Module 4 Part 3: Classification Approaches From Simple to Complex

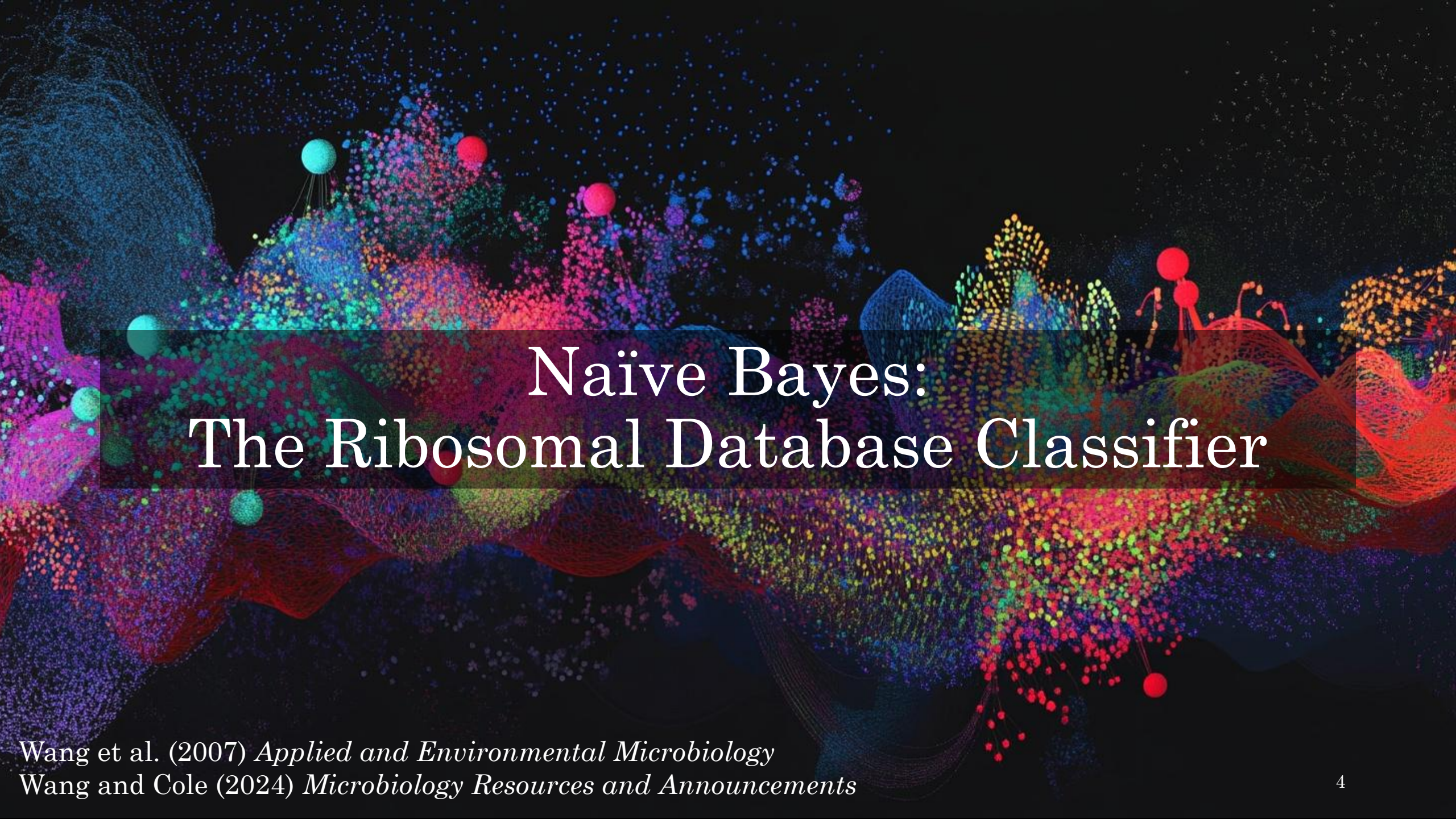
Remember our decision criteria

- Bias-variance tradeoff
- Tractability
- Interpretability

Examples

- Naïve Bayes – the Ribosomal Database Project classifier
- Random Forests – characterizing fish populations
- Fancy complicated transformer models (DNABERT and DNABERT-2)

DNABERT-2: Zhou et al. (2023) *arXiv* https://github.com/MAGICS-LAB/DNABERT_2

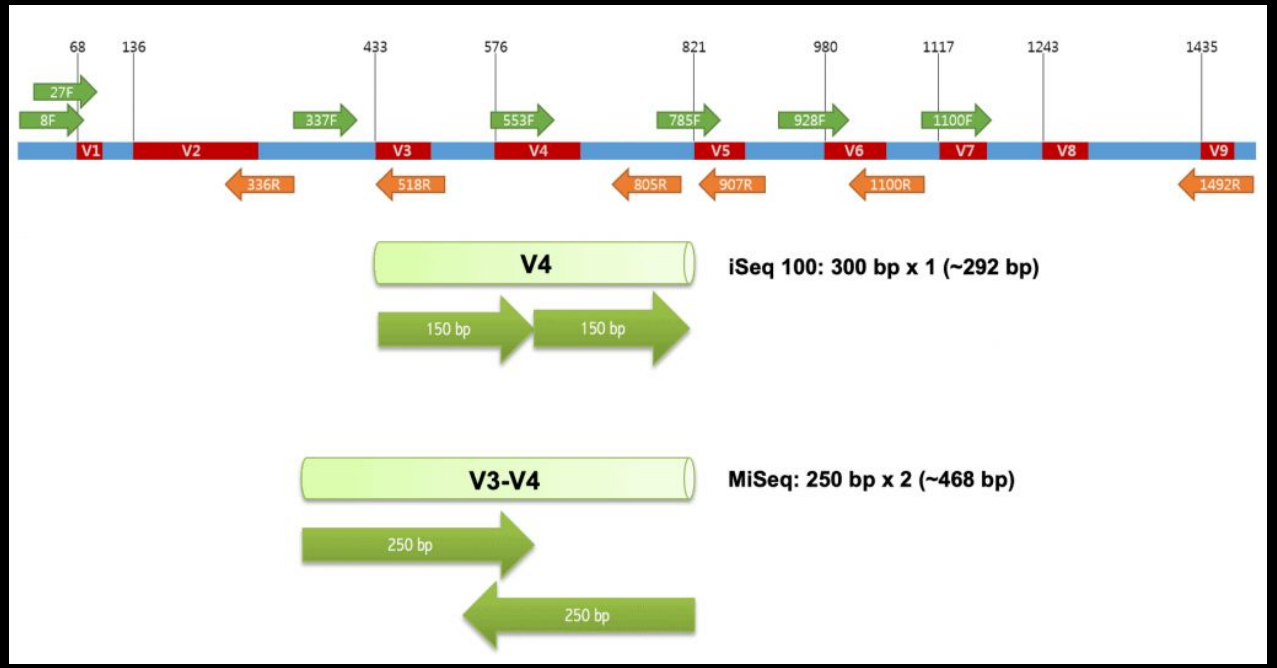


Naïve Bayes: The Ribosomal Database Classifier

Wang et al. (2007) *Applied and Environmental Microbiology*
Wang and Cole (2024) *Microbiology Resources and Announcements*

The Challenge: Classifying Microbial Communities

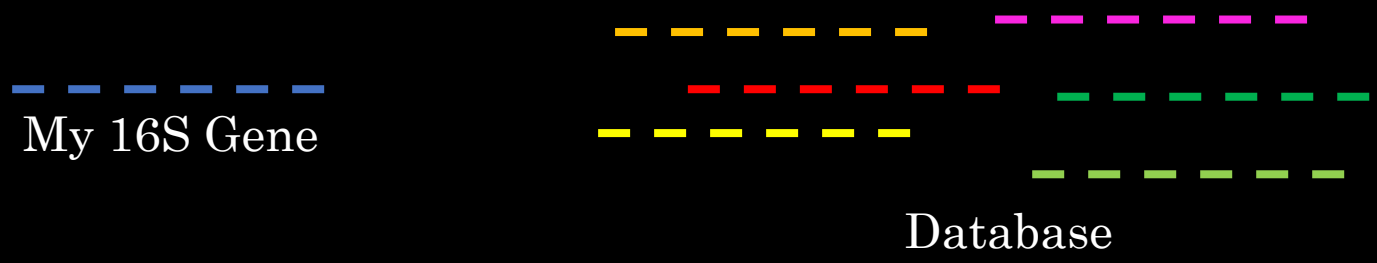
- Markers such as the 16S ribosomal RNA gene are universal, and differ enough to distinguish microbial species*
- So we sequence a bunch of 16S sequences from $HABITAT, now what?



*kind of
https://help.ezbiocloud.net/16s-rrna-and-16s-rrna-gene/

We can do best BLAST matching, but…

- BLAST isn't really set up for taxonomic classification



- What happens if:
  - The correct species isn't in the database?
  - The choice of cutoff isn't obvious?
  - We get equally good matches to sequences from multiple species?
- We need a classifier that can assign a 16S sequence to a given taxonomic group, if there is sufficient evidence

The RDP Classifier

1. Start with a big database of 16S sequences
2. Decompose each sequence into a series of 8-mers (4^8 = 65,536 possible words!)
3. Train the classifier to make predictions at the species level (and by extension, at every other taxonomic rank)
4. Compute some measure of support for the classification

1. Database

- Original 2007 paper:
  - 23,095 sequences from NCBI
  - 5014 reference “type strain sequences”

- Users can train the RDP classifier with their own sequences using e.g. RESCRIPt

Robeson et al. (2021) *PLoS Computational Biology*

2. k-mer decomposition

Each sequence S is represented by a vector x of its k-mer frequencies

S_i	x
E. coli	{ AA = 0.05, AC = 0.03, AG = 0.08, AT = 0.04, CA = …
Y. pestis	{ AA = 0.05, AC = 0.04, AG = 0.10, AT = 0.02, CA = …
C. difficile	{ AA = 0.01, AC = 0.05, AG = 0.08, AT = 0.02, CA = …
E. faecium	{ AA = 0.03, AC = 0.03, AG = 0.09, AT = 0.01, CA = …

3. Classification

Naïve Bayes: For a set of n classes C_1, C_2, C_3, …, C_n, and a problem instance x (usually represented with a feature vector),

Compute the probability of each C_i, given x

Posterior probability of membership in class i

$$p(C_i \mid x) = \frac{p(C_i)\, p(x \mid C_i)}{p(x)}$$

Prior probability of class i

Likelihood of x under model C_i

Probability of x

Predicted class: C_i that maximizes $p(C_i \mid x)$

The likelihood of x under C_i

k-mer decompositions are compositional models for each reference 16S gene

$$p(x \mid C_i) = \prod_{j=1}^{q} p(x_j \mid C_i)$$

For all k-mers in x…

The frequency of k-mer j in the genes representing C_i

Under a flat prior, the "winning" species is the one that maximizes the likelihood and therefore the posterior probability

A rank-flexible classifier

- Predicting the species implicitly predicts all higher taxonomic ranks too:
  - Species: *Escherichia coli*
  - Genus: *Escherichia*
  - Family: Enterobacteriaceae
  - Order: Enterobacteriales
  - Class: Gammaproteobacteria
  - Phylum: Proteobacteria (old school) / Pseudomonadota (new)
  - Domain: Bacteria

4. Support

- The RDP classifier does not use the Bayesian posterior as the measure of support

- Bootstrapping: sample a smaller subset (1/8 of all words) with replacement. How often do we get the same answer?

d__Bacteria; p__Firmicutes; c__Bacilli; o__Lactobacillales; f__Lactobacillaceae; g__Lactobacillus; s__Lactobacillus_crispatus = 0.99

4. Rank-flexible classification

- Set a bootstrap support threshold T

- If no species has support $\geq T$, check whether the aggregate support for the genus $\geq T$
- Can do this for higher ranks too
- If $T = 0.8$:

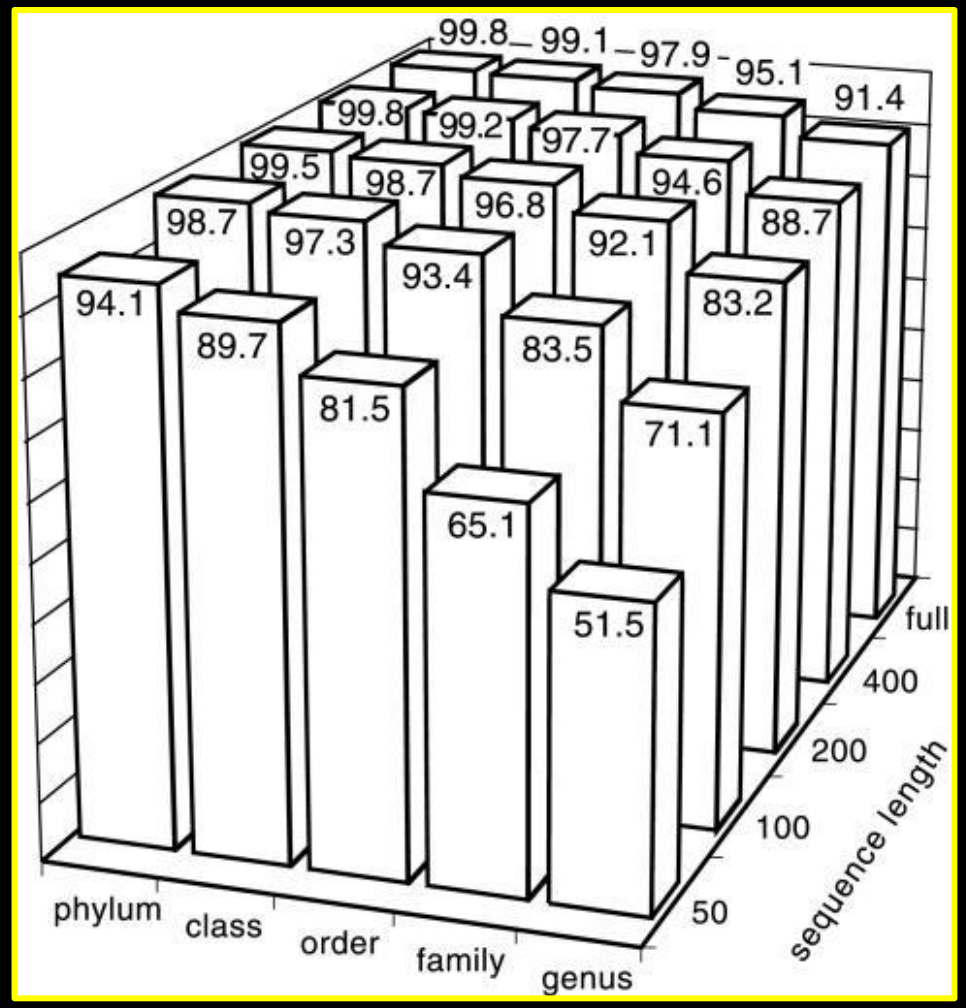
…; g__Lactobacillus; s__Lactobacillus_crispatus = 0.6 ✗
…; g__Lactobacillus; s__Lactobacillus_iners = 0.3 ✗

No classification at species level

But total support for *Lactobacillus* = 0.9 ✓

RDP Classifier Performance (2007 version)

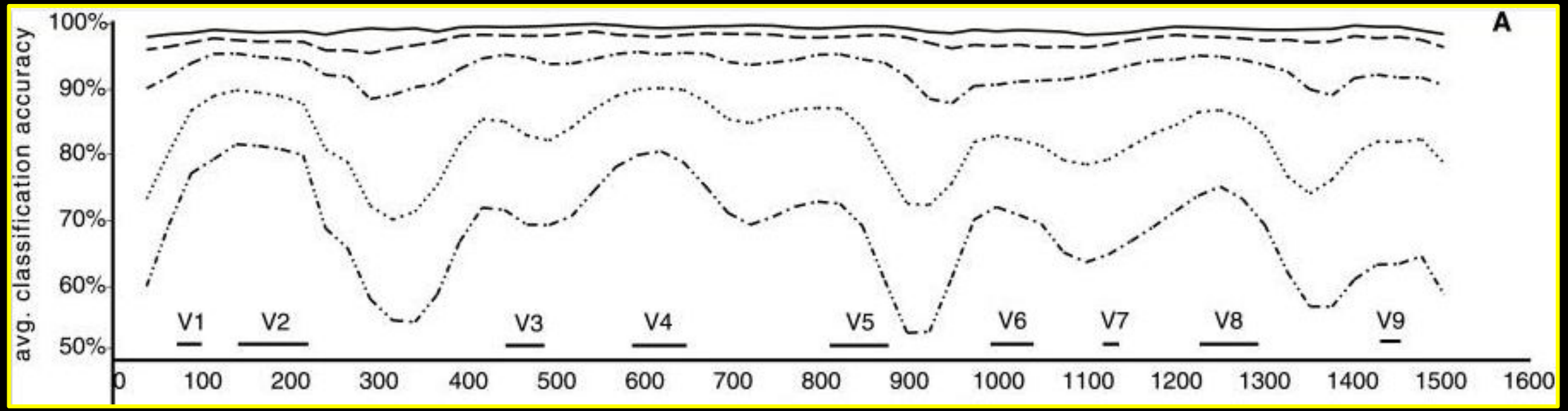
- Leave-one-out approach (train on all but one sequence, test on the last one)
- Classification to higher taxonomic ranks is more accurate (not surprising?)
- Shorter sequences = lower accuracy (definitely not surprising)



Wang et al. (2007) *Applied and Environmental Microbiology*

Performance by gene region

- Depends heavily on the region of the gene (different degrees of variation)
- V = variable regions, more sequence diversity



Wang et al. (2007) *Applied and Environmental Microbiology*

The point

- Assign samples to different classes using a probabilistic approach
- The k-mers in x_j are treated independently
  - A simplifying assumption that makes NB very fast and suited to large numbers of features
- Because we're using a k-mer decomposition, all positional information in the sequences is lost!

Further challenges in sequence classification

(1) The *k*-mer frequencies may not differ enough to support classification at the species / subspecies level

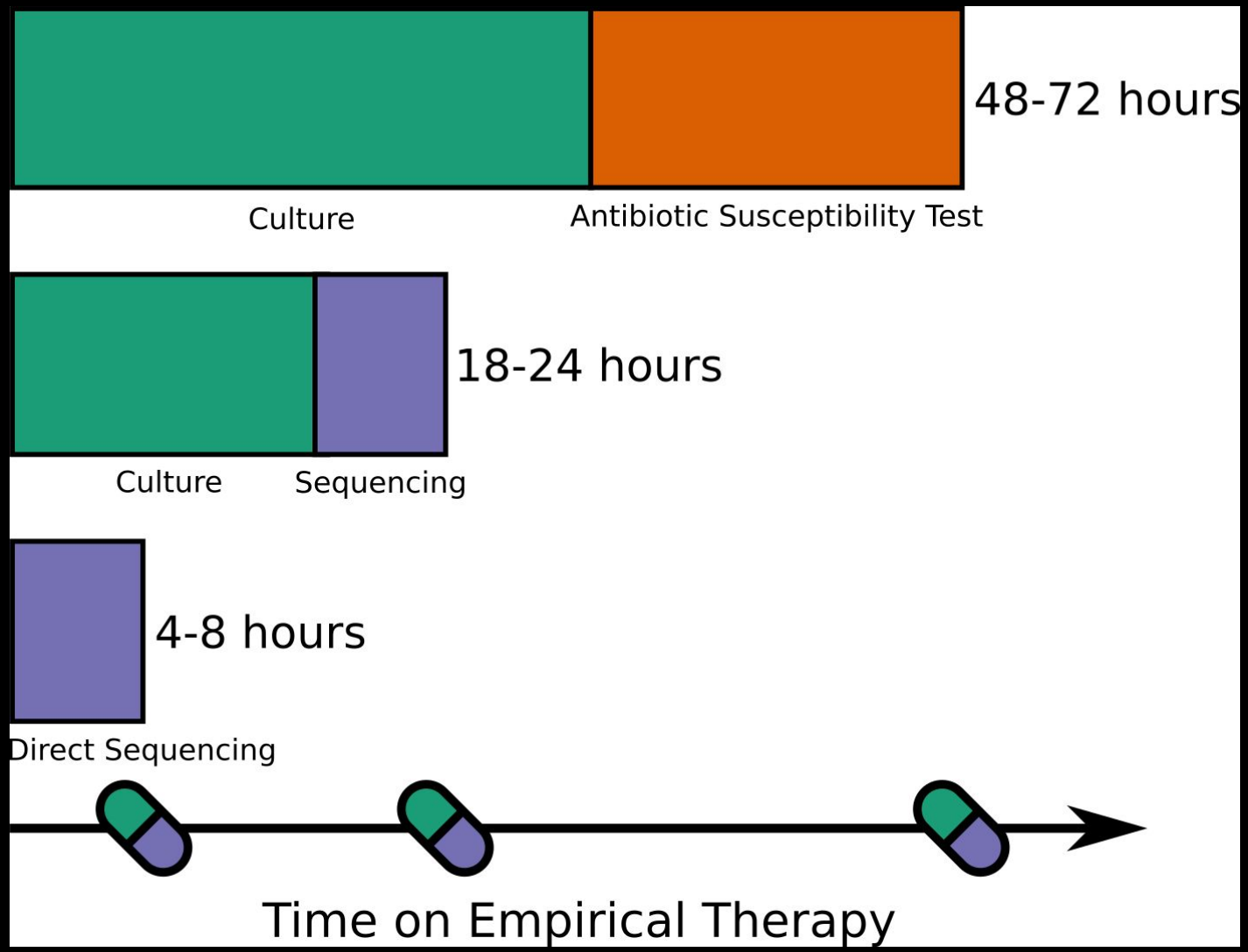
(2) We are restricted to the classes in our training set; new genomes are not modeled

- "rank-flexible" strategy can mitigate this to an extent

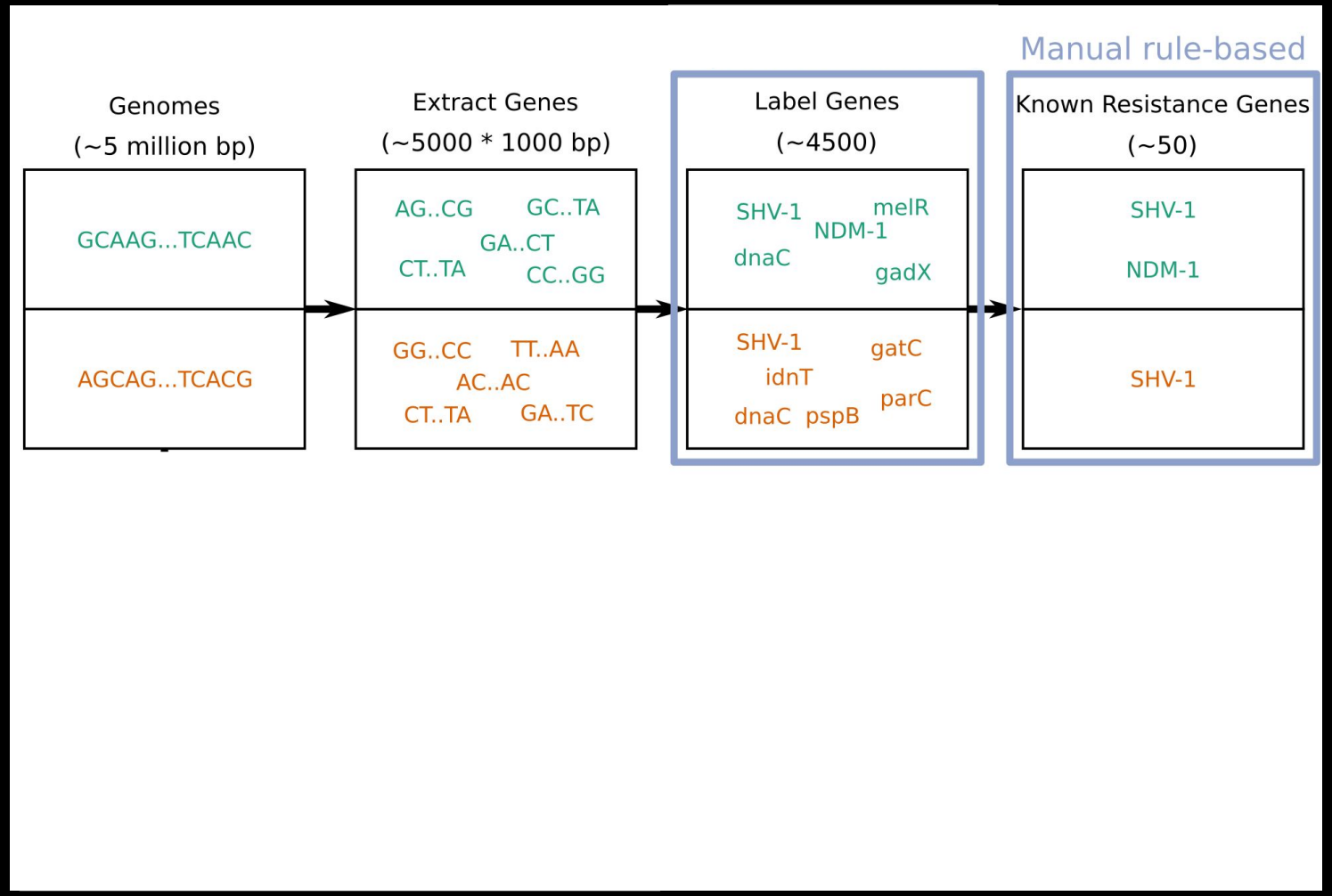
(3) Taxonomy is an awful mess

Learning from feature importances: Resistance Prediction

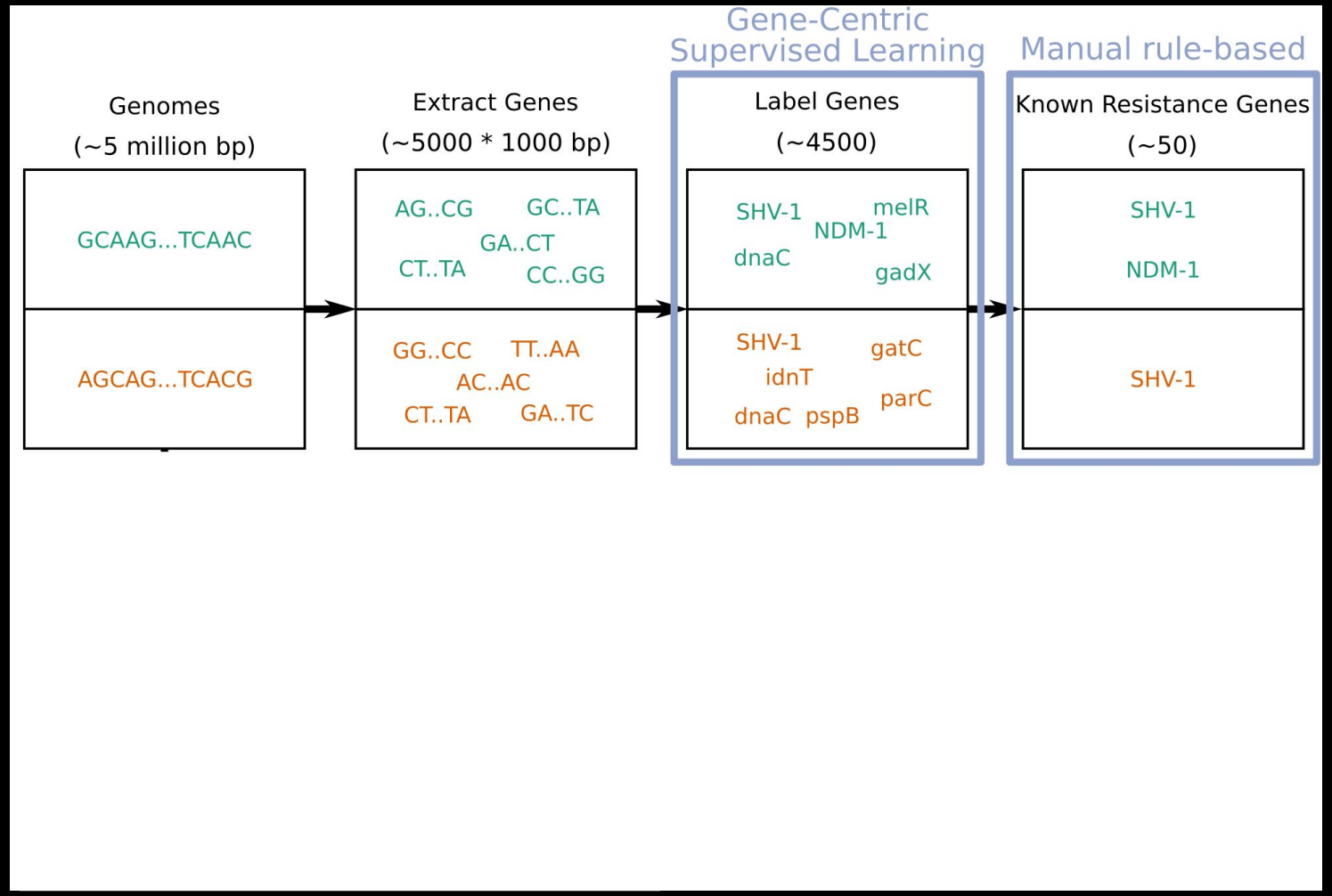
Predicting resistance from genomes



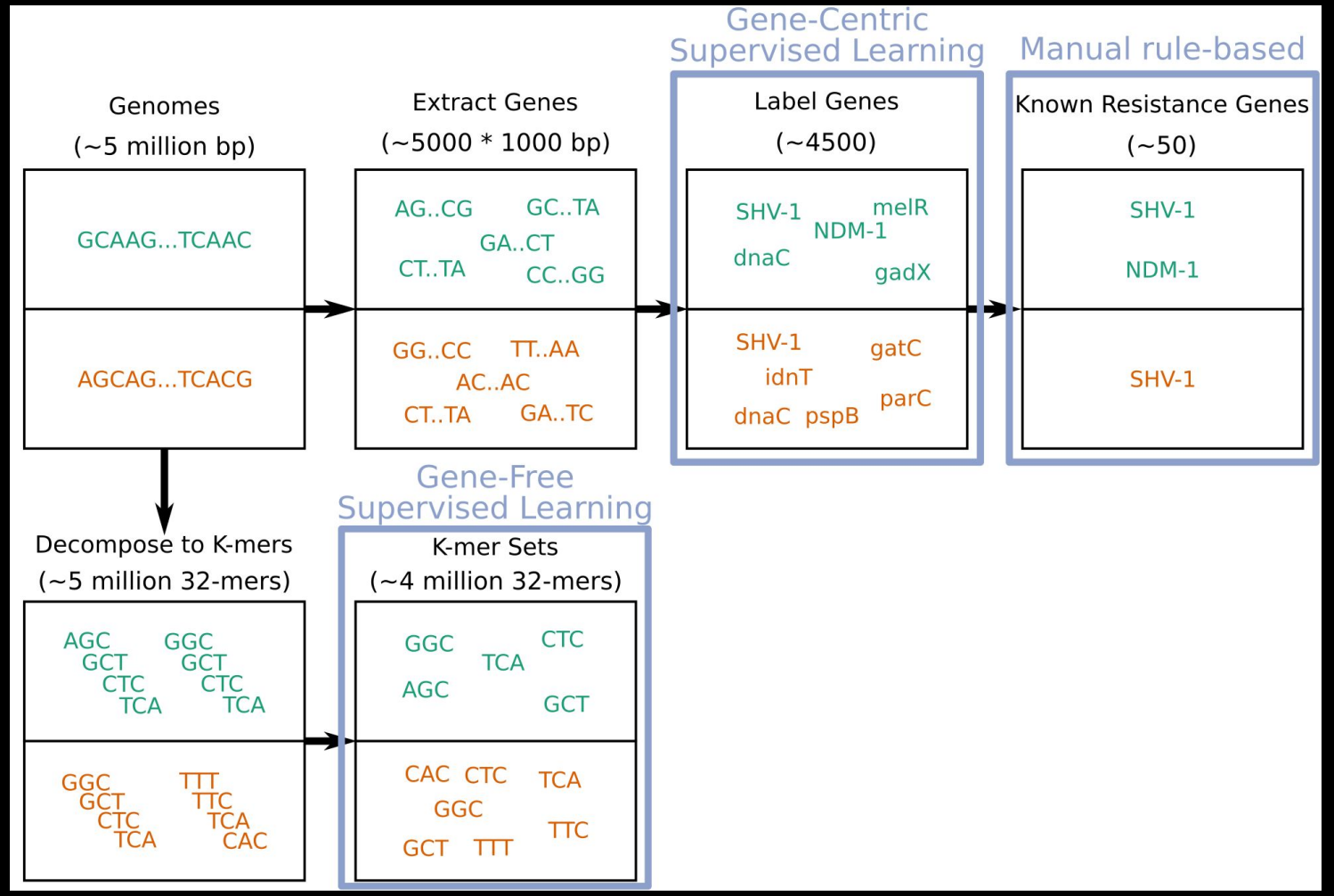
Several options for predicting resistance



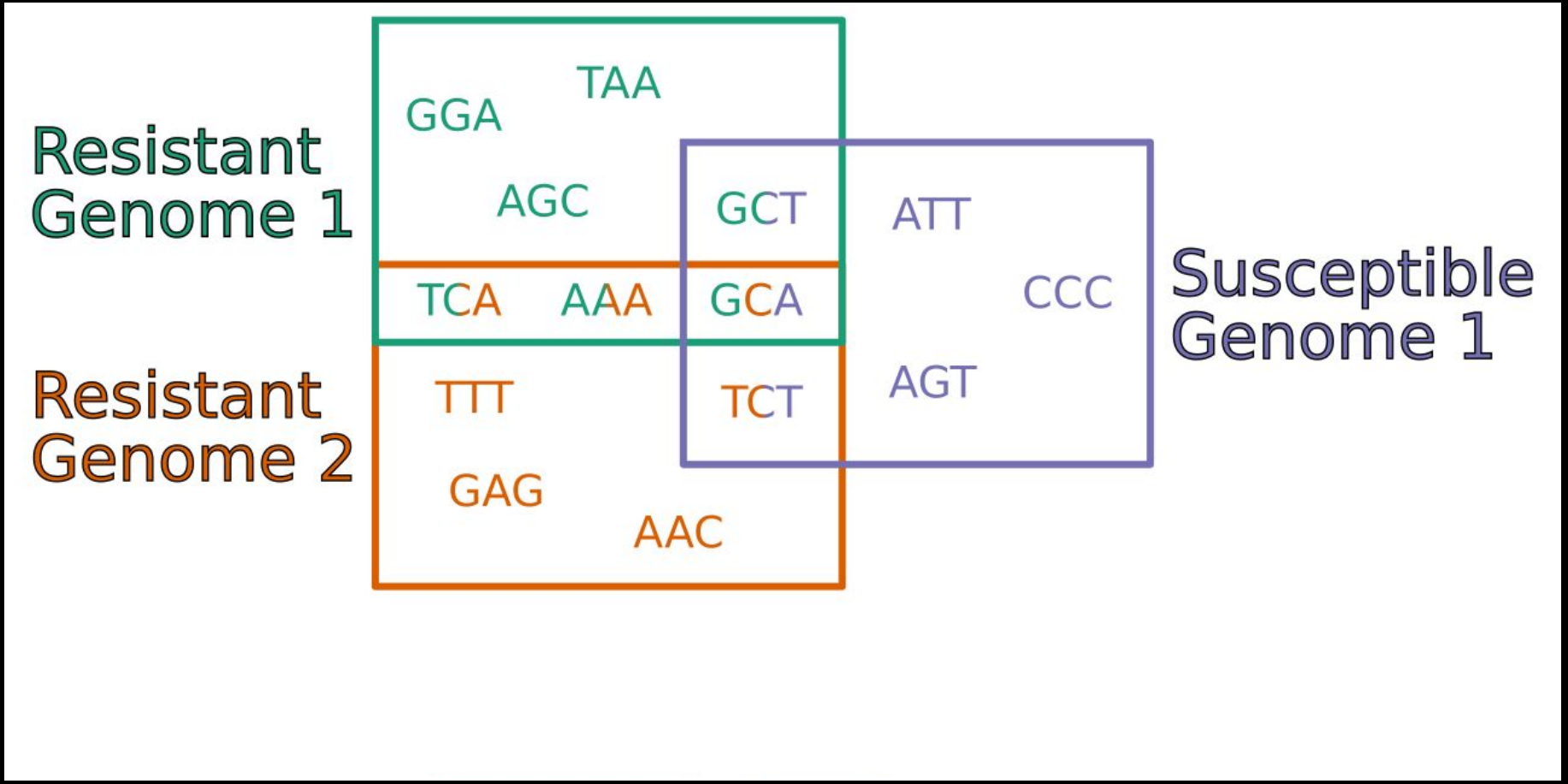
Several options for predicting resistance



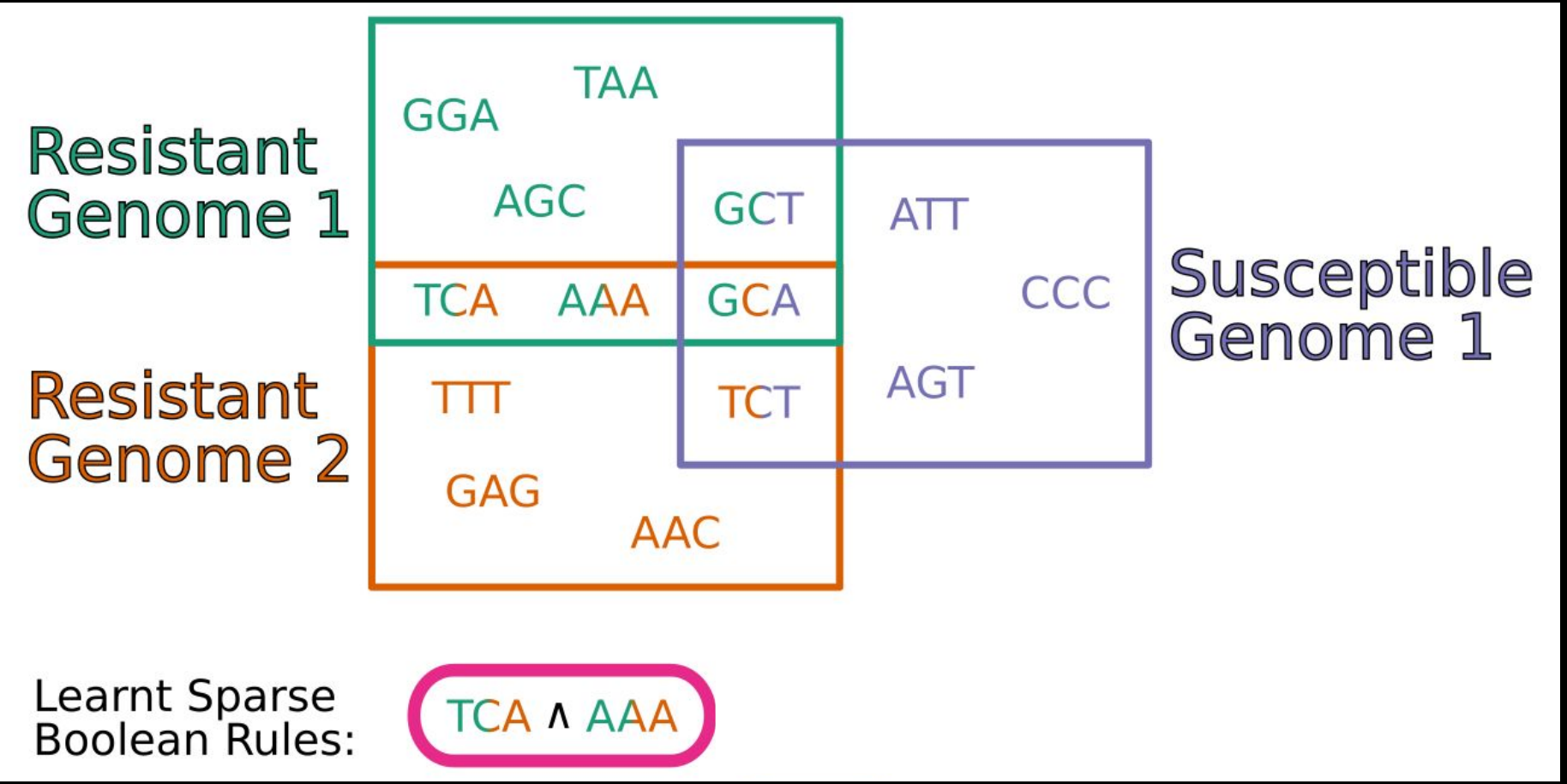
Several options for predicting resistance



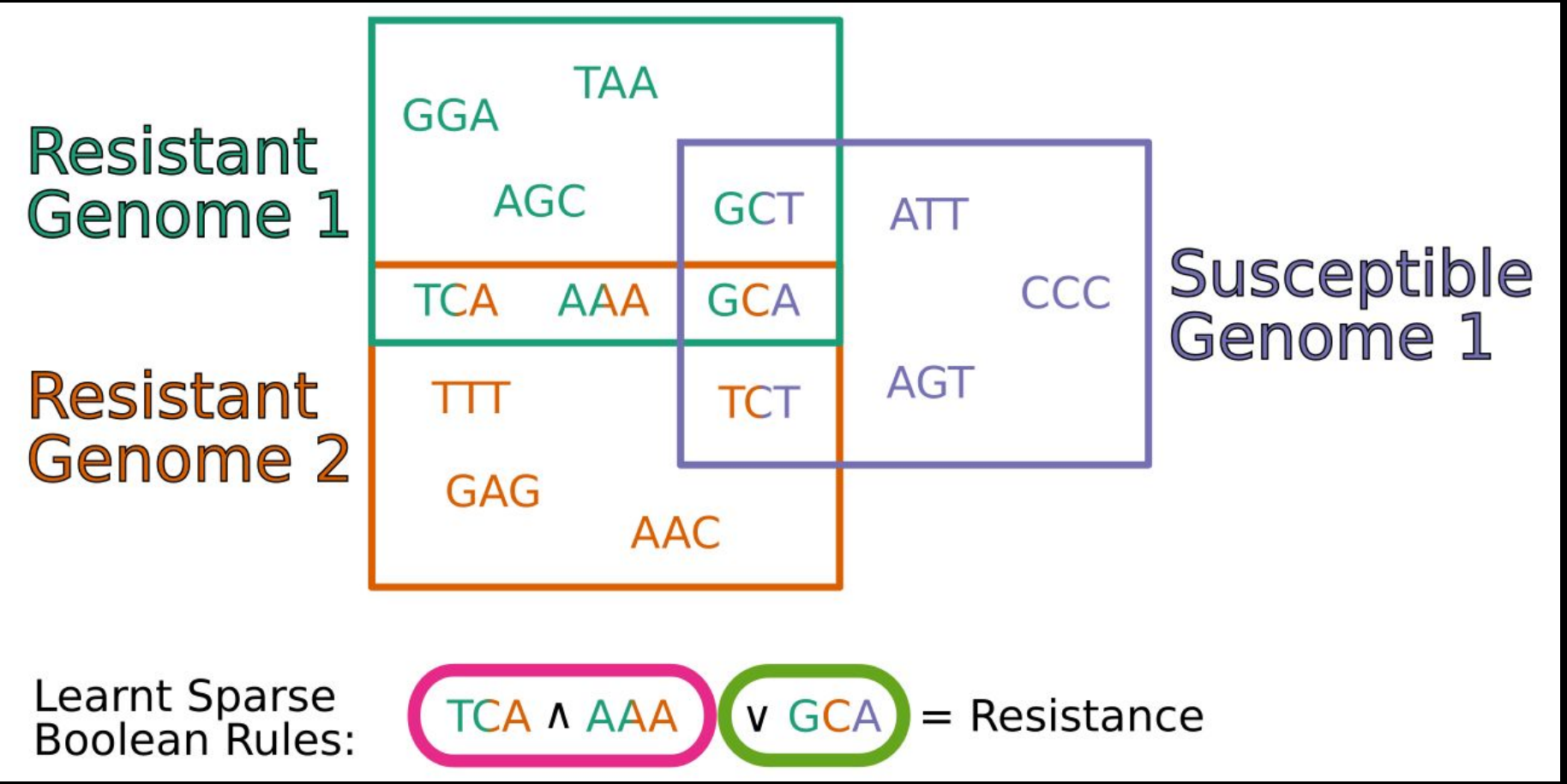
Set-Covering Machines for K-mers



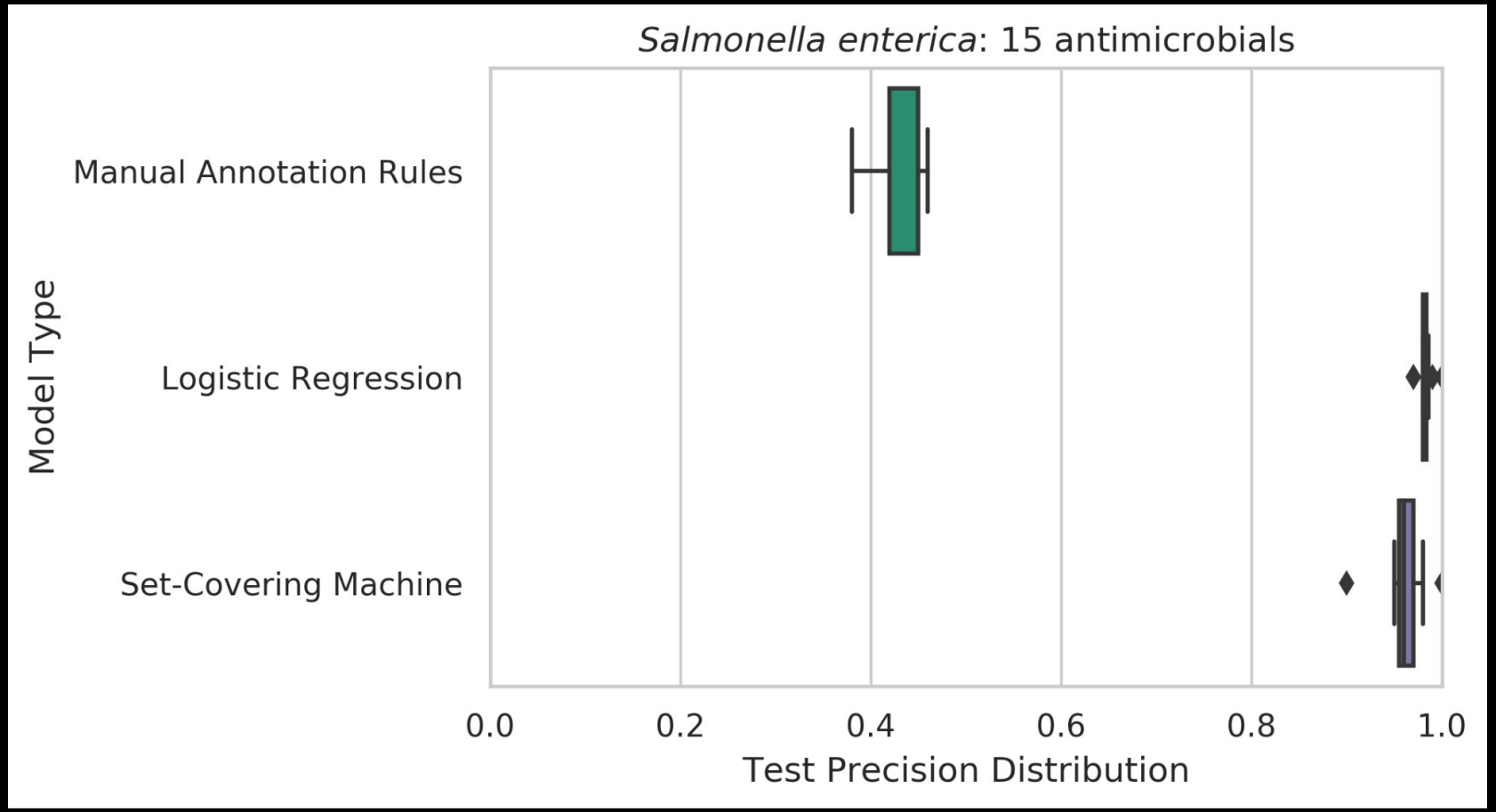
Set-Covering Machines for K-mers



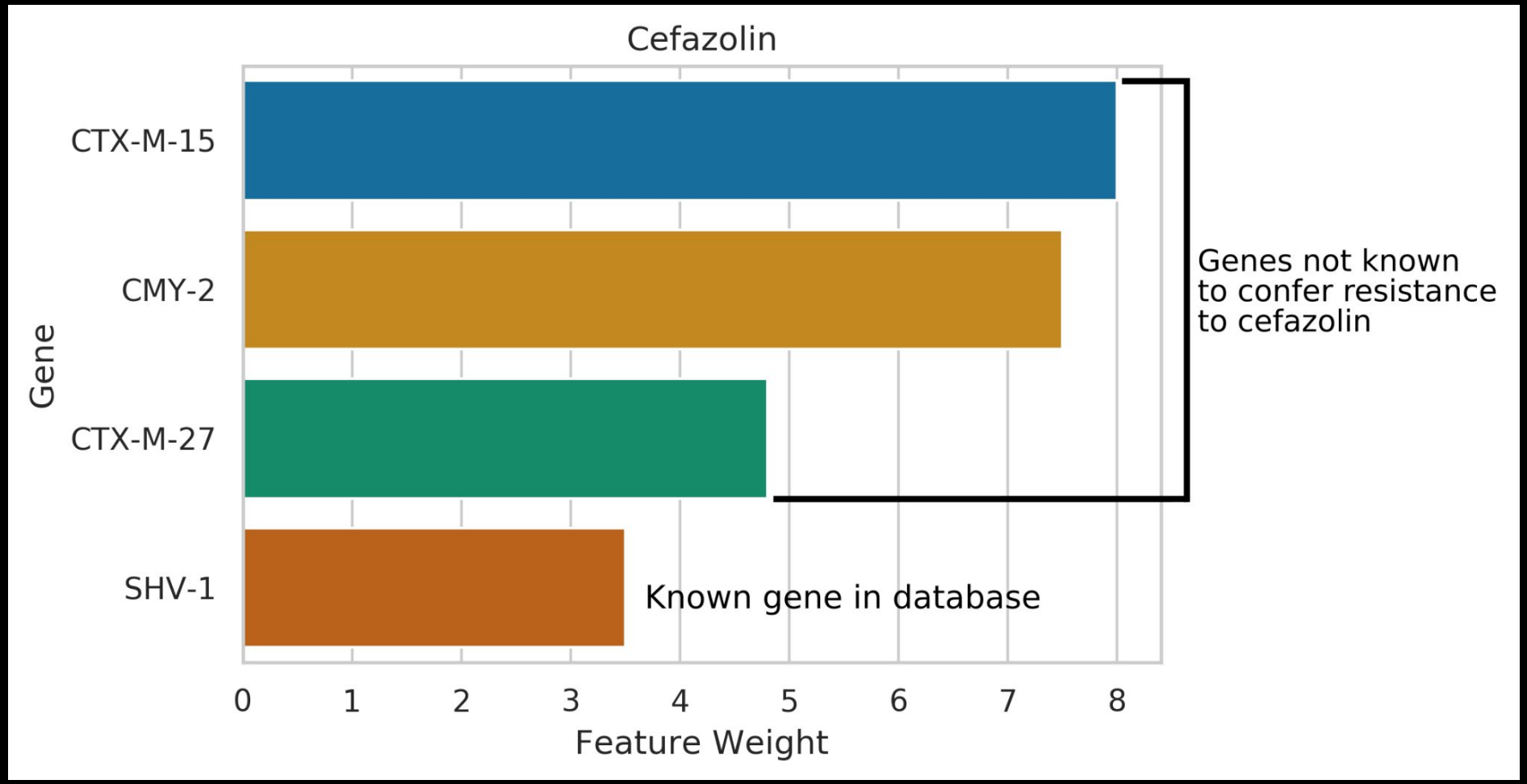
Set-Covering Machines for K-mers



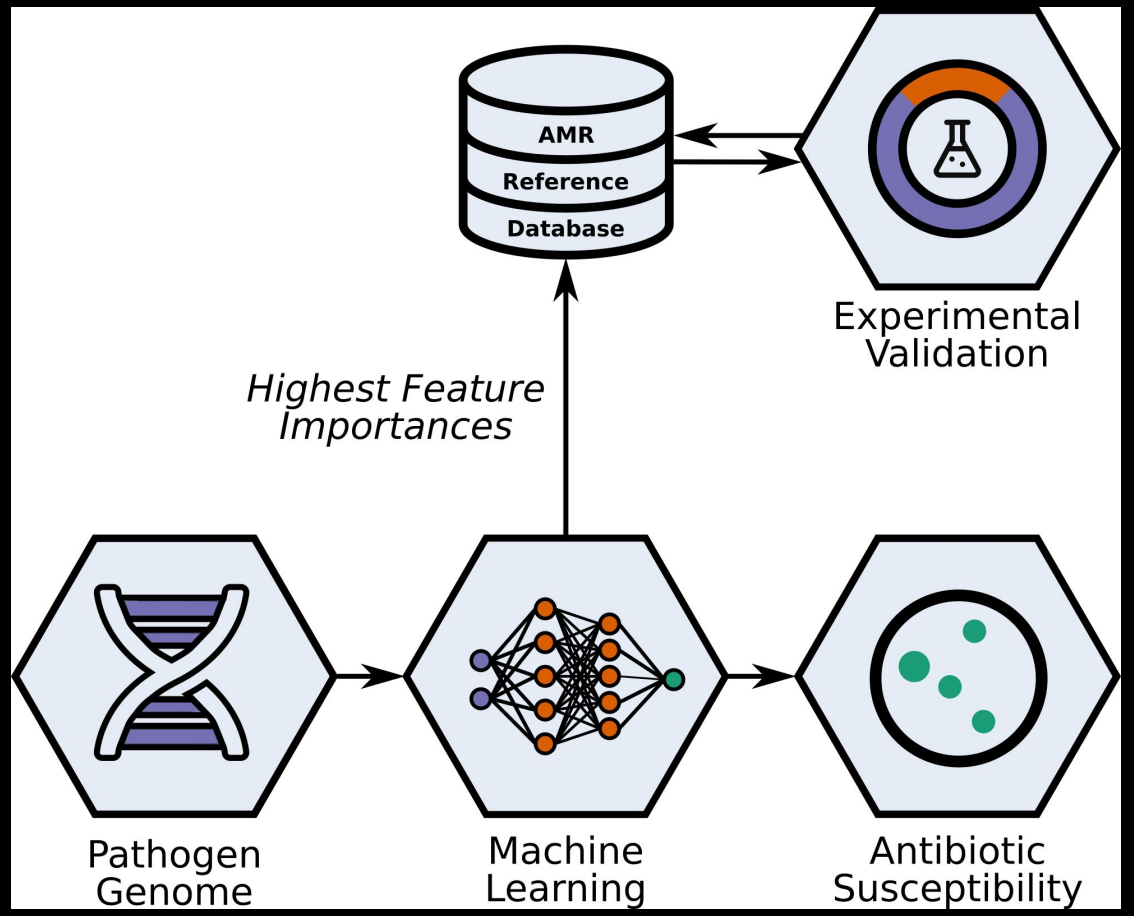
Reasonable precision for ML models



Inspect models for unexpected features



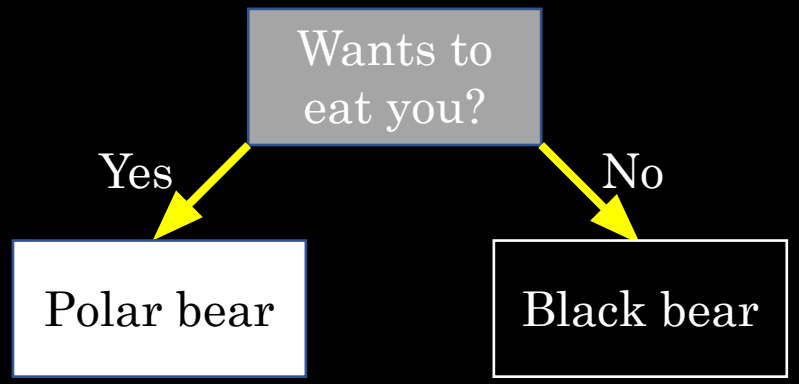
Test surprising features experimentally



Random Forest: Classifying salmon populations

Decision trees

- Classify two or more groups by creating *decision nodes* based on specific criteria

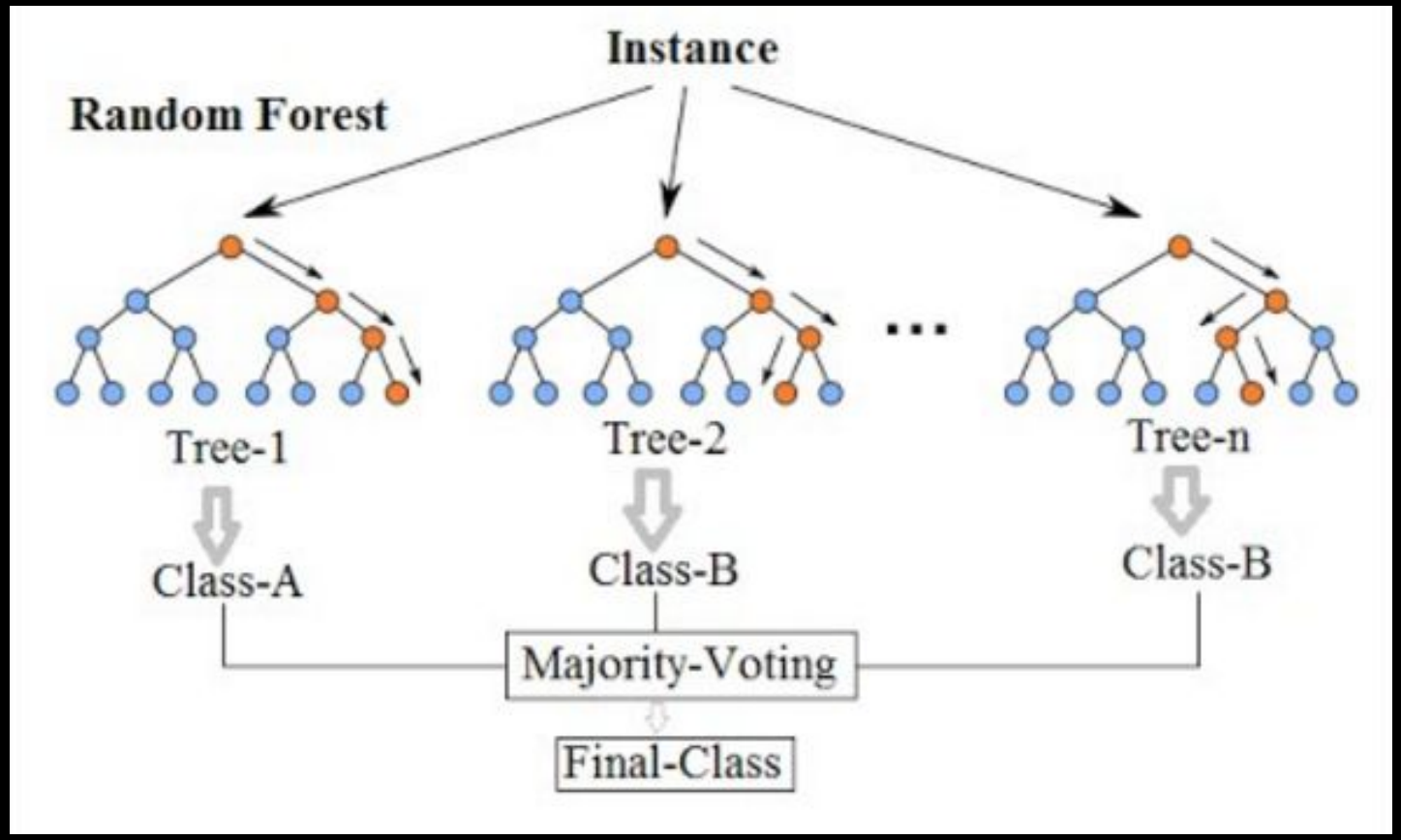


- Advantages: Simple and fast (greedy training)
- But overfitting is a serious problem (greedy training)

Random forests:

why use only one tree when you can use many?

- A series of weak classifiers can provide better generalization than DTs
- But how do we ensure we don't get the same tree *n* times?



By Venkata Jagannath - https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68995764

RFs part 1: trees, trees, trees!

- Each tree is trained on a randomly regenerated sample of the dataset, with replacement

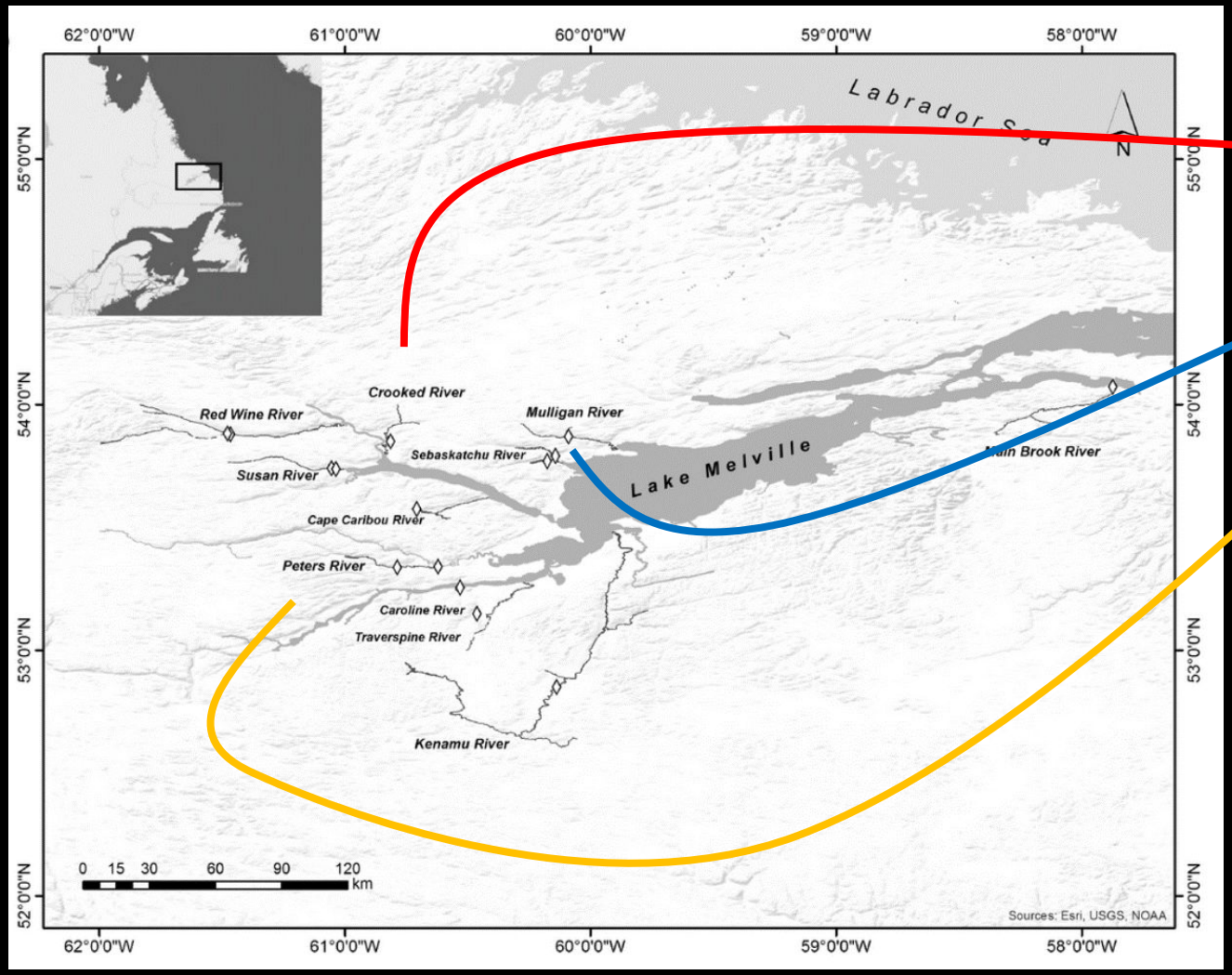
Original Dataset: $\{S_1, S_2, S_3, S_4, \ldots, S_n\}$
Bootstrapped dataset 1: $\{S_1, S_2, S_3, S_2, \ldots, S_n\}$
Bootstrapped dataset 2: $\{S_3, S_1, S_3, S_4, \ldots, S_n\}$
Bootstrapped dataset 3: $\{S_4, S_4, S_1, S_1, \ldots, S_n\}$
Each time we overrepresent some cases, and eliminate others = bootstrap aggregation (bagging)

RFs part 2: subsetting features

- If you use the same features every time, you'll get the same tree many times. Random subset selection addresses this.

- Each decision node in each tree can consider only a randomly selected subset of features. Individual trees will make poorer predictions overall, but in aggregate their generalization performance will be higher

Salmon populations in Labrador



Salmon breed here, here, here, etc.

Atlantic salmon are anadromous – they return to the same river where they hatched

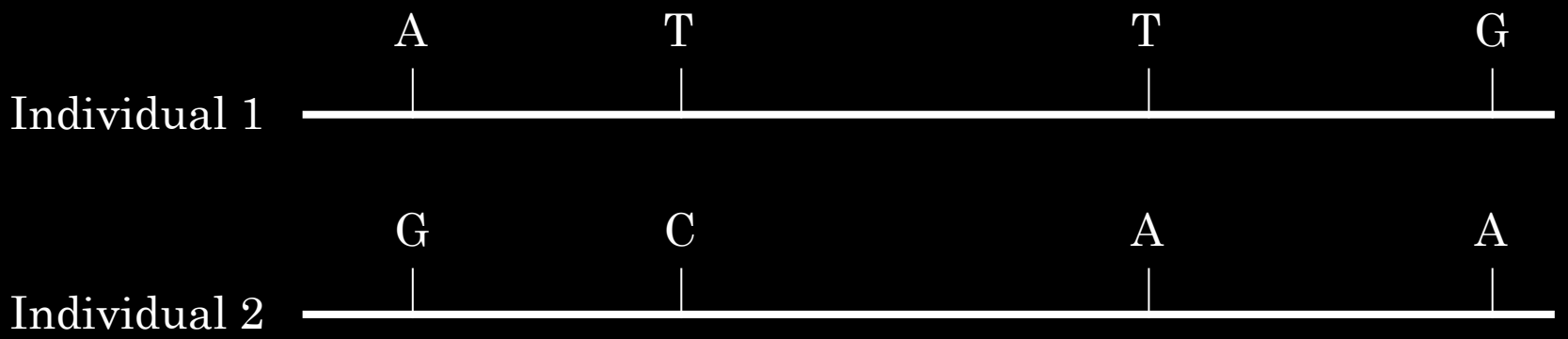
But are they the same population?

Sylvester et al. (Feb 2018) *Evol Applications*

One fish, two fish, Kenamu fish

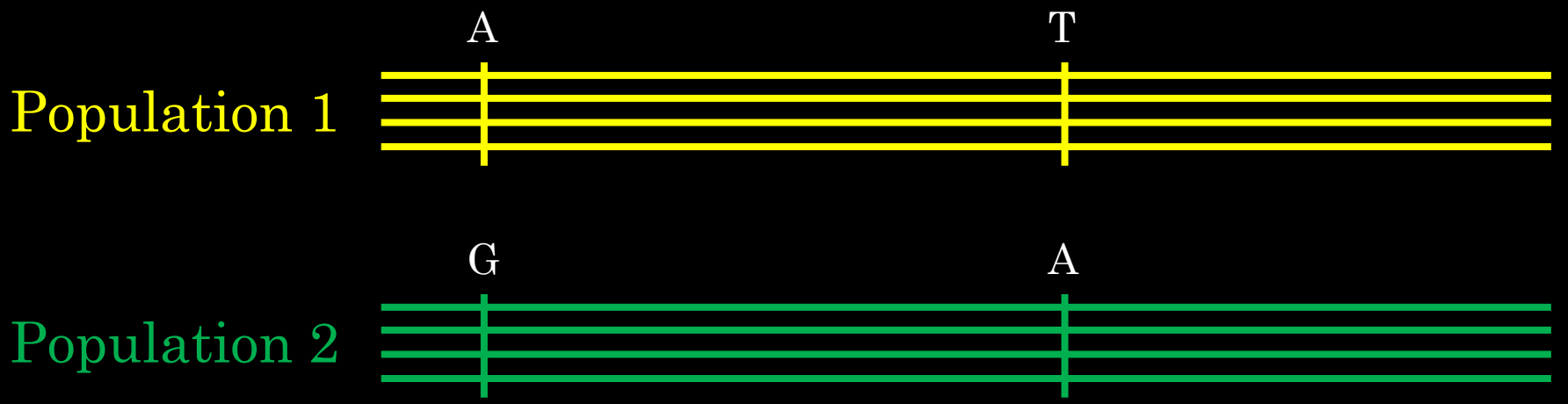
Every fish has its own distinctive genotype, with mutations that distinguish it from other members of the species

Single Nucleotide Polymorphisms (SNPs)



Can we distinguish populations?

Sets of features that distinguish individuals by their river of origin suggest that they constitute separate populations and should be treated as such



Of course it isn't that simple, which is why we need RFs!
Starting point: 220,000 salmon SNPs
Real starting point: 8000 salmon SNPs

Step 1: feature selection!

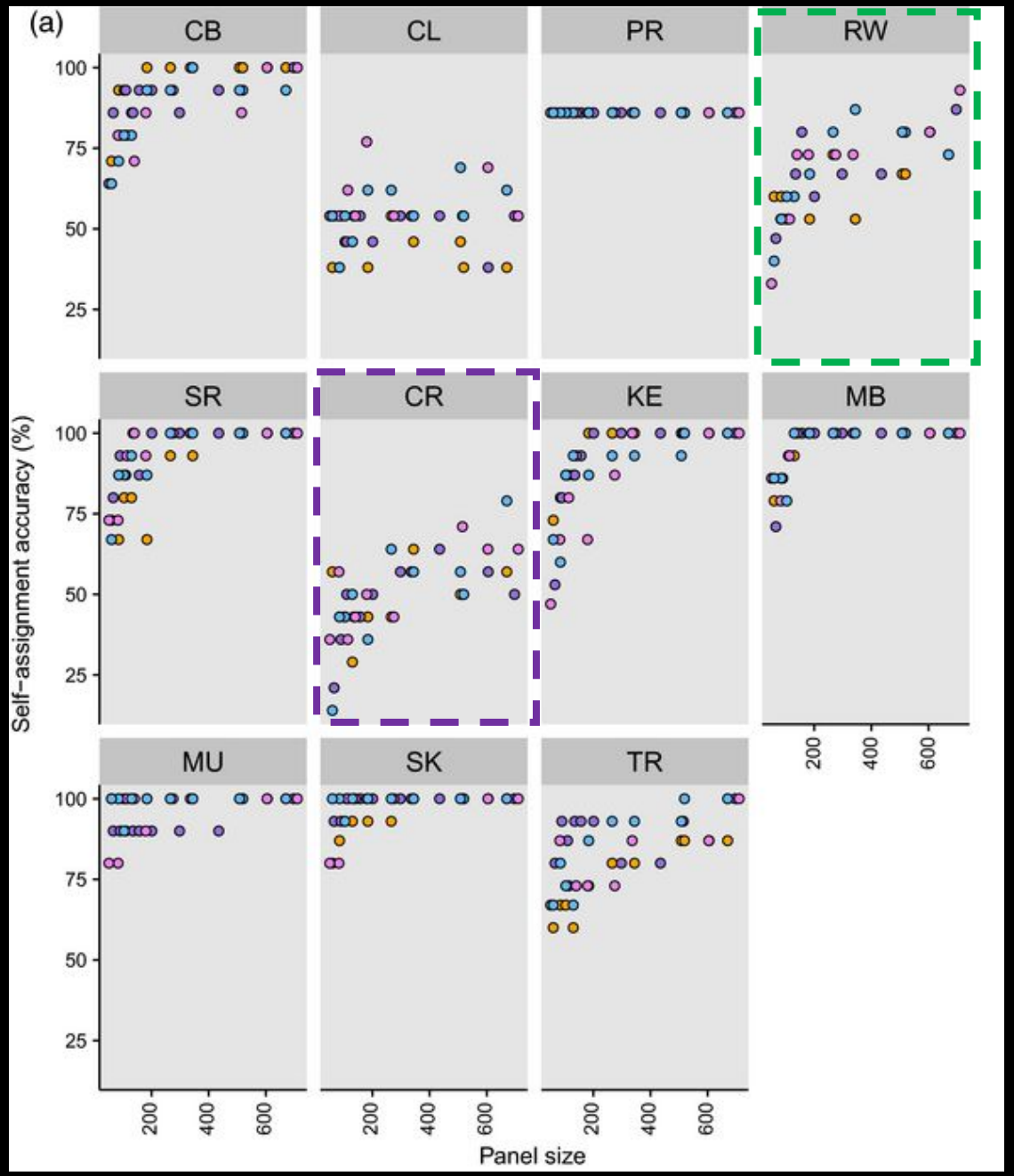
Mean decrease in accuracy (MDA): train RFs without a given SNP, and assess the impact on classification accuracy

Which features consistently gave substantial MDA across many trials?

Interestingly, relatively few features came up again and again (due to redundancy among features?)

Use this set of features for subsequent classification

RF, Regularized RF, and Guided Regularized RF were used for feature selection, as well as a basic population statistic (F_{ST})



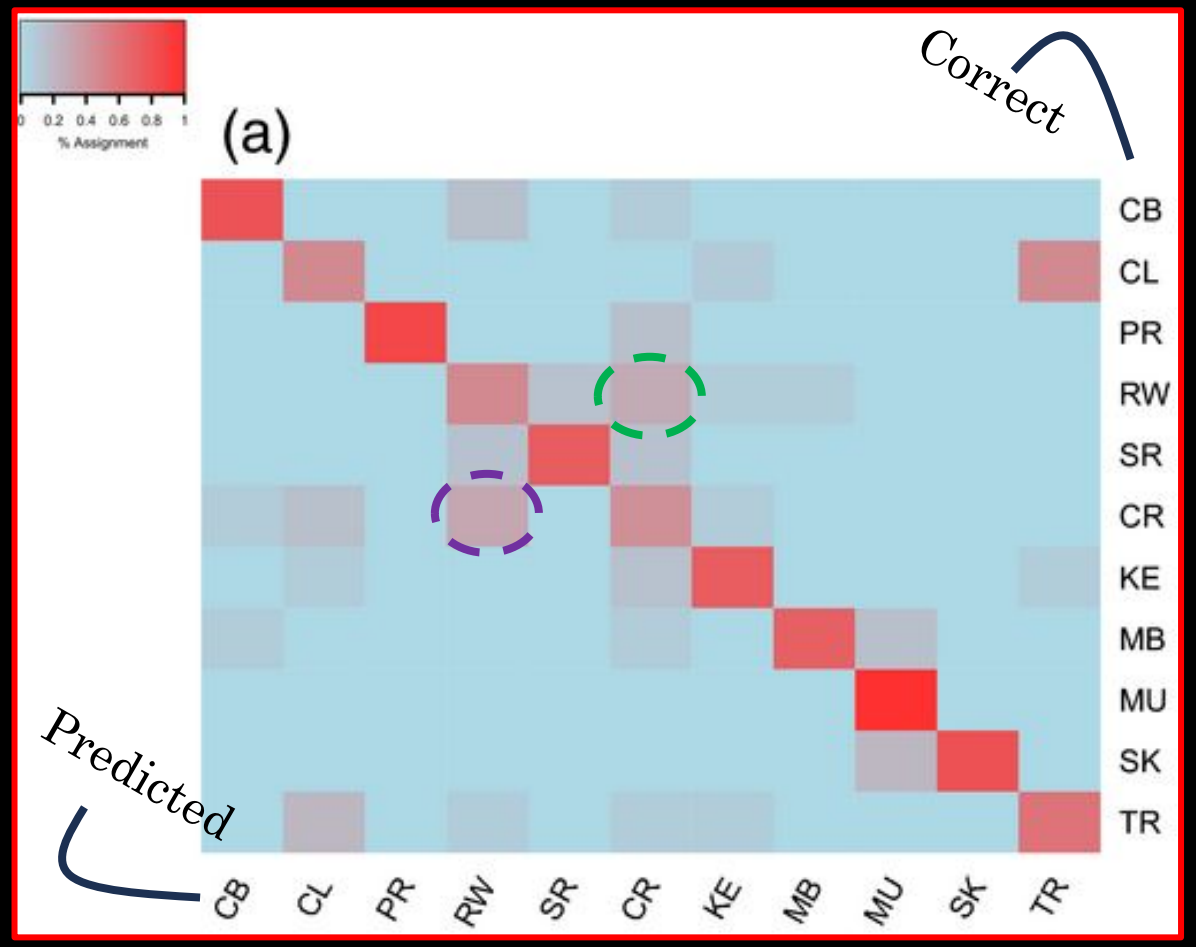
Step 2: actually classifying!

River by river, four types of classification algorithm (different coloured dots)

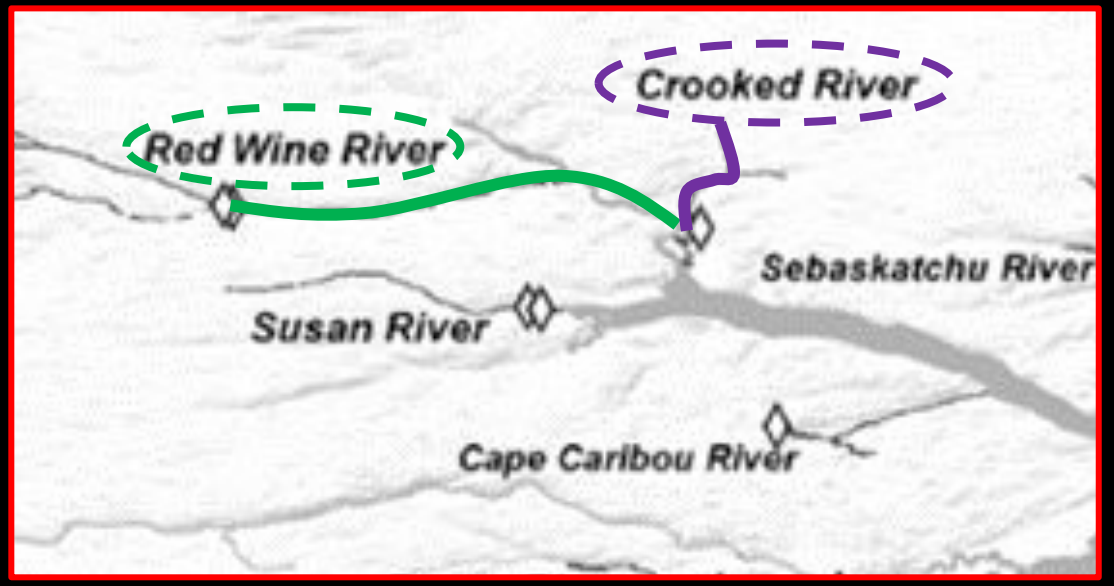
Note:

- Many rivers have near-perfect accuracy
- Some rivers (RW = Red Wine, CR = Crooked) are not so great

Misclassifications are not random



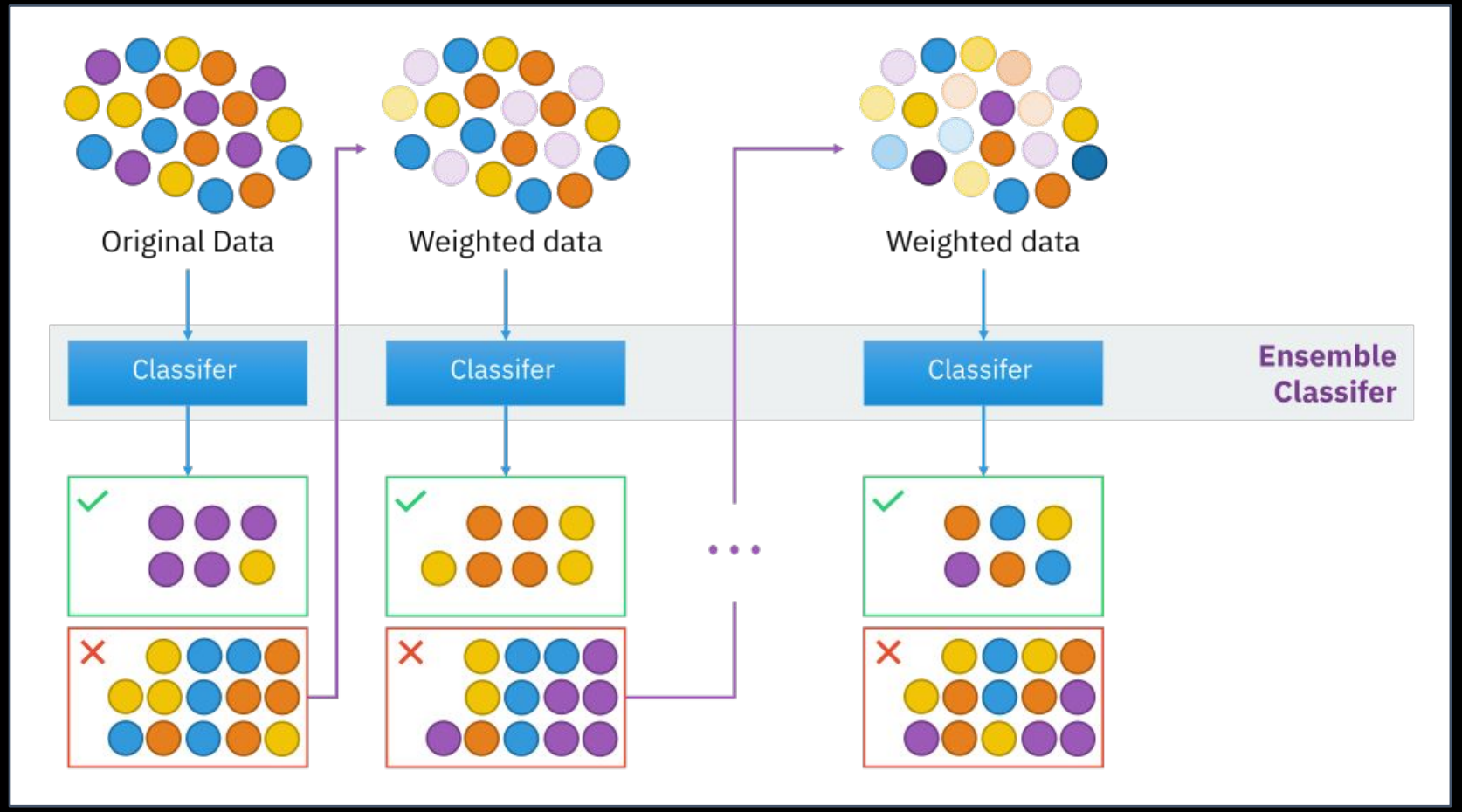
CR and RW are frequently misclassified as one another



Conclusions

- Most rivers were distinguishable, even with small-ish training sets
- Suggests that the populations are distinct and should be treated as such for management purposes
- Identified SNPs were used to design an array that can cheaply differentiate individual salmon caught in Lake Melville

Boosting: train model on what the last model got wrong



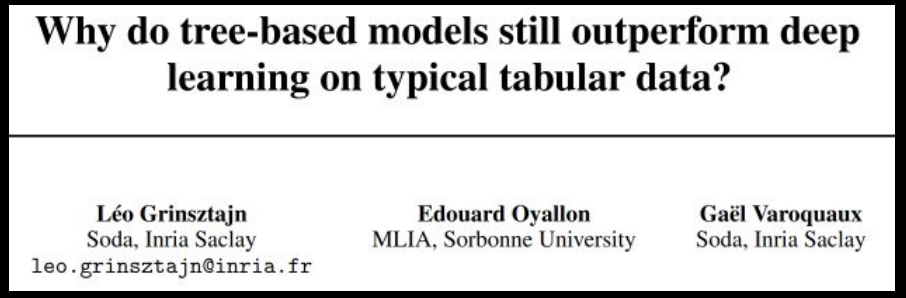
https://commons.wikimedia.org/wiki/File:Ensemble_Boosting.svg

Decision Trees methods regularly outperform deep learning on tabular data

Tree-based methods deal well with common features of tabular data (even compared to well-tuned neural networks):

- Heterogeneous data
- Ignoring uninformative data
- Non-smooth decision boundaries
- Moderate size & dimensionality
- Skewed or heavy-tailed feature distributions and other forms of dataset
- Rotational invariance (column/row order is not informative)

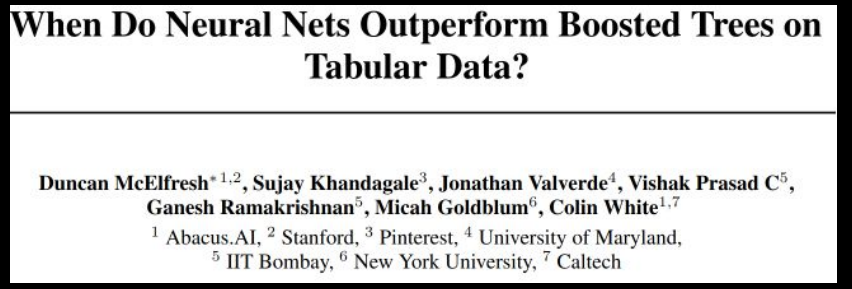
But: difference is often negligible (except in computational efficiency!)


Why do tree-based models still outperform deep learning on typical tabular data?

Léo Grinsztajn
Soda, Inria Saclay
leo.grinsztajn@inria.fr

Edouard Oyallon
MLIA, Sorbonne University

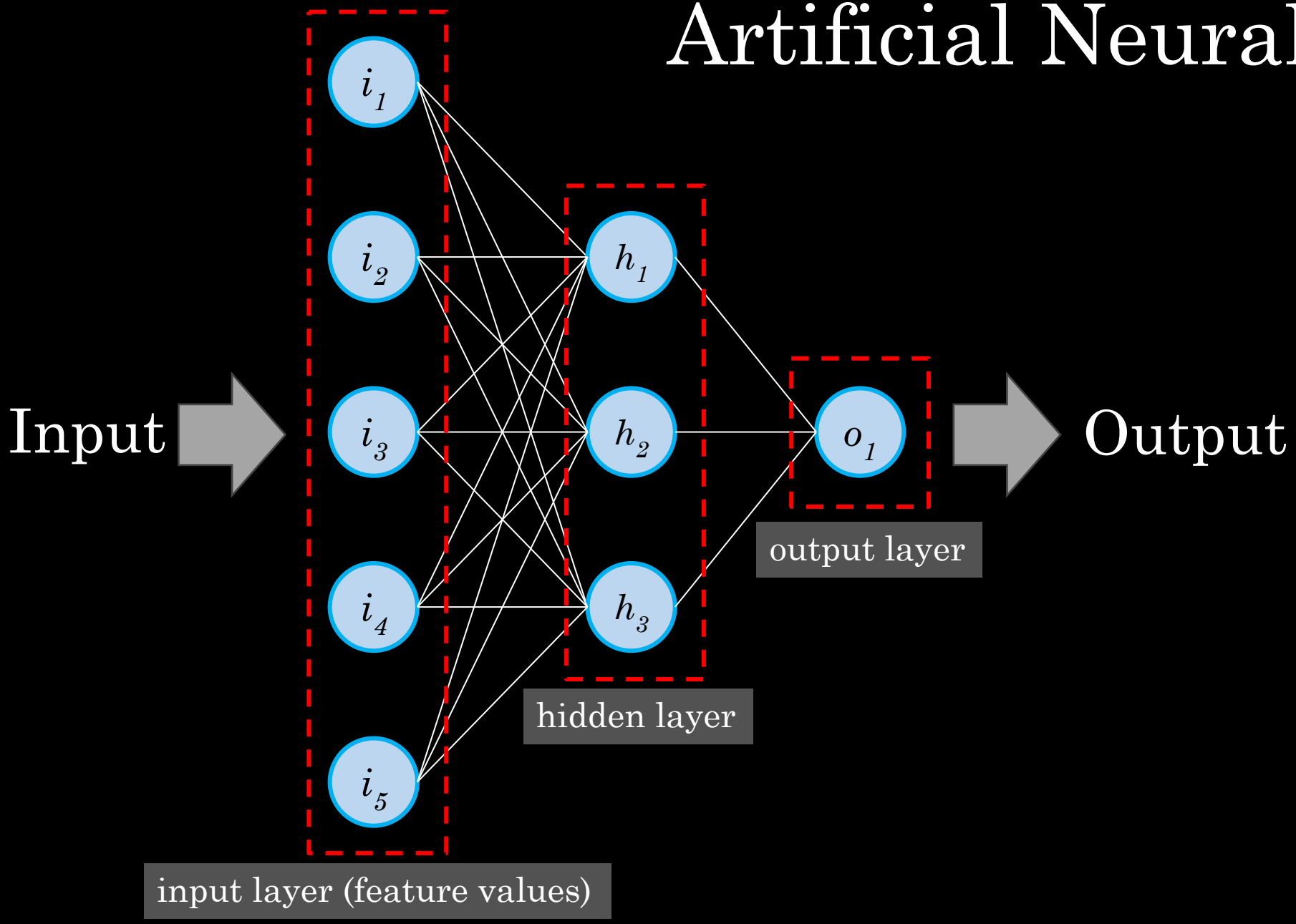
Gaël Varoquaux
Soda, Inria Saclay


When Do Neural Nets Outperform Boosted Trees on Tabular Data?

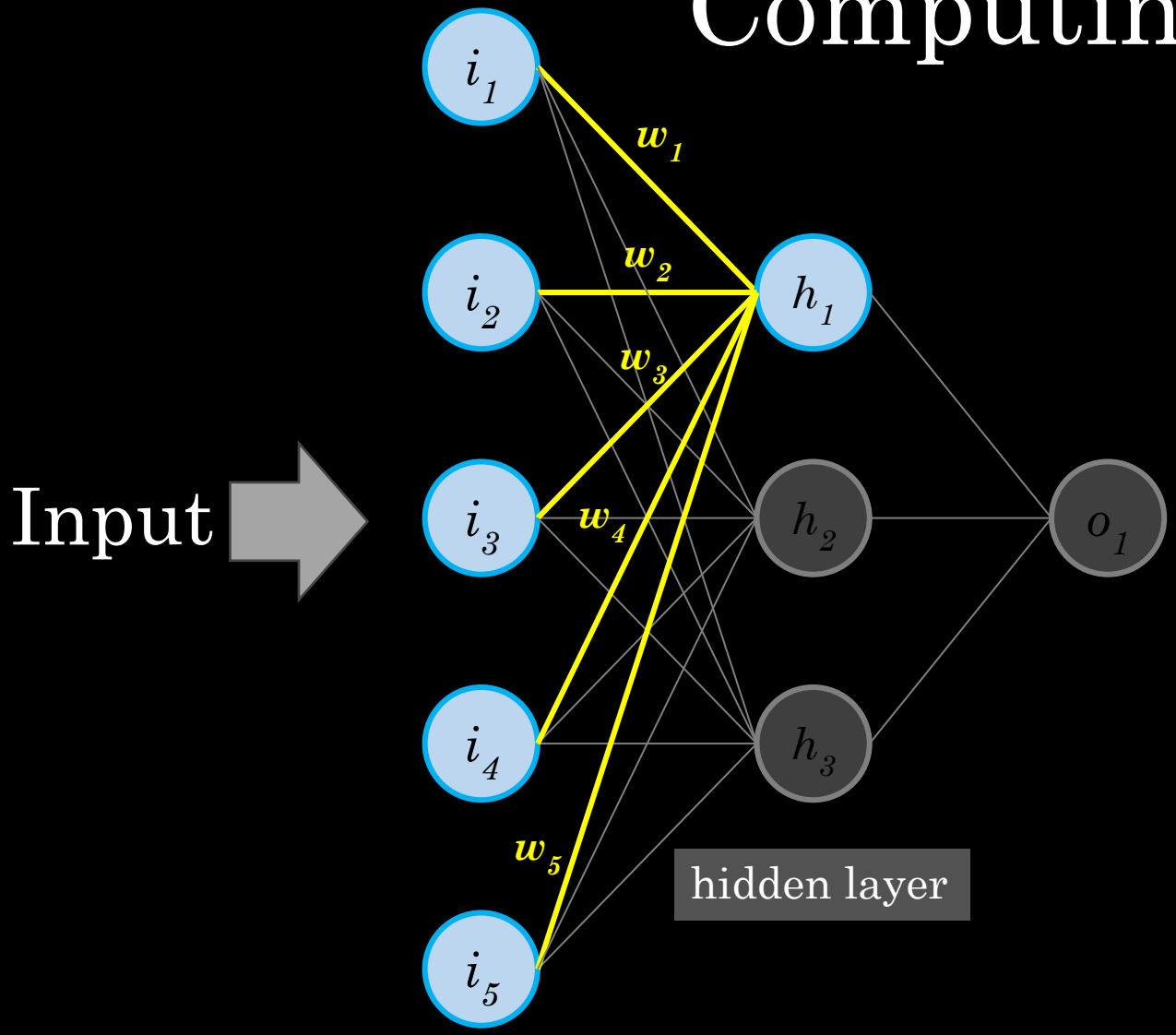
Duncan McElfresh[*1,2], **Sujay Khandagale**[3], **Jonathan Valverde**[4], **Vishak Prasad C**[5], **Ganesh Ramakrishnan**[5], **Micah Goldblum**[6], **Colin White**[1,7]
[1] Abacus.AI, [2] Stanford, [3] Pinterest, [4] University of Maryland, [5] IIT Bombay, [6] New York University, [7] Caltech

Transformer models: DNABERT-2

Artificial Neural Networks



Computing the value of a node

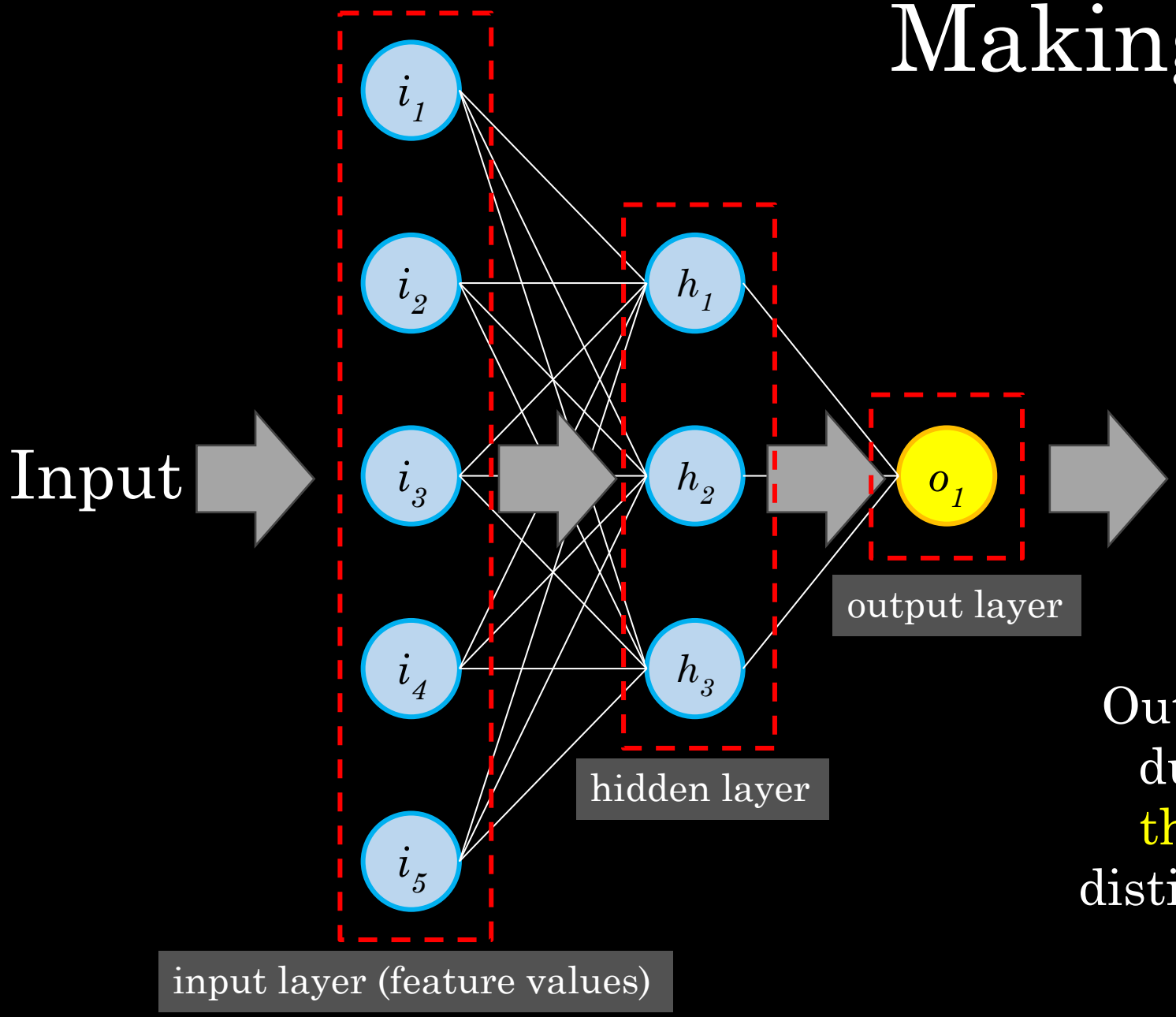


$$h_1 = \sigma(w_1 i_1 + w_2 i_2 + w_3 i_3 + w_4 i_4 + w_5 i_5)$$

$$= \sigma\left(\sum_{j=1}^{5} w_j i_j\right)$$

sigma is a squashing function that forces the summation to fit in a given range (e.g., 0 to 1)

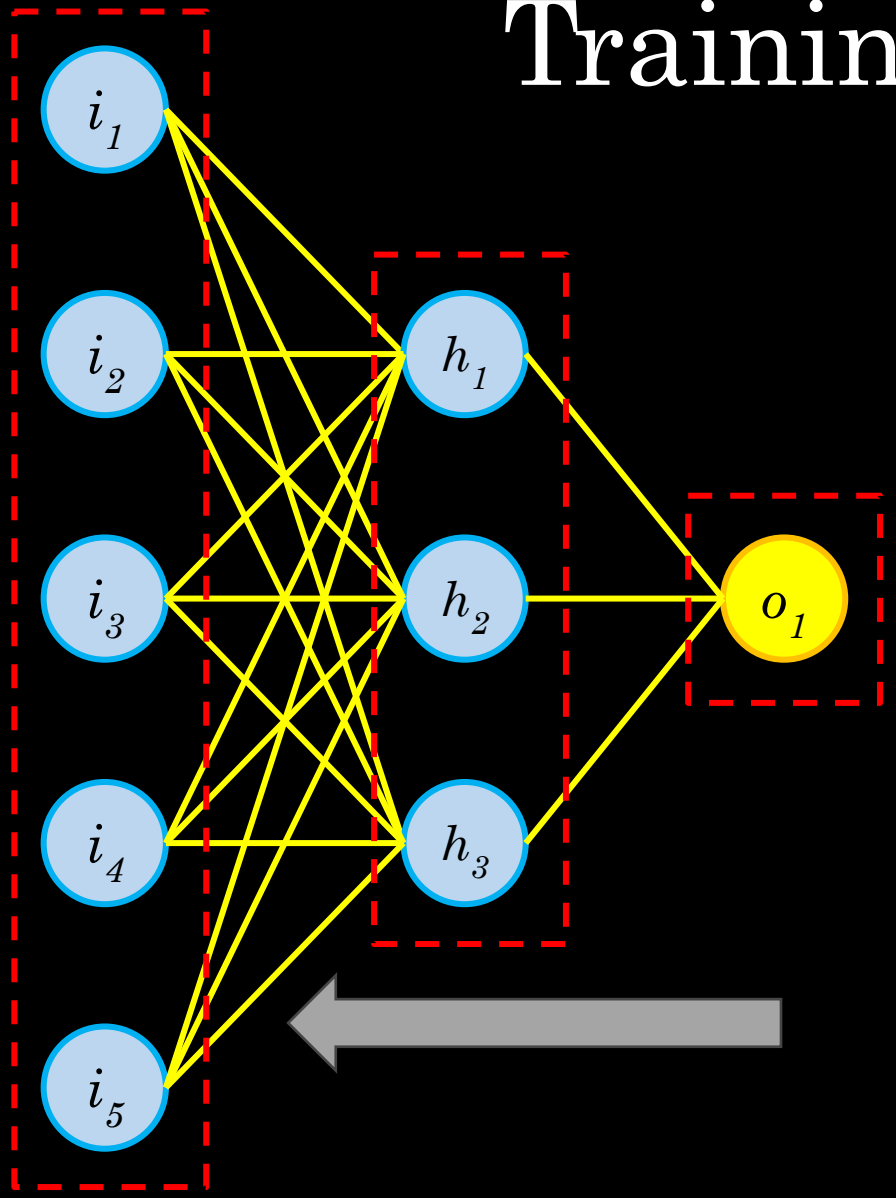
Making Predictions



≥ 0.5: Class 1
< 0.5: Class 2

Output falls within a given range due to the squashing function, thresholding can be applied to distinguish positive / negative class

Training a Neural Network



Predicted: 0.7
Actual: 1.0 (class 1)

Use the error (difference between predicted and true values) to backpropagate changes to connection weights based on prediction errors

Rumelhart et al. (1986) *Nature*

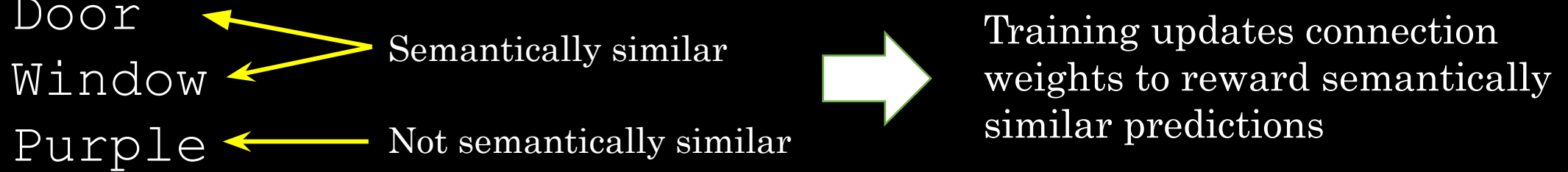
The structure of artificial neural networks

- Parameters: the connection weights that are optimized during training. Typically initialized randomly since we have no sweet clue what they should look like
- Regularization: e.g., weight decay which applies a penalty to the total magnitude of all connection weights. Small connection weights = small influence of input parameters = simpler model
- The size, layer arrangements, and other attributes of the neural network are hyperparameters that can be compared across runs

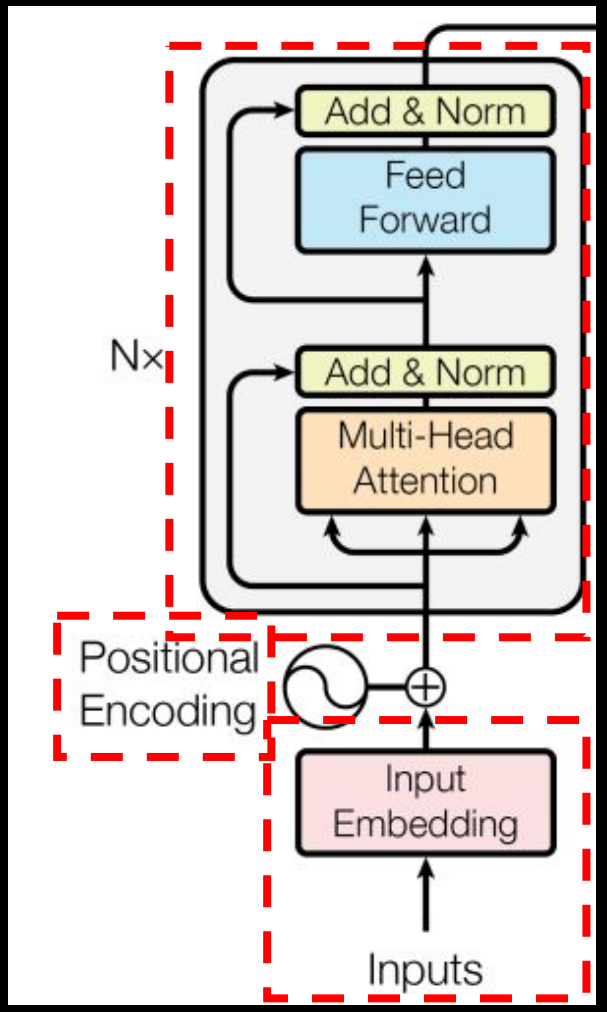
Transformer models

- Huge neural networks with many (often > 1,000,000,000) parameters
- Self-supervised learning: no human-provided labels, but instances from the data are used as prediction tasks

`A rather large deer entered the coffee shop through the ____.`



Transformer models



Provide positional information about tokens

(2) Feature encoder

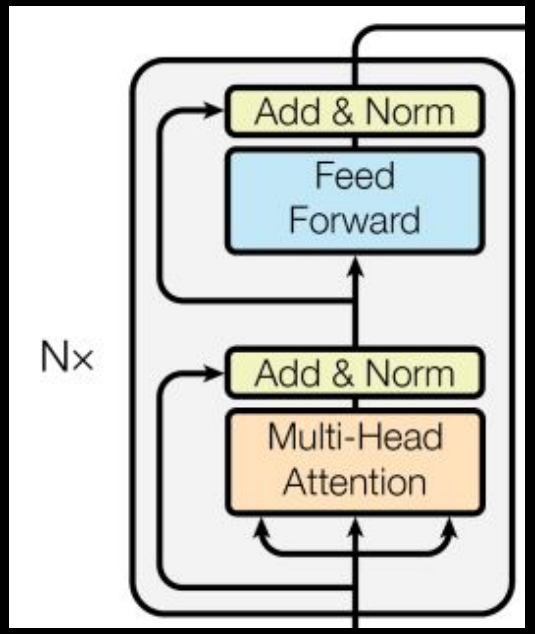
(1) Engineered features (tokens)

Vaswani et al. (2017) *31st Conference on Neural Information Processing Systems*

(1) Example tokens: *k*-mers

- *k*-mers are naïve and reflect nothing specific about the dataset.
- We don't know natural start / stop points of patterns of interest, so we use overlapping *k*-mers

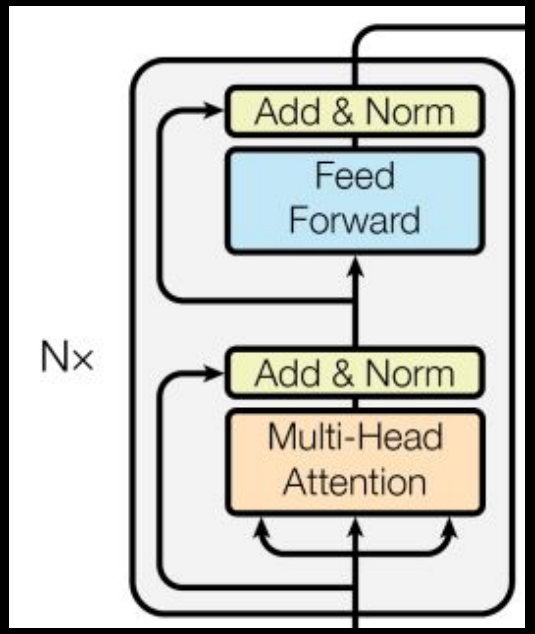
(2) Encoders



ATTENTION: Each token is examined in context with the other tokens.

I am AGGCTA at position x, which nearby tokens are of greatest importance?

(2) Encoders



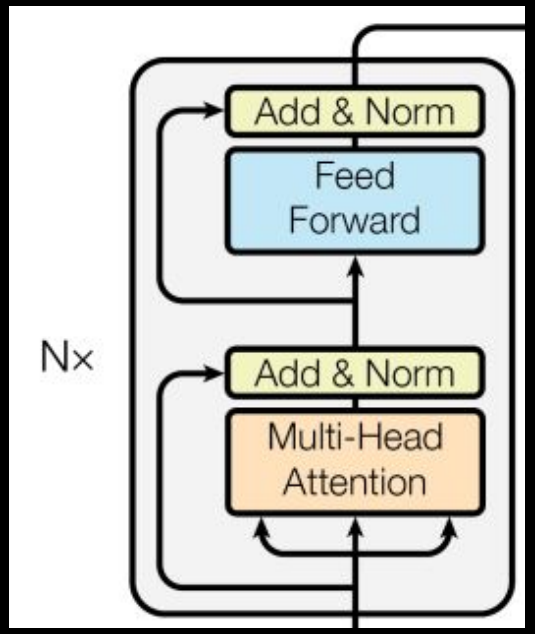
Nx: We can stack these encoder units on top of each other

This allows the transformer to learn higher-order associations in the data.

- Unit 1: important combinations of tokens in close proximity to one another
- Unit 2: combinations of combinations
- Unit 3: etc.

This is where the immense number of parameters comes from!

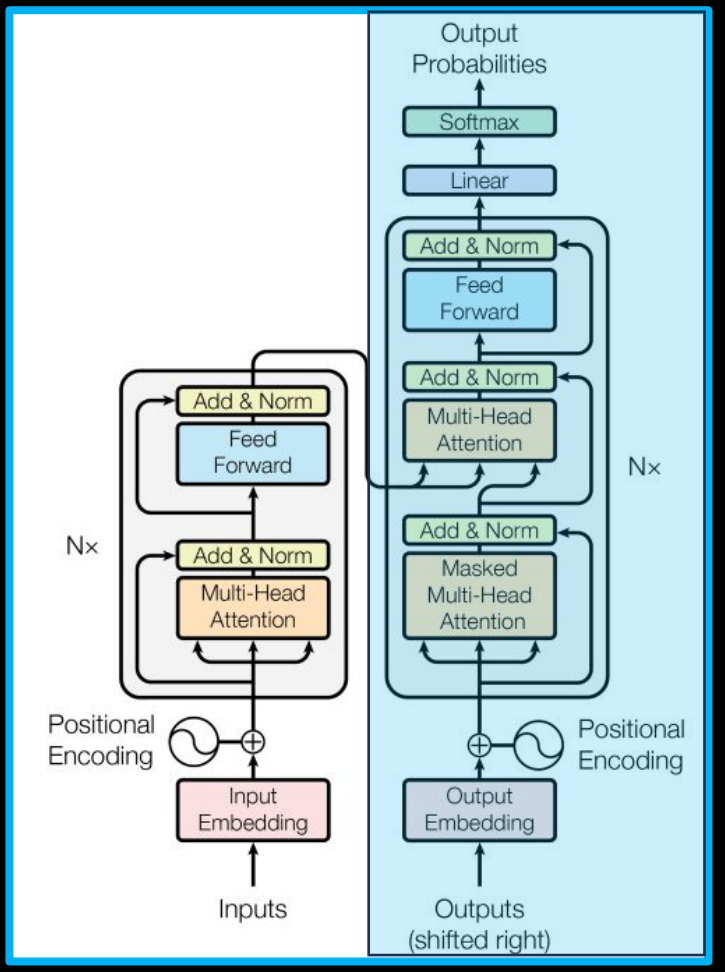
What is produced?



A complex matrix representation of the input sequence that captures higher-order relationships among the tokens

What do we do with a complex matrix representation??

- Feed it into a supervised learning model
- Compare encodings of different input sequences
- Language models: add a decoder that can generate new text including translations

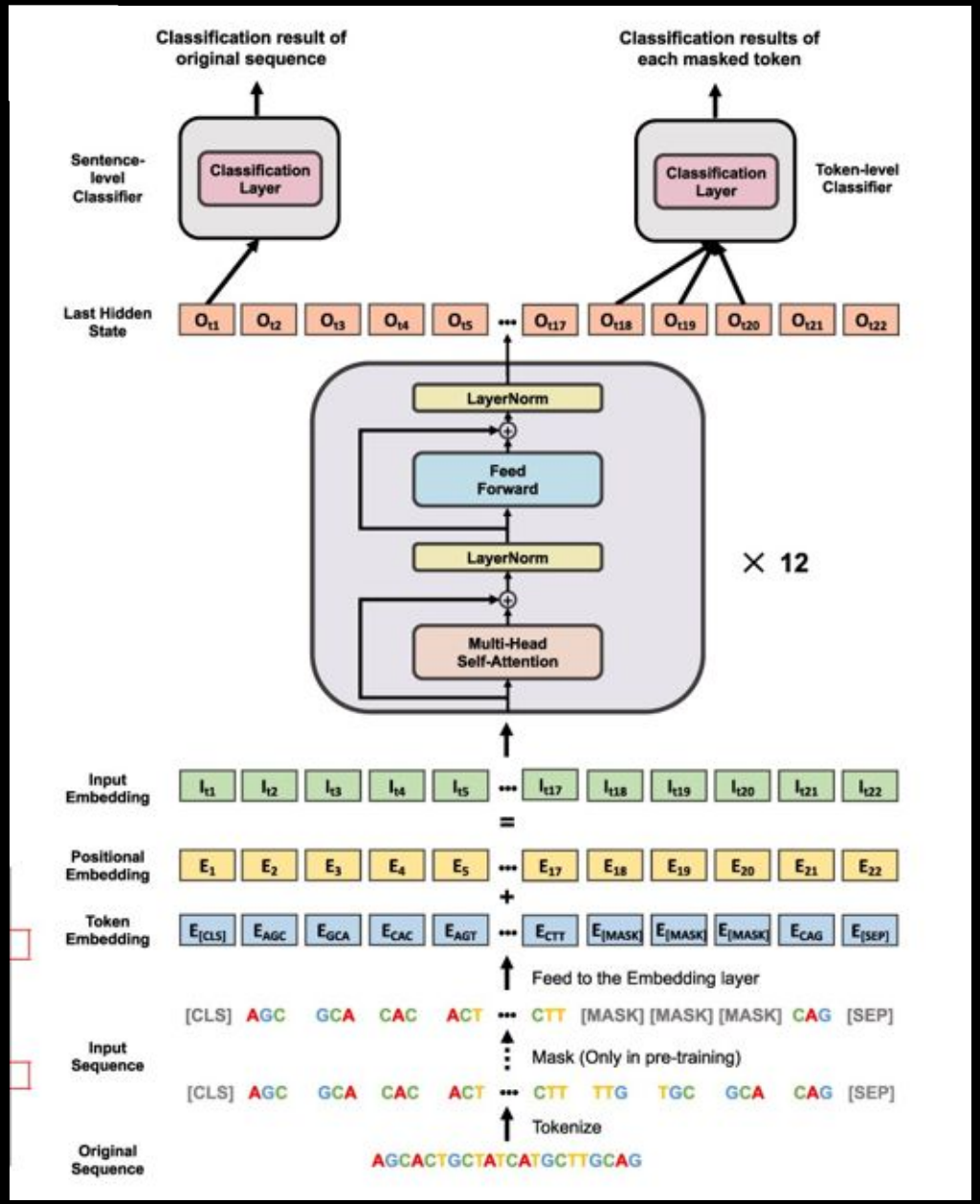


The DNABERT Architecture

Classifiers

12 layers of encoder units

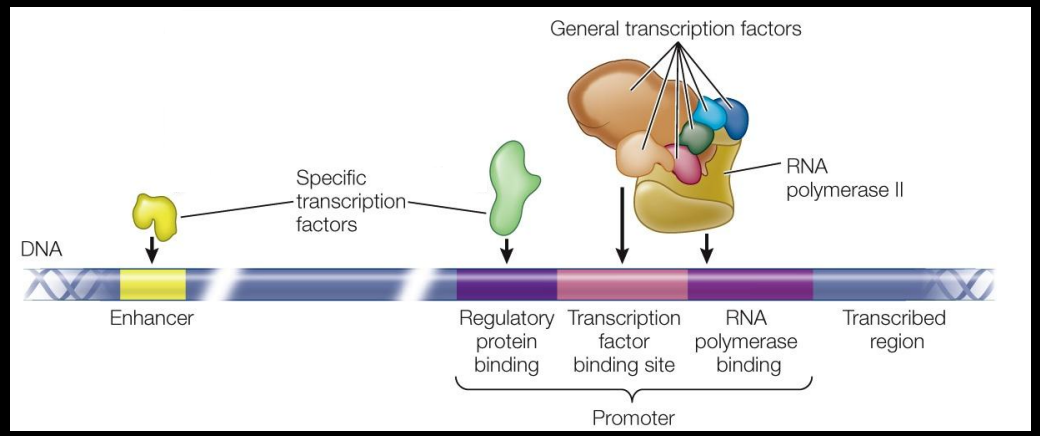
Token and positional embedding



Ji et al. (2021) *Bioinformatics*

Training DNABERT-1

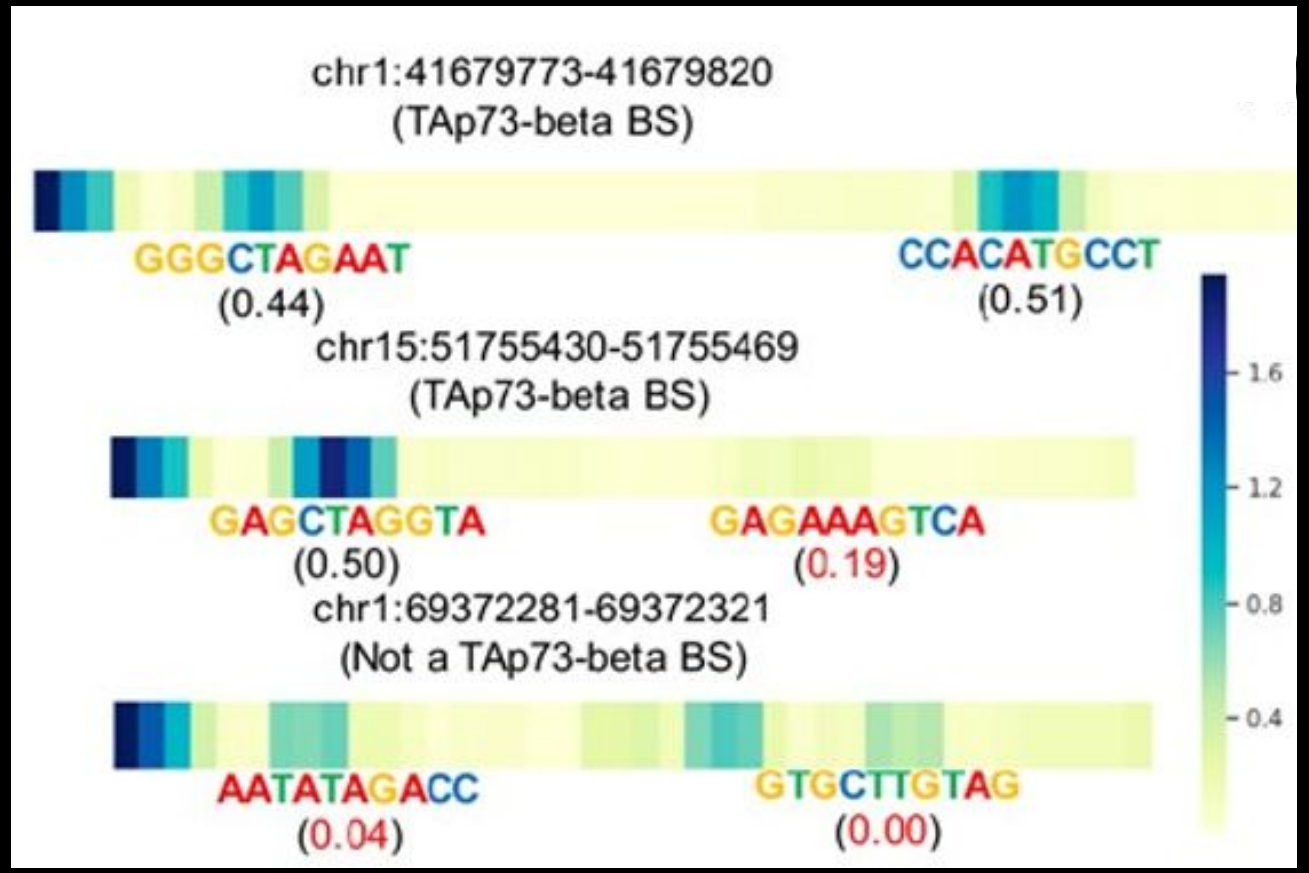
- Pre-train the sequence encoder
  - Use sequences of length 10 to 512 nucleotides
  - Overlapping k-mer decomposition, $3 \leq k \leq 6$
- Fine tune to specific problems
  - Promoters
  - Transcription factor binding sites
  - Splice sites in mRNA
  - …



https://digfir-published.macmillanusa.com/life11e/life11e_ch16_11.html

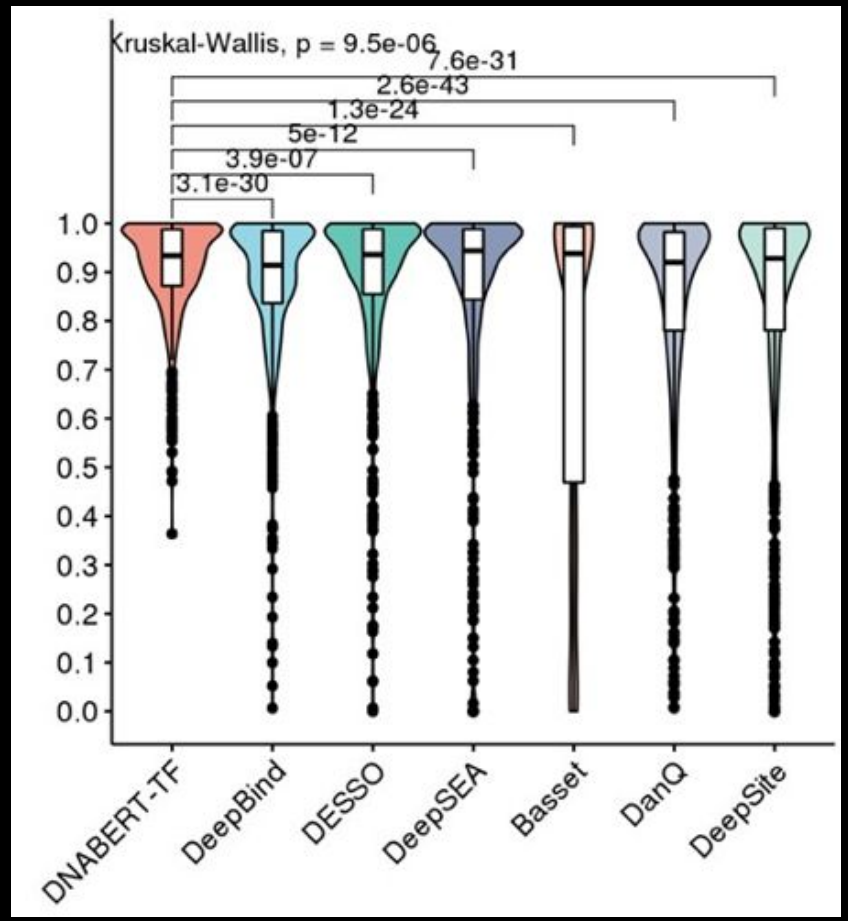
What attention looks like (sort of)

- Grainy (paper quality isn't great, sadly)
- Deep inside the encoder chain, the attention mechanism "recognizes" connections between tokens in different regions of the promoter
- Bottom example: not a promoter, attention is more diffuse
- Encoded representations enable highly effective fine tuning on the specific problem



DNABERT is accurate

- Fine tuning done specifically for transcription factor binding site (TFBS) prediction
- Matthews correlation coefficient for prediction of TFBSs
- Violin plot: distribution of MCC scores across many datasets
- Small *p*-values suggest that DNABERT-TF performs significantly better on average



I hope you like CO_2!

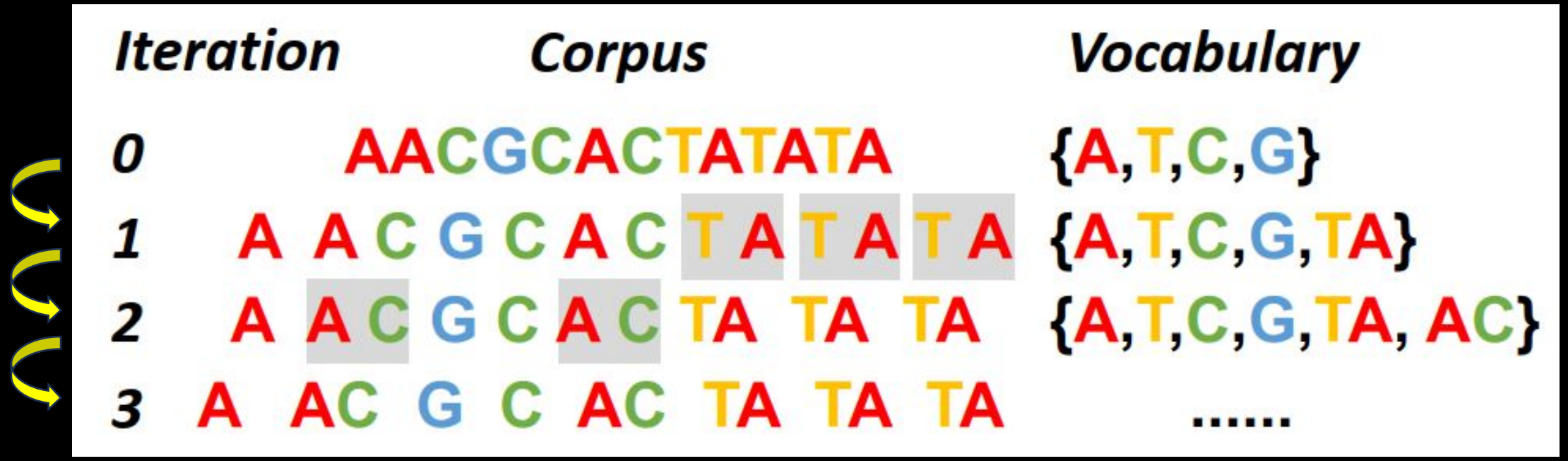
- DNABERT-1 has ~86 million parameters and 122 billion tokens
- Training required 25 days on 8 NVIDIA GPUs

DNABERT-2

- Modifications:
  - Different tokenization strategy
  - More parameters (117M vs 86M)
  - Train on larger data sets (Human / Bunch of Humans / Many Species)
  - Updated attention mechanism – supports longer input sequences (5000 -10,000 nt)
  - Lots of compute optimizations under the hood. More parameters, but over 3x faster than DNABERT-1

DNABERT-2: Zhou et al. (2023) *arXiv* https://github.com/MAGICS-LAB/DNABERT_2

DOWN WITH *K*-MERS



SentencePiece: Byte-pair encoding – start with simplest possible *k*-mers ($k = 1$). Build longer and longer representations based on their frequency in the dataset

Tokens are longer on average, so sequence representations are shorter

Zhou et al. (2023) *arXiv*

Performance

- No pretty visualizations like in first DNABERT paper
- Nucleotide transformer: *k*-mers, way more parameters, slower
- No diamond: encoder pre-training independent of specific classification tasks
- Diamond: further task-specific encoder training

Species	Yeast	Mouse	Virus	Human			
Task	EMP	TF-M	CVC	TF-H	PD	CPD	SSP
DNABERT (3-mer)	49.54	57.73	62.23	64.43	84.63	**72.96**	84.14
DNABERT (4-mer)	48.59	59.58	59.87	64.41	82.99	71.10	84.05
DNABERT (5-mer)	48.62	54.85	63.64	50.46	84.04	72.03	84.02
DNABERT (6-mer)	49.10	56.43	55.50	64.17	81.70	71.81	84.07
NT-500M-human	45.35	45.24	57.13	50.82	85.51	66.54	79.71
NT-500M-1000g	47.68	49.31	52.06	58.92	86.58	69.13	80.97
NT-2500M-1000g	50.86	56.82	66.73	61.99	86.61	68.17	85.78
NT-2500M-multi	58.06	67.01	**73.04**	63.32	**88.14**	71.62	**89.36**
DNABERT-2	55.98	67.99	71.02	**70.10**	84.21	70.52	84.99
DNABERT-2♦	**58.83**	**71.21**	68.49	66.84	83.81	71.07	85.93

Table 4: The models' averaged performance on the 7 tasks in the GUE benchmark, including Epigenetic Marks Prediction (EMP), Transcription Factor Prediction on the Human genome and the Mouse genome (TF-H and TF-M), Covid Variants Classification (CVC), Promoter Detection (PD), Core Promoter Detection (CPD), and Splice Site Prediction (SSP).

Summary

- Models can be as simple or as complex as you like

- Important considerations:
  - Interpretability
  - Data representations
  - Do you have enough training cases?
  - Do you have the resources of Microsoft, Amazon, or Google?

- DNABERT-2 and other deep-learning methods build abstract representations we can't understand, but these can have real biological significance


The End