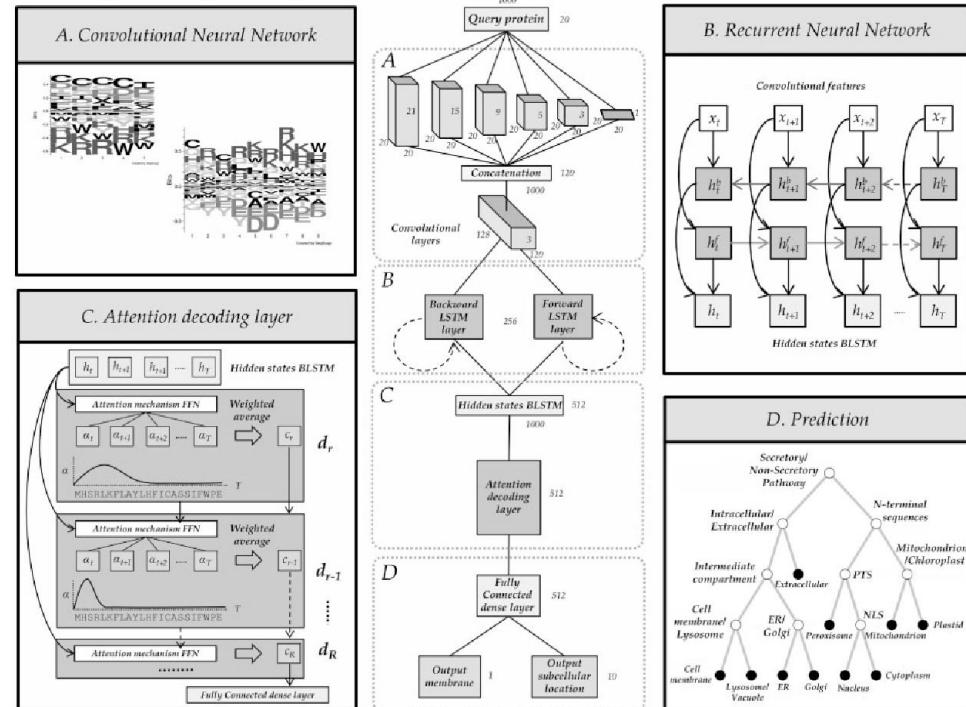
SignalP5.0

Prediction of signal peptides in proteins



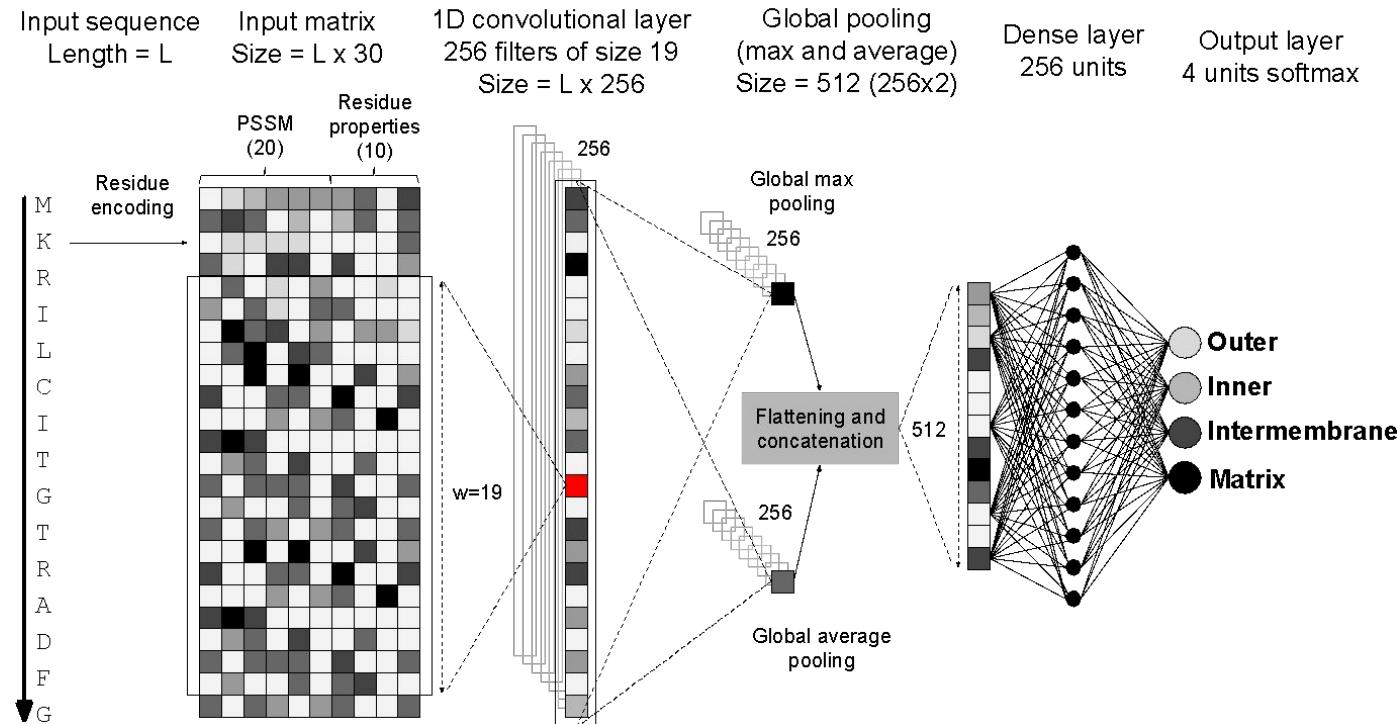
DeepLoc

Prediction of protein subcellular localization



DeepMito

Prediction of protein
sub-mitochondrial
localization



Summary of CNNs

- MLP are not efficient in modelling image data
- CNNs have been introduced to cope with image processing specificity, including translation invariance
- In CNN architectures, two types of layers are interleaved: convolutions to detect patterns and pooling to aggregate them
- Variants of CNNs exist to model sequences (1D-CNNs) or 3D objects
- 1D-CNNs are well-suited to analyse biological sequences for many applications, including motif detection and sequence classification

Examples

SCOP class prediction

- [https://colab.research.google.com/drive/1V7jv5S8Gj-QwrBq_r_4I0YVQZ6CDLX
BU?usp=sharing](https://colab.research.google.com/drive/1V7jv5S8Gj-QwrBq_r_4I0YVQZ6CDLXBU?usp=sharing)

vonHeijne using a single CNN filter

- [https://colab.research.google.com/drive/12e_KBtjnN0S-IIIalrMa7OkmAy2nPTN
H?usp=sharing](https://colab.research.google.com/drive/12e_KBtjnN0S-IIIalrMa7OkmAy2nPTNH?usp=sharing)

References

1. LeCun, Y. et al. (1990) Handwritten digit recognition with a back-propagation network, In Proc. of NIPS, 396-404.
2. Hubel and Weisel (1968) Receptive fields and functional architecture of monkey striate cortex, *The Journal of Physiology*, 195(1): 215-243.
3. Alipanahi, B. et al. (2015) Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning, *Nat Biotechnol.*, 33(8):831-8
4. Zhou, J., & Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model, *Nature Methods*, 12(10):931–934.
5. Angermueller, C. et al. (2017), DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning, *Genome Biology*, 18:67.
6. Ciresan, D.C. et al. (2013) Mitosis detection in breast cancer histology images with deep neural networks, In MICCAI 2013, 411-418.
7. Savojardo C, Martelli PL, Fariselli P, Casadio R. DeepSig: deep learning improves signal peptide detection in proteins. *Bioinformatics*. 2018 May 15;34(10):1690-1696.
8. Almagro Armenteros JJ, Tsirigos KD, Sønderby CK, Petersen TN, Winther O, Brunak S, von Heijne G, Nielsen H. SignalP 5.0 improves signal peptide predictions using deep neural networks. *Nat Biotechnol*. 2019 Apr;37(4):420-423.
9. Almagro Armenteros JJ, Sønderby CK, Sønderby SK, Nielsen H, Winther O. DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics*. 2017 Nov 1;33(21):3387-3395.
10. Savojardo C, Bruciaferri N, Tartari G, Martelli PL, Casadio R. DeepMito: accurate prediction of protein submitochondrial localization using convolutional neural networks. *Bioinformatics*. 2019 Jun 20.

Introduction to deep-learning methods

Neural Networks for structured data

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025

Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Outline

- Introduction
- Multi-layer perceptrons (MLPs)
- Convolutional neural networks (CNNs)
- **Recurrent neural networks (RNNs)**
- **Graph neural networks (GNNs)**
- Attention mechanisms
- Transformers
- Protein language models
- Applications
- Conclusions

Sequence data processing

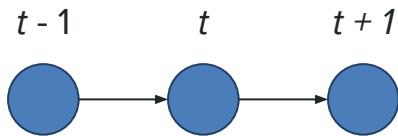
Arise in different domains:

- Time-series analysis (economics, weather forecasting)
- Natural language processing (tagging, machine translation, text generation)
- Signal processing (e.g., audio)
- Video analysis
- **Biological sequence analysis (proteins/DNA/RNA)**

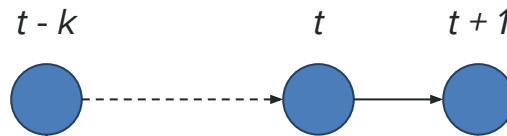
Typical prediction tasks:

- Sequence labelling (one label per timestep)
- Sequence classification (one label per sequence)
- Sequence generation (next observation/whole sequence)
- Sequence-to-sequence (translation)

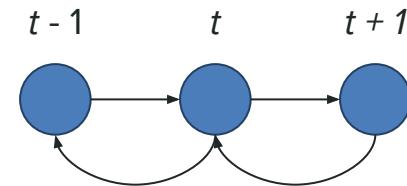
Observation inter-dependencies



Observation at time t
depends on observation
at time $t - 1$ (Markov)



Long-range
dependencies involving
time-step far away
between each other



Bi-directional
dependencies (from past
to future and vice versa)

Models for sequential data

Markov models:

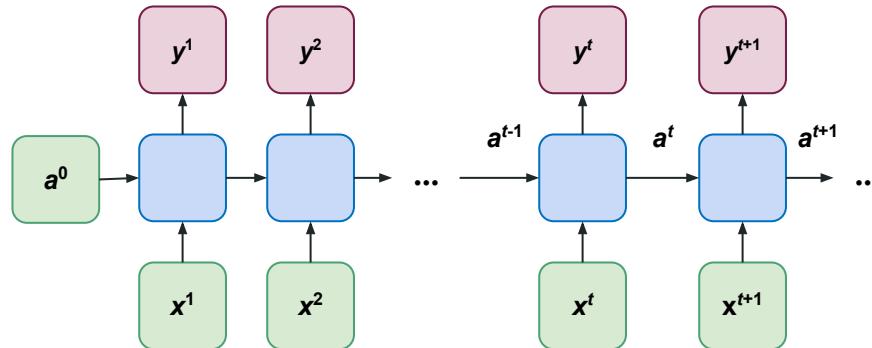
- First-order Hidden Markov Models (HMMs)
- Higher-order HMMs (modelling of long-range dependencies in the past)
- Conditional Random Fields (well-suited for sequence labelling)

Neural networks:

- MLPs treat data observations independently
- CNNs can model sequences, but lack an explicit modelling of observation inter-dependencies
- **Recurrent Neural Networks (RNNs)**

Recurrent Neural Network

- RNNs, are a class of neural networks that allow previous outputs to be used as inputs
- Standard architectures are defined as follows:

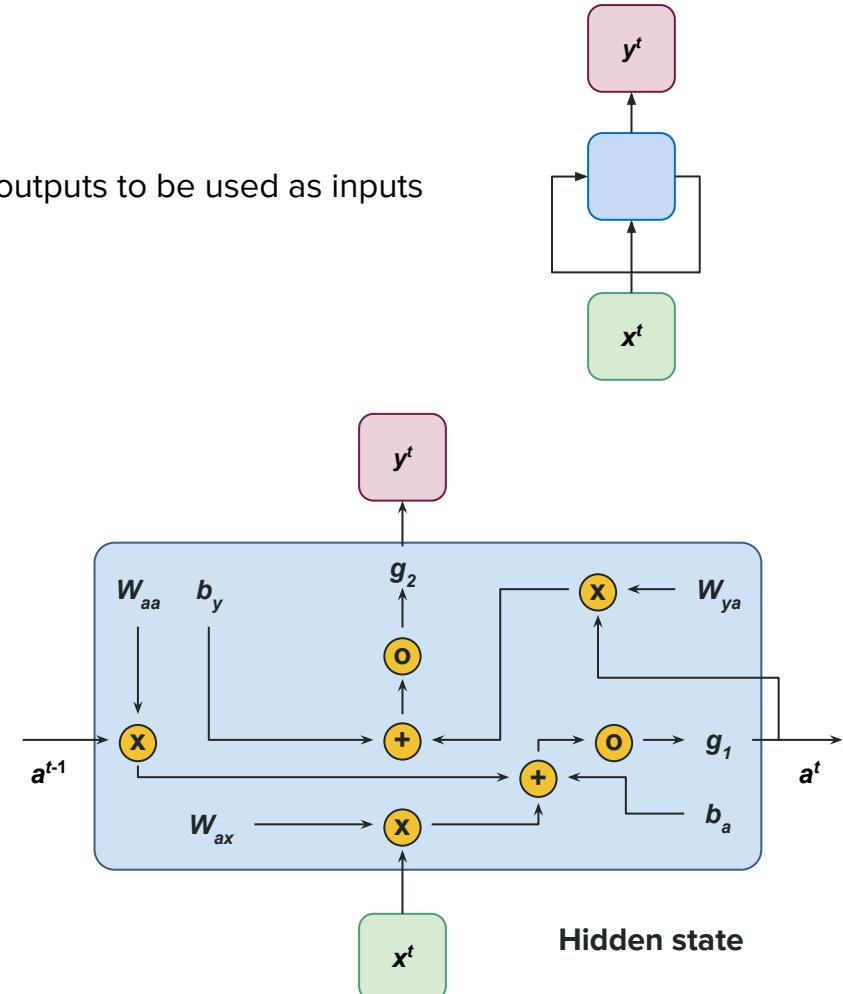


Unrolled architecture (over time)

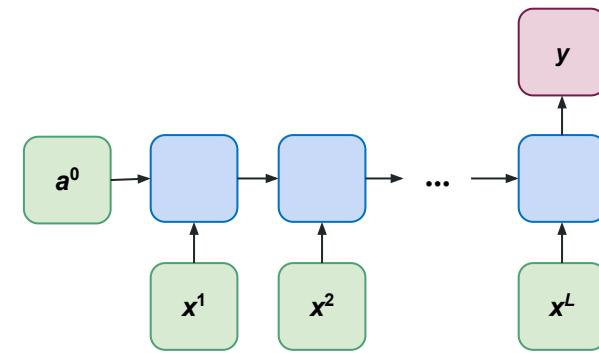
For each timestep t , the activation a^t and the output y^t are expressed as follows:

$$a^t = g_1(W_{aa}a^{t-1} + W_{ax}x^t + b_a)$$

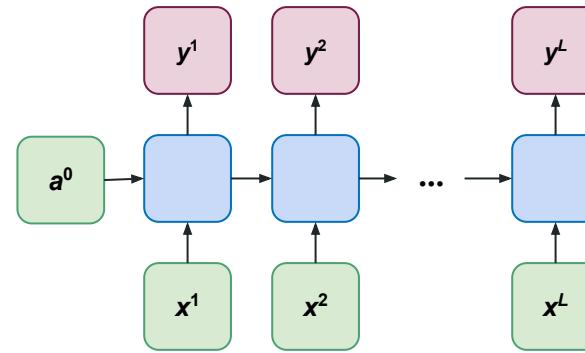
$$y^t = g_2(W_{ya}a^t + b_y)$$



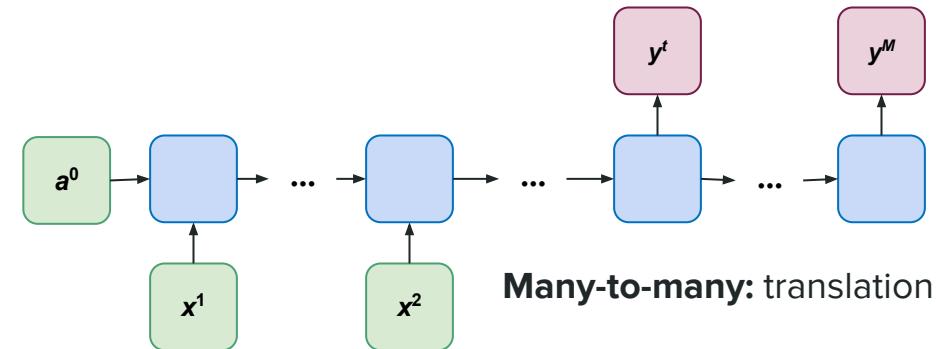
RNN applications



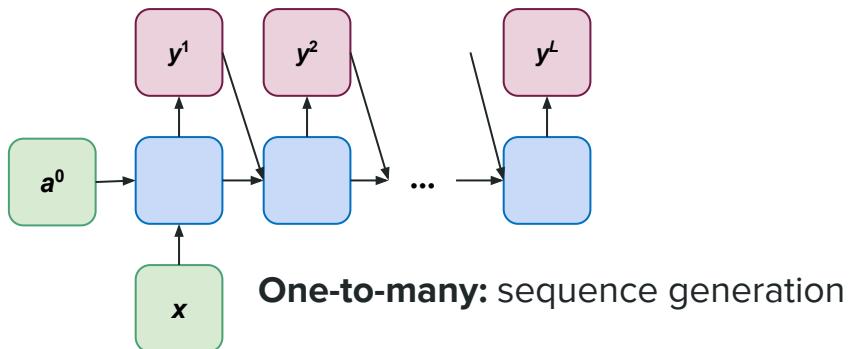
Many-to-one: sequence classification



Many-to-many: sequence labelling

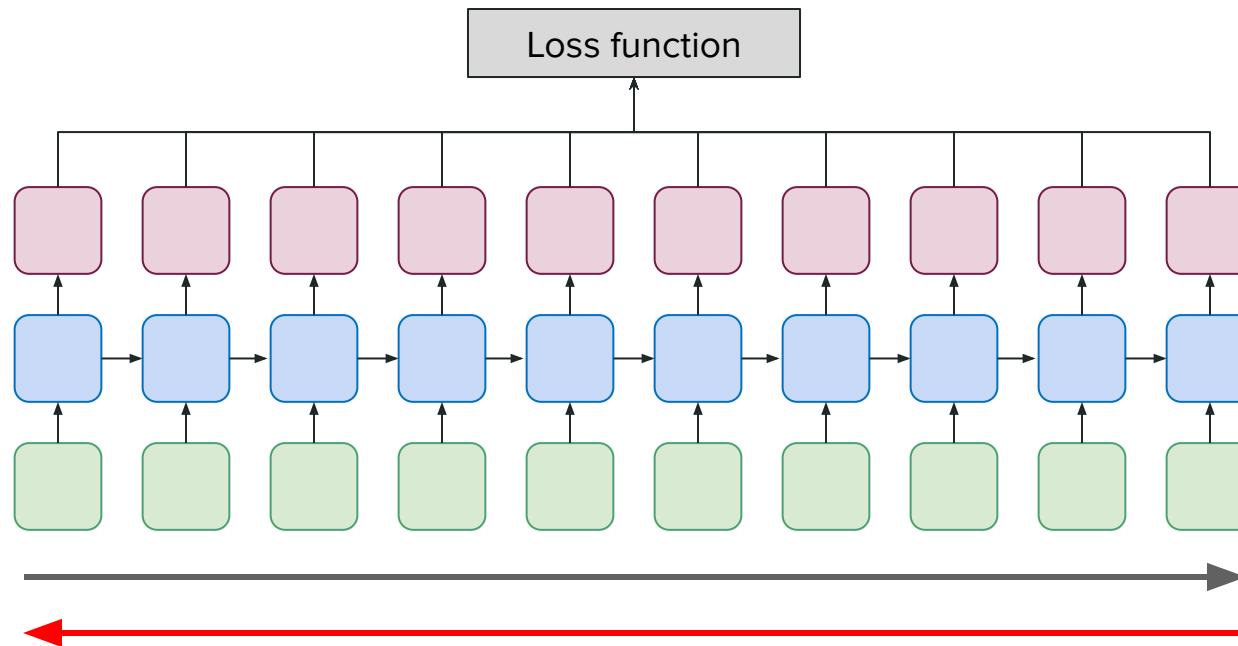


Many-to-many: translation



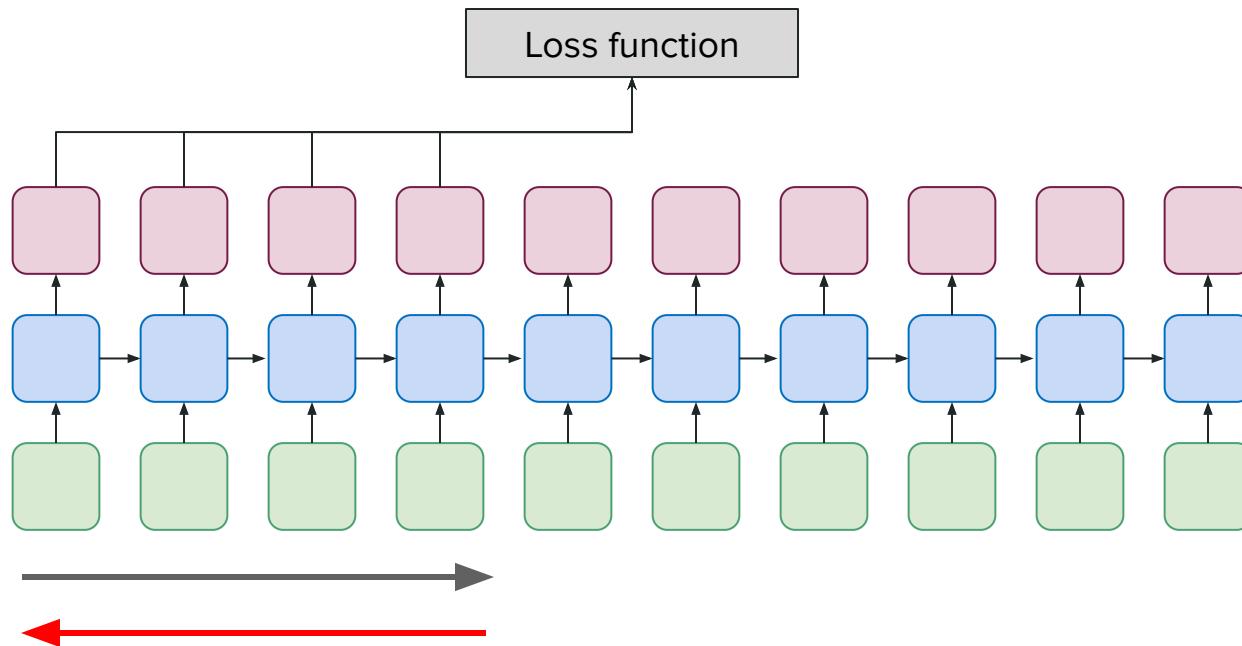
One-to-many: sequence generation

Training RNNs: backpropagation through time



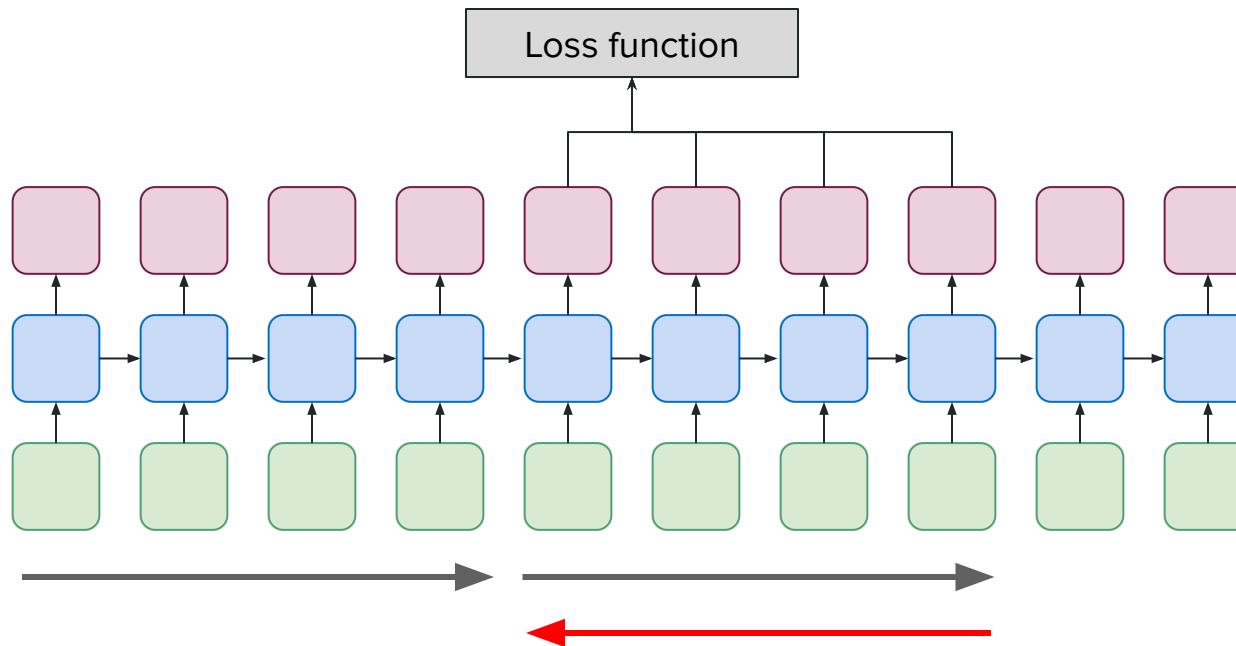
Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

Training RNNs: truncated backpropagation through time



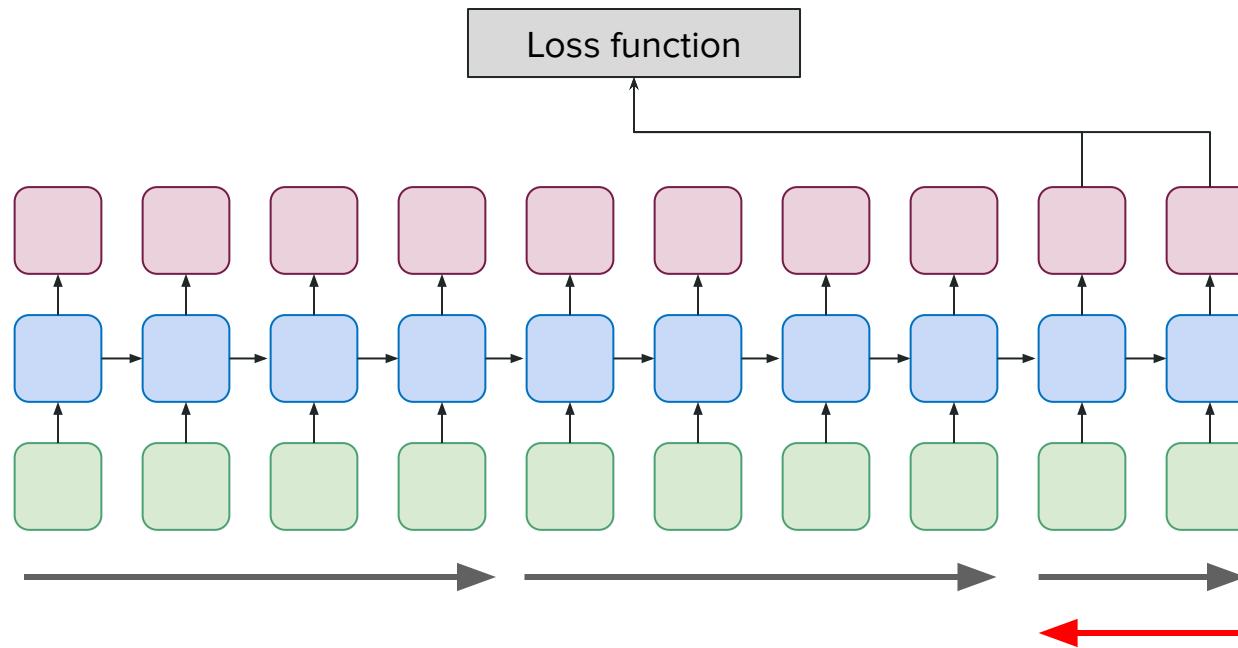
Run forward and backward through chunks of the sequence instead of whole sequence

Training RNNs: truncated backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Training RNNs: truncated backpropagation through time

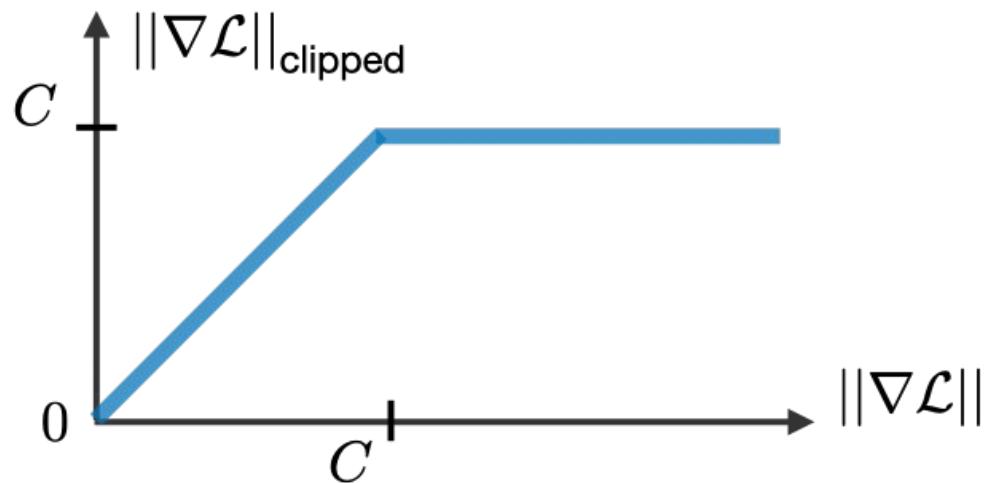


Standard RNN issues

- **Vanishing/exploding gradients:** the vanishing and exploding gradient phenomena are often encountered in the context of RNNs
- In the same way a product of k real numbers can shrink to zero or explode to infinity, so can a product of matrices
- Being λ_1 is the largest singular value of W :
 - $\lambda_1 < 1 / \gamma$ sufficient for vanishing gradients problem to occur
 - $\lambda_1 > 1 / \gamma$ necessary for exploding gradients
 - $\gamma = 1$ for the tanh activation and $\gamma = 1 / 4$ for the sigmoid activation

Gradient clipping for exploding gradient

It is a technique used to cope with the exploding gradient problem. By capping the maximum value for the gradient, this phenomenon is controlled in practice.



RNN with special gates

In order to remedy the vanishing gradient problem, specific **gate units** are used in some types of RNNs and usually have a well-defined purpose.

| Type of gate | Role | Used in |
|----------------|----------------------------------|-----------|
| Update gate | How much past should matter now? | GRU, LSTM |
| Relevance gate | Drop previous information? | GRU, LSTM |
| Forget gate | Erase a cell or not? | LSTM |
| Output gate | How much to reveal of a cell? | LSTM |

Variants of RNNs

Long-Short Term Memory (LSTM) networks

- The key component is a **memory cell** that acts like an accumulator over time
- Instead of computing new state as a matrix product with the old state, it rather computes the difference between them. Expressivity is the same, but gradients are better behaved

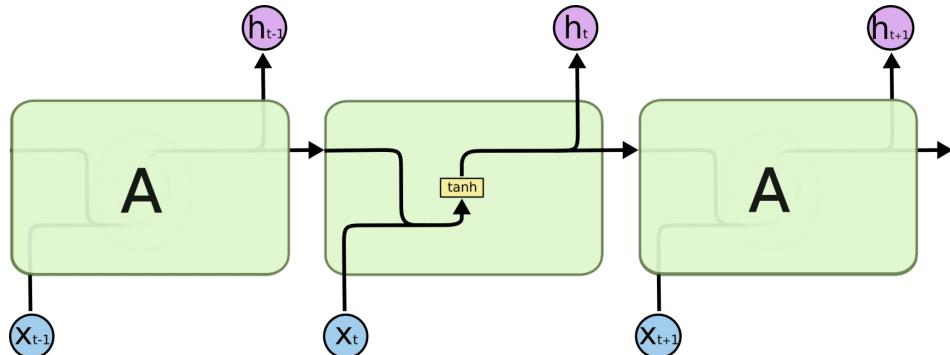
Gated Recurrent Units (GRUs)

- A very simplified version of the LSTM with less parameters

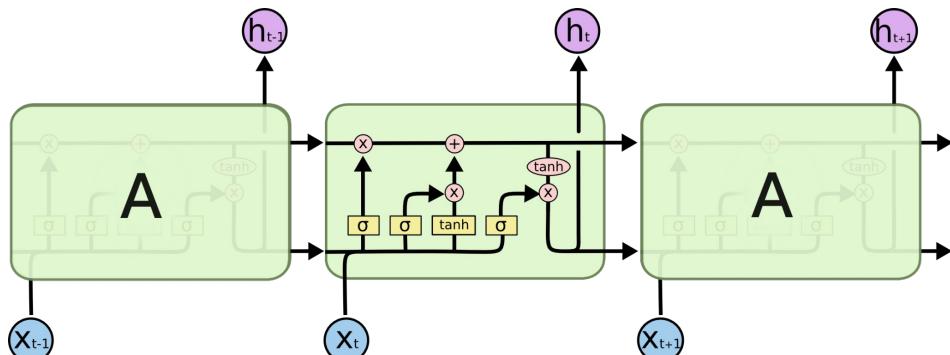
LSTMs

- In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer
- In LSTMs repeating module there are four layers (**gates**), interacting in a very special way

Standard RNN repeating module

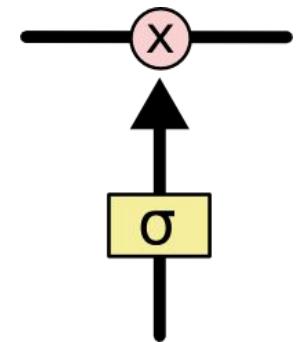


LSTM repeating module



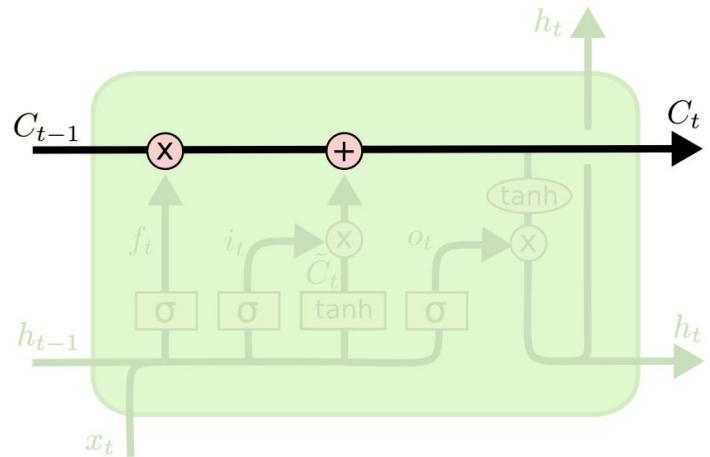
LSTM gate

- Gates are a way to optionally let information through
- Composed out of a sigmoid neural net layer and a pointwise multiplication operation
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”



LSTM: cell state

- Responsible of capturing and retain long-term dependencies in sequential data
- Act as a conveyor belt though the the entire chain
- Three LSTM gates control the cell state:
 - **Forget gate**
 - **Input gate**
 - **Output gate**

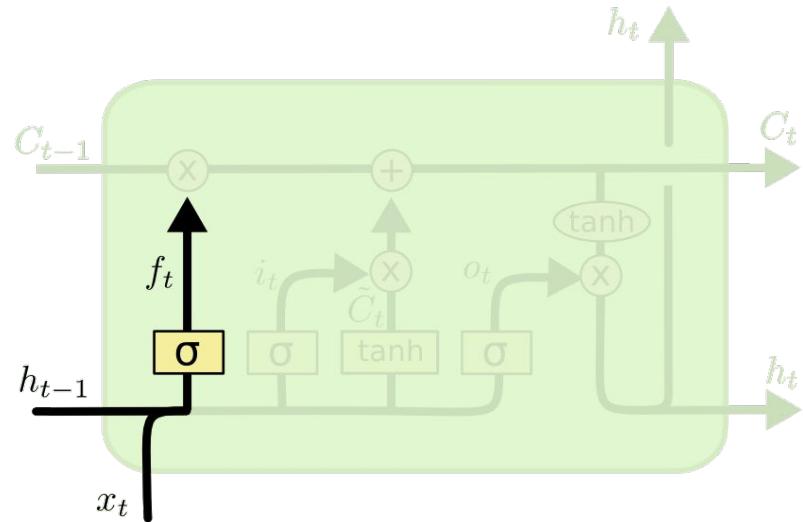


Cell state vs hidden state

- Cell state (C) can be seen as an accumulator providing an aggregation of data from all previous time-steps
- Cell state is useful to model long-range dependencies
- Hidden state (h) characterize only the previous time step (i.e., short-range dependencies)
- Hidden state is not directly interpretable as a “label” for current time step. It is just an encoding of the current state of the LSTM.
- Mapping this encoding to the actual label is performed usually outside the LSTM (e.g., using a standard feed-forward net)
- In general, relevant time-steps as well as optimal encoded states are determined during training in relation to the task at hand

Step 1: forget gate

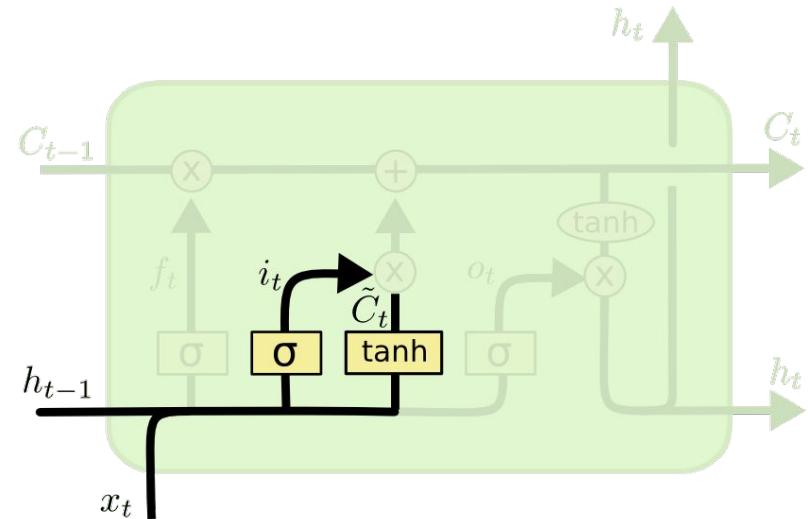
- The first step in our LSTM is to decide what information we're going to throw away from the cell state.
- The forget gate looks at h_{t-1} and the input x_t and outputs a number between 0 and 1 for each number in the cell state C_{t-1}
- 1 = completely keep the value, 0 = completely get rid of the value
- Weights W_f perform feature extraction to determine how important the memory (C) is in relation to the current time step



$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Step 2: input gate and candidate new value for C

- Next step is to decide what new information we're going to store in the cell state.
- The input gate looks at previous hidden state (h_{t-1}) and the current input (x_t) and outputs a number between 0 and 1 to decide which values we'll update
- A tanh layer create a vector of candidate values that could be added to the cell
- Weights W_C and W_i perform feature extraction to, respectively, compute an encoding of the data and to identify remember-worthy components of h and x

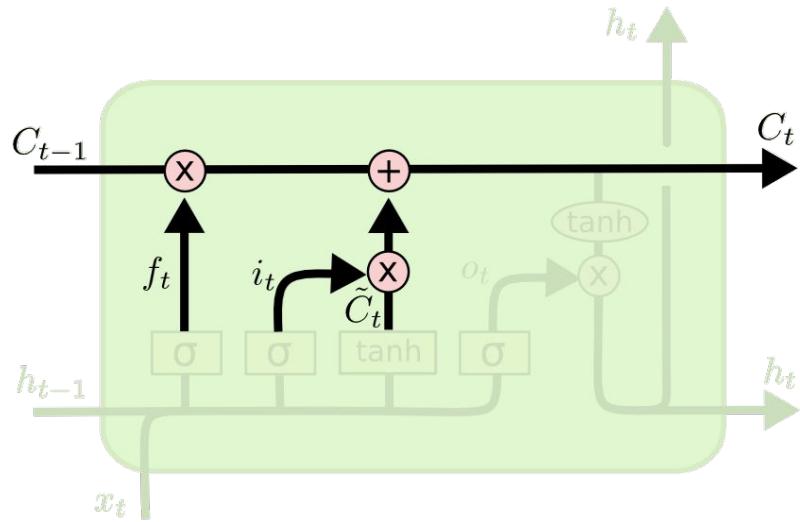


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Step 3: update the cell state

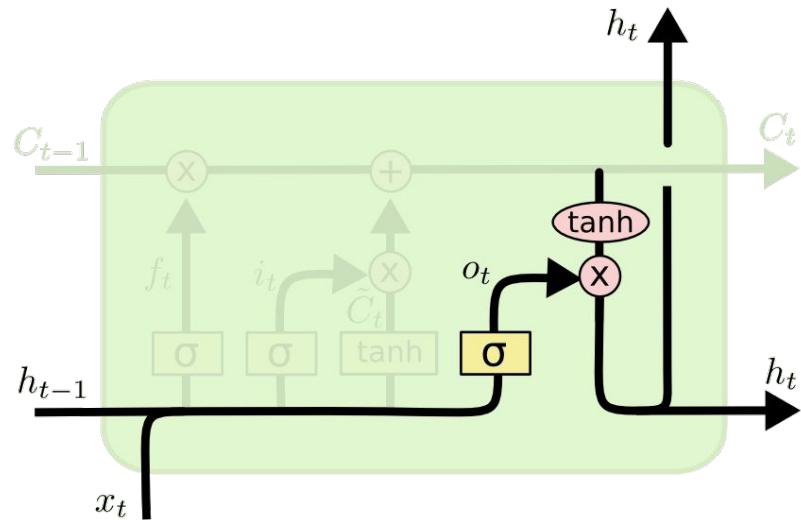
- The old cell state (C_{t-1}) is firstly multiplied by the output of the forget gate (f_t)
- Then, we add the new candidate values for the cell, scaled by how much we decided to update each state value
- Together, this operations implement the selective forgetting of information and the storage of new values to be remembered from now on



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Step 4: output gate

- Output will be based on our cell state, but will be a filtered version
- A first sigmoid layer decides what parts of the cell state we're going to output
- The cell state is put through a tanh and multiplied by the output of the sigmoid

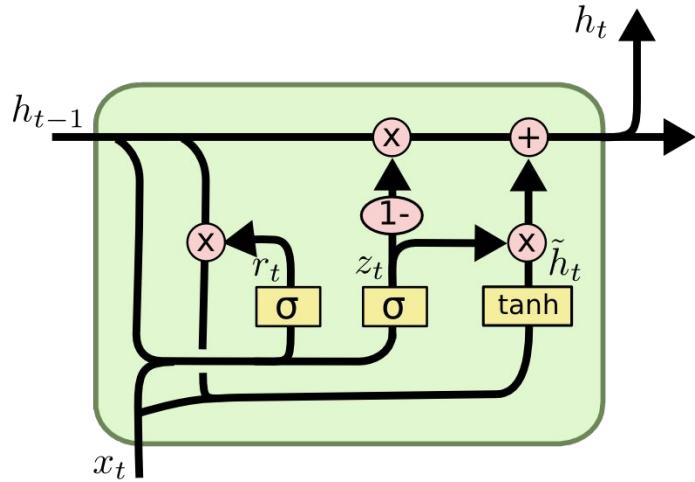


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

Gated Recurrent Units (GRUs)

- Simpler model than LSTM
- Combine forget and input gate into a single update gate
- Merges cell state (C) and hidden state (h)
- Less parameters to be trained
- In some applications perform better than LSTM



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Advantages of LSTM

- The gating mechanisms of LSTMs reduces the risk of the vanishing gradient problem
- The key is the update rule for the LSTM cell state, which is additive rather than multiplicative
- In other words, for standard RNN, the gradient decays with:

$$\prod w\sigma'(\cdot)$$

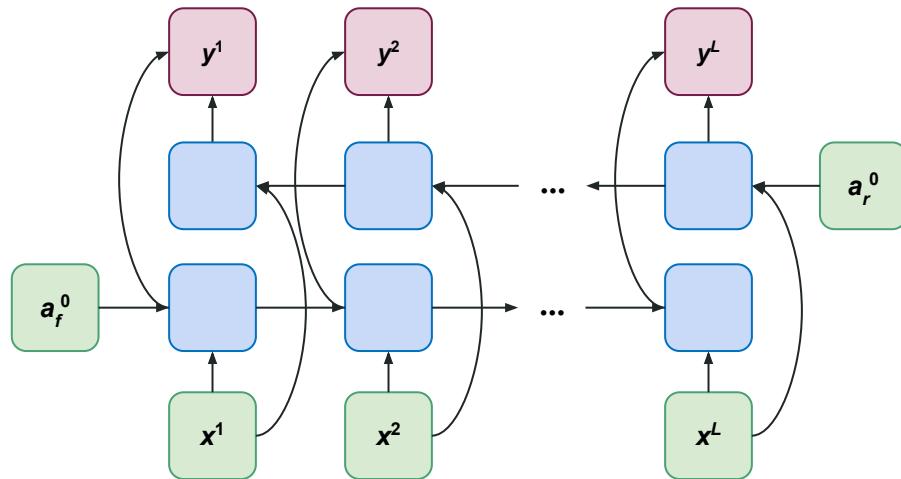
- In LSTM, the gradients flow through time only by the cell state, and decays with:

$$\prod \sigma(\cdot)$$

- In both cases, the gradient **can** vanish. However, in LSTM, there is no exponential fast decaying factor involved (w)

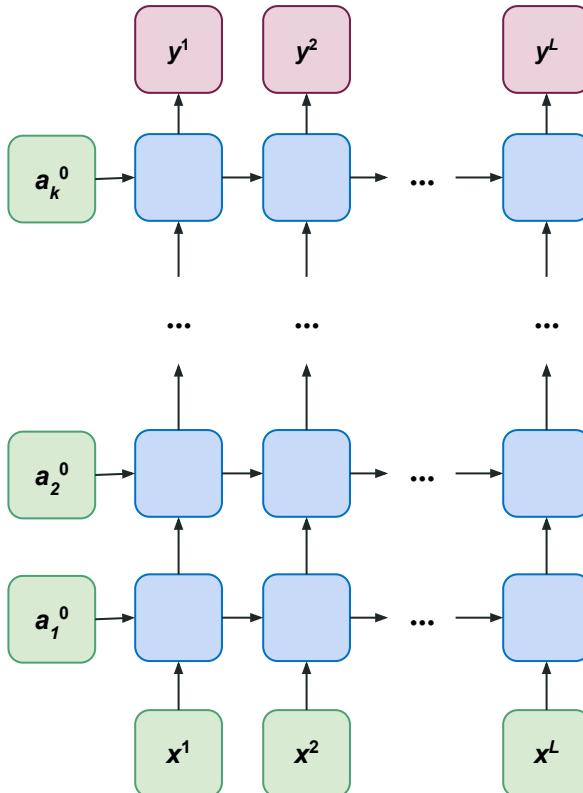
Bi-directional RNN/LSTM

- Motivation: output at time t may depend on the whole input sequence
- Two layers processing the sequence forward and backward
- Output hidden layer is usually a concatenation of the two hidden layers
- All types of RNNs can be implemented as bidirectional



Deep RNNs

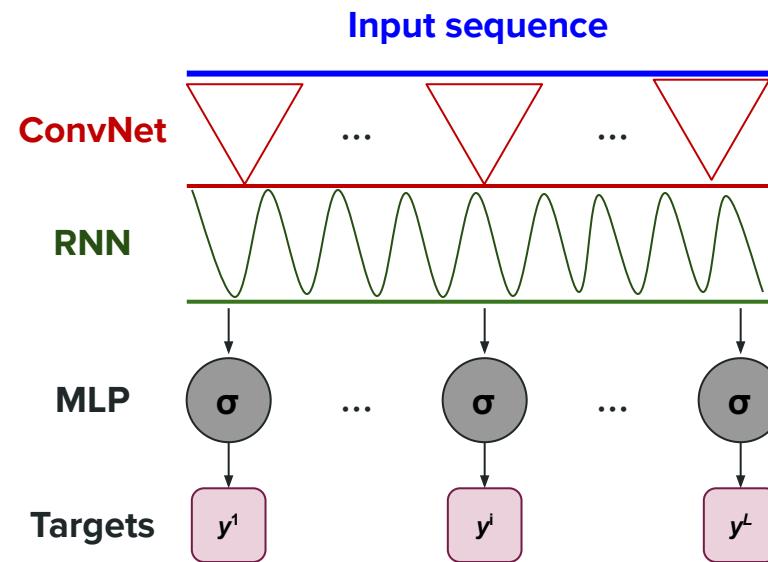
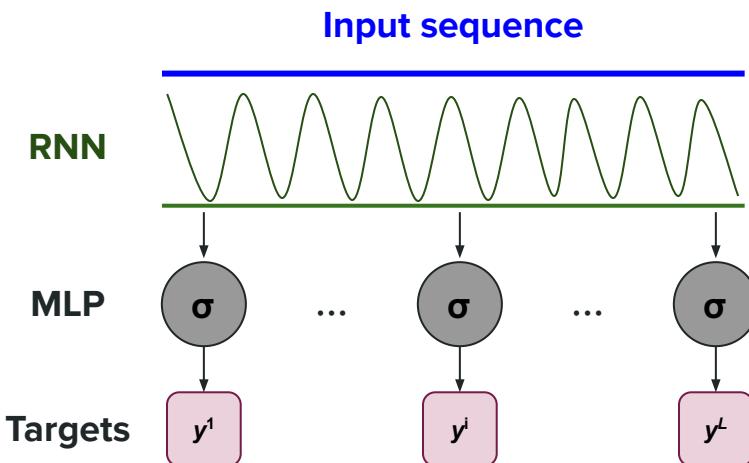
- Multiple stacked layers of RNNs
- Output of the current layers is used as input to the next layer
- Each layer can be also bidirectional



Common architectures

RNNs can be used alone or combined with other types of layers (e.g, convolutional or fully-connected MLPs)

Typical architectures include:



Summary of RNNs

- RNN and variants are well-suited to model sequential data
- RNNs can be used to solve different tasks: sequence labelling, classification, generation and translation
- RNNs are trained using back-propagation through time, i.e., treating the network as an MLP of depth equal to the sequence length and weight sharing
- Standard RNNs suffer of vanishing/exploding gradients
- Gated RNNs, such as LSTM and GRU can mitigate the problem and better learn long-term dependencies
- RNNs can be combined with standard MLPs and CNNs when applied to real-world tasks

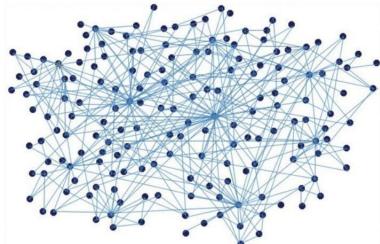
Examples

- Helix prediction using ConvNet+LSTM

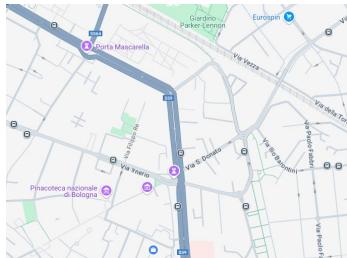
<https://colab.research.google.com/drive/1htCOpZrmq5hBon95lvXFI2iZHzadY2L9?usp=sharing>

Neural Networks on graph data

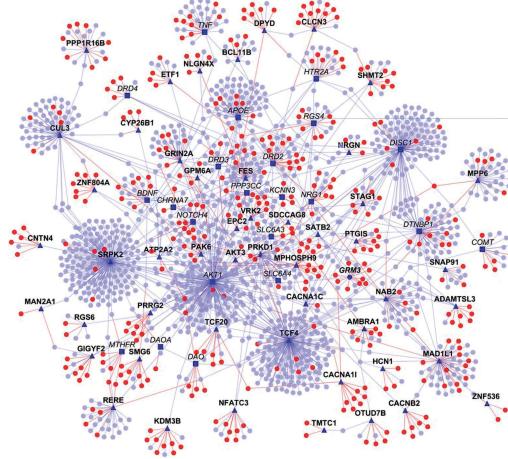
In many domains, data are not living in regular grids (e.g., sequences)



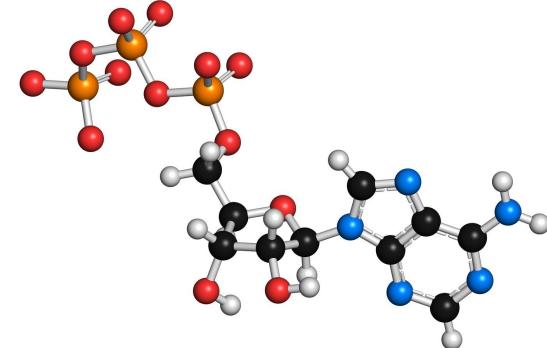
Social or communication networks



Road maps



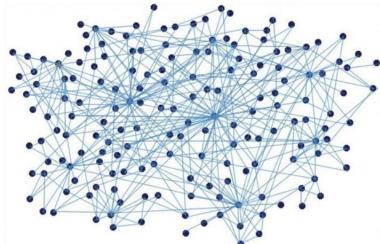
Protein interaction networks



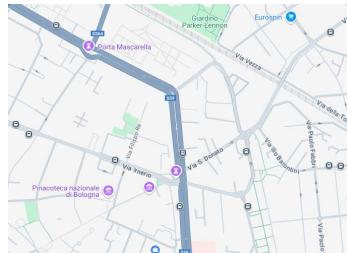
Molecules

Neural Networks on graph data

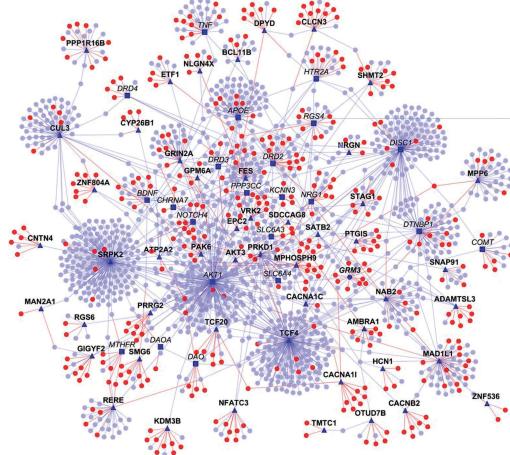
In many domains, data are not living in regular grids (e.g., sequences)



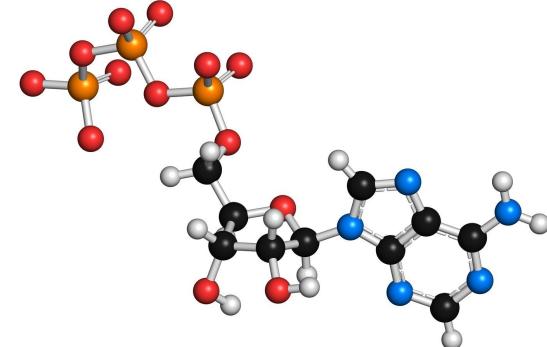
Social or communication networks



Road maps



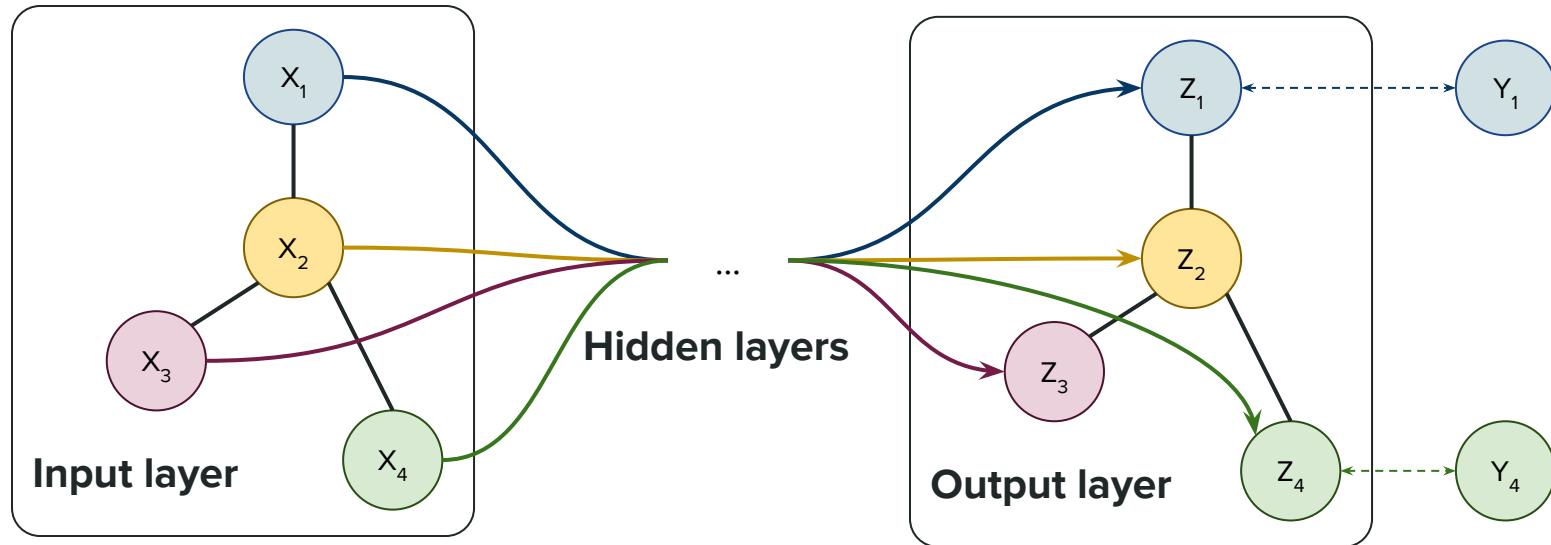
Protein interaction networks



Molecules

Standard CNN or RNN do not work on this data

Neural Networks on graph data



- Learn feature representations for nodes
- Use information from neighboring nodes to update the current node representation

Graph “convolution”

- Aggregates information from neighbors to update information on node
- Inspired by convolution on pixels in CNNs
- Differs from CNN convolution since in general graphs the neighborhood is of variable size

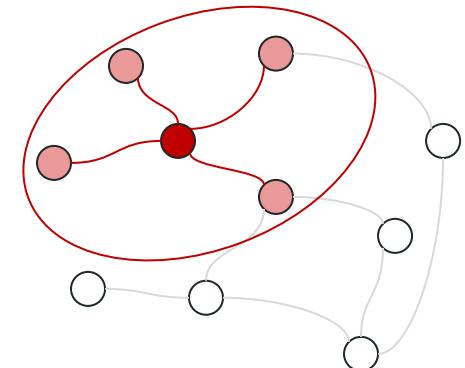
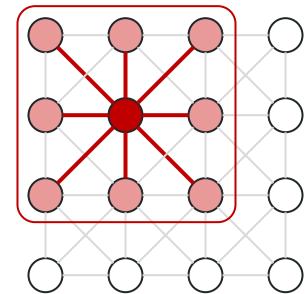
$$h_{\mathcal{N}(v)}^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\})$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, h_{\mathcal{N}_v}^{(k)})$$

k : index of the network layer

$h_v^{(k)}$: k -th hidden layer representation of node v

\mathcal{N}_v : set of neighbors of node v



Graph “convolution”: example

- Mean aggregation:

$$h_{\mathcal{N}(v)}^{(k)} = \text{MEAN}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\})$$

- Max aggregation (after transforming representations):

$$h_{\mathcal{N}(v)}^{(k)} = \text{MAX}^{(k)}(\{\sigma(W_{pool}^{(k)} h_u^{(k-1)} + b) : u \in \mathcal{N}(v)\})$$

- Combine operation: concatenation + linear mapping + sigmoid:

$$h_v^{(k)} = \sigma(W^{(k)}[h_v^{(k-1)}; h_{\mathcal{N}(v)}^{(k)}])$$

Message Passing Neural Networks

- General formulation including both node and edge features and learned aggregation and combination functions
- Aggregation from neighbouring nodes:

$$m_v^{(k)} = \sum_{u \in \mathcal{N}(v)} M^{(k-1)}(h_v^{(k-1)}, h_u^{(k-1)}, e_{vu})$$

- Update node representation:

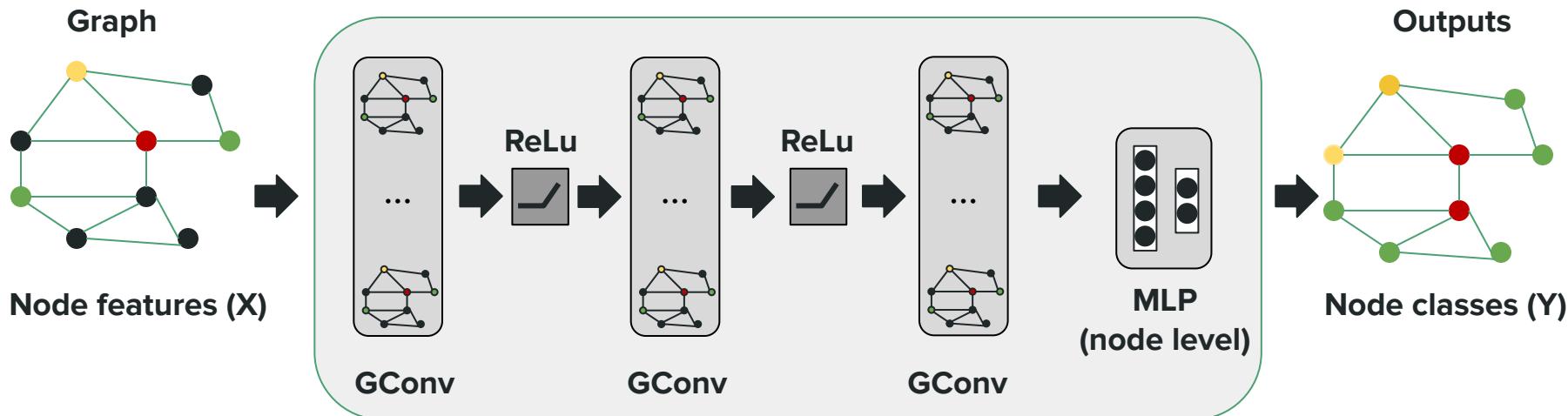
$$h_v^{(k)} = U^{(k)}(h_v^{(k-1)}, m_v^{(k)})$$

e_{vu} : features associated to edge (v,u)

$M^{(k-1)}$: **message function** (e.g., a MLP) computing message from neighbour

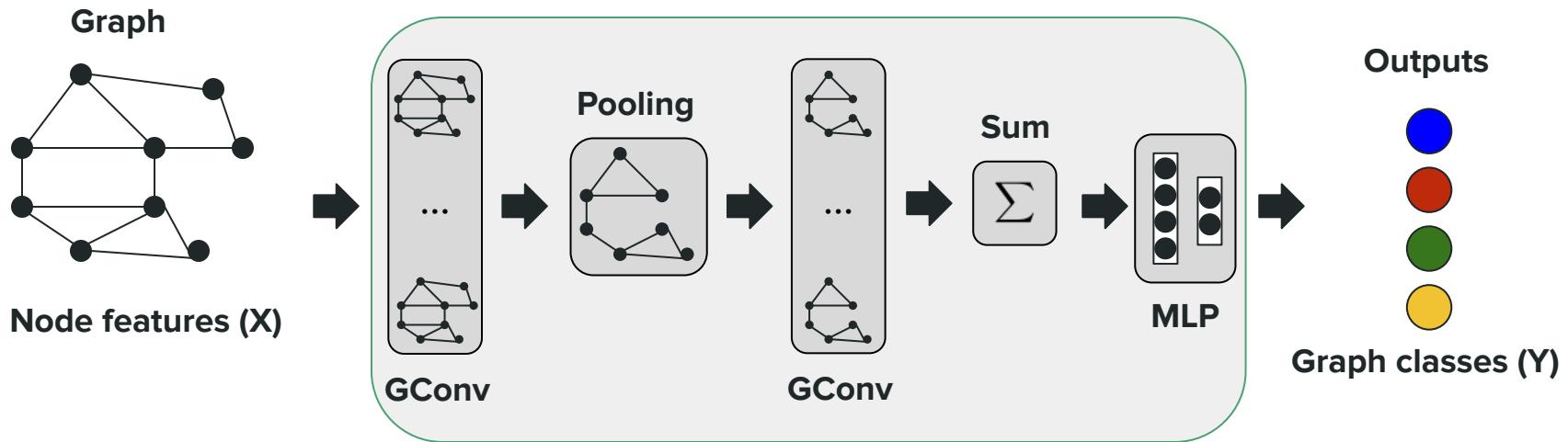
$U^{(k)}$: **update function** (e.g., a MLP) combining message and local information

Node classification



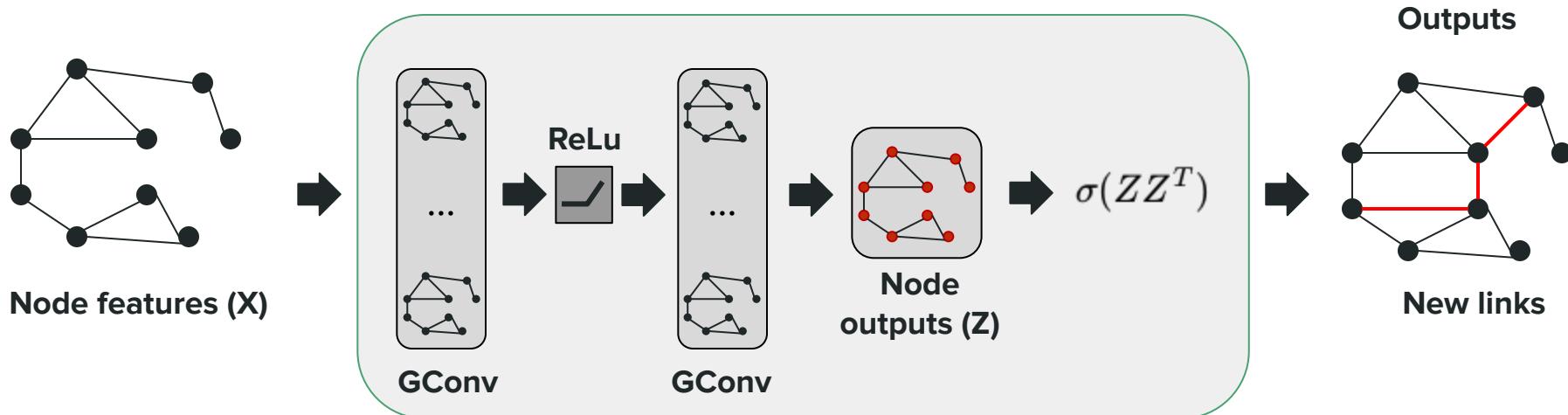
- Semi-supervised setting: some labels are known at training time
- Compute node representations with layerwise architecture
- Add output layer to map node representations to labels (MLP+softmax, MLP+sigmoid)

Graph classification



- Alternate convolution and pooling layers as in CNN
- Pooling has the effect of progressively reducing the number of nodes
- Use a final aggregation function (e.g., Sum) and then an MLP to map the whole graph into a set of classes

Link prediction



- Estimate the probability of all links as $p(A_{i,j}) = \sigma(z_i z_j^T)$
- Z is the graph representation computed by GConvs
- New links are those having $p > 0.5$

Summary of GNNs

- RNN/CNN cannot handle data structured as irregular graphs
- GNNs are a generalization of CNNs to non regular grids
- GNNs with multiple layers progressively compute representations of graph nodes (aka node embeddings)
- A given node embedding is computed by aggregation and combination of embeddings from neighboring nodes (message passing)
- GNNs can solve different graph-based tasks: node or graph classification, link prediction
- A PyTorch implementation is available through the PyTorch Geometric package:

<https://pytorch-geometric.readthedocs.io/en/latest/>

References

1. Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2): 179–211, 1990
2. Pascanu, R. et al. (2013) On the difficulty of training Recurrent Neural Networks, arXiv 1211.5063.
3. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (November 15, 1997), 1735–1780
4. Cho et al (2014) Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, arXiv, 1406.1078v3
5. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
6. William L. Hamilton, Rex Ying, Jure Leskovec (2018) Inductive Representation Learning on Large Graphs, arXiv, 1706.02216
7. Kipf & Welling (2017) Semi-Supervised Classification with Graph Convolutional Networks, arXiv 1609.02907
8. Duvenaud et al. (2015) Convolutional Networks on Graphs for Learning Molecular Fingerprints, In proc. of NIPS 2015
9. Kipf & Welling (2016) Variational Graph Auto-Encoders, arXiv 1611.07308

Introduction to deep-learning methods

Attention and Transformers

Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

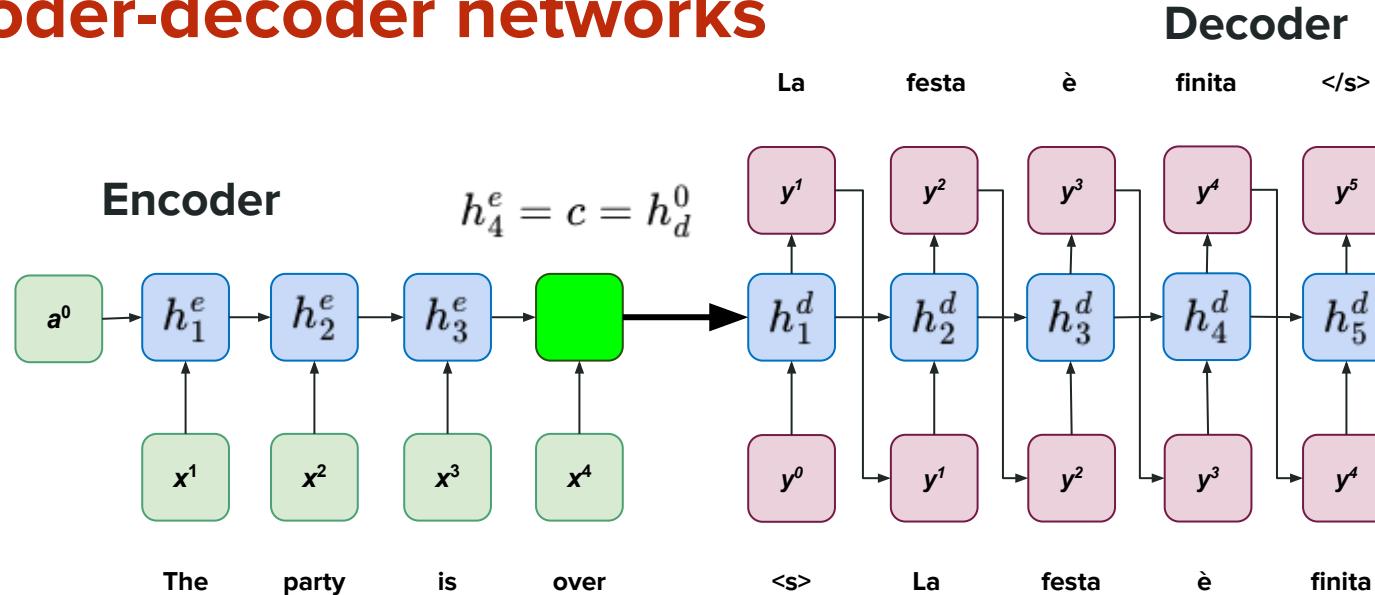
Outline

- Introduction
- Multi-layer perceptrons (MLPs)
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Graph neural networks (GNNs)
- **Attention mechanisms**
- **Transformers**
- Protein language models
- Applications
- Conclusions

Sequence-to-sequence models

- Models capable of generating an arbitrary length output sequence given an input sequence (of potentially different length)
- The key idea involve two components
- The **encoder** takes an input sequence and create a contextualized representation of it, called the **context**
- The **decoder** takes the context and generates a task specific output sequence
- Many applications:
 - Machine translation, summarization, question answering
 - Bioinformatics applications: definition of **protein language models**

Encoder-decoder networks



- Encoder/decoder are implemented using RNNs
- The final state of the encoder serves as the context and used as initial state of the decoder
- Commonly, the context is provided to all time-steps in the decoder (not shown in figure)
- The decoder outputs the probability of next token given previous token, decoder state and the context
- Training of the network is performed end-to-end

Encoder-decoder networks

- The simplicity of the encoder-decoder model is its clean separation of the encoder — which builds a representation of the source text — from the decoder, which uses this context to generate a target text
- This context vector is the hidden state of the last (nth) time step of the source text
- This final hidden state is thus acting as a bottleneck: it must represent absolutely everything about the meaning of the source text, since the only thing the decoder knows about the source text is what's in this context vector
- Information at the beginning of the sentence, especially for long sentences, may not be equally well represented in the context vector

Attention

- The **attention mechanism** is a way of allowing the decoder to get information from all the hidden states of the encoder, not just the last hidden state
- In the attention mechanism, as in the standard encoder-decoder model, the context vector c is a single vector that is a function of the hidden states of the encoder:

$$c = f(h_1^e, \dots, h_n^e)$$

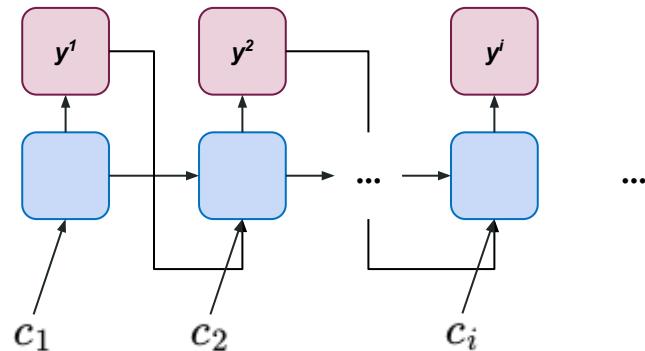
Attention

- Because the number of hidden states varies with the size of the input, we can't use the entire tensor of encoder hidden state vectors directly as the context for the decoder.
- The idea of attention is instead to create the single fixed-length vector c by taking a weighted sum of all the encoder hidden states.
- The weights focus on ('attend to') a particular part of the source text that is relevant for the token the decoder is currently producing.
- Attention thus replaces the static context vector with one that is dynamically derived from the encoder hidden states, different for each token in decoding.

Attention

- This context vector, c_i , is generated anew with each decoding step i and takes all of the encoder hidden states into account in its derivation.
- We then make this context available during decoding by conditioning the computation of the current decoder hidden state on it (along with the prior hidden state and the previous output generated by the decoder):

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$



Dot product attention

- The first step in computing c_i is to compute how relevant each encoder state is to the decoder state at time $i-1$
- We capture relevance by computing – at each i during decoding – a score between the current decoder state and each encoder state j
- The score **dot product attention** implements relevance as similarity: measuring how similar the decoder hidden state is to an encoder hidden state by computing the dot product:

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$

- The result of the dot product is a scalar that reflects the degree of similarity between the two vectors.
- The vector of these scores across over all the encoder hidden states gives us the relevance of each encoder state to the current step of the decoder.

Dot product attention

- To make use of these scores, we'll normalize them with a softmax to create a vector of weights, α_{ij} , that tells us the proportional relevance of each encoder hidden state j to the current decoder state:

$$\alpha_{ij} = \text{softmax}(\text{score}(h_{i-1}^d, h_j^e)), \forall j \in e$$

$$= \frac{\exp(\text{score}(h_{i-1}^d, h_j^e))}{\sum_k \exp(\text{score}(h_{i-1}^d, h_k^e))}$$

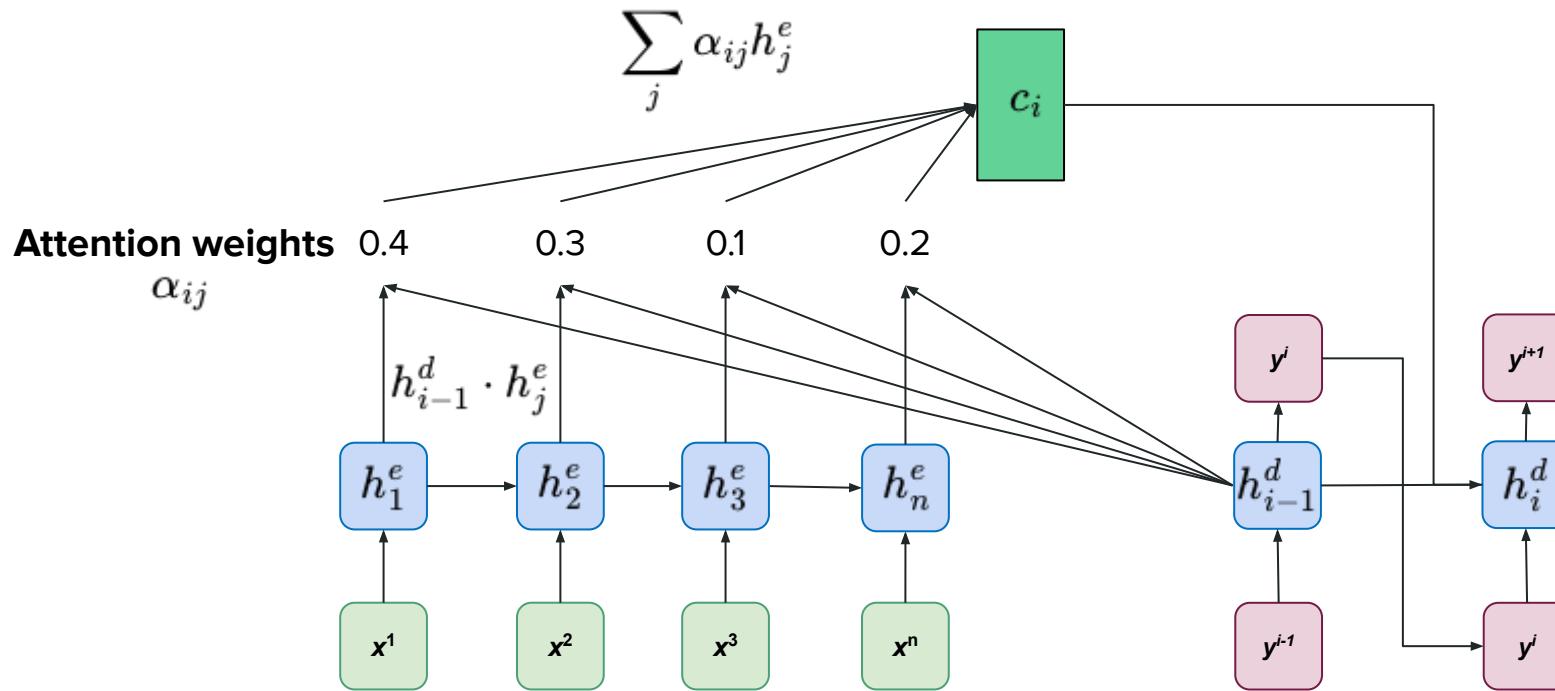
Computing the context

- Finally, given the distribution in α , we can compute a fixed-length context vector for the current decoder state by taking a weighted average over all the encoder hidden states:

$$c_i = \sum_j \alpha_{ij} h_j^e$$

- With this, we finally have a fixed-length context vector that takes into account information from the entire encoder state that is dynamically update to reflect the needs of the decoder at each step of decoding.

Encoder-decoder with dot product attention



More powerful attentions

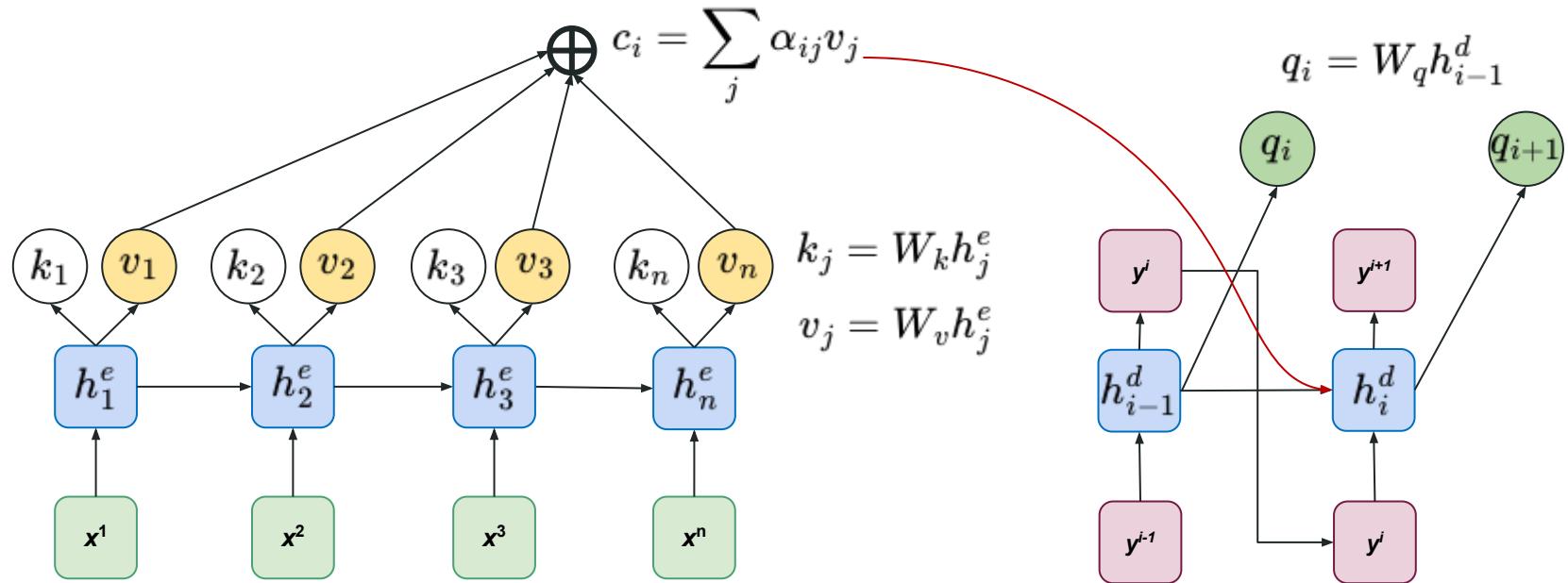
- Instead of simple dot product attention, we can get a more powerful function that computes the relevance of each encoder hidden state to the decoder hidden state by parameterizing the score with its own set of weights, W_s

$$\text{score}(h_{i-1}^d, h_j^e) = h_{i-1}^d W_s h_j^e$$

- The weights W_s , which are then trained during the normal end-to-end training, give the network the ability to learn which aspects of similarity between the decoder and encoder states are important to the current application.
- This bilinear model also allows the encoder and decoder to use different dimensional vectors.

More powerful attentions: query, key, value

$$\alpha = \text{softmax}(\mathbf{k}q_i^T)$$



More powerful attentions: query, key, value

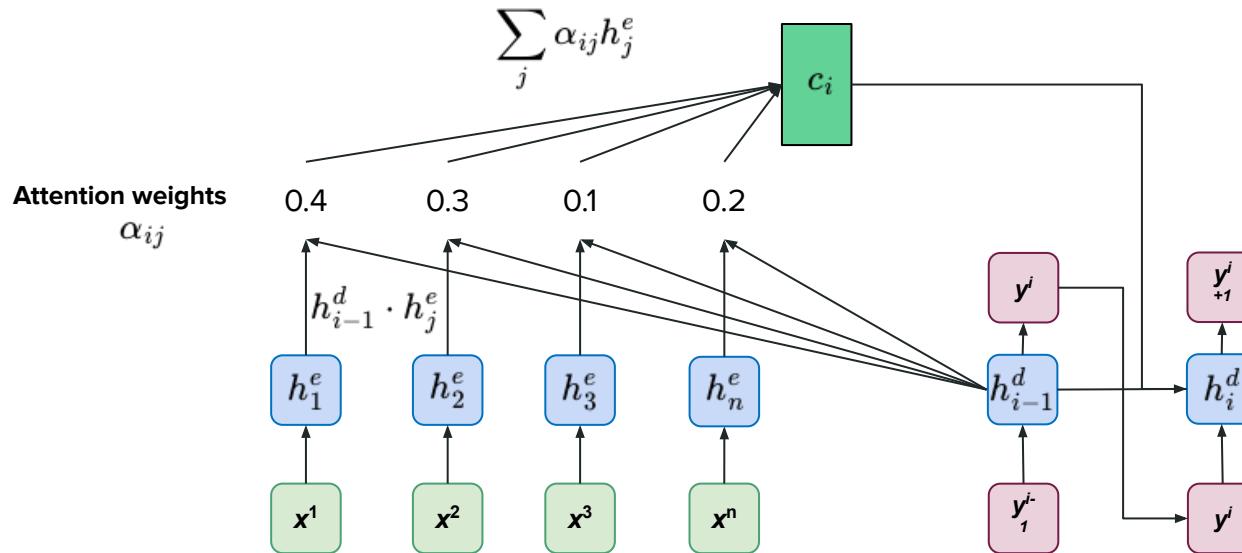
- Encoder outputs explicit “key” and “value” vectors at each time step
 - key: used to evaluate the importance of the different values at that time
- Decoder outputs an explicit “query” vector at each output time
 - Query: used to evaluate which input to pay attention to
- The attention weights are a function of key and query
- The context is weighted sum of value
- When:

$$k_j = v_j = h_j^e$$

$$q_i = h_i^d$$

we recover the standard case presented before

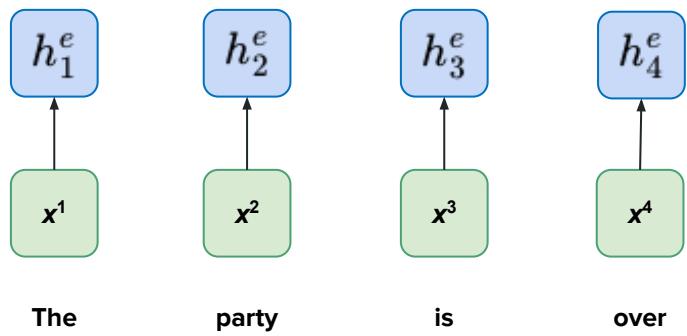
Non-recurrent encoders



- We get rid of recurrency at the encoder
- Problem: we eliminate context-specificity at the encoder output
- Solution: use the attention framework to directly compute context-aware encoder outputs (i.e., context-aware embeddings)
- Based on the self-attention mechanisms

Self attention

- First, for every word in the input sequence we compute an initial representation
 - E.g., using a single MLP layer



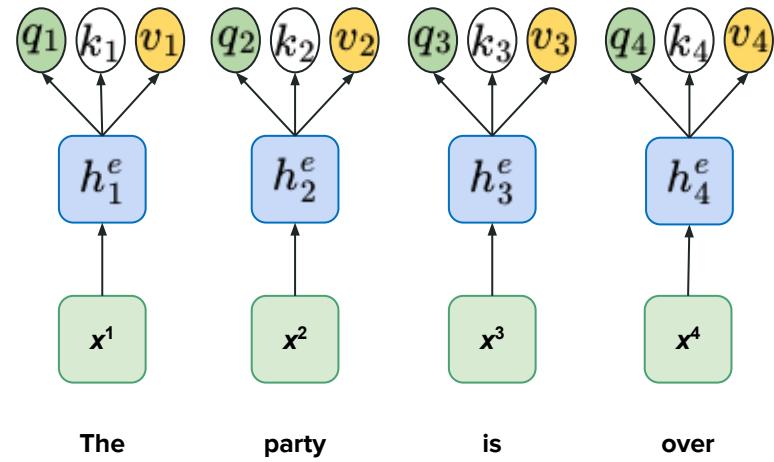
Self attention

- Then, from each representation, we compute a query, a key and a value
 - Using separate linear transforms
 - The weight matrices W_q , W_k and W_v are learnable parameters

$$q_j = W_q h_j^e$$

$$k_j = W_k h_j^e$$

$$v_j = W_v h_j^e$$



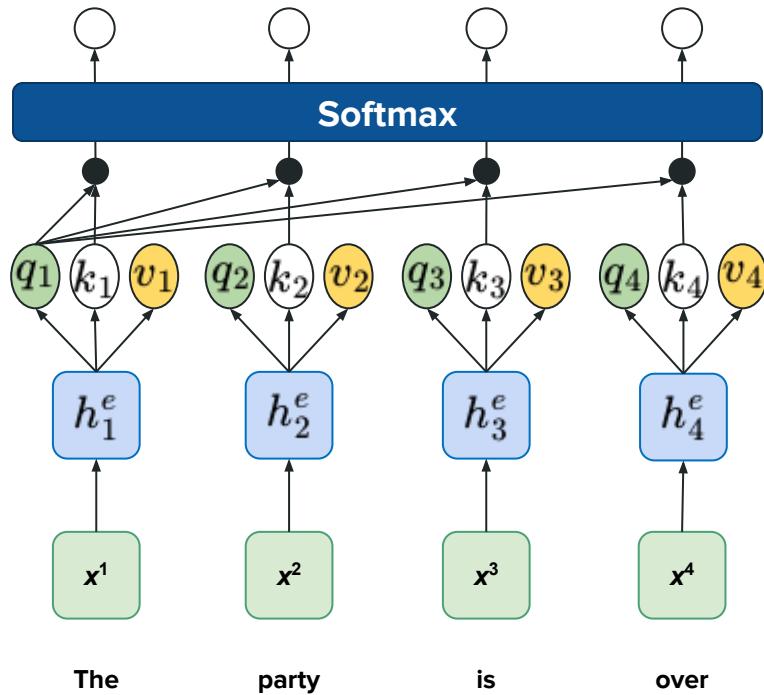
Self attention

- For each word, we compute an attention weight between that word and all other words
 - The raw attention of the i -th word to the j -th word is a function of q_i query and key k_j
 - The raw attention values are put through a softmax to get the final attention weights

$$\alpha_{ij} = \text{attn}(q_i, k_{0:N})$$

$$e_{ij} = q_i^T k_j$$

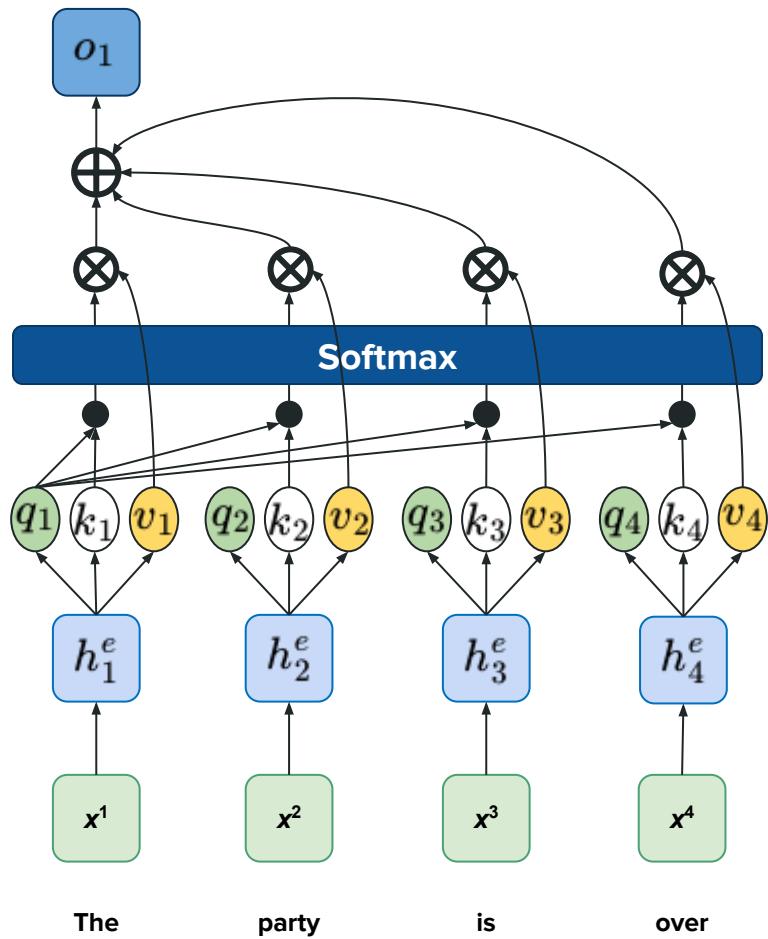
$$\alpha_{i0}, \dots, \alpha_{iN} = \text{softmax}(e_{i0}, \dots, e_{iN})$$



Self attention

- The updated representation for the word is the attention-weighted sum of the values for all words, including itself

$$o_i = \sum_j \alpha_{ij} v_j$$

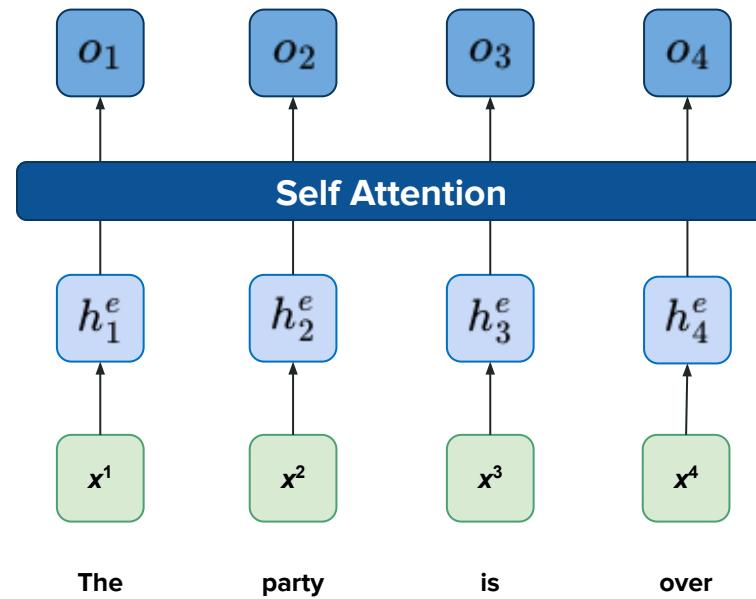


Self attention

- Obtain the context-aware representation for all input time steps

$$o_i = \sum_j \alpha_{ij} v_j$$

- This is a single-head attention block



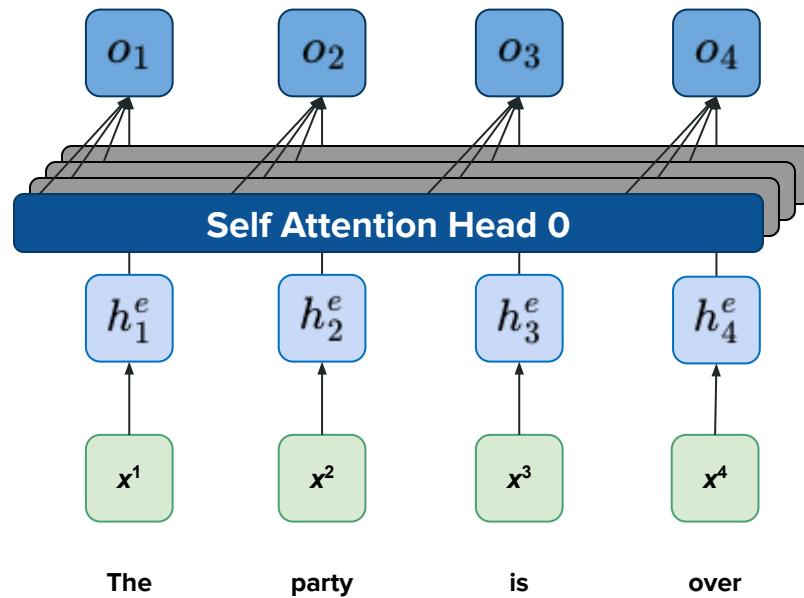
Multi-head Self attention

- We can have multiple such attention “heads”
 - Each will have an independent set of queries, keys and values
 - Each will obtain an independent set of attention weights
 - Potentially focusing on a different aspect of the input than other heads
 - Each computes an independent output
- The final output is the concatenation of the outputs of these attention heads

$$q_j^a = W_q^a h_j^e \quad k_j^a = W_k^a h_j^e \quad v_j^a = W_v^a h_j^e$$

$$\alpha_{ij}^a = attn(q_i^a, k_{0:N}^a) \quad o_i^a = \sum_j \alpha_{ij}^a v_j^a$$

$$o_i = [o_i^1; o_i^2; \dots; o_i^H]$$



Multi-head Self attention block

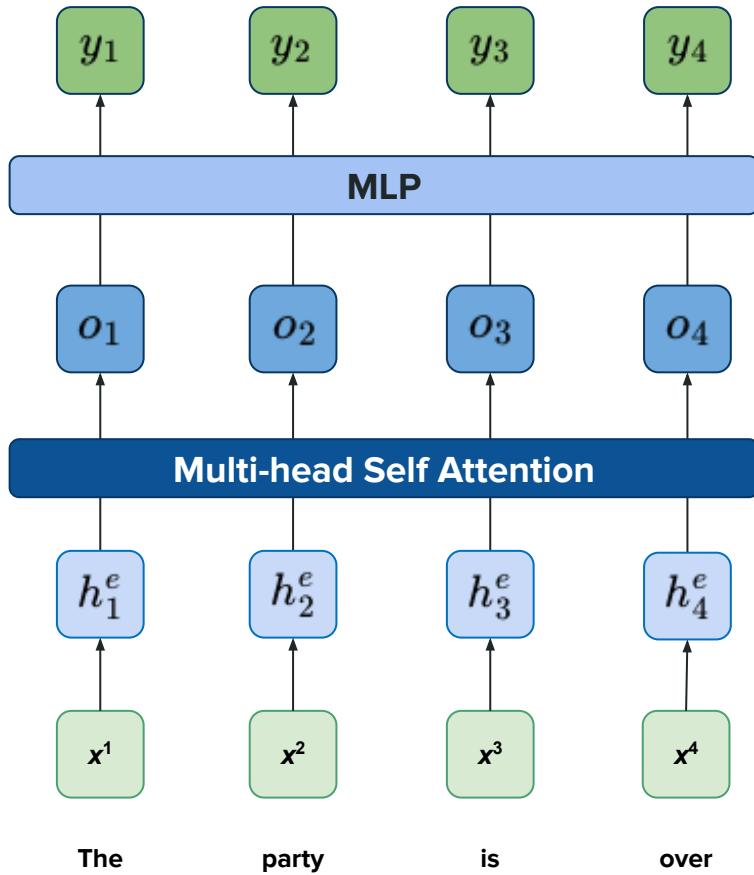
- The entire unit, including multi-head self-attention module followed by MLP is a multi-head self-attention block

$$q_j^a = W_q^a h_j^e \quad k_j^a = W_k^a h_j^e \quad v_j^a = W_v^a h_j^e$$

$$\alpha_{ij}^a = \text{attn}(q_i^a, k_{0:N}^a) \quad o_i^a = \sum_j \alpha_{ij}^a v_j^a$$

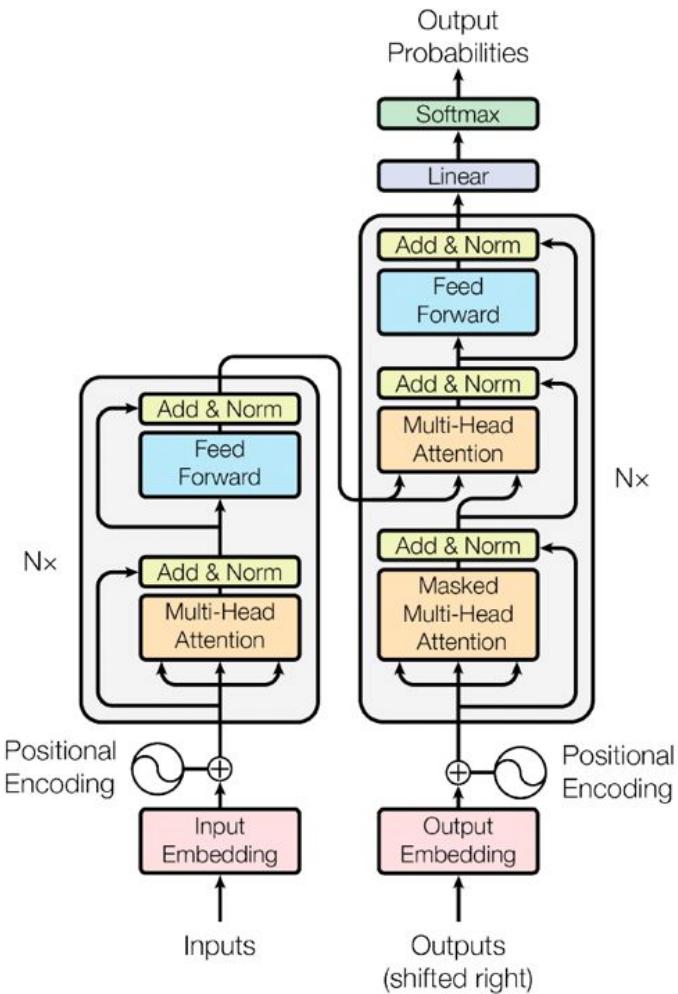
$$o_i = [o_i^1; o_i^2; \dots; o_i^H]$$

$$y_i = \text{MLP}(o_i)$$



Transformer

- Multi-head attention is at the basis of the Transformer architecture
- A sequence-to-sequence model that replaces recurrence with positional encoding and multi-head self attention
- A Transformer comprises two components:
 - A stack of encoder layers, adopting Multi-Head Self Attention blocks to produce a context-aware embedding of the input text
 - A stack of decoder layers, adopting Multi-Head Cross Attention blocks, producing the output sequence



Transformer performance

From “Attention is all you need”

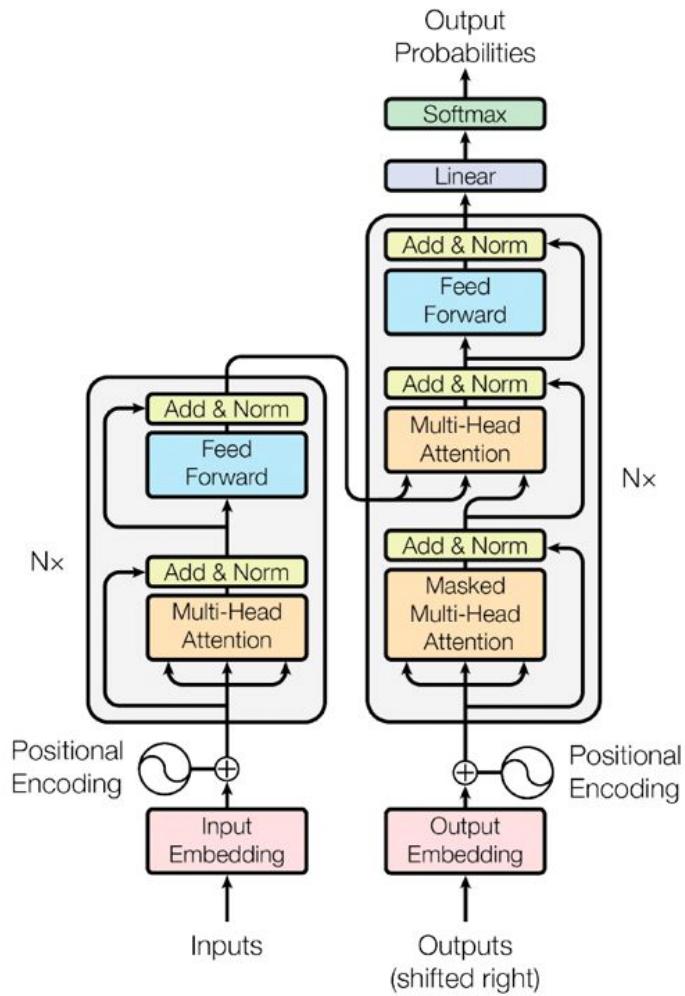
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---------------------------------|-------------|--------------|-----------------------|---------------------|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | 41.29 | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $3.3 \cdot 10^{18}$ | |
| Transformer (big) | 28.4 | 41.8 | $2.3 \cdot 10^{19}$ | |

Very good

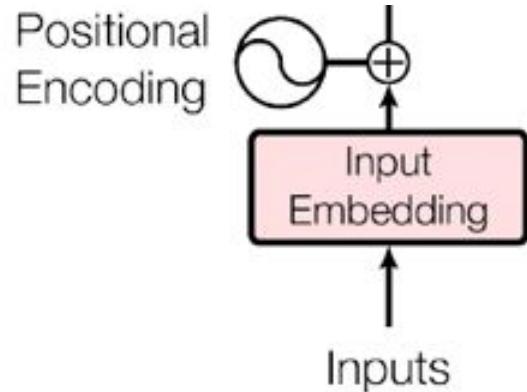
Very fast

- Tremendous decrease in model computation for similar performance as state-of-art translation models
- The last row in the table shows transformer performance
- The final two columns show computational cost



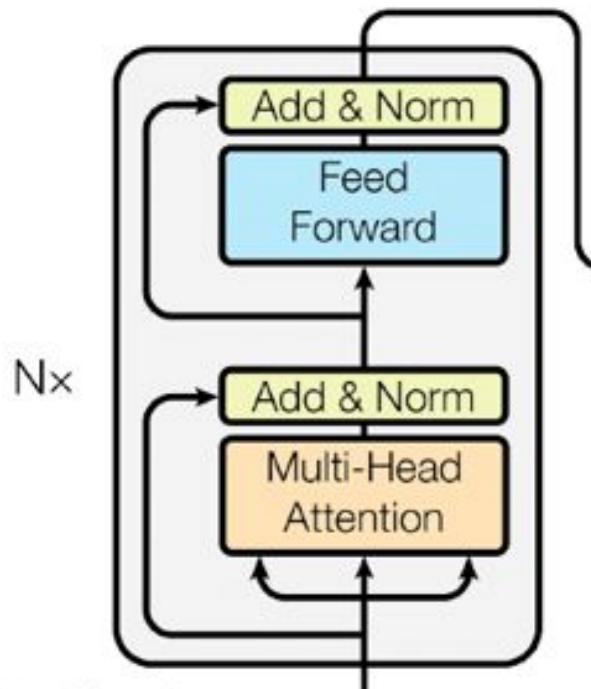
Transformer

- Input text is firstly positionally encoded using a embedding layer
- Positional encoding produces a preliminary numerical representation of the input text taking care of exploiting the relative positions of the words in the text



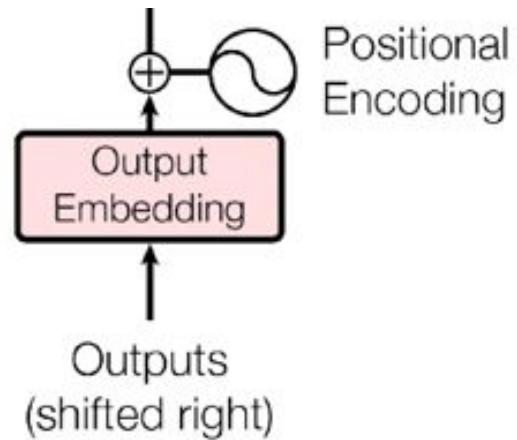
Transformer

- A single encoder layer internally adopts:
 - Multi-Head Self Attention block
 - Residual connections
 - Batch normalization
 - Output MLP
- The final output of the encoder layer is a contextualized embedding of the input sequence
- N such encoders layers are stacked in the final architecture



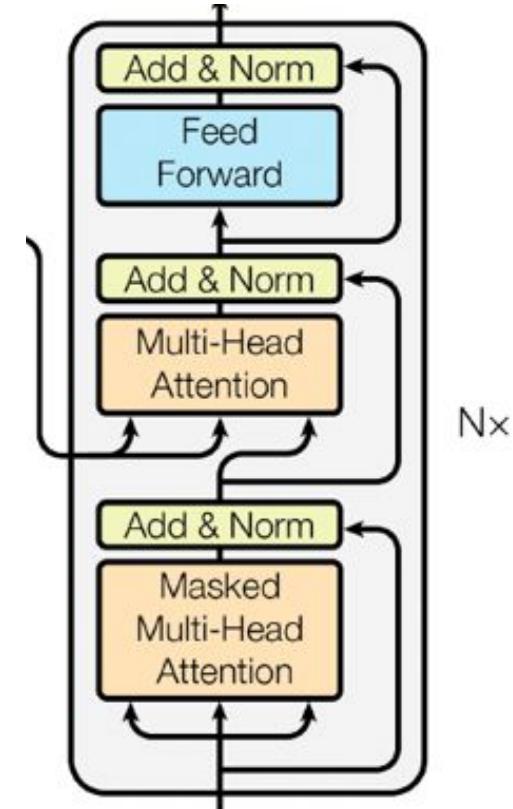
Transformer

- Positional encoding is also applied at the decoder side, to provide position aware embedding of the output sequence
- The output sequence is shifted right to let the decoder predict the next word given the previous (autoregressive decoder)



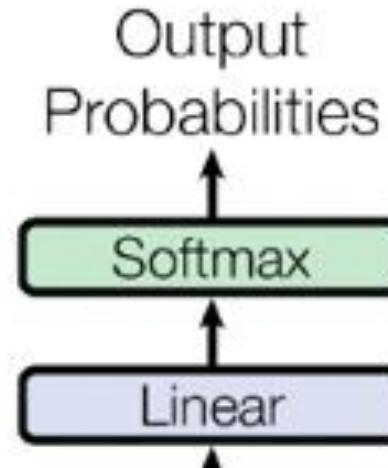
Transformers

- The decoder layer adopt two attention blocks
- A Masked Multi-Head Self Attention block to contextualize the output sequence from time 0 to t-1
- A Encoder-Decoder Multi-Head Attention (EDMHA) block which compute cross attentions between input and output sequence
- In practice, EDMHA takes the encoder output as key and queries, and the decoder internal state as values



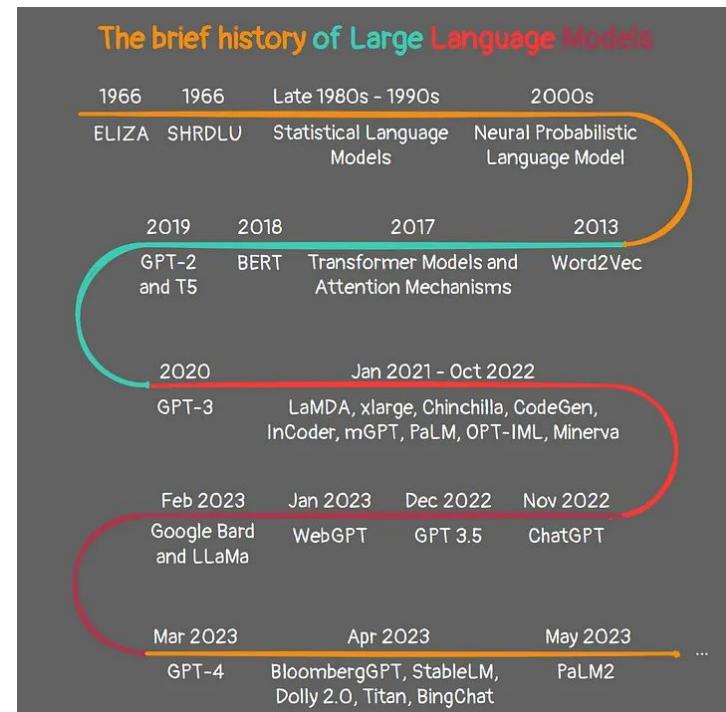
Transformer

- The decoder generates probabilities over all possible output words
- Outputs are usually generated one at a time, taking into consideration words generated in previous time steps



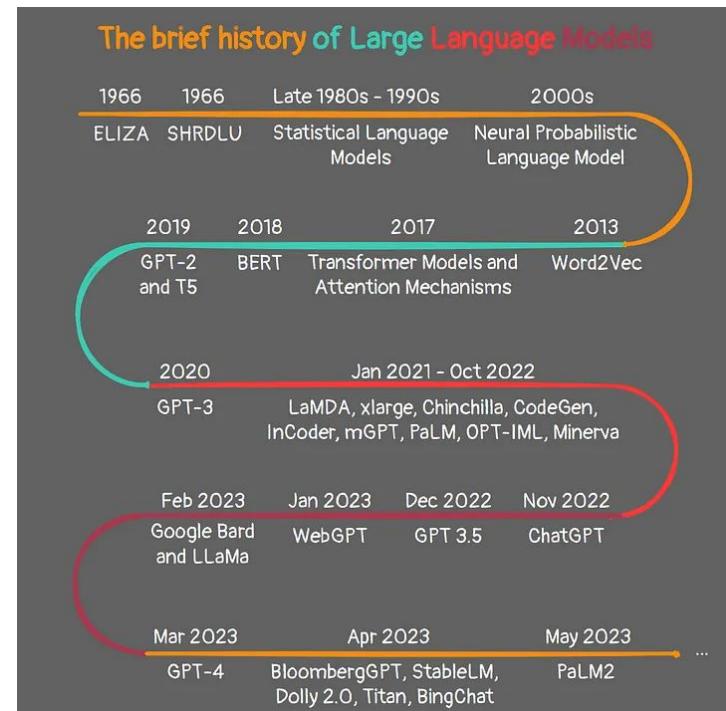
Language models

- Enable analysing patterns in language by predicting words
- Can be used to:
 - Compute the probability of a given token sequence
 - Generate sequences from the distribution of the language



Language models

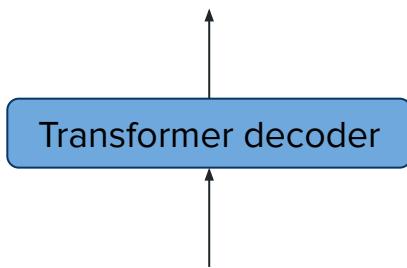
- Best performing models are based on Transformer architectures
- Many different categories:
 - Autoregressive models (or decoder-only models)
 - Autoencoding models (or encoder-only models)
 - Sequence-to-sequence models (or encoder-decoder models)



Transformers for LMs

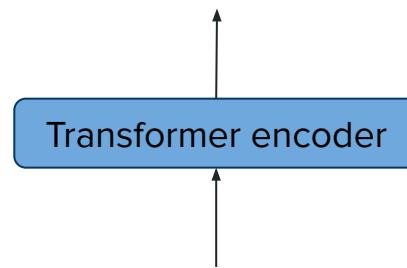
Autoregressive LM

Predict next word given the previous



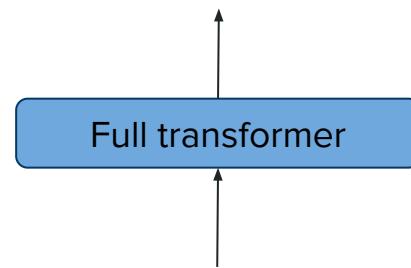
Autoencoding LM

Predict missing words



Sequence-to-sequence LM

Predict missing/next words



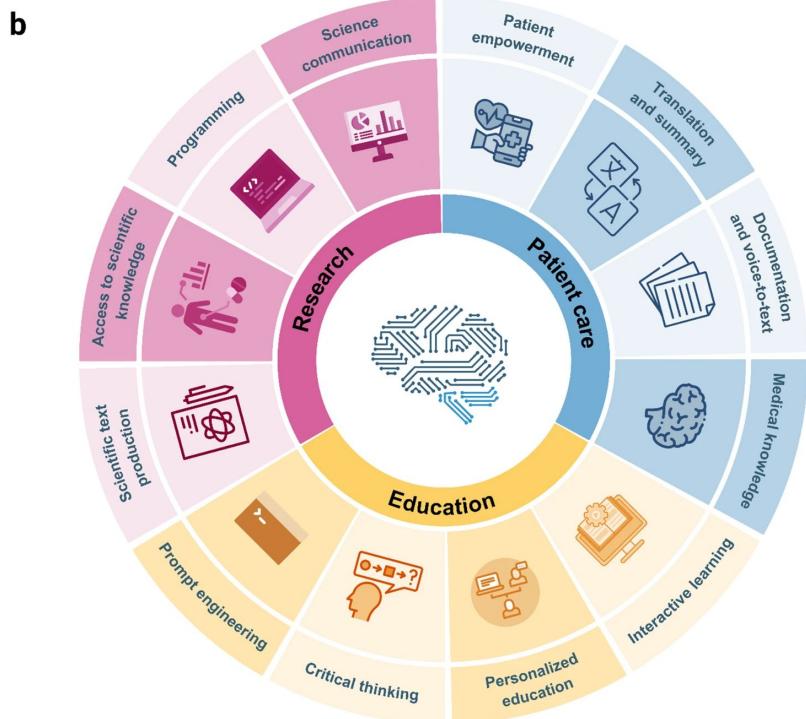
- Use masked attention in the decoder focusing only on previous words
- Examples: **GPT**

- Use masked input and try to reconstruct the text in output
- Examples: **BERT**

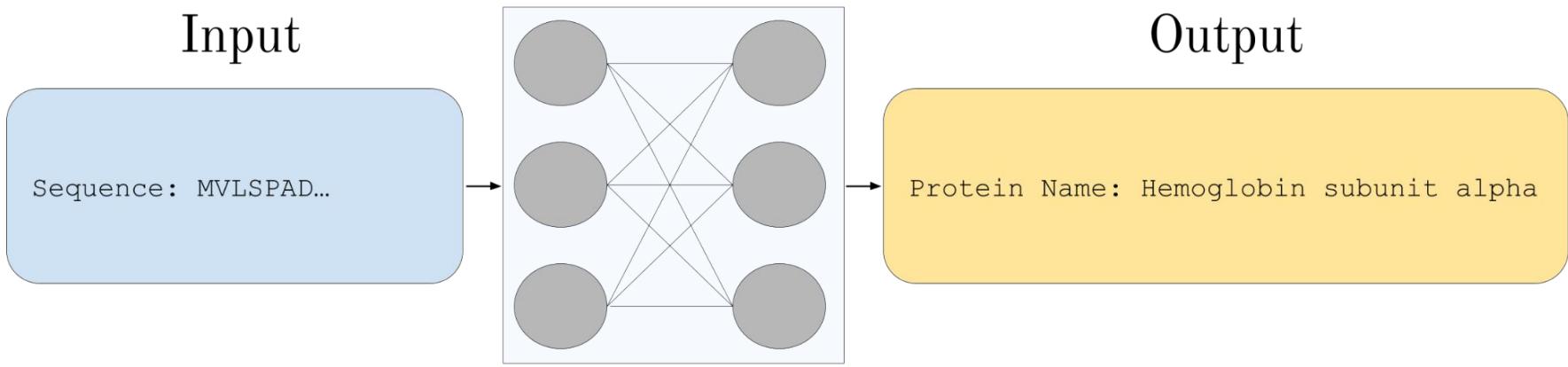
- Use the whole Transformer architecture
- Can be trained either as masked LM or autoregressively
- Examples: **T5**

Application of LMs in medicine

- Many potential applications in medicine
 - Research
 - Patient care
 - Education
- Fine-tuning of existing LMs might be needed to properly adopt them in the biomedical domain



UniProt ProtNLM



ProtNLM is a method developed by Google used by UniProt to automatically annotate uncharacterized protein sequences. This method works by predicting a short textual description for proteins based solely on their amino acid sequence, using a sequence-to-sequence model (T5X framework).

https://storage.googleapis.com/brain-genomics-public/research/proteins/protnlm/uniprot_2022_04/protnlm_preprint_draft.pdf

UniProt ProtNLM

A

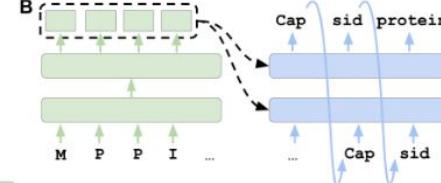


Caption: Dog with flowers

Minor capsid protein VP1

MPPIKRQPRGVLPGYRYLGPFPNPLDN...
VNNADRAAQQLHDHAYSELIKSGKNPFLYFN
KADEFKIDDLKDDWSIGGIIGSSFFKIRKA
VAPALGNKERAKRHFYFANSNKGAKTKK
SEPKPCTSKMSDTDIQDQQPDVTDAFQNTS
GGGTGSIGGGKGSGVGIVSTGGWVGGS... FSD
KYVVTKNTRQFITTION...

B



C

"Which protein family is: MPPIKRQPRGVLPGYRYLGP..."

"What's the name of the protein: MPPIKRQPRGVLPGYRYLGP..."

"Which EC number is associated with: MPPIKRQPRGVLPGYRYLGP..."

"Provide the organism classification associated with: MPPIKRQPRGVLPGYRYLGP..."

T5

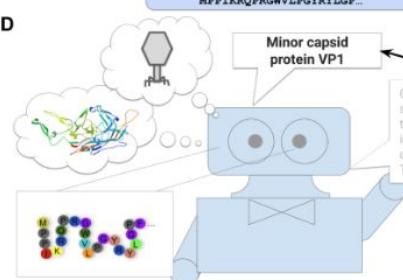
"Polyomavirus coat protein"

"Minor capsid protein VP1"

"3.1.1.4"

"Viruses > Monodnaviria > Parvoviridae ..."

D



Minor capsid protein VP1

Capsid proteins self-assembles to form an icosahedral capsid with a T=1 symmetry.

<https://www.uniprot.org/uniprot/Q3YPH4>

UniProt

Q3YPH4 · CAPSD_HBOC1

Minor capsid protein VP1 · Private lineage-specific 1 (strain Human Bocavirus 1 type 1) [HbAV1]

Unstable isoform PPE-1C3.1.1.4 · Gene: VP1 - 613 amino acids · Evidence at protein level · Annotations

Entry · Feature viewer · Publications · External links · History

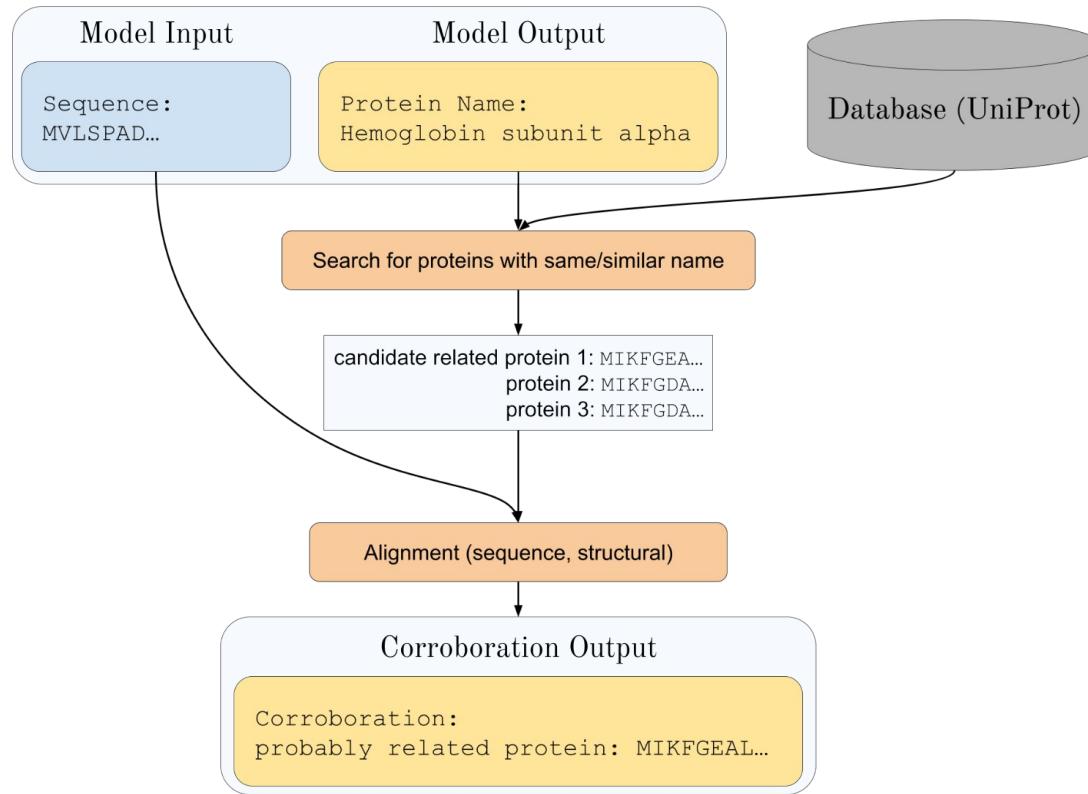
BLAST · Help · J. Download · Add · Add classification · Entry feedback

Function

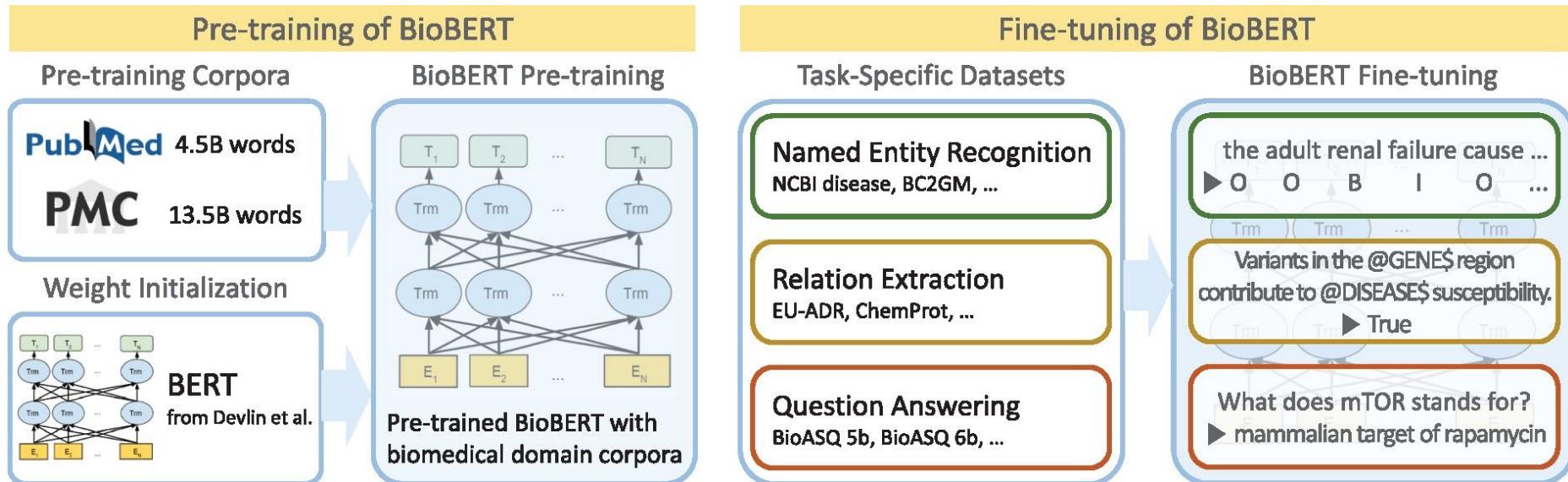
Capsid proteins self-assemble to form an icosahedral capsid with a T=1 symmetry, about 26 nm in diameter, and consisting of 60 copies of three size variants of the capsid proteins, VP1, VP2, and VP3. The capsid contains the viral genome (single-stranded DNA). The capsid is surrounded by a lipid bilayer that is termed as the S-hat shell and there are densities extending the S-hat layer into the interior of the capsid (PubMed:28233046).

The capsid encapsulates the genomic ssDNA (Probable). Binding to the host receptors also induces

UniProt ProtNLM: curation of predictions

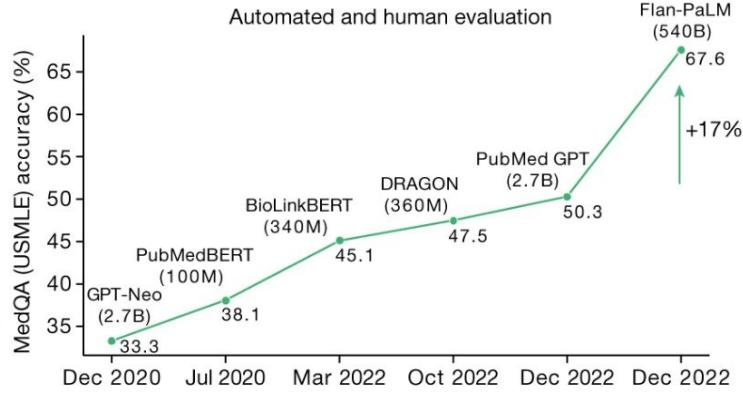
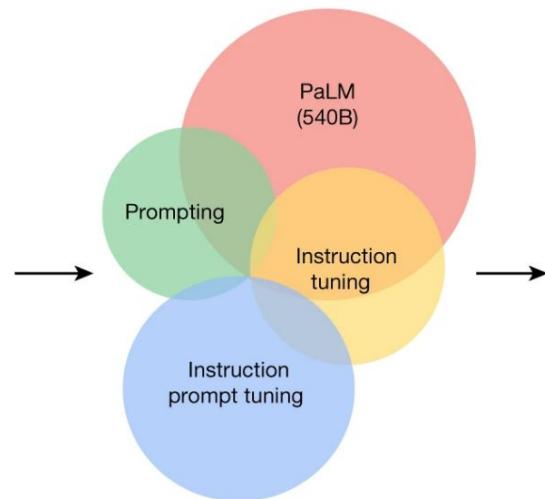
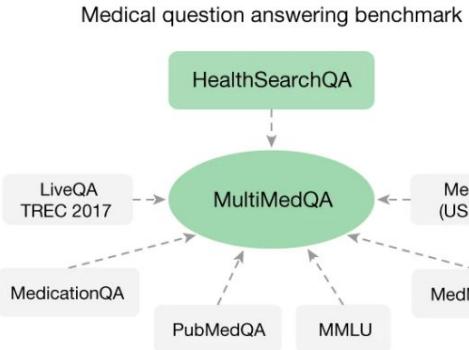


BioBERT



Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics*, Volume 36, Issue 4, February 2020, Pages 1234–1240

Med-PaLM



Q: How long does it take for newborn jaundice to go away?

Med-PaLM: Newborn jaundice is when a newborn baby's skin and eyes look yellow. It is caused by a buildup of a substance called bilirubin in the baby's blood. Bilirubin is a byproduct of the normal process of red blood cells breaking down. Newborn jaundice is common and typically harmless, but it can be a sign of a more serious condition. The jaundice typically goes away on its own within a few weeks. In some cases, it may last longer or require treatment. The time it takes for the jaundice to go away can vary depending on the cause and the severity of the jaundice. If the jaundice is severe or lasts longer than a few weeks, the doctor may recommend testing or treatment to determine the cause and prevent complications.

Med-PaLM performs encouragingly on consumer medical question answering

Singhal, K., Azizi, S., Tu, T. et al. Large language models encode clinical knowledge. *Nature* **620**, 172–180 (2023).

- Instruction prompt tuning to further align PaLM to the medical domain, producing Med-PaLM. Med-PaLM compares favourably with answers given by clinicians under our human evaluation framework
- Instruction tuning: a technique for fine-tuning large language models (LLMs) on a labeled dataset of instructional prompts and corresponding outputs. It improves model performance not only on specific tasks.

The Transformers library

Transformers provides APIs and tools to easily download and train state-of-the-art pretrained language models

These models support common tasks in different modalities, such as:

- Natural Language Processing: text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.
- Computer Vision: image classification, object detection, and segmentation.
- Audio: automatic speech recognition and audio classification.
- Multimodal: table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.
- **Protein language models**

Transformers support different frameworks including PyTorch, TensorFlow, and JAX

Documentation available at: <https://huggingface.co/docs/transformers/index>

Models are available at the Huggingface collaboration platform (<https://huggingface.co/>)

Examples

- Toy example of text generation pipeline

[https://colab.research.google.com/drive/10R1hQNGfH7rt6veZewBFq1wRpOQmc2th
#scrollTo=c2SzIV34mHbU](https://colab.research.google.com/drive/10R1hQNGfH7rt6veZewBFq1wRpOQmc2th#scrollTo=c2SzIV34mHbU)

- Example of model fine tuning

[https://colab.research.google.com/drive/1l9fnawy8-Pv7pGGZnrJt4pevj6B5naQV?us
p=sharing](https://colab.research.google.com/drive/1l9fnawy8-Pv7pGGZnrJt4pevj6B5naQV?usp=sharing)

Summary

- Attention is a mechanism to let a model understand what is important during the inference process
- Attention is at the core of Transformers, which are used to define large language models
- Language model: a model trained on large datasets of text for sequence generation/translation
- Many application, even in the biomedicine field:
 - Automatic protein annotation
 - Medical text mining
 - Medical question answering

References

1. Vaswani et al. "Attention is all you need." Advances in neural information processing systems. 2017
2. Clusmann, J., Kolbinger, F.R., Muti, H.S. et al. The future landscape of large language models in medicine. Commun Med 3, 141 (2023)
3. Gane et al. ProtNLM: Model-based Natural Language Protein Annotation
4. Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics, Volume 36, Issue 4, February 2020, Pages 1234–1240
5. Singhal, K., Azizi, S., Tu, T. et al. Large language models encode clinical knowledge. Nature 620, 172–180 (2023).

Introduction to deep-learning methods

Modelling the language of proteins

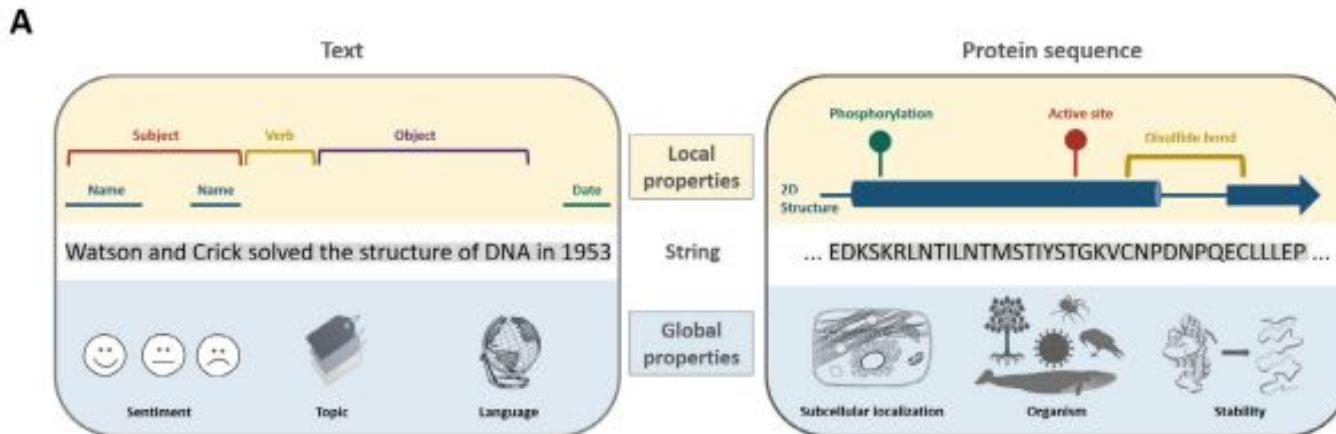
Laboratory of Bioinformatics II – Module 2
International Bologna Master in Bioinformatics
A.A. 2024-2025
Castrense Savojardo - Biocomputing Group, Dept. of Pharmacy and Biotechnology
castrense.savojardo2@unibo.it

Outline

- Introduction
- Multi-layer perceptrons (MLPs)
- Convolutional neural networks (CNNs)
- Recurrent neural networks (RNNs)
- Graph neural networks (GNNs)
- Attention mechanisms
- Transformers
- **Protein language models**
- Applications
- Conclusions

The “language” of proteins

Proteins are similar to natural languages



The “language” of proteins

But not exactly the same:

- Proteins cannot be “read” as texts
- Vocabulary of proteins is much smaller (20 different “words”) than vocabularies of natural languages (thousands of words)
- Small changes (single-residue mutations) in a protein are semantically negligible with respect to small changes in natural languages:
 - I love you vs I loved you
- No punctuation no differentiation of words, sentences and paragraphs
- NL have fewer distant interactions — proteins 3D structure
- Letters of proteins can be modified (e.g., post-translational modifications)

Tokenizing proteins

Natural languages

String: The cat sat on the mat

Possible tokenizations: [*start*] [T] [h] [e] [*space*] [c] [a] [t] ...
[*start*] [The] [cat] [sat] [on] [the] [mat]
[*start*] [The] [cat sat on] [the] [mat]

Proteins

MSTIYSTGKVCNP...

[*start*] [M] [S] [T] [I] [Y] [S] [T] [G] ...
[*start*] [MS] [TI] [YS] [TG] ...
[*start*] [M] [STI] [YST] [GK] [VCN] ...

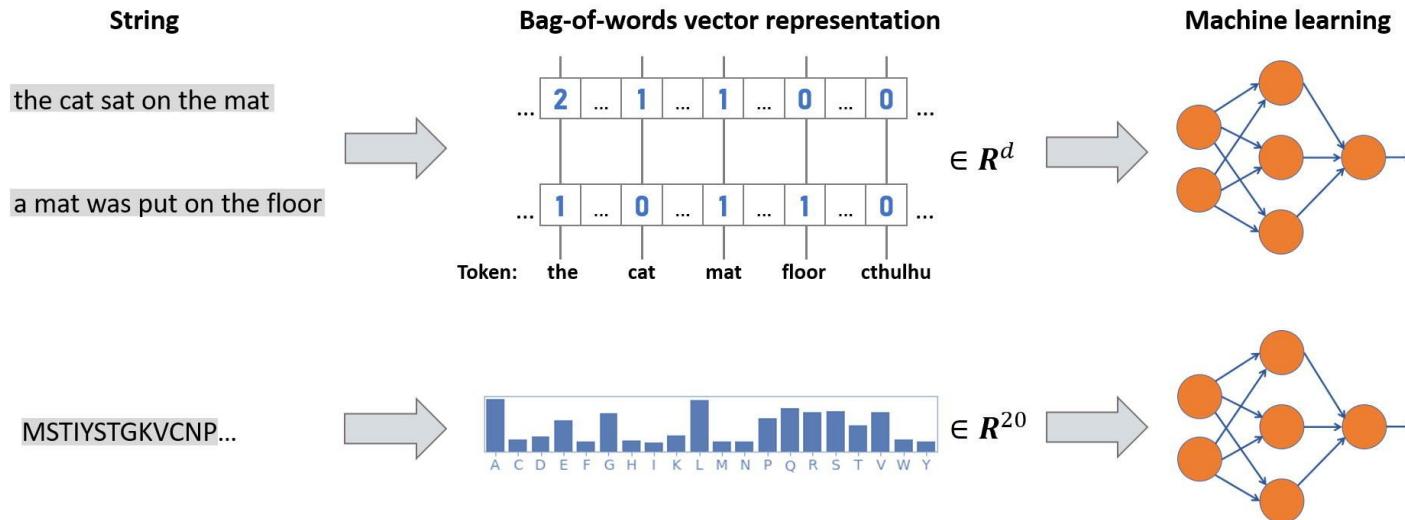
Modelling the language of proteins

- Define models that are able to capture protein structural and functional features only analysing a large set of proteins sequences
- It makes sense, since sequence determines structure that determines function
- Are proteins learnable?

Linguistic hypothesis:

“The space of naturally occurring proteins occupies a learnable manifold. This manifold emerges from evolutionary pressures that heavily encourage the reuse of components at many scales: from short motifs of secondary structure, to entire globular domains”

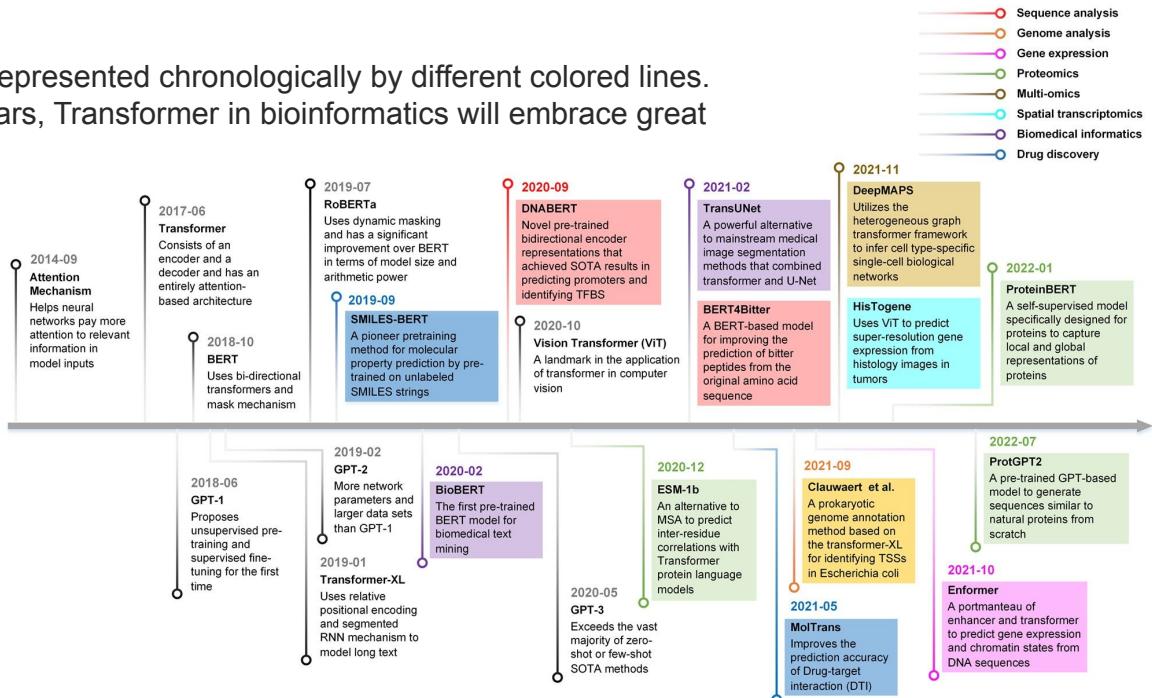
Naive approach to model (protein) languages



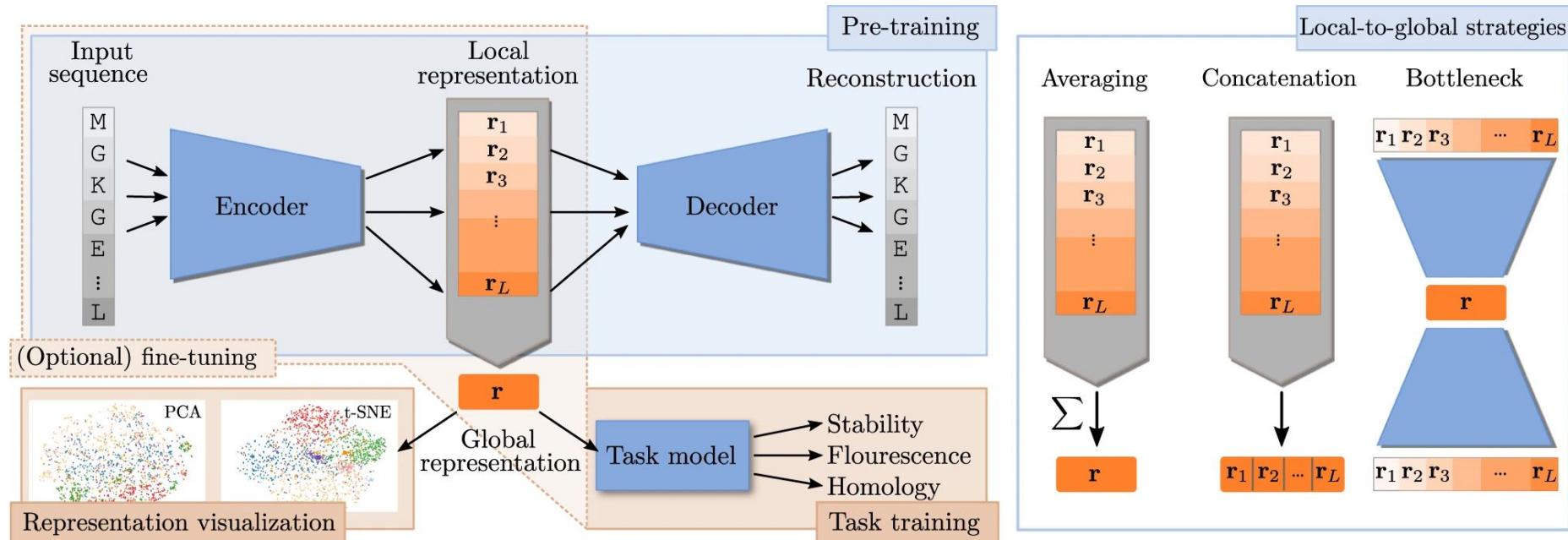
Protein Language Models (pLMs)

Different bioinformatics application models are represented chronologically by different colored lines. Following the prominent progress in the past years, Transformer in bioinformatics will embrace great advancement in the upcoming years.

Pre-training: use transformer-based architectures similar to NLP models. These architectures leverage self-attention mechanisms to learn complex relationships and patterns within protein sequences. The model is trained using self-supervised learning, aiming to predict missing parts of protein sequences given the context of the surrounding sequences. This context-based prediction allows the model to capture meaningful representations of proteins.



pLMs: basic idea



pLMs: terminology

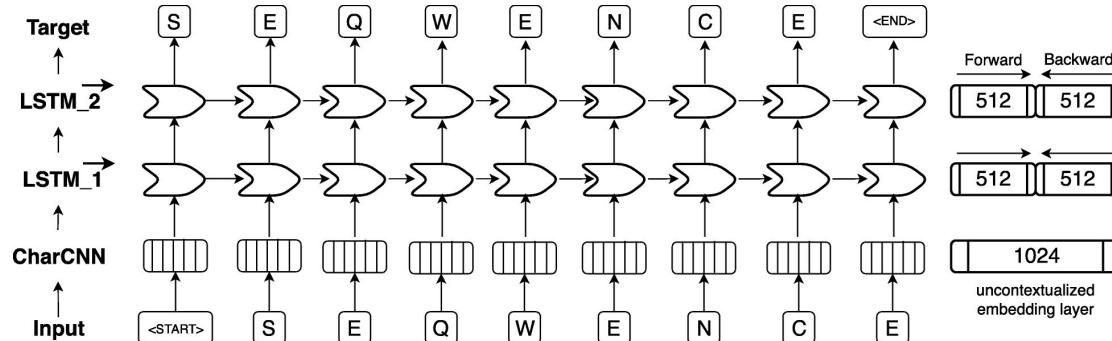
- **Pre-training:** the process of training a model on a large-scale dataset comprising millions/billions of protein sequences
 - Typical datasets: UniRef50/90, BFD
 - Training is self-supervised: the pLM is trained in reconstructing corrupted input sequences in output
- **Fine-tuning:** the process of refining parameters of pre-trained pLM on a task-specific dataset
 - The pLM is extended with an additional **classification/regression head** acting on the pLM representations
 - Refinement is performed in a supervised learning setting on the specific task
- **Transfer learning:** using embeddings as produced by the pLM to solve a specific task

pLMs: history

- First wave of pLMs were based on simple RNN-based architectures
 - SeqVec (Heinzinger *et al.*, 2019)
 - UniRep (Alley *et al.*, 2019)
- SOTA pLMs now based on Transformers
 - ESM (Rives *et al.*, 2021; Lin *et al.*, 2023)
 - ProtTrans (Elnaggar *et al.*, 2022)
 - ProstT5 (Heinzinger *et al.*, 2023)
- Generative pLMs
 - ProtGPT (Ferruz *et al.*, 2022)

SeqVec

- Developed at the Rostlab (Technical University of Munich)
- One of the first attempts of modelling protein sequences as a language
- Based on the ELMo NLP architecture: CNN + LSTM networks
- Model trained autoregressively: predicting the next residue in the sequence
- Training dataset: UniRef50 database comprising 33M proteins



SeqVec

Tested on different standard tasks:

- Visualization of embedding space w.r.t. SCOP, EC, Kingdoms
- Per-protein prediction tasks: Localization, membrane proteins
- Per-residue prediction tasks: secondary structure, disorder

Results:

- Embeddings encode protein properties
- Performance are better than single-sequence and worse than profile-based encodings

Visualizing embedding spaces

Using dimensionality reduction techniques to project n-dimensional embedding vectors (either residue- or protein-level) data into 2 or 3 dimensions

Common techniques adopted in the context of pLMs:

- t-distributed stochastic neighbor embedding (tSNE)
- Uniform Manifold Approximation and Projection (UMAP)

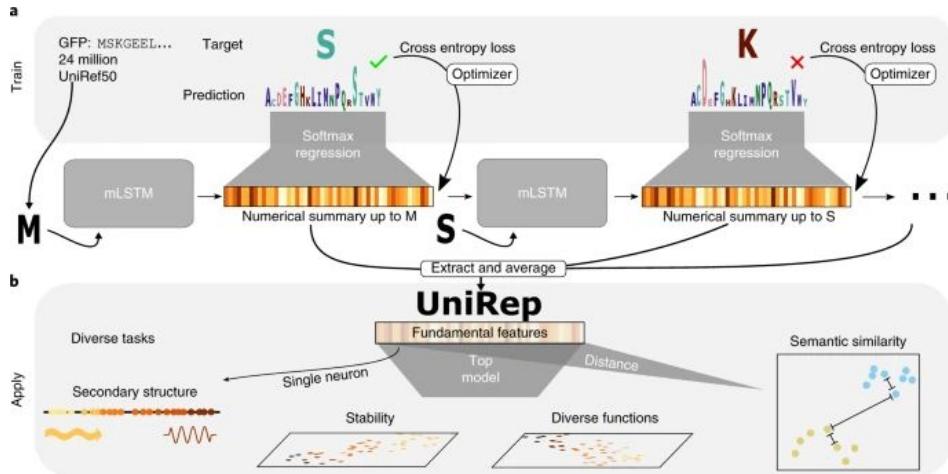
tSNE

- Input: a set of N point in some high-dimensional space $\mathbf{x}_1, \dots, \mathbf{x}_N$
- First estimate a probability p_{ij} that the two points x_i and x_j are clustered together in the original space
 - p_{ij} is estimated as proportional to the similarity of the involved points in the original space
- Then, learn a lower-dimensional map $\mathbf{y}_1, \dots, \mathbf{y}_N$ reflecting the similarities p_{ij}
- For computing the map, the algorithm estimate and refine a probability q_{ij} in the lower-dimensional space, reflecting similarities between points i and j
- The locations of the point y are estimated minimizing the Kullback-Liebler divergence between P and Q

$$\text{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

UniRep

- Developed at the George Church lab in the USA (Harvard Medical School)
- Architecture based on LSTM similar to SeqVec



ESM

Family of models developed by the MetaAI research team

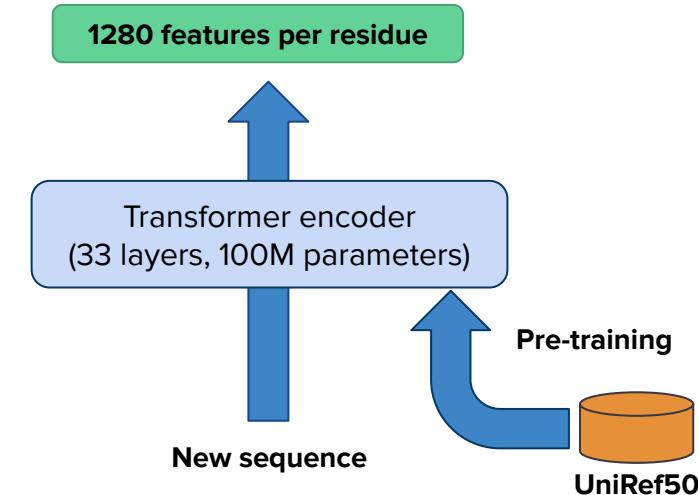
ESM1-b

- Based on BERT-style encoder only architecture including 33 layers
- Pre-trained on UniRef50 (about 60M sequences) on masked input sequences
- Produces 1280-dimensional residue embeddings

ESM2

- Improved ESM1 architecture
- Pre-trained on UniRef50, masked input
- A family of models of increasing complexity (i.e., number of encoder layers from 6 to 48)
- Embeddings of variable sizes, from 320 to 5120
- Used within the ESMFold protein structure prediction method

ESM1-b

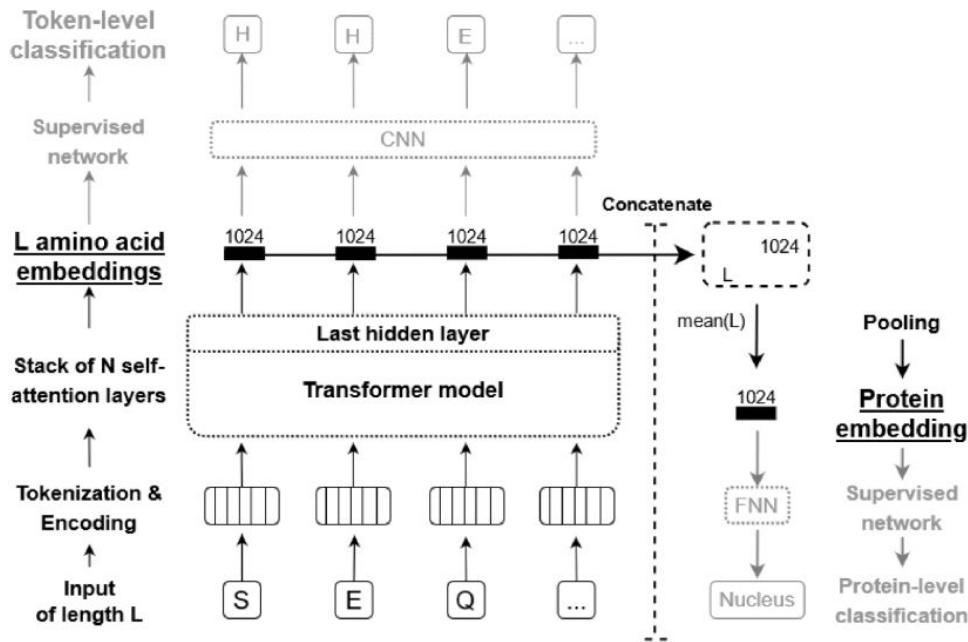


Rives et al., Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci U S A. 2021 Apr 13;118(15):e2016239118.

Lin et al., Evolutionary-scale prediction of atomic-level protein structure with a language model. Science 379 ,1123-1130 (2023). DOI:10.1126/science.adc2574

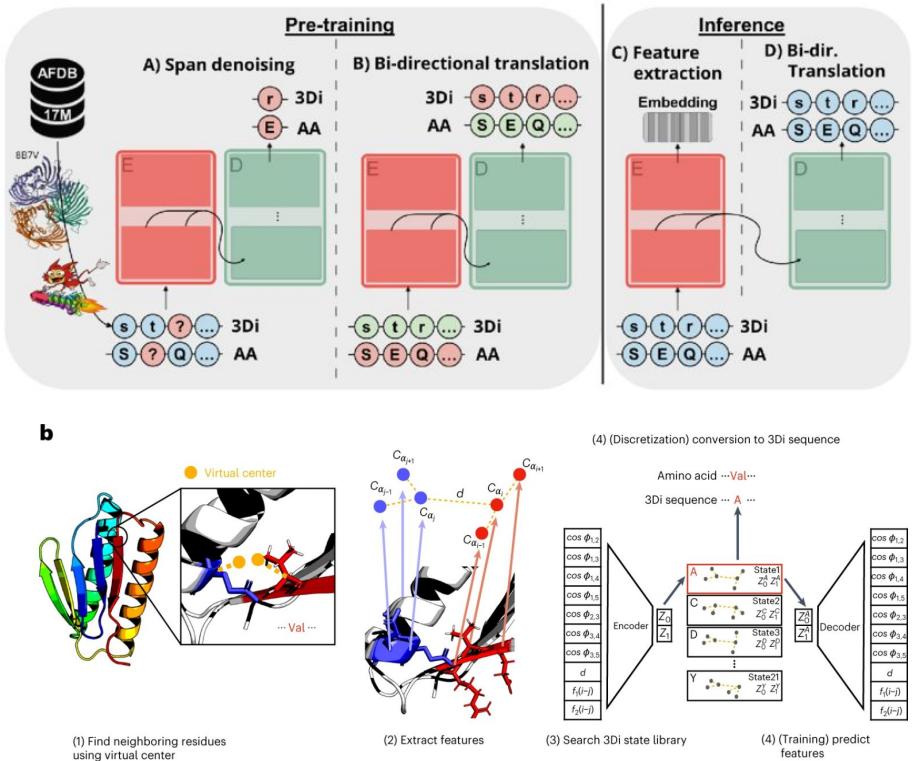
ProtTrans

- Transformer-based family of models developed at the Rostlab
- Based on the sequence-to-sequence architecture T5
- Pre-training performed in two steps:
 - First pass using a the very large BFD database comprising 3B sequences
 - Refinement using UniRef50
- Producing embedding of 1024 features per residue



ProstT5

- Bilingual protein language model developed at Rostlab
 - Based on T5 architecture
 - Sequence to structure mapping treated as a translation task
 - Input language: protein sequences
 - Output language: structural states derived from Foldseek 3Di strings
 - Can be used to:
 - Build structure aware embeddings
 - “Translate” protein sequences to 3Di
 - “Translate” 3Di to protein sequences



Heinzinger et al. (2023) Bilingual Language Model for Protein Sequence and Structure, bioRxiv 2023.07.23.550085

van Kempen, M., Kim, S.S., Tumescheit, C. et al. Fast and accurate protein structure search with Foldseek. Nat Biotechnol 42, 243–246 (2024).

ProtGPT2

- Autoregressive model based on GPT2 architecture
- Can generate protein sequences conserving natural protein properties (amino acid propensities, secondary structure, globular/disorder domains) while exploring new regions of the protein sequence space
- Two ways of using ProtGPT2:
 - Using the pre-trained model, generate a dataset of synthetic proteins
 - After fine tuning on a dataset of interest, can focus on properties of a specific protein set (e.g., a family) during the generation process
- Important applications in de-novo protein design

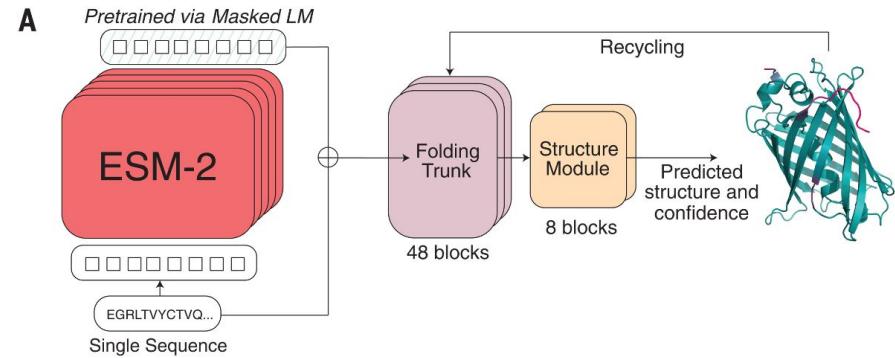
pLM applications

pLMs have been applied in recent years to almost all traditional areas of protein structural and functional Bioinformatics

- Protein structure prediction
- Protein homology detection and functional annotation
- Protein-protein interactions at different scales (network, PPI interface)
- Evaluation of the structural and functional impact of protein variations

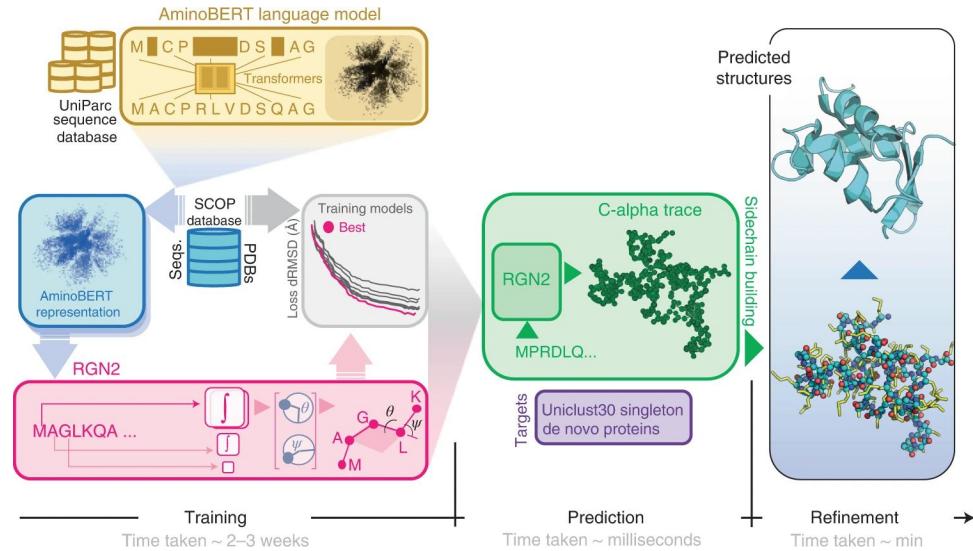
Protein structure prediction using pLMs

- Deep-learning architecture similar to AF2
- **Not based on MSA input**
- Input: **ESM2 embeddings**
- **No structural templates used as in AF2**
- Much faster than AF2 with only a slight loss in prediction performance
- **ESM Metagenomics Atlas:** 772 M predicted metagenomic protein structures from sequence in the MGnify database
- Well-suited for orphan proteins (no good MSA) and designed proteins



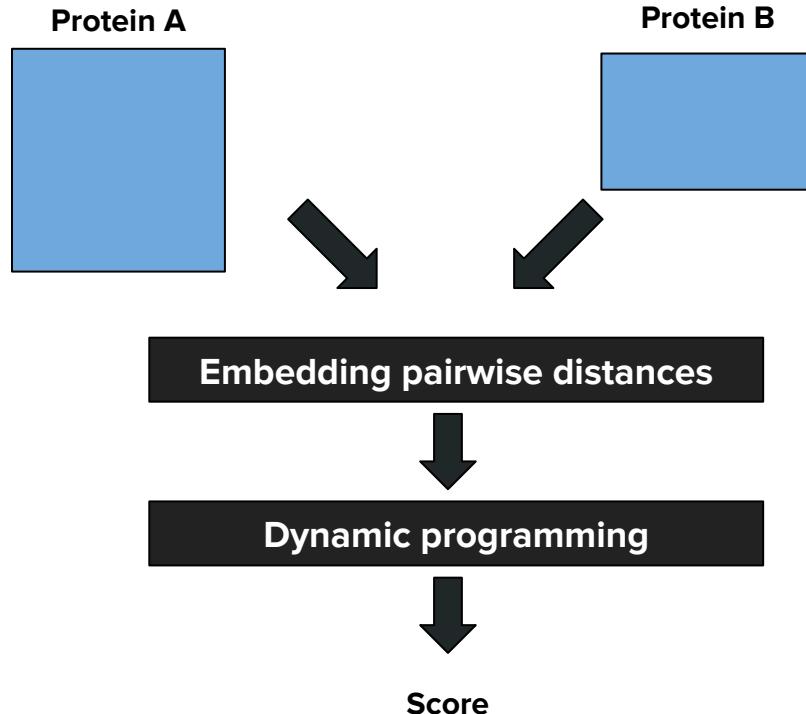
Protein structure prediction using pLMs

- Another single-sequence based approach based on a pLM developed ad-hoc, called AminoBERT
- 3D structure predicted using Frenet-Serret model from embeddings



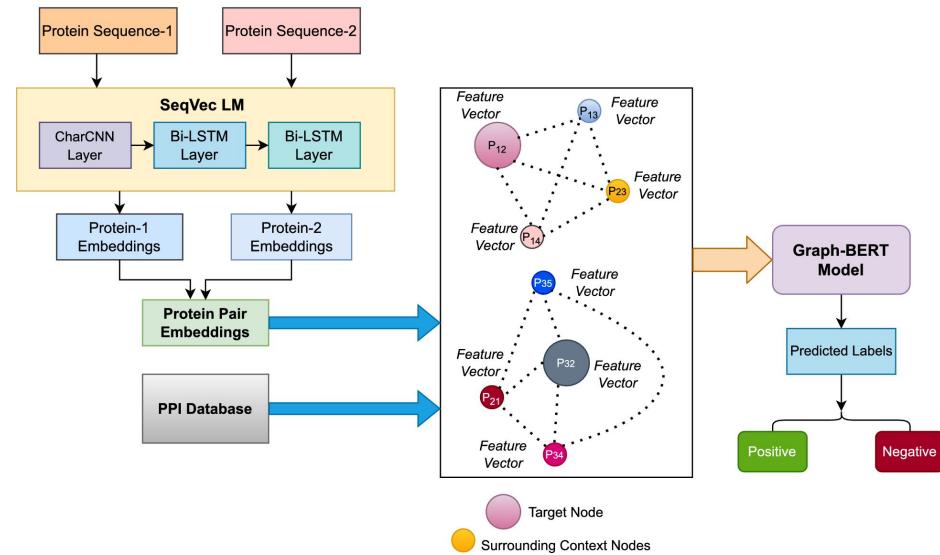
Protein function prediction using pLMs

- Remote homology detection and function prediction resulting from comparison of pLM embeddings
- Can be done using average embeddings or directly on the whole matrix comprising residue-level embeddings
- EBA: use dynamic programming algorithms to “align” embedding matrices
- Homology detected from alignment score



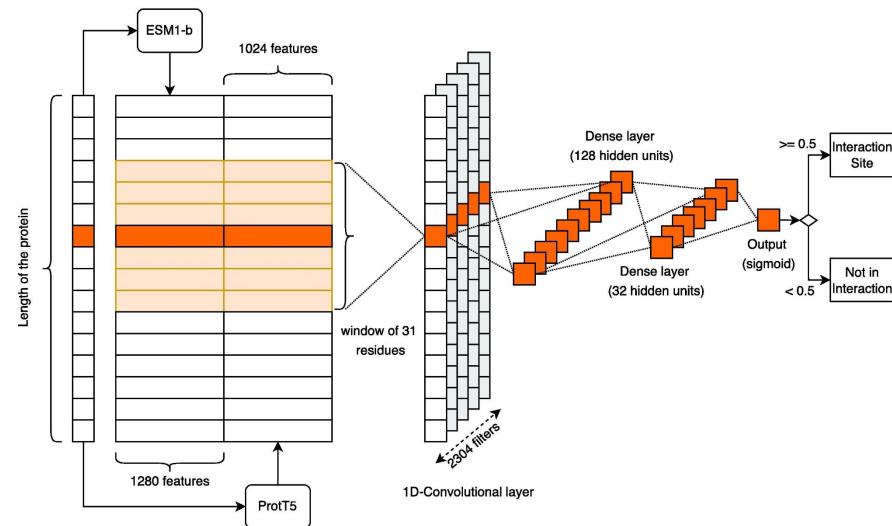
Protein-protein interaction prediction using pLMs

- Infer new PPIs using a combination of pLM and GNN
- SeqVec embeddings used for generating graph node features
- GNN used to infer new links



Protein-protein interaction prediction using pLMs

- ISPRED-SEQ: Residue-level prediction of protein-protein interface
- Works on unbound protein sequences to infer putative interaction sites (partner unspecific)
- Protein sequences is embedded using two pLMs: ESM1-b and ProtT5
- A simple convolutional network is then adopted for predicting whether a residue is part or not of the interface



Impact of mutations using pLMs

VESPA method:

- Based on ProtT5 embeddings predicts the functional effect of protein variants
- Trained on the about 100K variations endowed with binary effects (neutral, effect)
- Tested on different datasets, including large deep-mutational scanning test sets

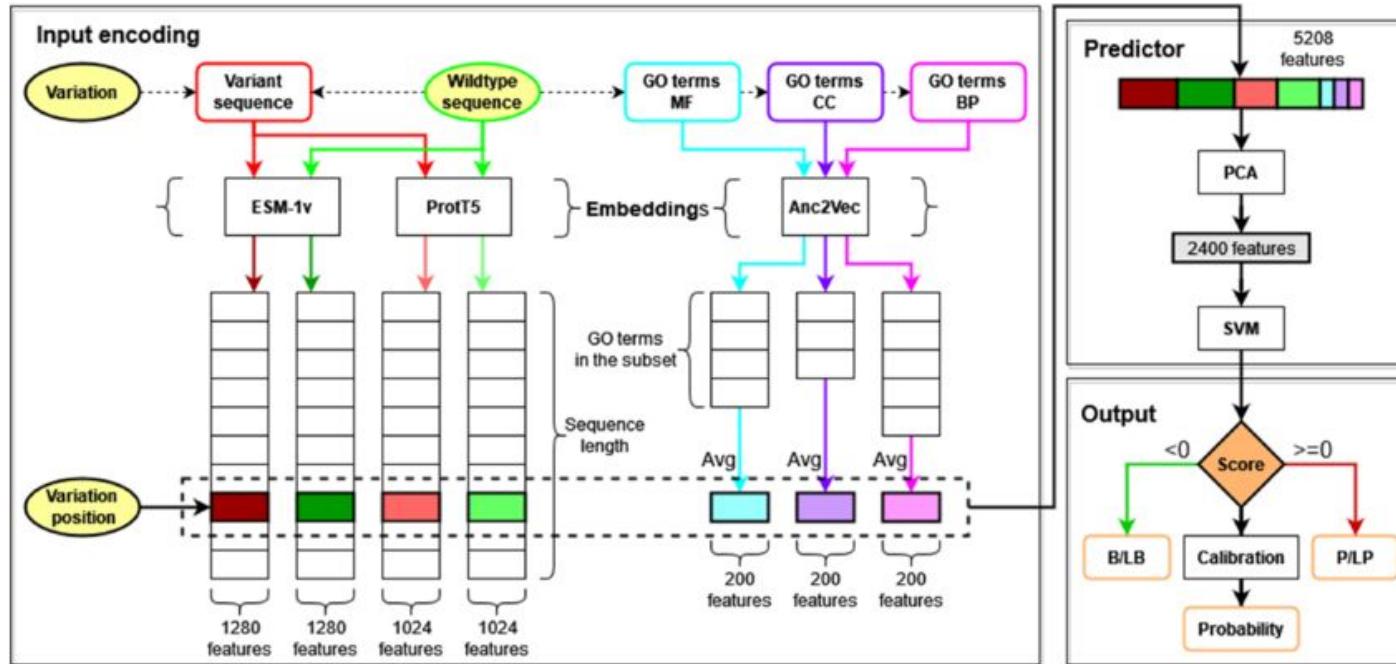
Marquet, C., Heinzinger, M., Olenyi, T. et al. Embeddings from protein language models predict conservation and variant effects. *Hum Genet* **141**, 1629–1647 (2022).

E-SNPs&GO:

- Predict variant pathogenicity using two different pLM embeddings (ProtT5 and ESM1-v)
- Also adopts the Anc2Vec method to produce embeddings of Gene Ontology terms

Manfredi M, Savojardo C, Martelli PL, Casadio R. E-SNPs&GO: embedding of protein sequence and function improves the annotation of human pathogenic variants. *Bioinformatics*. 2022 Nov 30;38(23):5168-5174

E-SNPs&GO



Manfredi M, Savojardo C, Martelli PL, Casadio R. E-SNPs&GO: embedding of protein sequence and function improves the annotation of human pathogenic variants. Bioinformatics. 2022 Nov 30;38(23):5168-5174

Examples

Embed and visualize using ESM2:

<https://colab.research.google.com/drive/1qsl8rltMRsJOfZR2ilaKlyOL13FZQmMG?usp=sharing>

Exercise

- Implement a NN model to predict SCOP classes using ESM2 embeddings
 - Use a small ESM2 model (e.g. esm2_t33_650M_UR50D)
- Datasets (training, validation, testing) are available on Virtuale
- The model can be either:
 - Based on full-sequence embedding representations (residue-level) and an architecture that can handle variable-length inputs (e.g., CNN, LSTM)
 - Based on pooled embedding representations (e.g., average embedding vector) and simple MLP architecture
- Optimize the model architecture to get the best performance on validation set
- Use the testing to evaluate generalization
- Use previously presented example Colabs as reference
- Create a single colab (or script if you prefer) and share it with me