Machine learning in population and phylogenetics

Megan L Smith
Woods Hole Workshop on Molecular Evolution

34th Workshop on Molecular Evolution

M

Do you think people should use machine learning for population and phylogenetics?

As an AI language model, I don't have personal opinions or beliefs. However, I can provide you with some information on the topic.

Machine learning techniques can be valuable tools in the fields of population genetics and phylogenetics. These areas of study involve analyzing genetic data to understand the relationships, evolutionary history, and diversity within populations and species.

Machine learning algorithms can aid in tasks such as genotype imputation, population structure analysis, genetic variant classification, and phylogenetic tree construction. They can help researchers handle large volumes of data, detect patterns, make predictions, and identify relevant features that might not be immediately apparent.

Using machine learning for population and phylogenetic analyses can offer several benefits. It can enhance computational efficiency, improve accuracy in data analysis, and help uncover hidden patterns or relationships that may be challenging to detect manually. Additionally, machine learning approaches can provide opportunities for automation and scalability, allowing researchers to analyze more extensive datasets and explore complex genetic interactions.

However, it's important to note that machine learning is a tool and not a substitute for domain expertise. It should be used in conjunction with traditional analytical methods and biological knowledge to ensure accurate interpretation of results. Careful validation and verification are essential to ensure that the machine learning models are reliable and appropriately applied to the specific biological questions at hand.

Moreover, machine learning techniques should be transparent, interpretable, and subject to rigorous scrutiny to ensure that the results are scientifically meaningful. Ethical considerations, such as potential biases in the training data or interpretations, should also be taken into account.

In summary, machine learning can be a valuable tool for population genetics and phylogenetics, but it should be used judiciously, in conjunction with domain expertise, and with careful consideration of potential limitations and ethical implications.

Goals

- Why might we use machine learning in population and phylogenetics?
- How do some supervised machine learning algorithms work?
- How have supervised machine learning approaches been applied in population and phylogenetics?
- What are the challenges and limitations of these approaches, and how do we move forward?

Table of contents

Introduction and Motivation

What is Machine Learning?

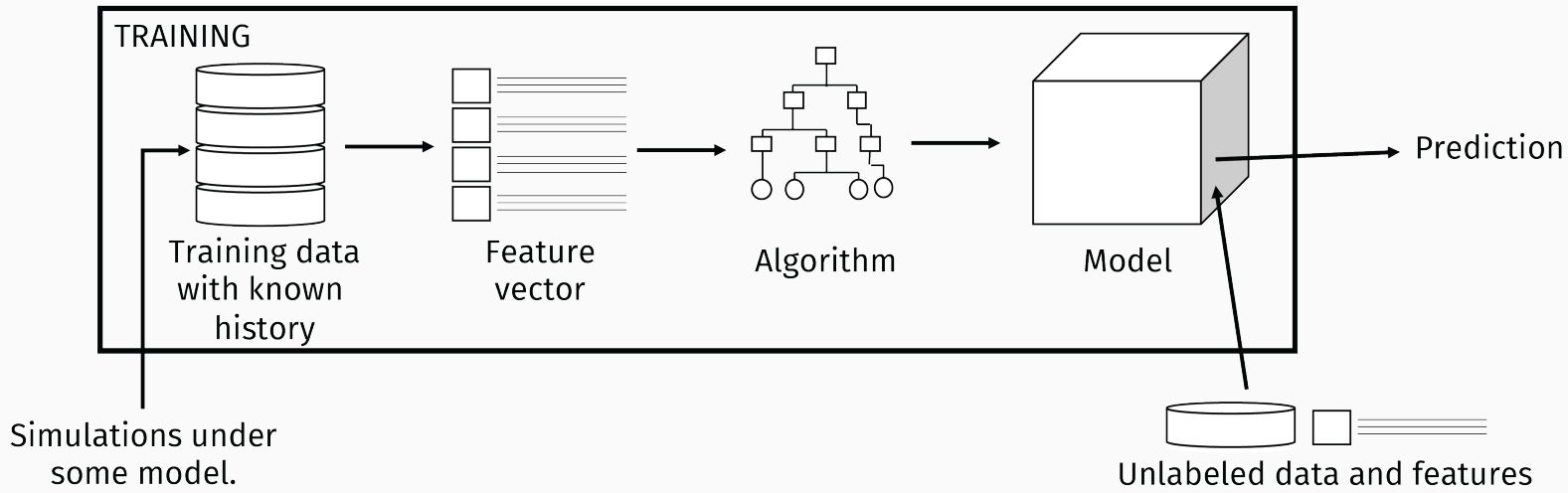
- A subfield of Artificial Intelligence.
- Uses data to perform inference without explicit mathematical models.
- Identifies patterns, which can be used to predict unknown outcomes.
- Major classes: unsupervised vs. supervised

Unsupervised Machine Learning

- Finds patterns within data.
- No notion of prediction.
- example: Principle Component Analyses

Supervised Machine Learning

- **Goal:** To learn to predict an output from some input.
- Requires training data to learn the mapping between input and output.
- Tunes parameters to maximize prediction accuracy



Motivation

- The growing availability of genomic data and theoretical advances have led to an increased appreciation of the need to consider complex models when analyzing genomic data in evolutionary contexts.

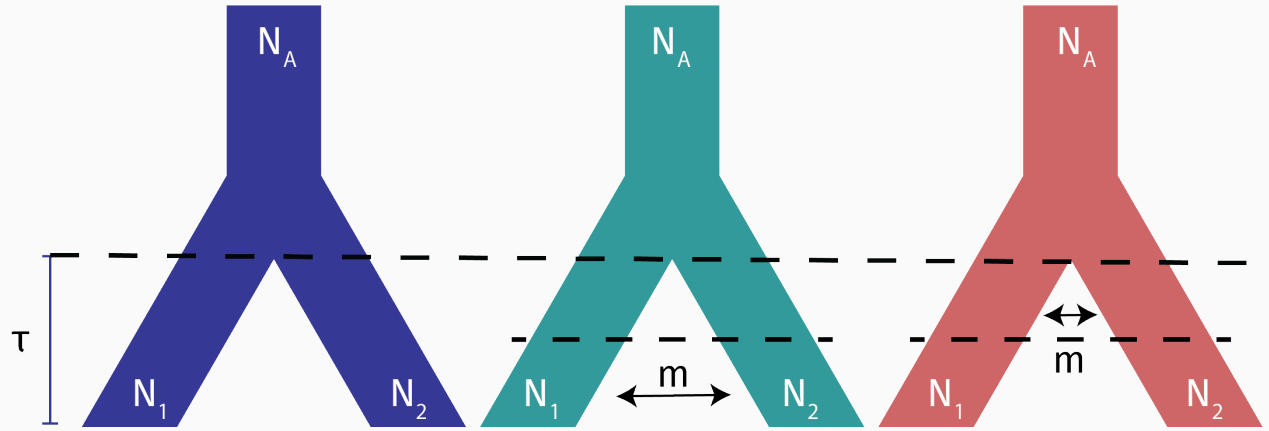
- Sometimes, we cannot use full Likelihood and Bayesian methods to analyze large genomic datasets under complex models due to intractable likelihood functions and computational challenges.

- This has led to a growing popularity of likelihood-free approaches.

Approximate Bayesian Computation

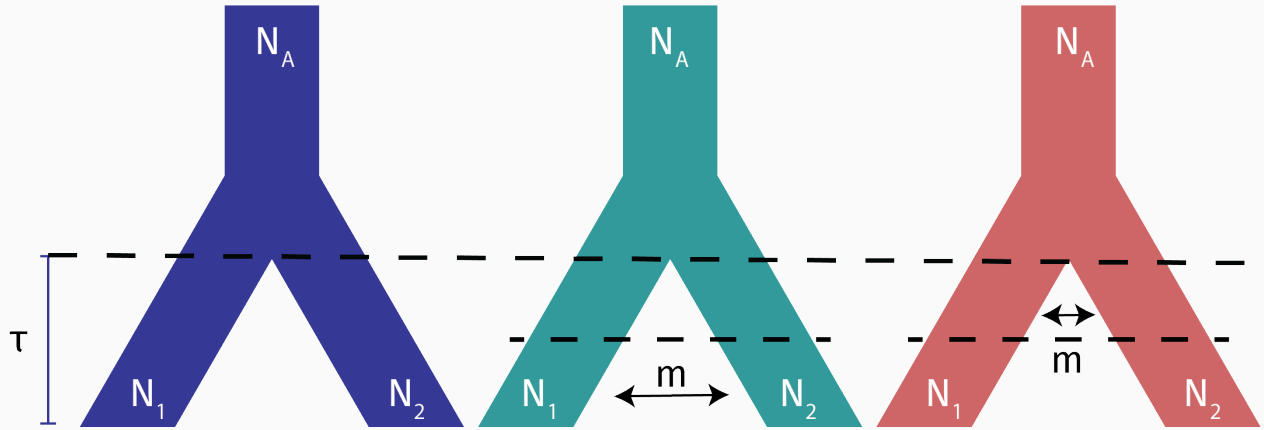
Approximate Bayesian Computation (ABC) was one of the earliest such approaches used in population genomics.

Imagine we want to use ABC to select the best demographic model given our observed sequence data.



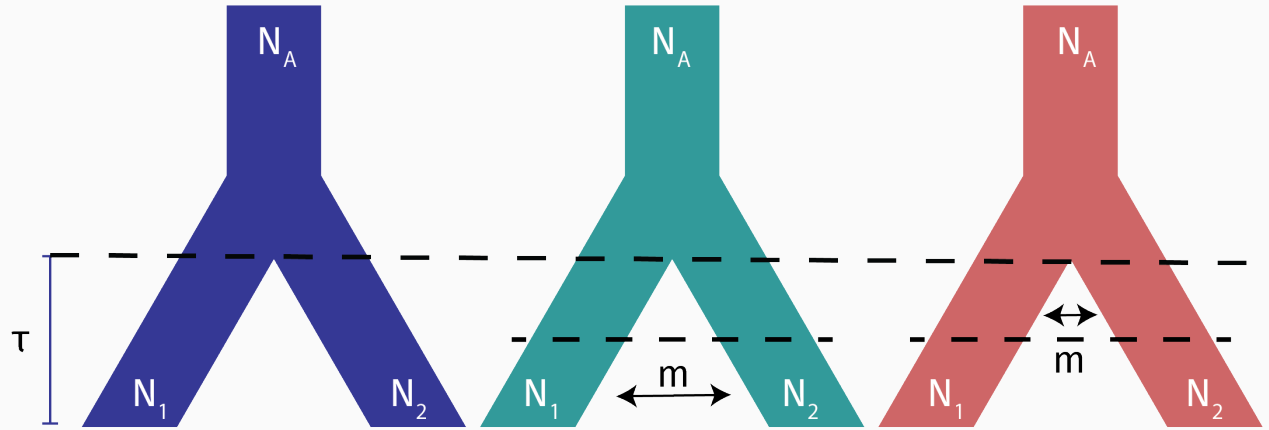
Approximate Bayesian Computation

Step 1: Simulate a prior distribution.



Approximate Bayesian Computation

Step 1: Simulate a prior distribution.



Model 1, Replicate 1:

N_A : U(5000, 100000); Sample: 30,587

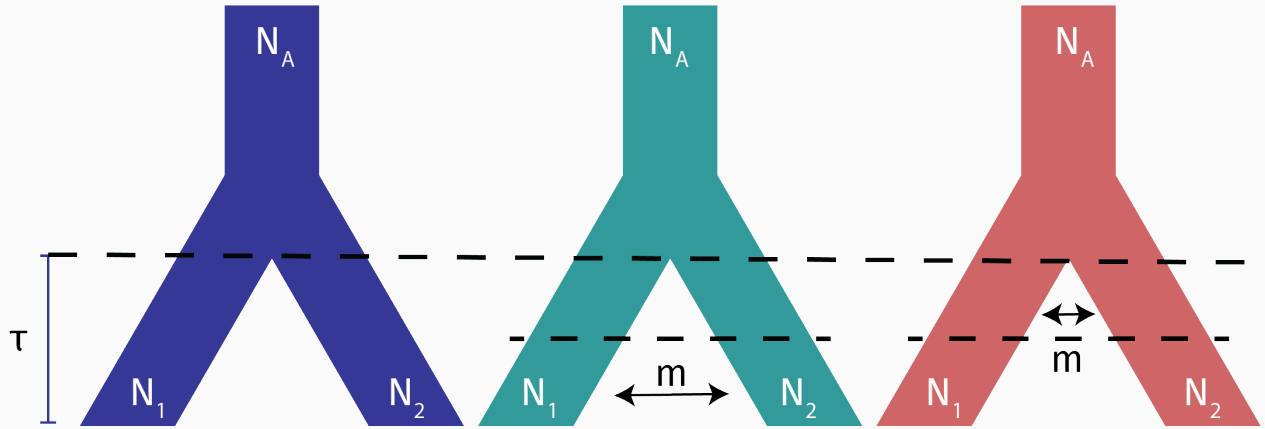
N_1 : U(5000, 100000); Sample: 52,765

N_2 : U(5000, 100000); Sample: 41,582

τ : U(1000, 100000); Sample: 75,283

Approximate Bayesian Computation

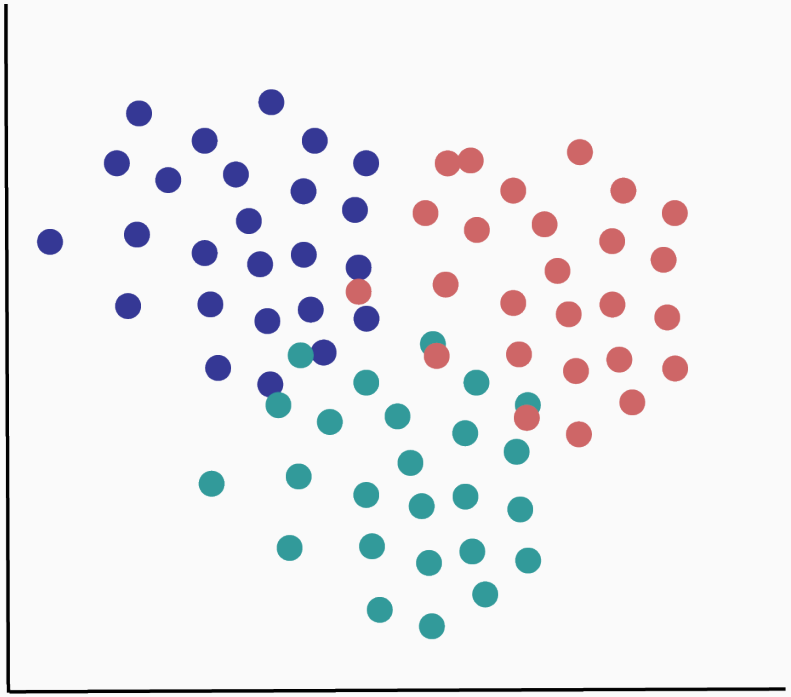
Step 1: Simulate a prior distribution.



Repeat this process tens of thousands of times for each model to generate a **prior**!

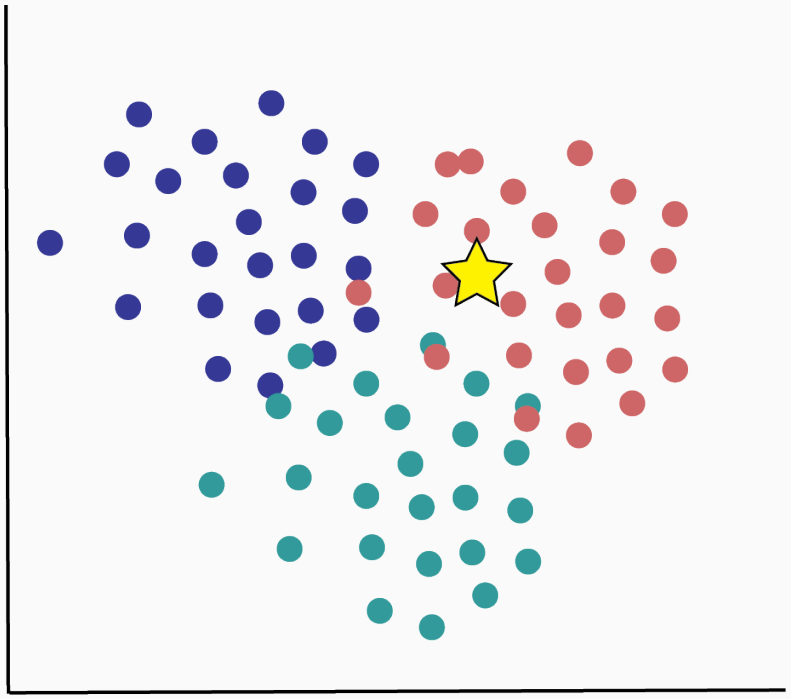
Approximate Bayesian Computation

Step 2: Calculate summary statistics for each simulated dataset in the prior (e.g., π, F_{ST}).



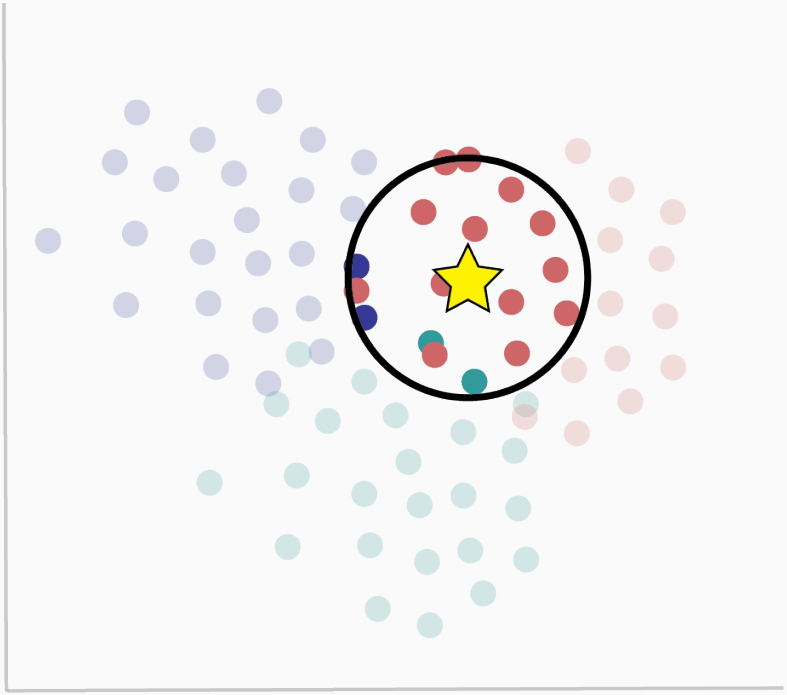
Approximate Bayesian Computation

Step 3: Calculate the same summary statistics for your empirical data.



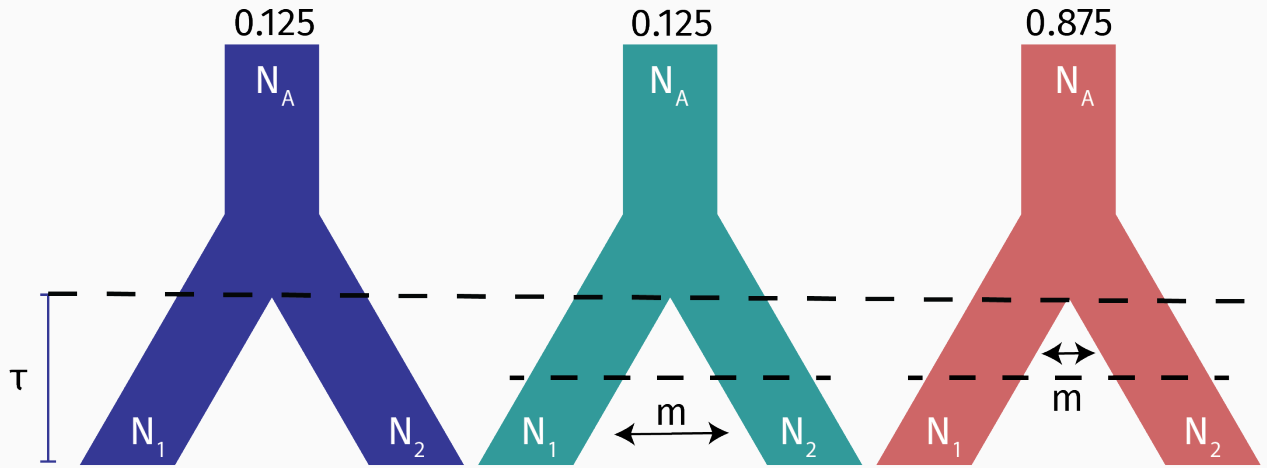
Approximate Bayesian Computation

Step 4: Calculate the distance $\rho(D, D_i)$ between empirical and simulated data, and keep simulated data within some distance of the empirical data.



Approximate Bayesian Computation

Step 5: Calculate the approximate posterior probability of each model.



Strengths of ABC

- It's very flexible!
- We can consider any model under which we can simulate data.
- We can carefully select summary statistics that we expect will help to distinguish amongst models.

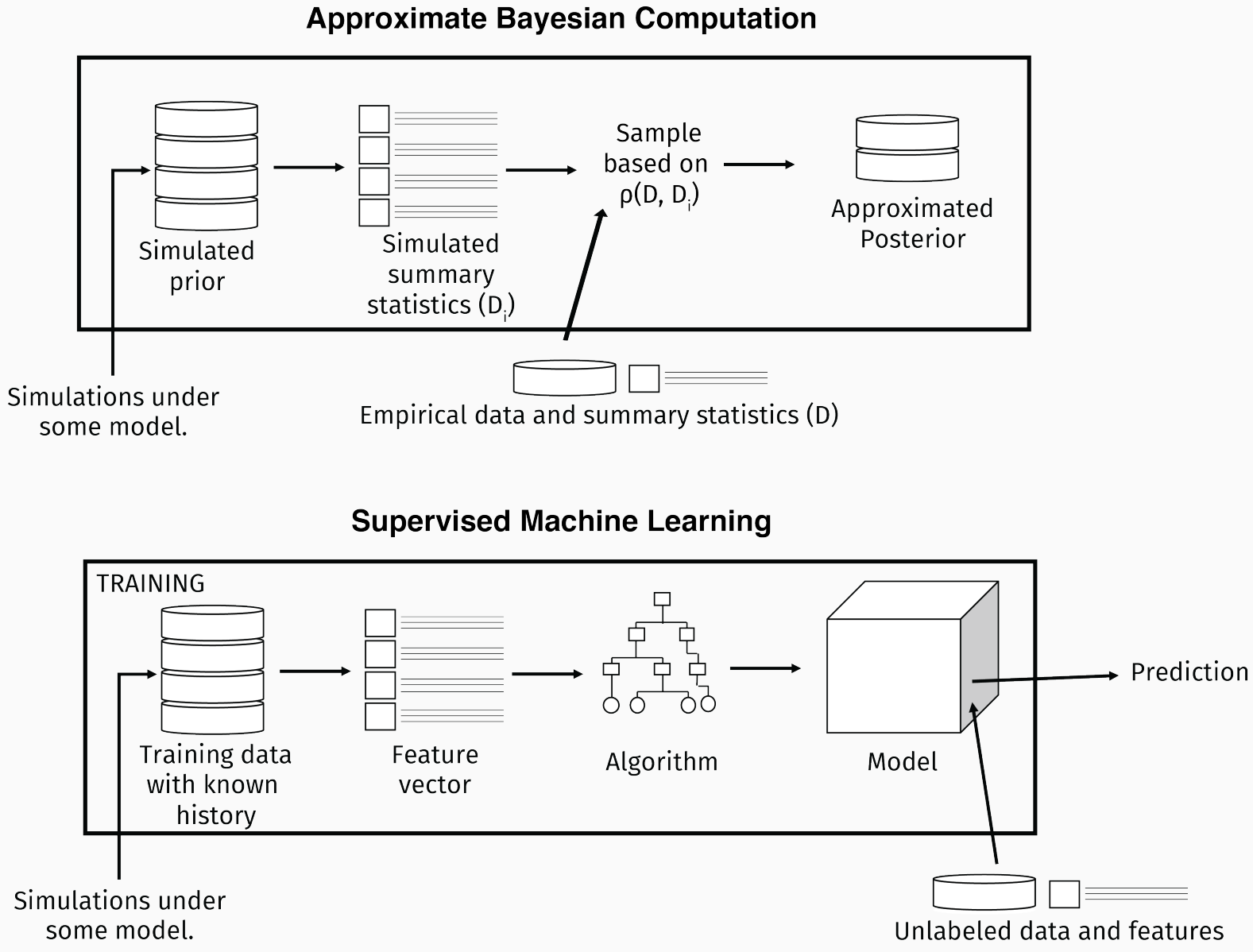
Weaknesses of ABC

- We have to summarize our data using summary statistics, causing information loss. This is especially true when we have genomic-scale data!

- Using too many summary statistics leads to a decrease in performance (i.e., the curse of dimensionality).

- Considering too many models leads to a decrease in performance.

Supervised Machine Learning versus ABC



Supervised Machine Learning versus ABC

- We are not as limited by the number of statistics we can use, meaning we don't have as much information loss.
- In some cases, we can use data directly as input, avoiding any apriori statistic selection.
- In practice, we can compare more models with greater accuracy.

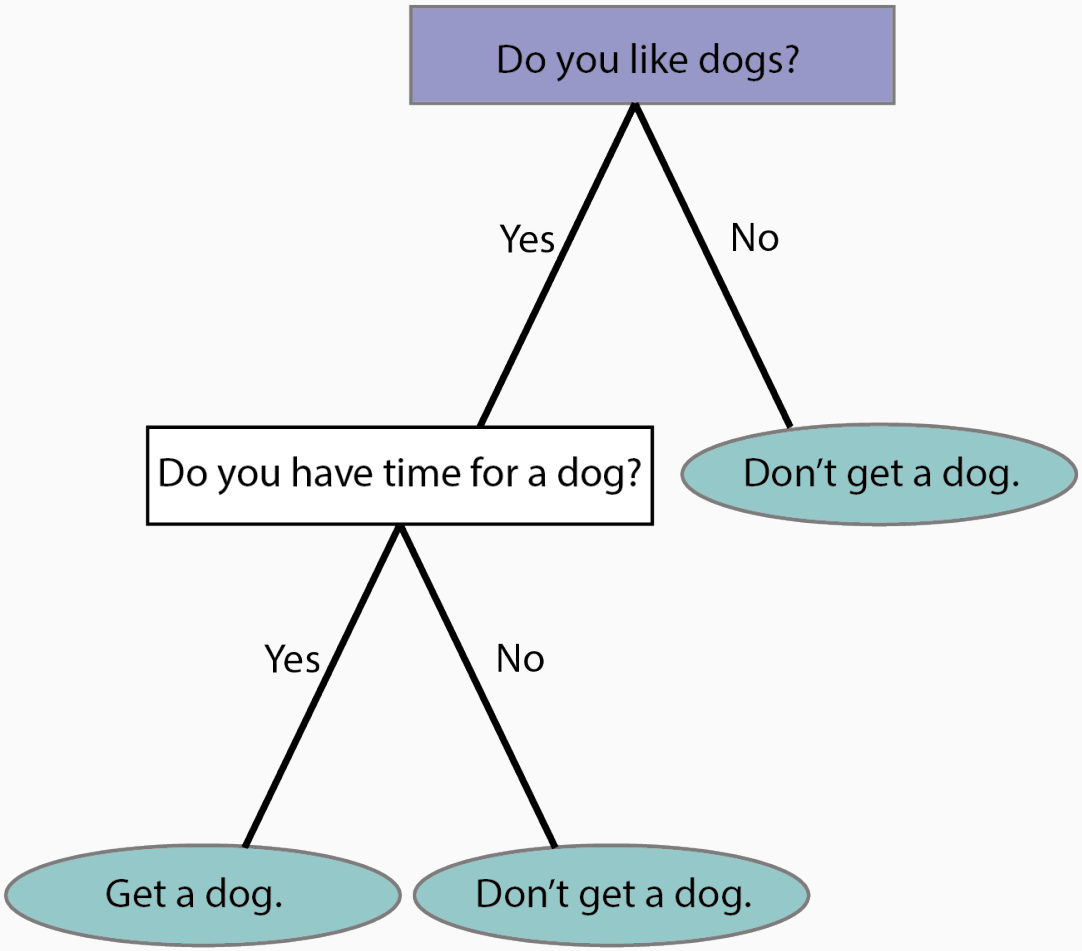
Why do we use machine learning?

- We cannot always use full Likelihood or Bayesian approaches to compare complex models or estimate parameters from our large genomic datasets.
- Traditional likelihood-free approaches have limitations that are more pronounced for large-scale genomic datasets.
- Supervised machine learning allows us to compare complex models with improved computational effeciency while using our data more effectively.

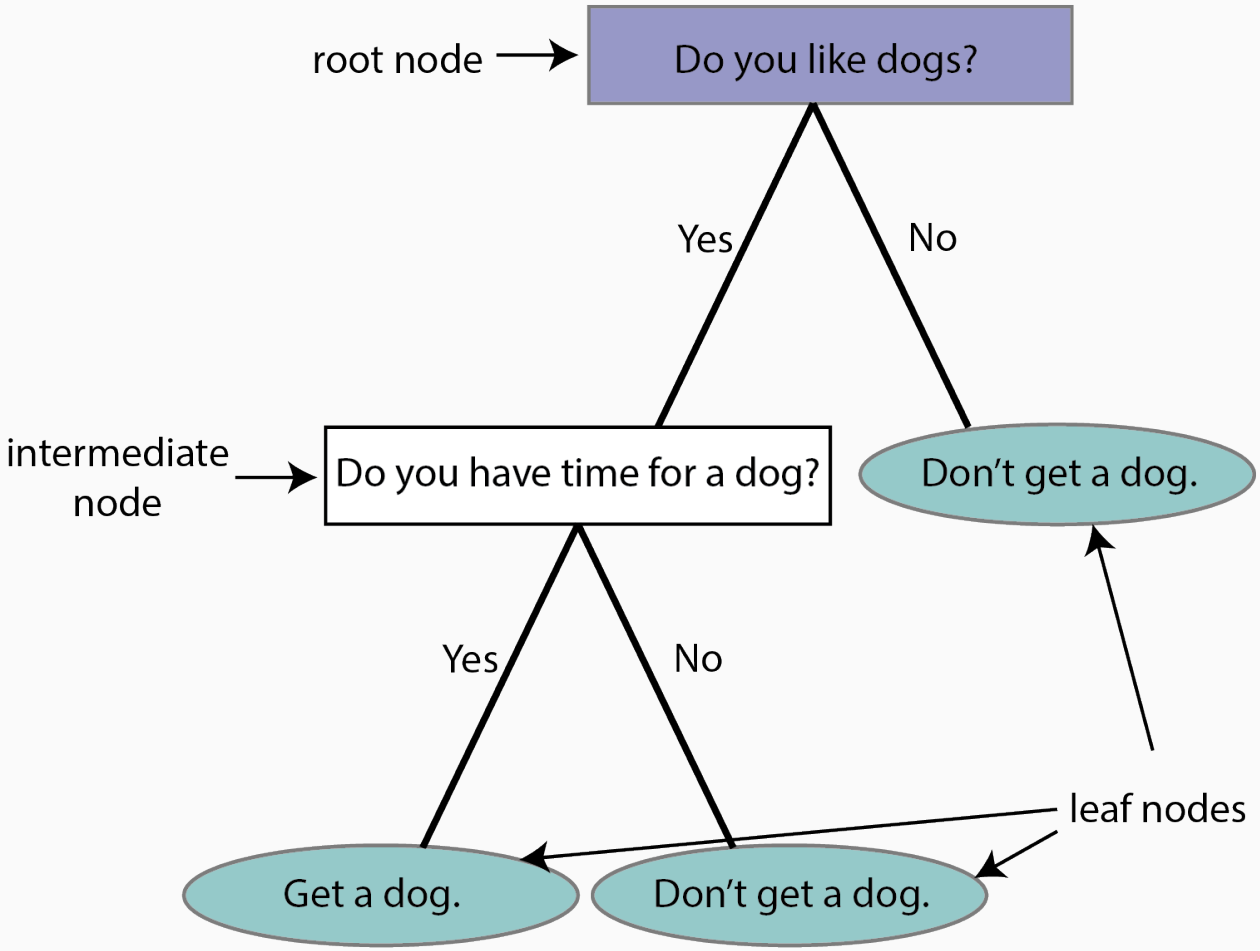
Overview of Supervised Machine Learning Algorithms

Outline

What is a decision tree?



What is a decision tree?



How do we build decision trees?

Goal: Build a decision tree that can predict whether I will find a slug at a particular sampling site.

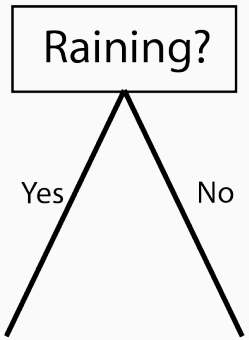
Will I find a slug?

Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**

How do we build decision trees?

Will I find a slug?

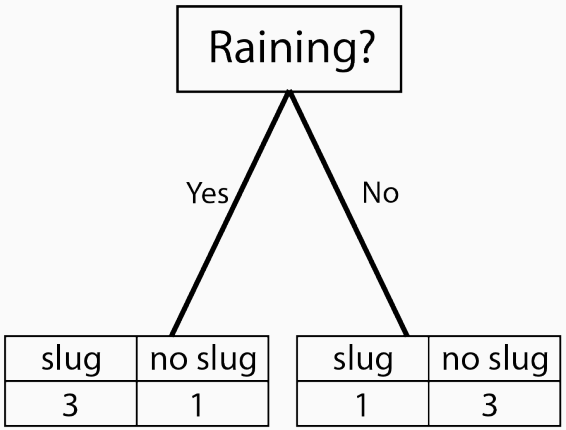
Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



How do we build decision trees?

Will I find a slug?

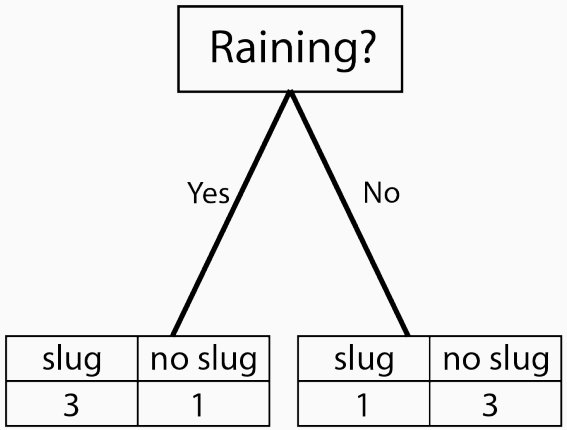
Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



How do we build decision trees?

Will I find a slug?

Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



GINI Index

measure of node purity

0: perfectly pure

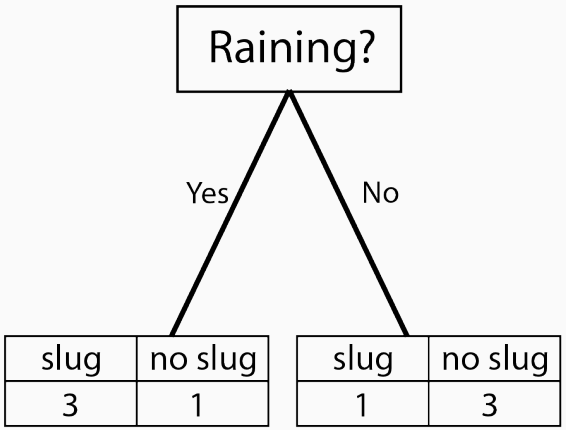
1: elements randomly distributed

$$GINI = 1 - \sum_{i=1}^{n} {p_i}^2$$

How do we build decision trees?

Will I find a slug?

Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



GINI Index

measure of node purity

0: perfectly pure

1: elements randomly distributed

yes: GINI = $1 - (3/4)^2 - (1/4)^2 = 0.375$

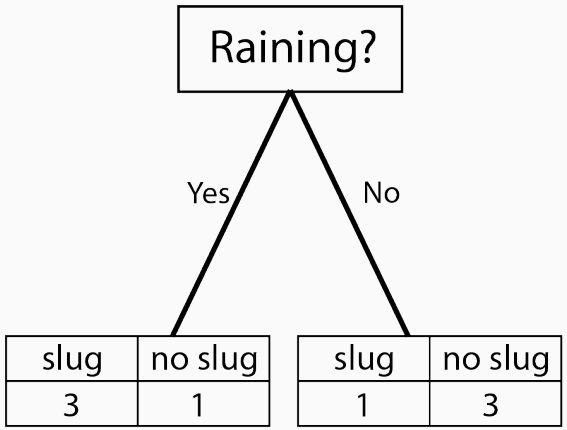
no: GINI = $1 - (1/4)^2 - (3/4)^2 = 0.375$

average = $(4/8) \times 0.375 + (4/8) \times 0.375 = 0.375$

How do we build decision trees?

Will I find a slug?

0.375	0.4667	0.4286	
Raining?	**Stream?**	**# Logs**	**Slug?**
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



GINI Index

measure of node purity

0: perfectly pure

1: elements randomly distributed

yes: GINI = $1 - (3/4)^2 - (1/4)^2 = 0.375$

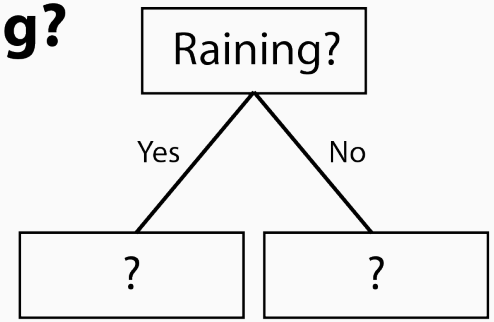
no: GINI = $1 - (1/4)^2 - (3/4)^2 = 0.375$

average = $(4/8) \times 0.375 + (4/8) \times 0.375 = 0.375$

How do we build decision trees?

Will I find a slug?

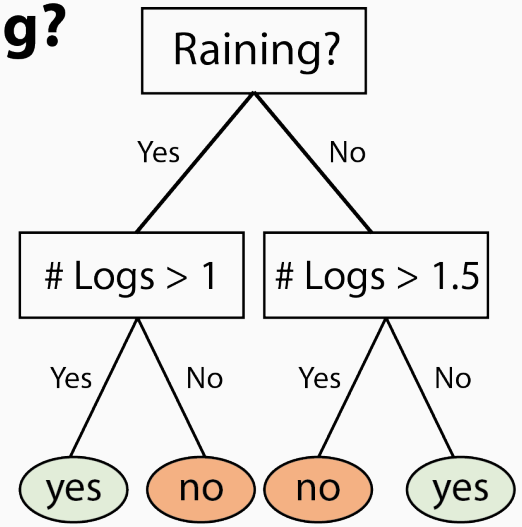
Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



How do we build decision trees?

Will I find a slug?

Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



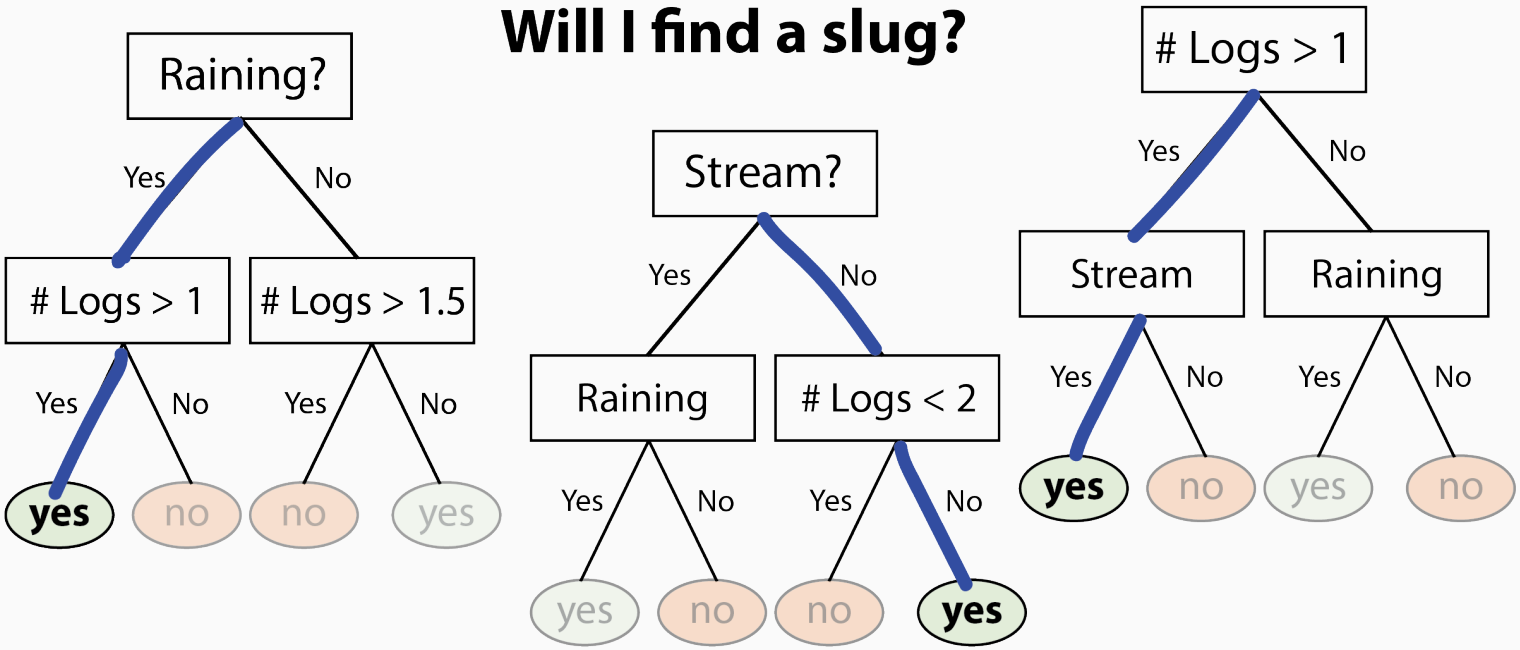
How do we build decision trees?

- We recursively split our training data using the features that perform best.
- We stop when we meet some criterion (maximum depth, minimum number of samples in leaf, minimum decrease in error metric).
- We can use decision trees for classification or regression.
- We can learn about which predictors drive classification from our decision trees!
- Decision trees are prone to overfitting!

Random Forests!

- To address this issue, we can use a collection of decision trees!
- We construct a Random Forest Classifier (or regressor) by doing the following.
    - For each iteration *i*:
        - Generate a bootstrapped dataset.
        - Create a decision tree using the bootstrapped dataset, and only use a random subset of variables when constructing each node.
    - When classifying data, use all trees, and consider the class that the most decision trees vote for to be the predicted class.
    - This is called Bagging (bootstrapping + aggregating).

Random Forests!

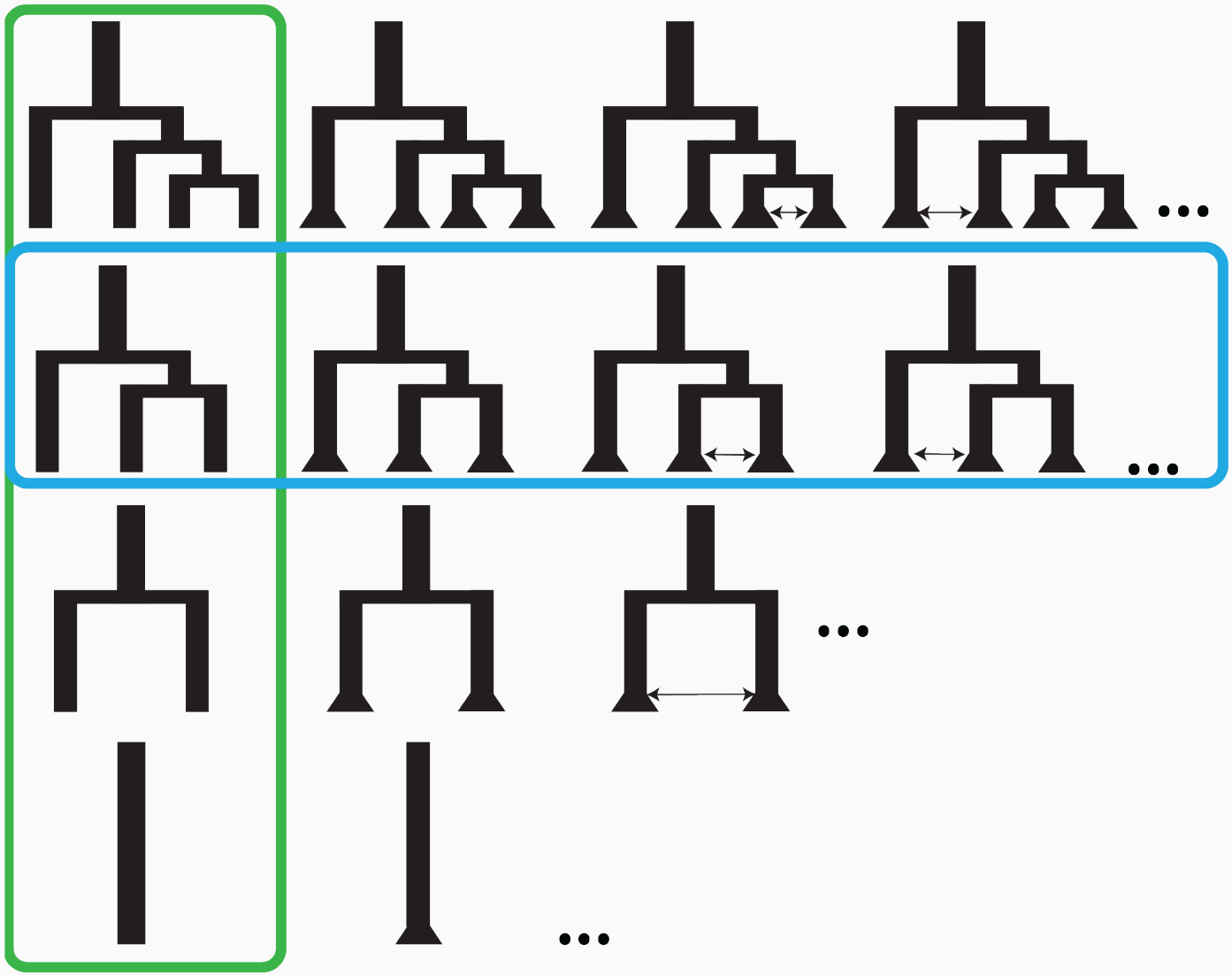


Advantages of Random Forests

- Less prone to overfitting.
- We can estimate 'out-of-bag' error rates very easily to assess power.
  - For each element of the training data:
    - Predict the element's class using the decision trees constructed without reference to that element.
  - Calculate how often elements of each class are accurately predicted.

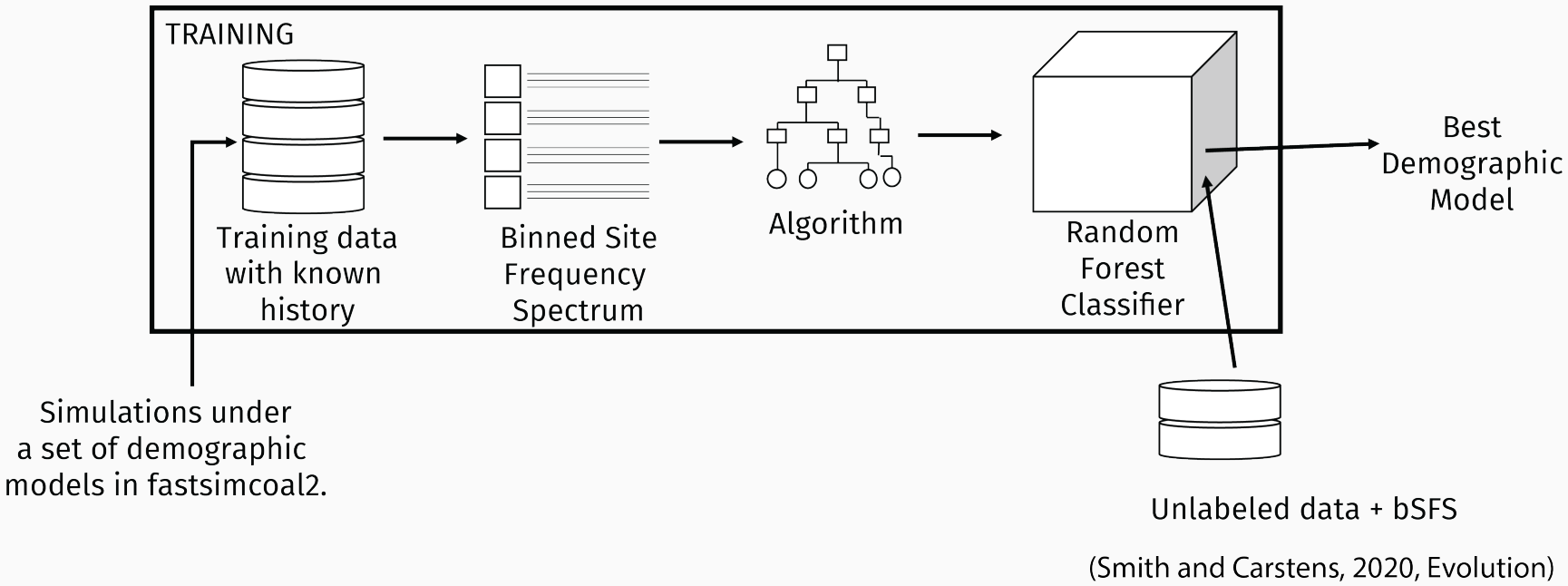
Examples: delimitR

Goal: To delimit species while considering population-level processes.



Examples: delimitR

Goal: To delimit species while considering population-level processes.



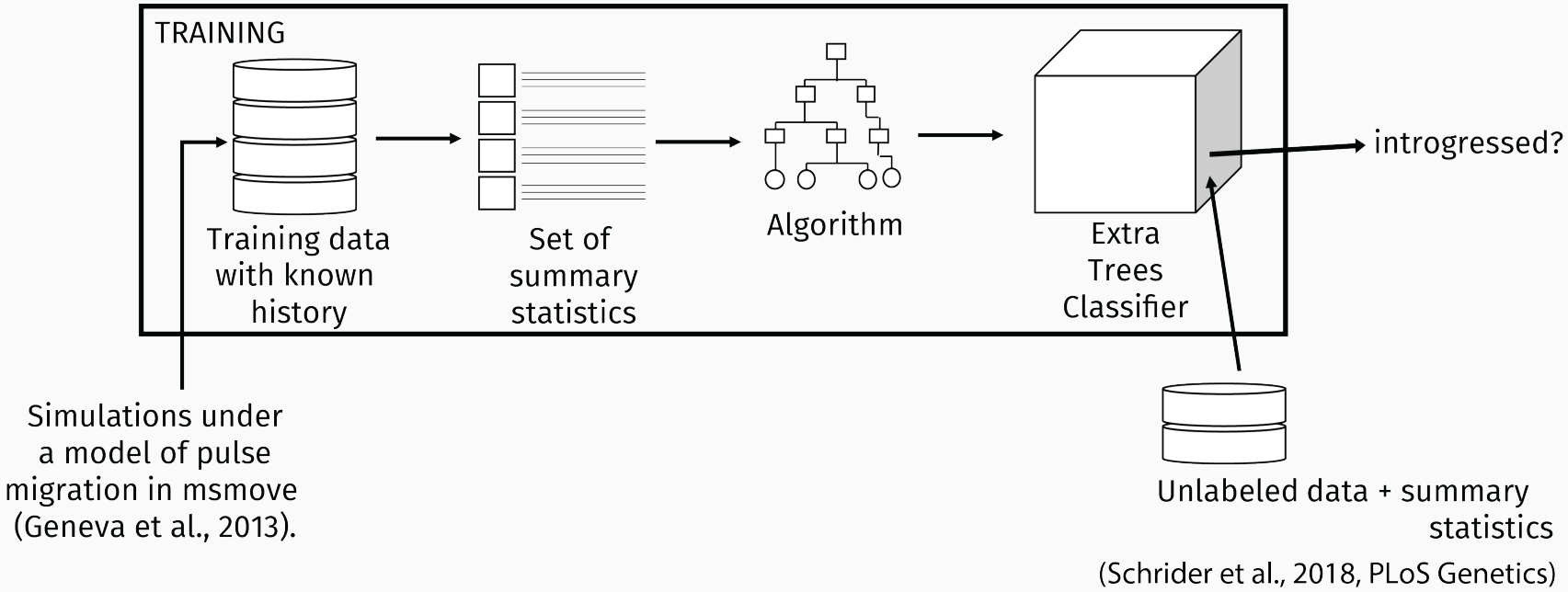
(Smith and Carstens, 2020, Evolution)

Examples: delimitR

- delimitR predicts the best demographic model using SNP data.
- delimitR can consider any model that can be defined in fastsimcoal2 (Excoffier et al., 2013).
- delimitR uses the SFS to summarize the data.
- delimitR achieves error rates < 5% for hundreds of models.

Examples: FILET

Goal: To classify genomic windows as introgressed or not, and to determine the direction of introgression when it occurs.



(Schrider et al., 2018, PLoS Genetics)

Examples: FILET

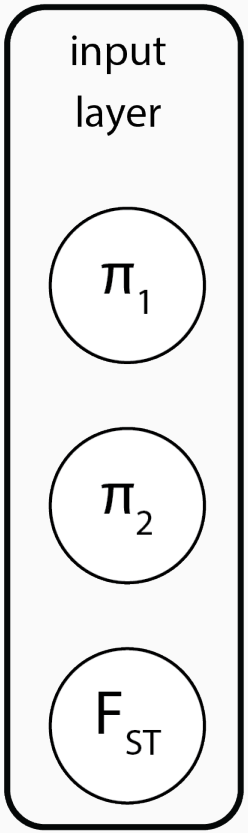
- FILET predicts whether a genomic region has introgressed across population or species boundaries.
- FILET uses a suite of hand-crafted summary statistics to summarize the data.
- FILET achieves reasonable error rates that depend on the divergence time and time of introgression.

Decision Trees and Random Forests

- Decision trees learn how to use features to split data in meaningful ways, and then can be used to make predictions on unseen data.
- Random Forests are collections of decision trees.
- Decision-tree based classifiers are highly interpretable and outperform traditional ABC in many cases.

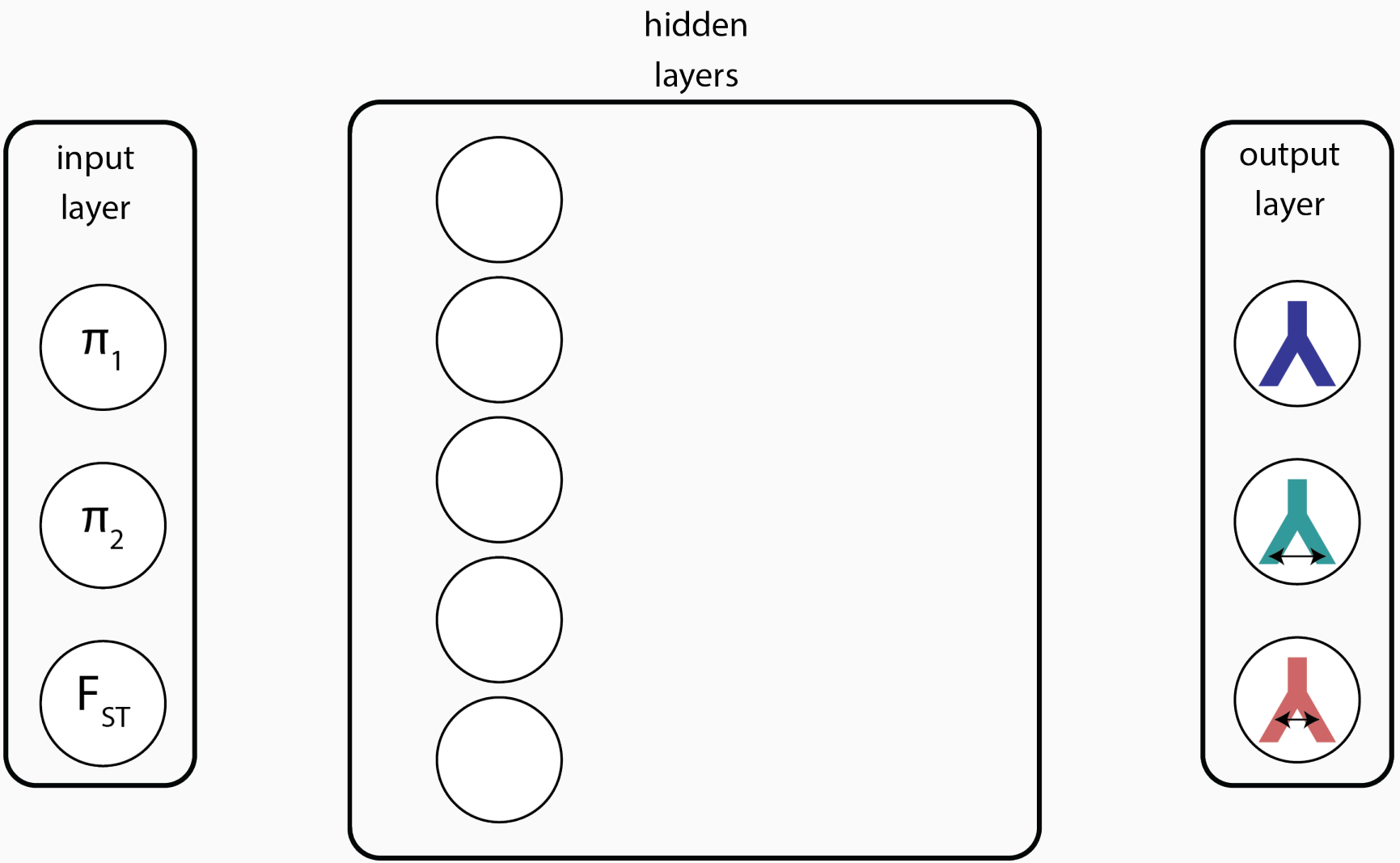
Outline

What are neural networks?

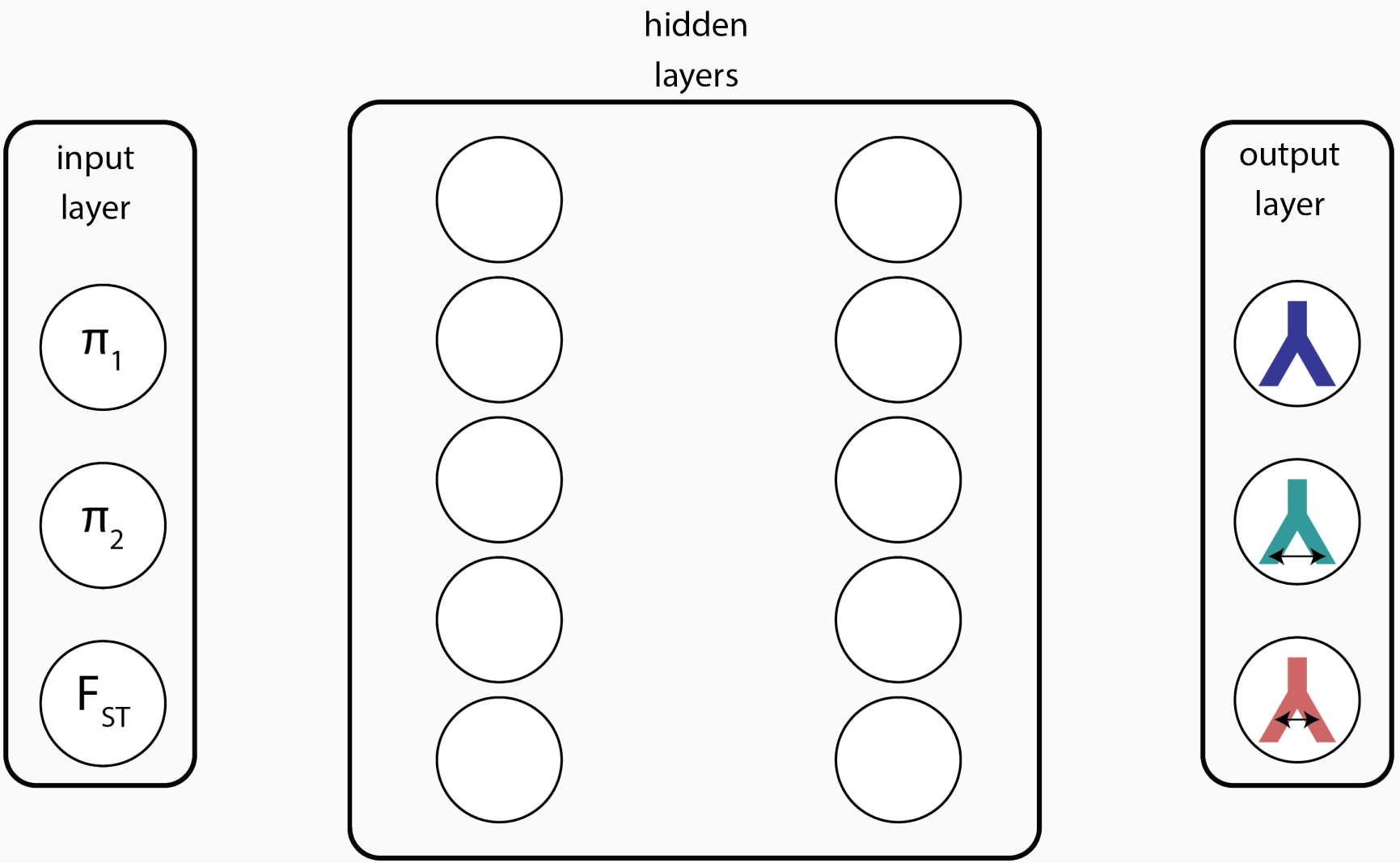


What are neural networks?

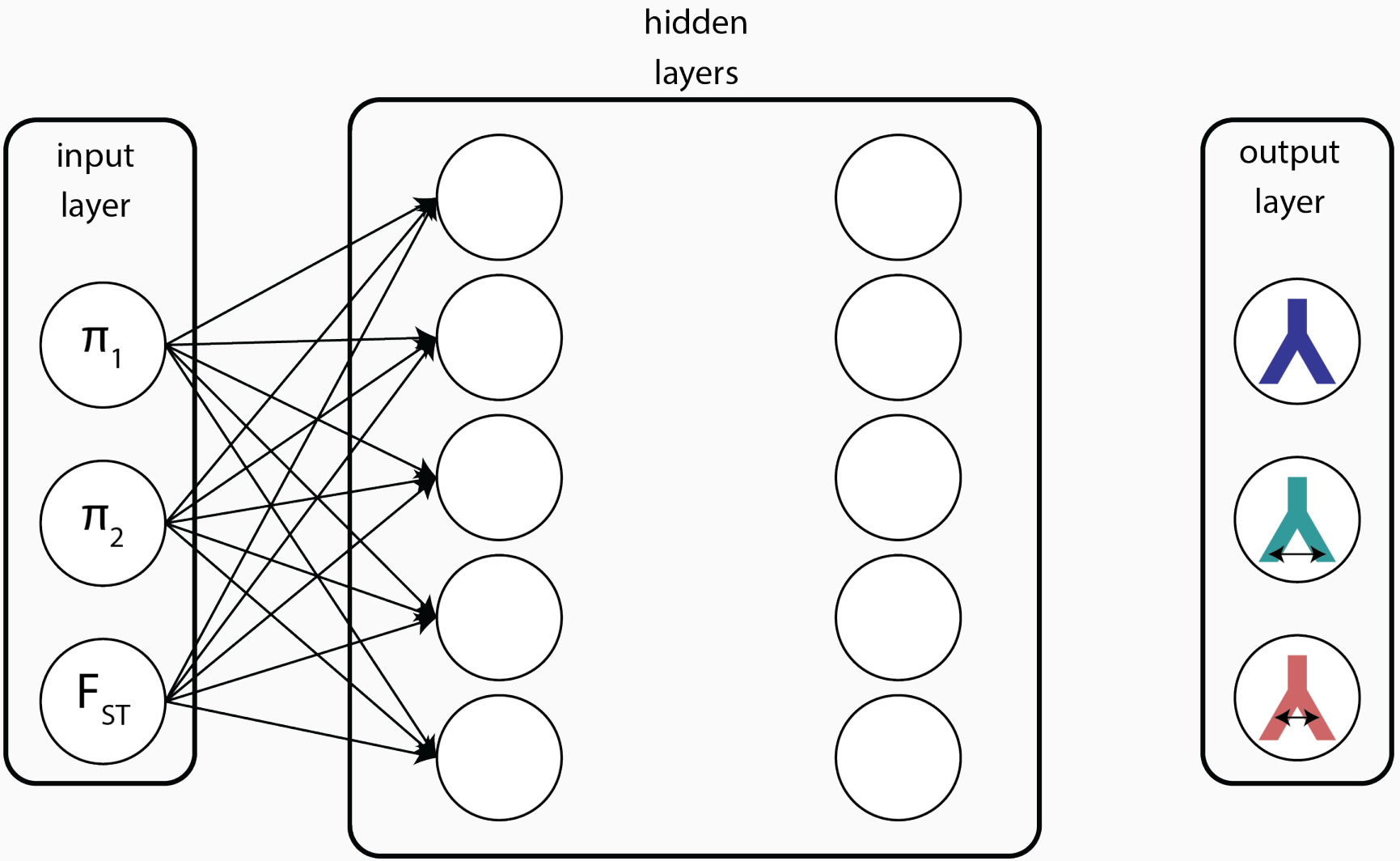
What are neural networks?



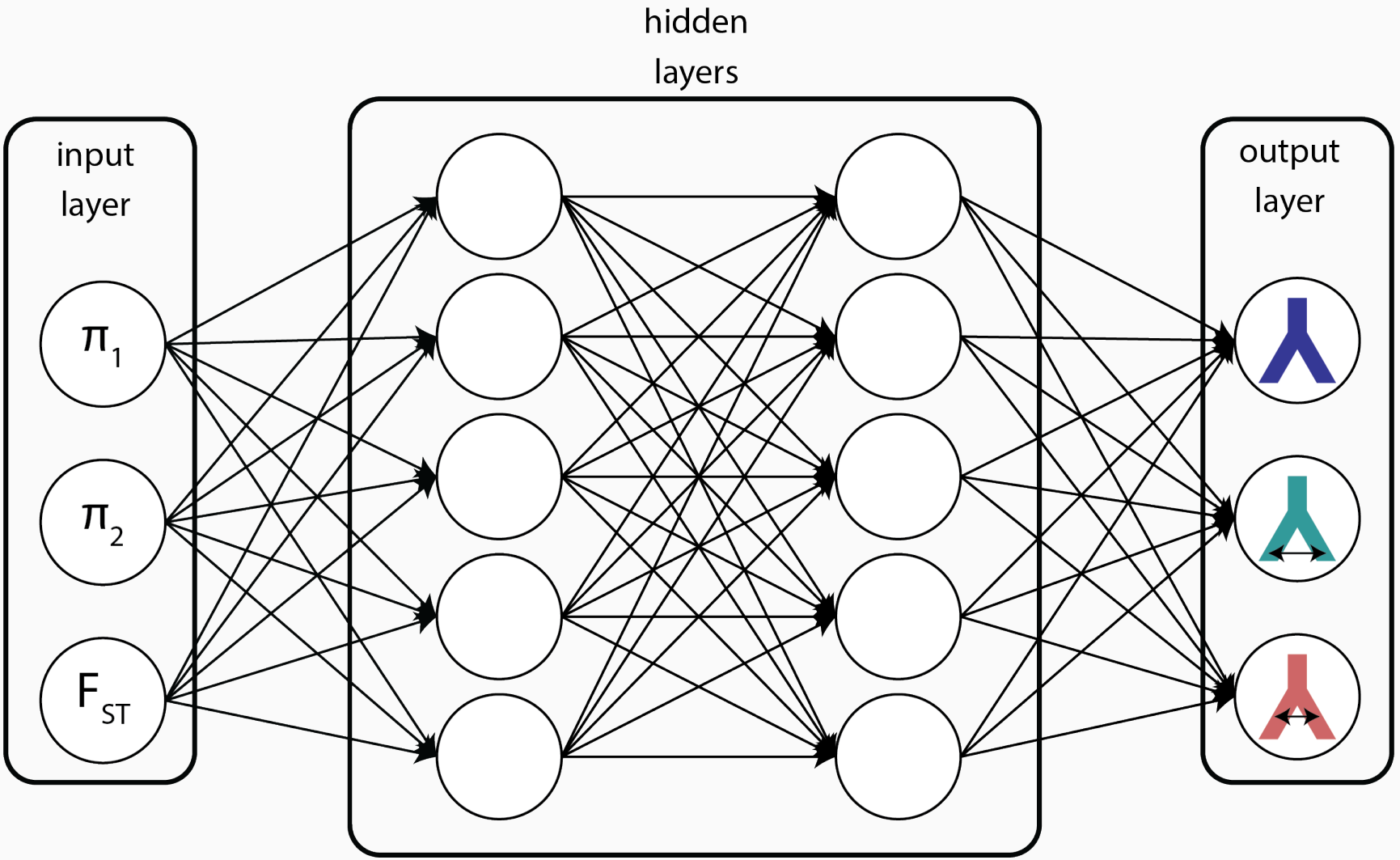
What are neural networks?



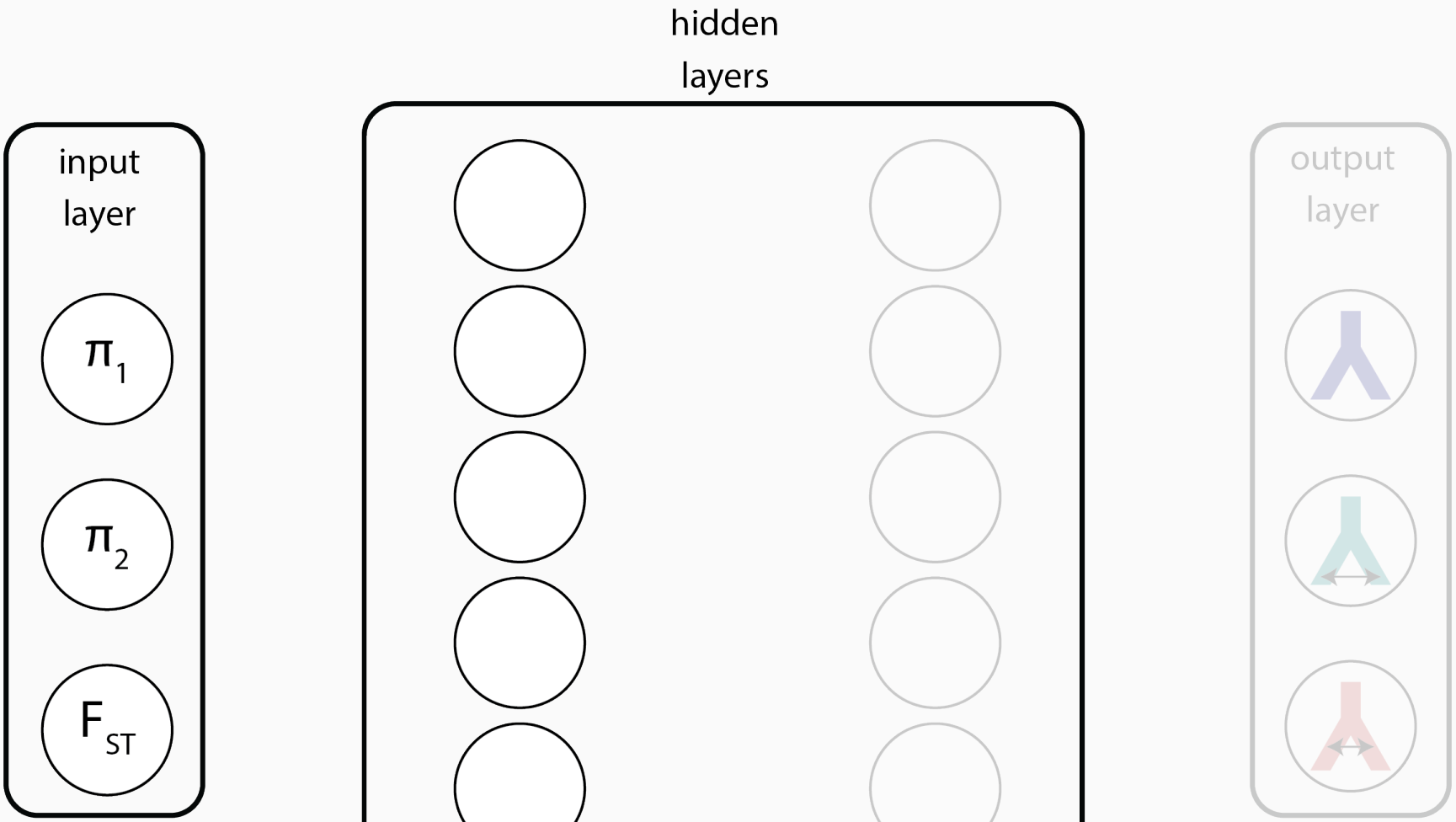
What are neural networks?



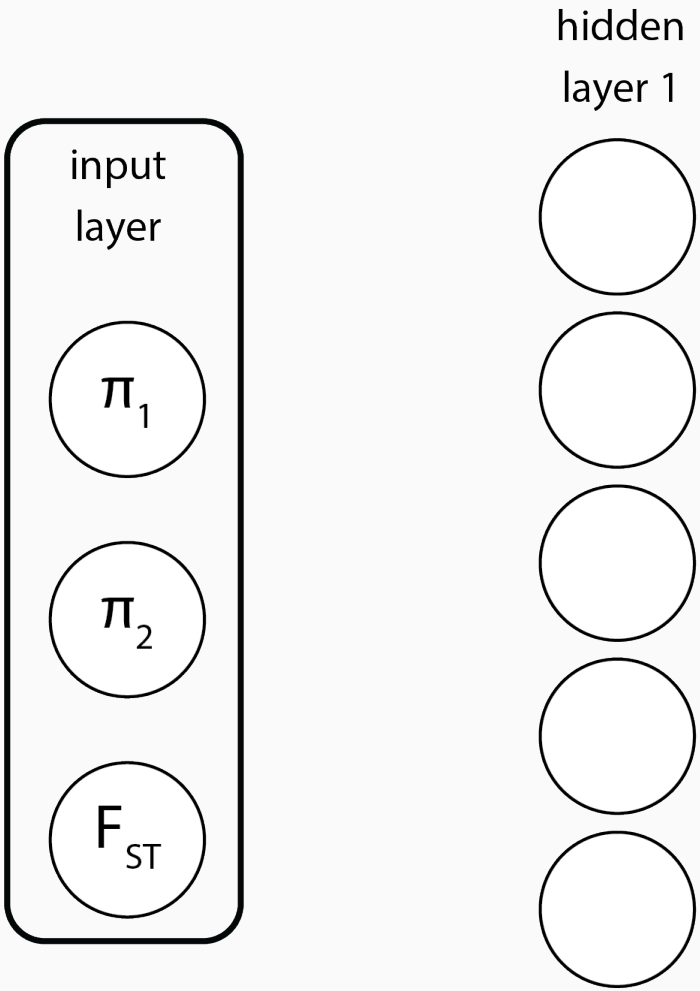
What are neural networks?



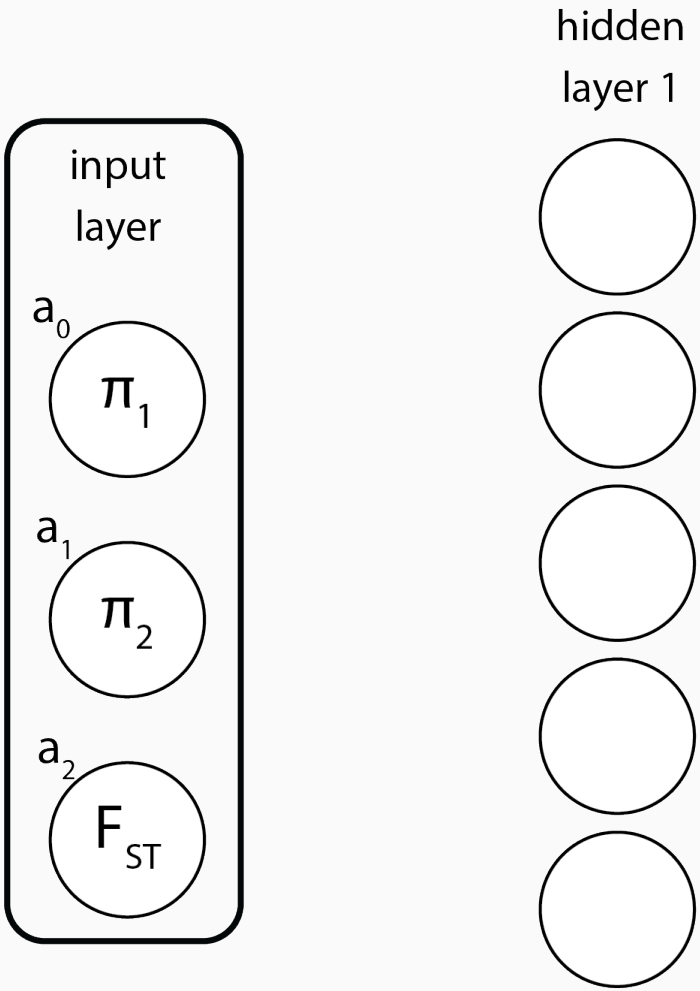
How do neural networks work?



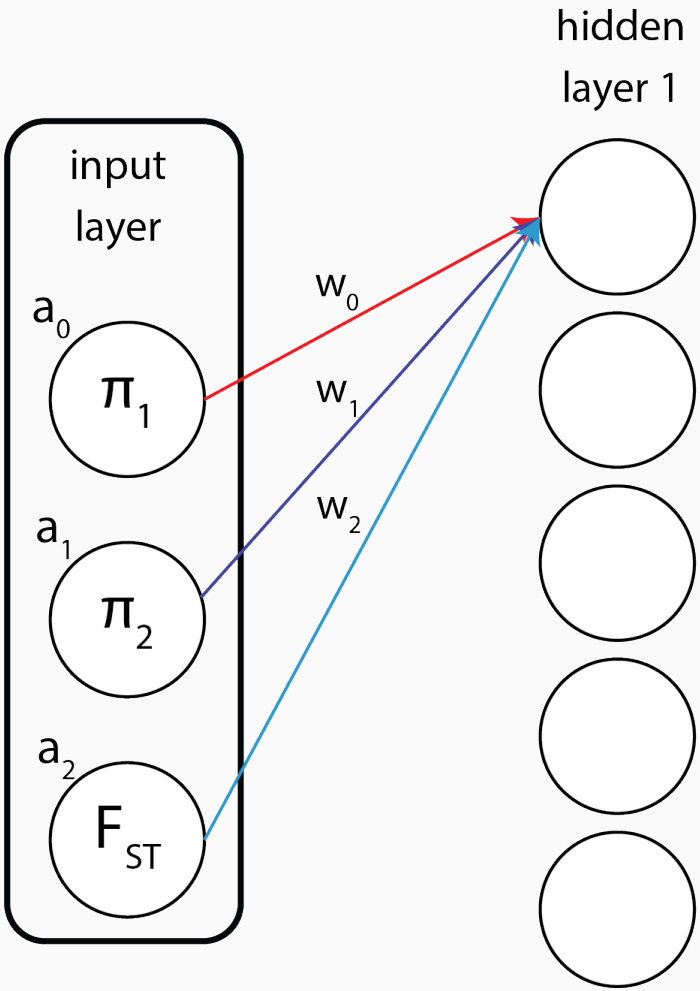
How do neural networks work?



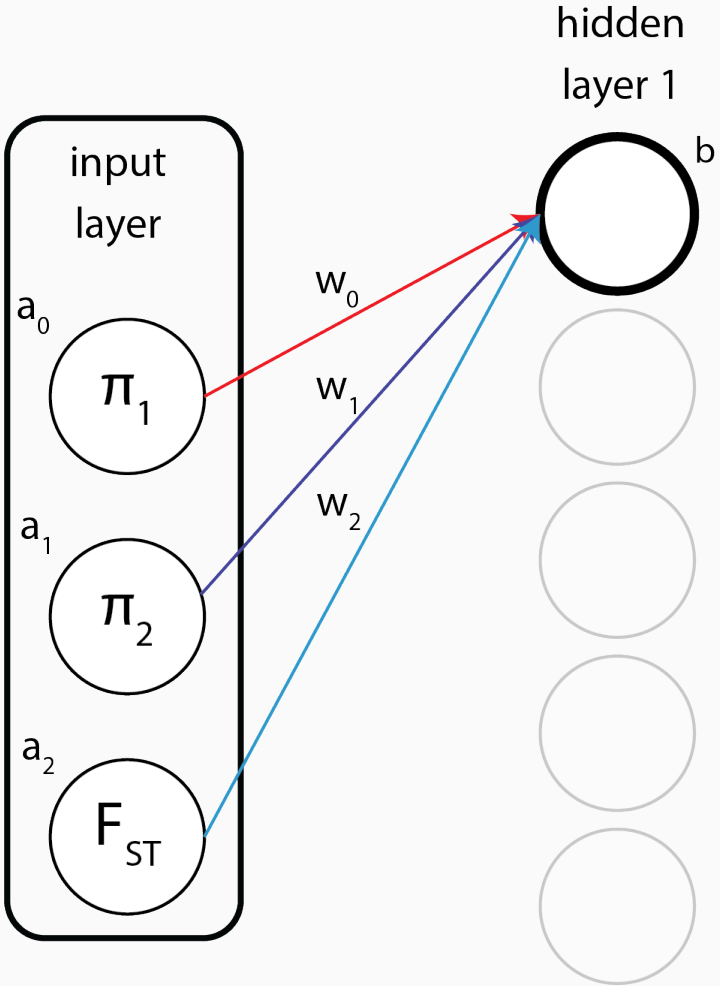
How do neural networks work?



How do neural networks work?



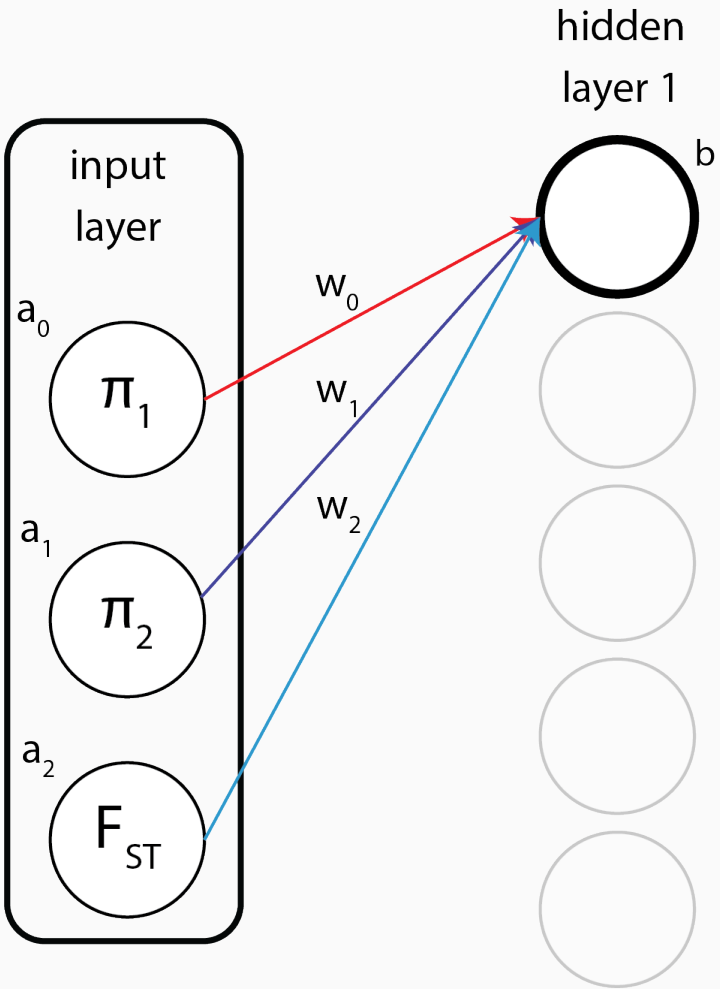
How do neural networks work?



$$= a_0 w_0 + a_1 w_1 + a_2 w_2$$

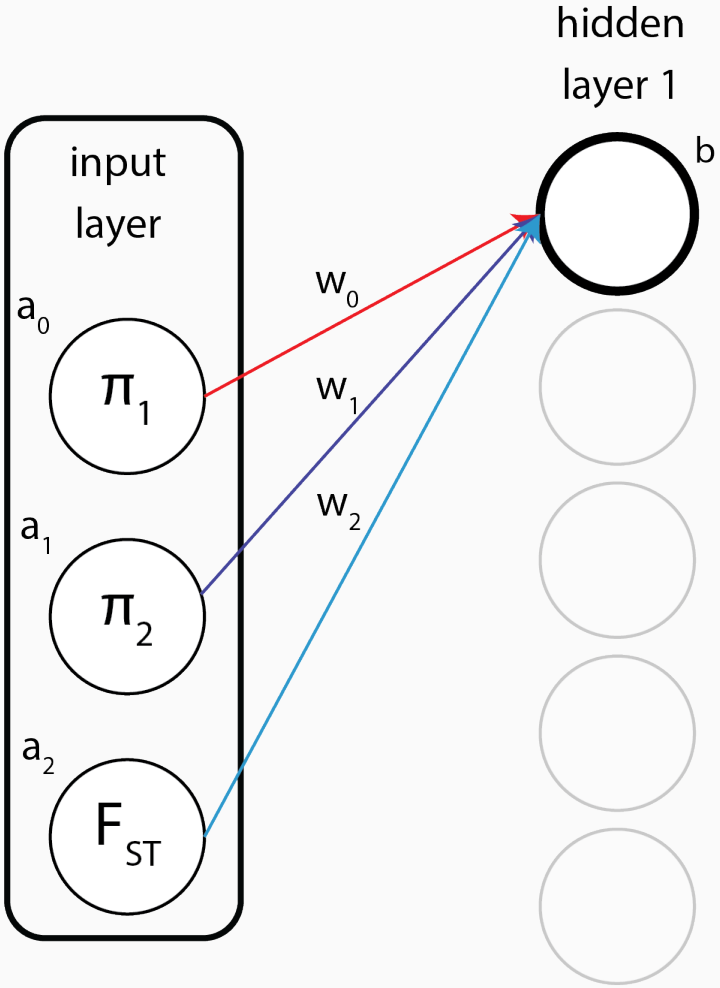
$$= \sum_i a_i w_i$$

How do neural networks work?



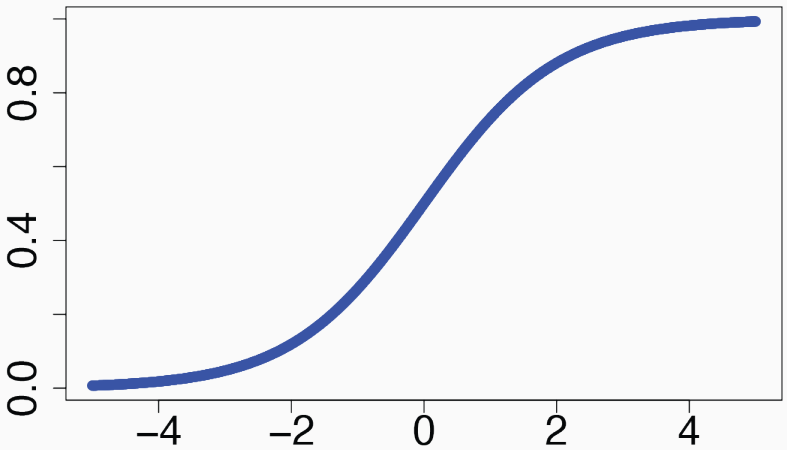
$$= \sum_i a_i w_i + b$$

How do neural networks work?



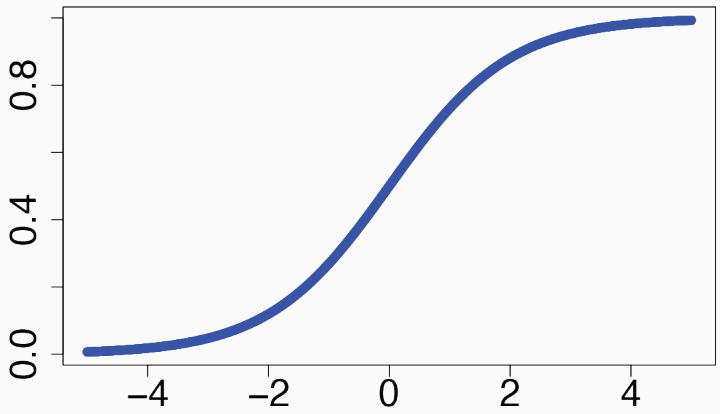
$$= \sum_i a_i w_i + b$$

Sigmoid activation function

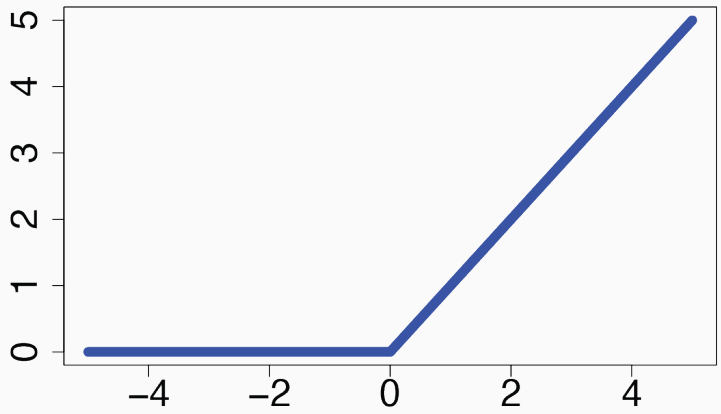


What are activation functions?

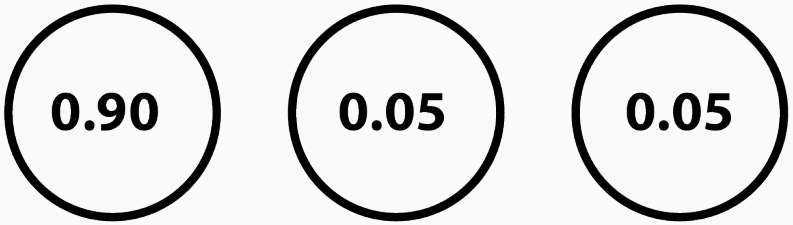
Sigmoid



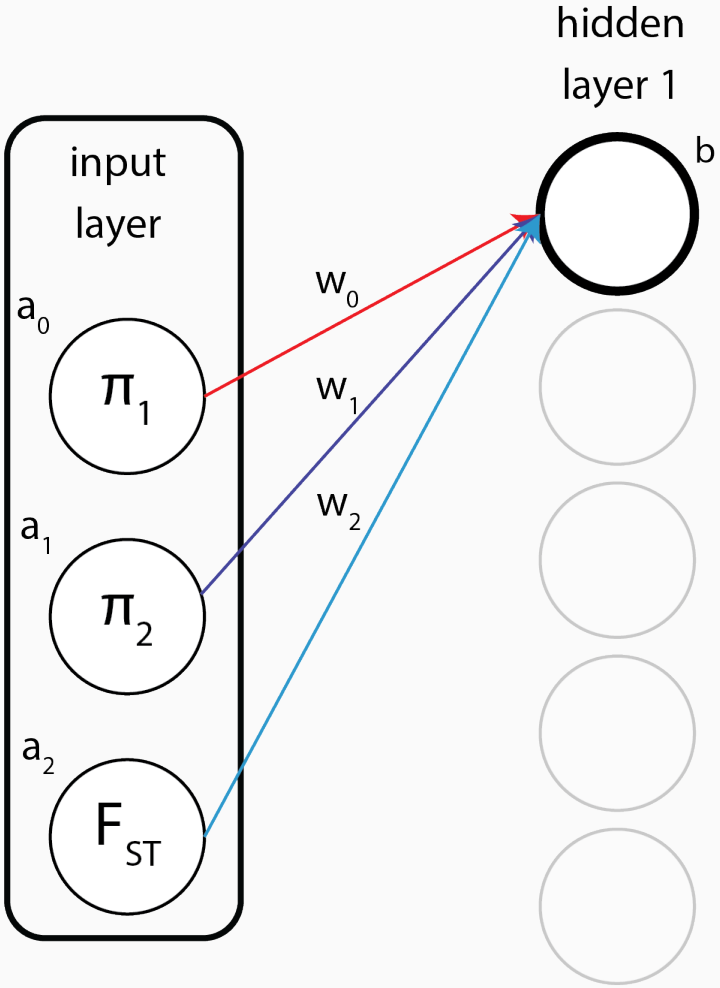
ReLU



Softmax

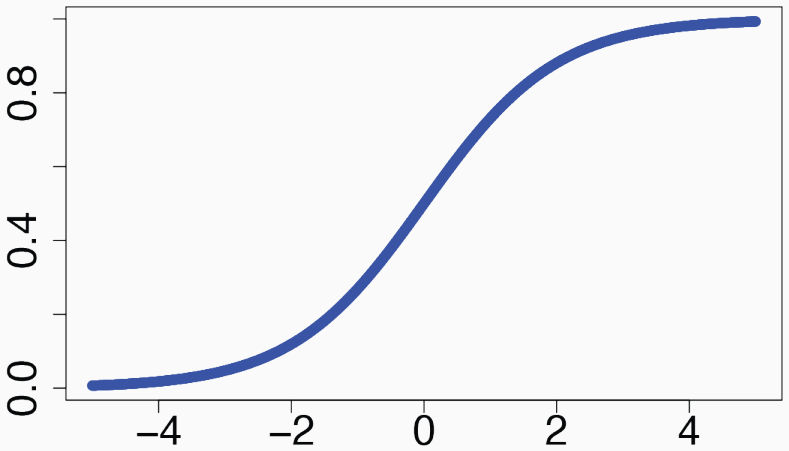


How do neural networks work?

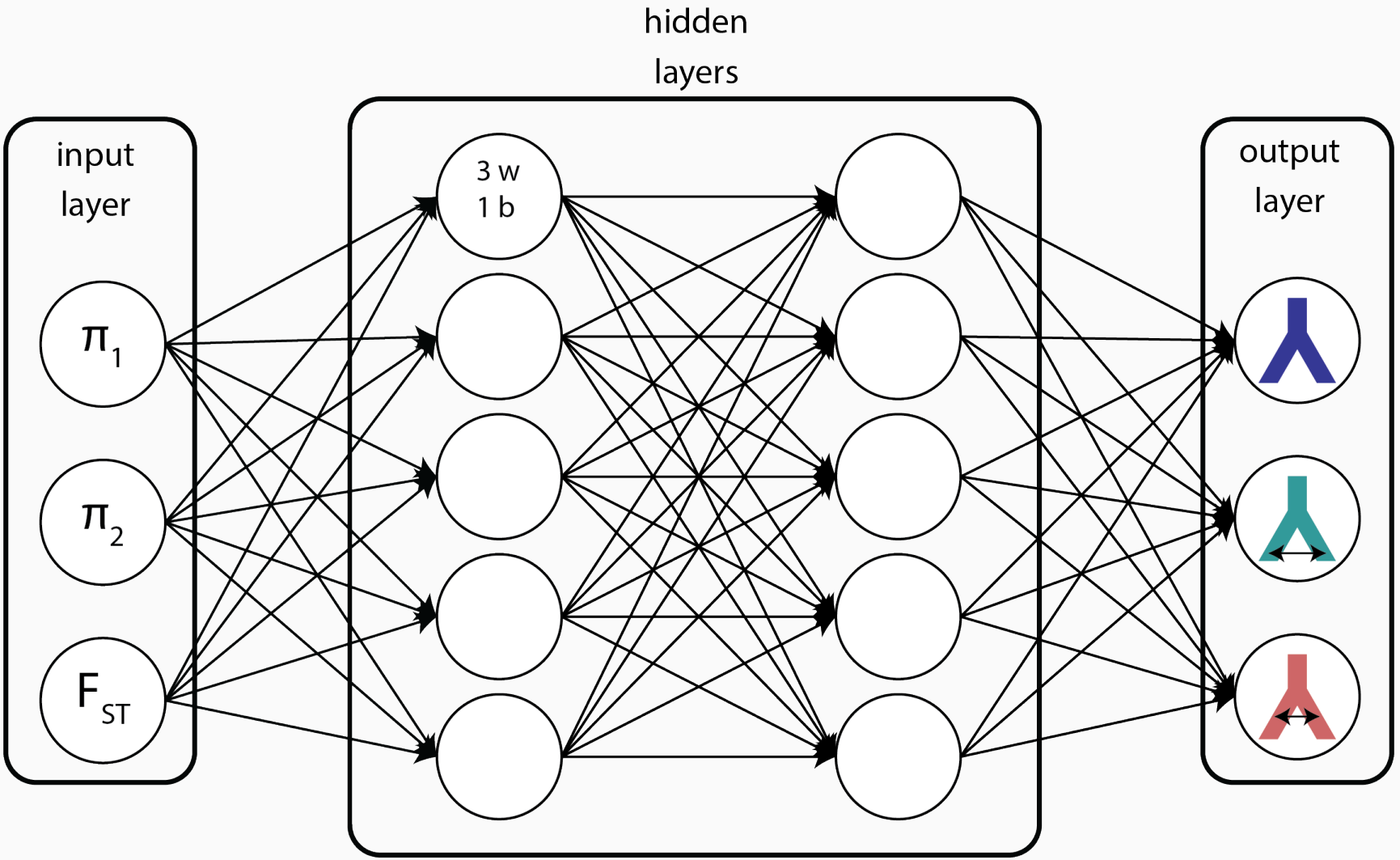


$$= \sum_i a_i w_i + b$$

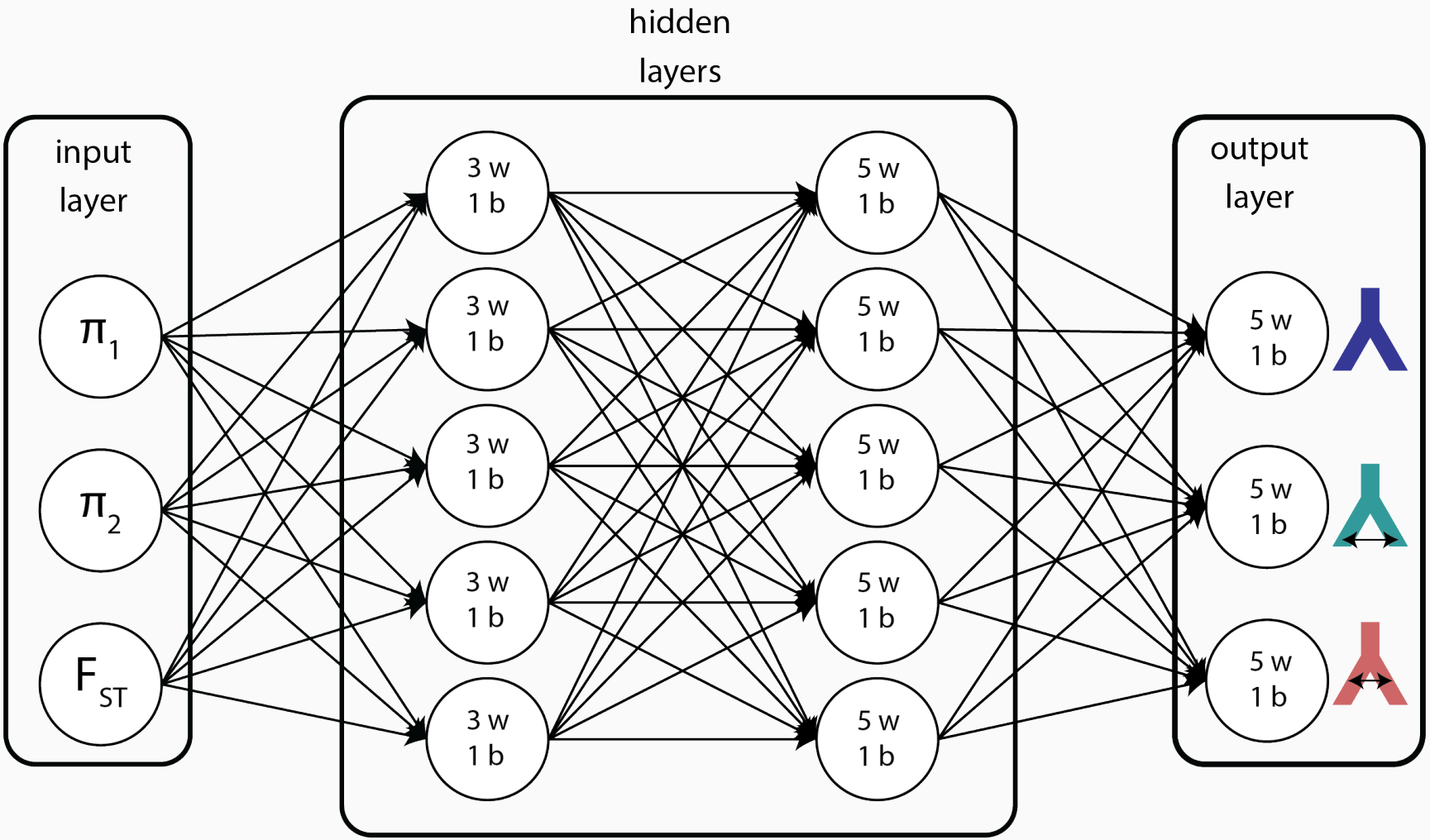
Sigmoid activation function



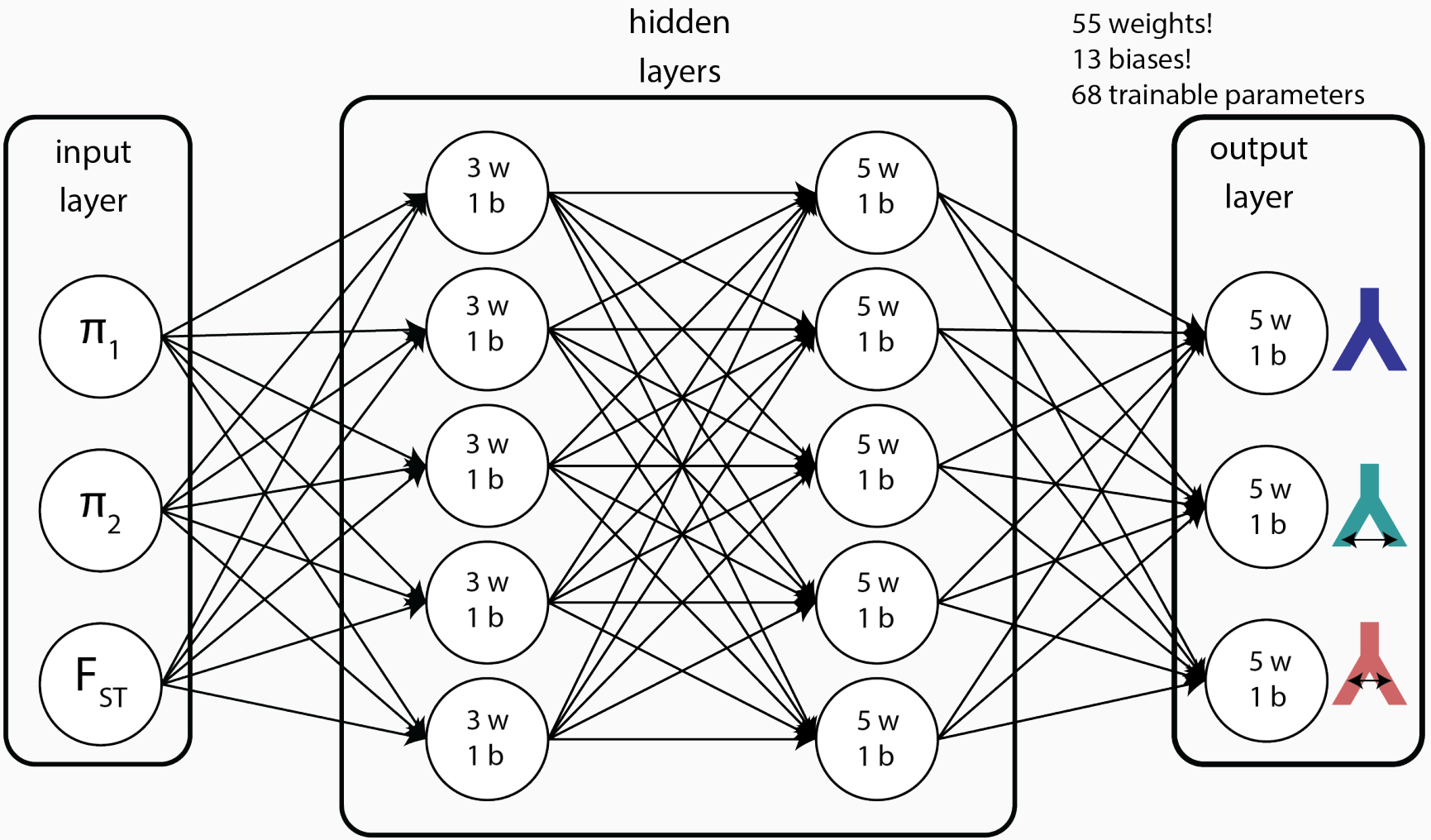
How do neural networks work?



How do neural networks work?



How do neural networks work?



How do neural networks learn?

- We need to learn the weights and biases.
- To do this we use **backpropogation**.
- Backpropogation uses gradient descent (calculus) to find the values of the weights and biases that minimize prediction error!

How do we train neural networks?

1. Initialize the weighs and biases of the network.
2. For each epoch *i*:
    - Shuffle the training data and divide it into minibatches.
    - For each minibatch *j*:
        - Pass the minibatch through the neural network.
        - Calculate the loss (i.e., some measure of prediction error.)
        - Update weights and biases to improve prediction accuracy.
3. Evaluate prediction accuracy on an independent dataset.

How do we train neural networks?

- **epochs**: the number of training iterations
- **minibatches**: subsets of the training data used during training
- **learning rate**: controls how quickly weights and biases are updated

How do we build neural networks?

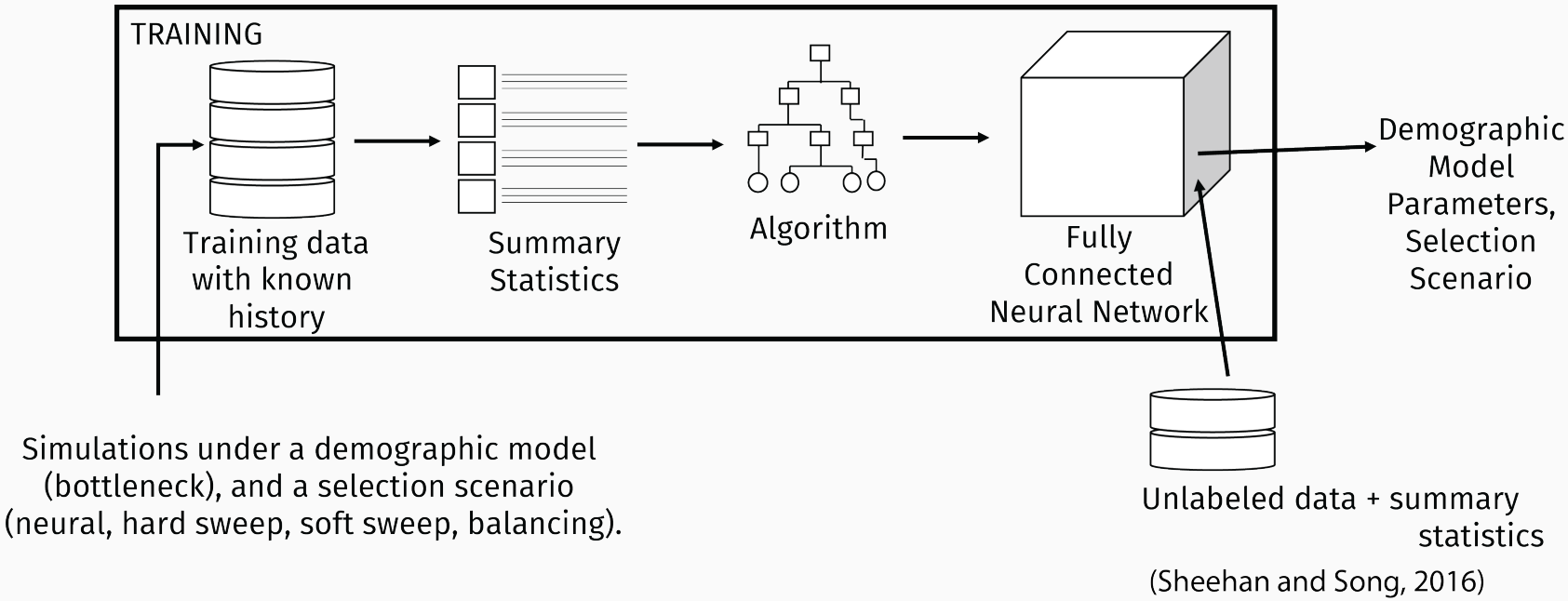
- We decide how to structure our input data.
- We decide what we want to output.
- We decide how many **hidden layers** to include, how many **neurons** to include in each layer, and which **activation functions** to use.
- We decide how many **epochs** to use, how many **mini-batches** to use, and select a **learning rate.**
- The number of hidden layers, the numbers of neurons, the choice of activation functions, the numbers of epochs and mini-batches, and the learning rate are all referred to as **hyperparameters** of our network, and we usually try to optimize these!

How do we build evaluate neural networks?

- When training a neural network, we hold out some percentage of our data as **validation data**.
- The **validation data** helps us to assess whether our model is only memorizing the **training data**, or whether it will be able to generalize.
- Since we may use error on this validation set to manually (or automatically) tune our hypterparameters, it is important to also hold out some of our data as an independent **test data** set, which should only be used to evaluate performance after all training, tuning, and adjustments are complete!

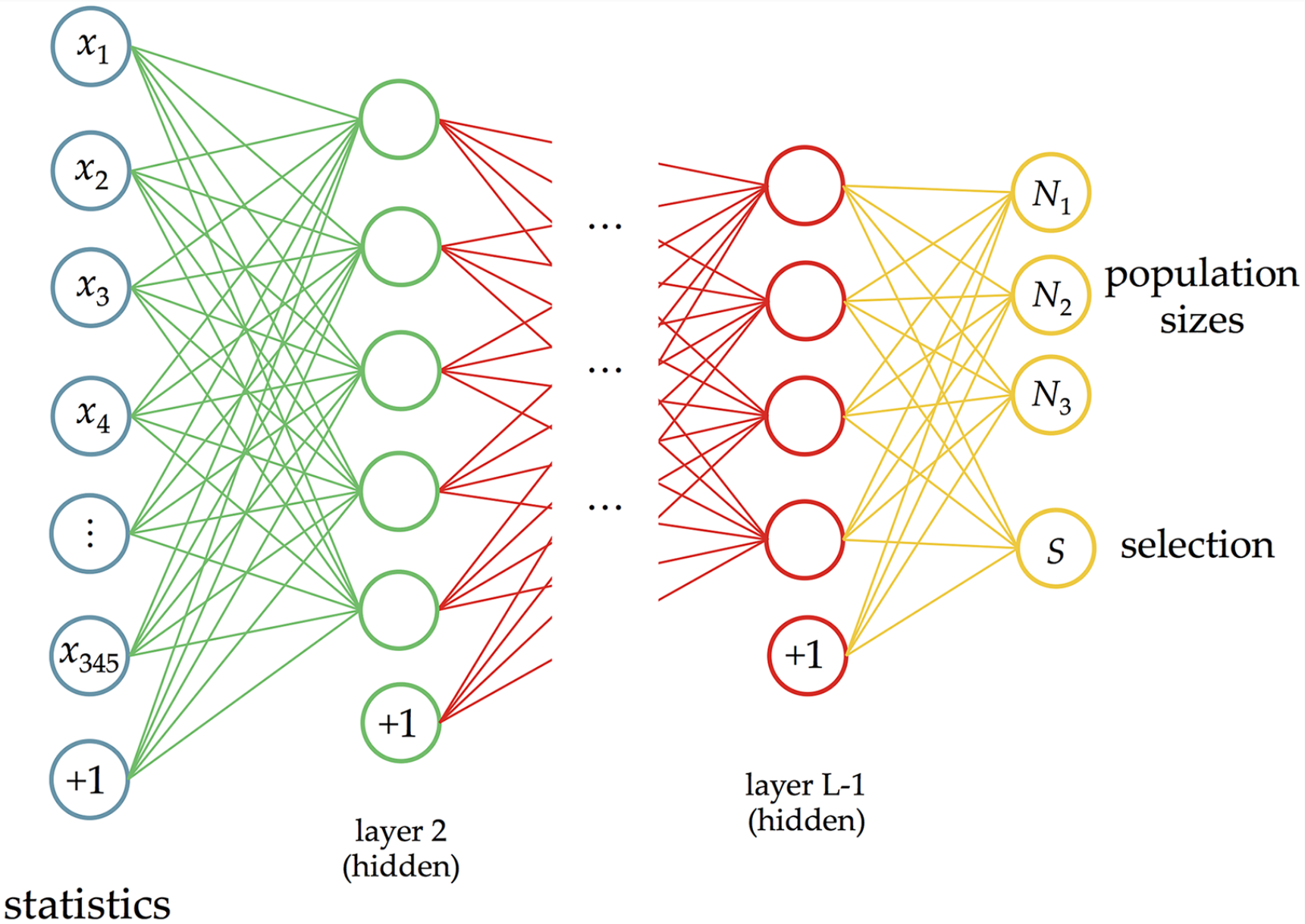
Examples: evoNet

Goal: to estimate historical population sizes and detect selection.



(Sheehan and Song, 2016)

Examples: evoNet

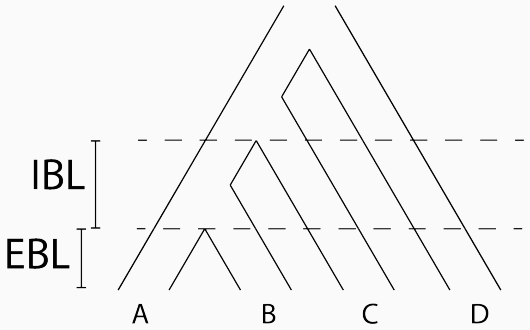


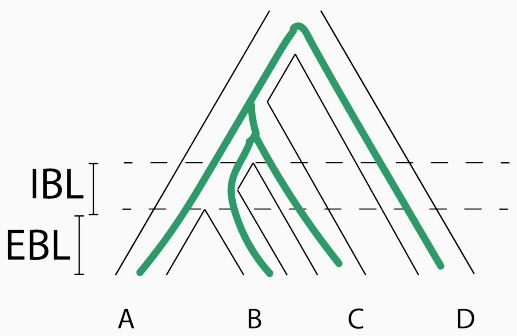
Examples: evoNet

- evoNet predicts population sizes through time and a selection history.
- To do this, evoNet includes two rather different outputs!
- evoNet uses a large number of summary statistics.
- evoNet estimates selection class more easily than population sizes.

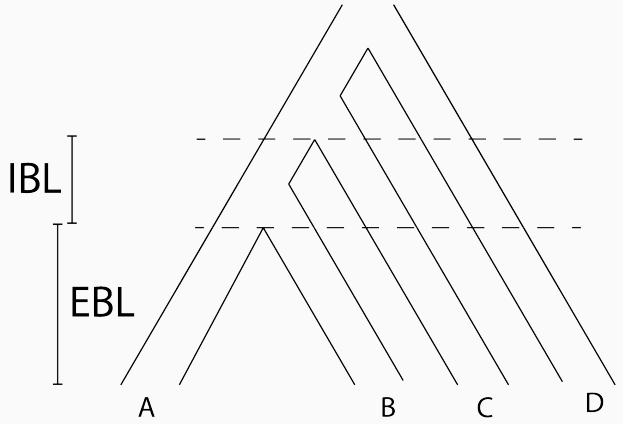
Examples: ml4ils

Goal: to estimate the frequency of concordant quartet topologies from a set of alignments and gene trees.





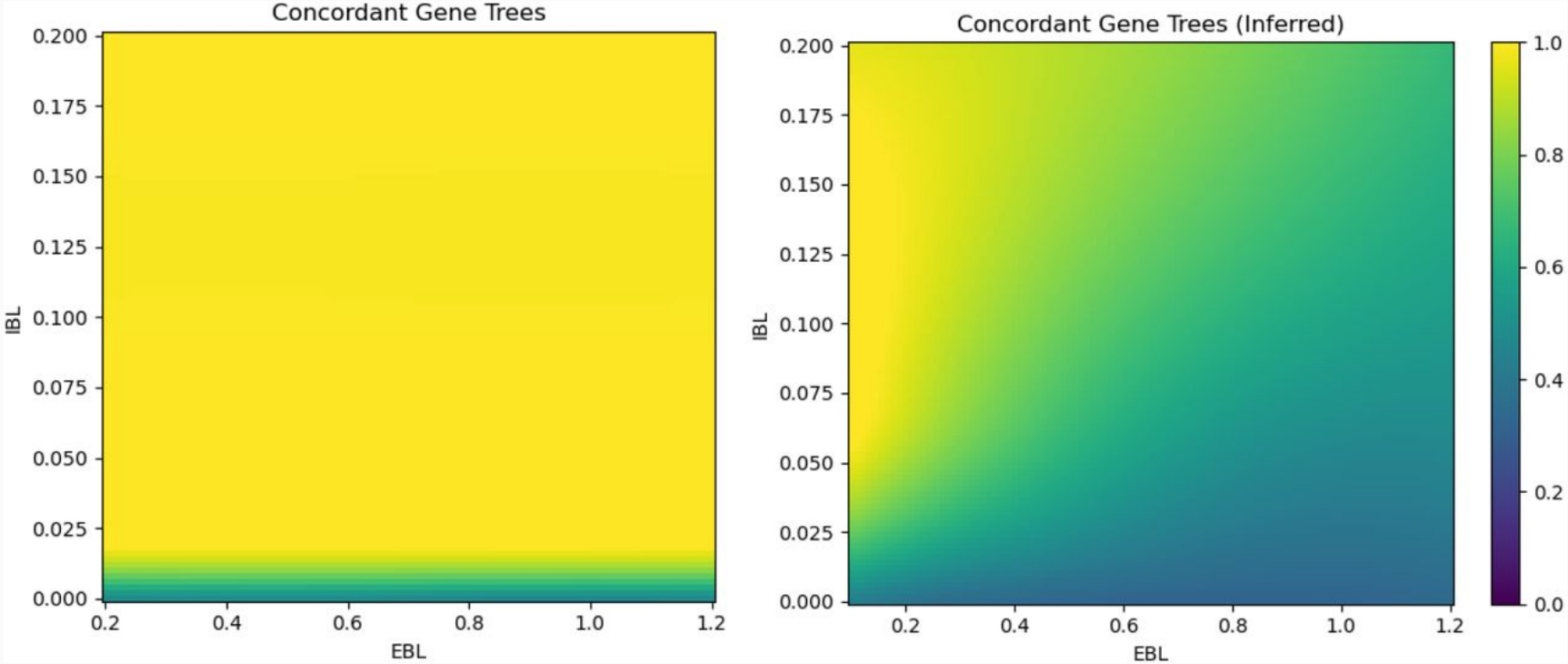
shorter IBL = more discordance due to ILS



longer EBL = more discordance due to gene tree estimation error

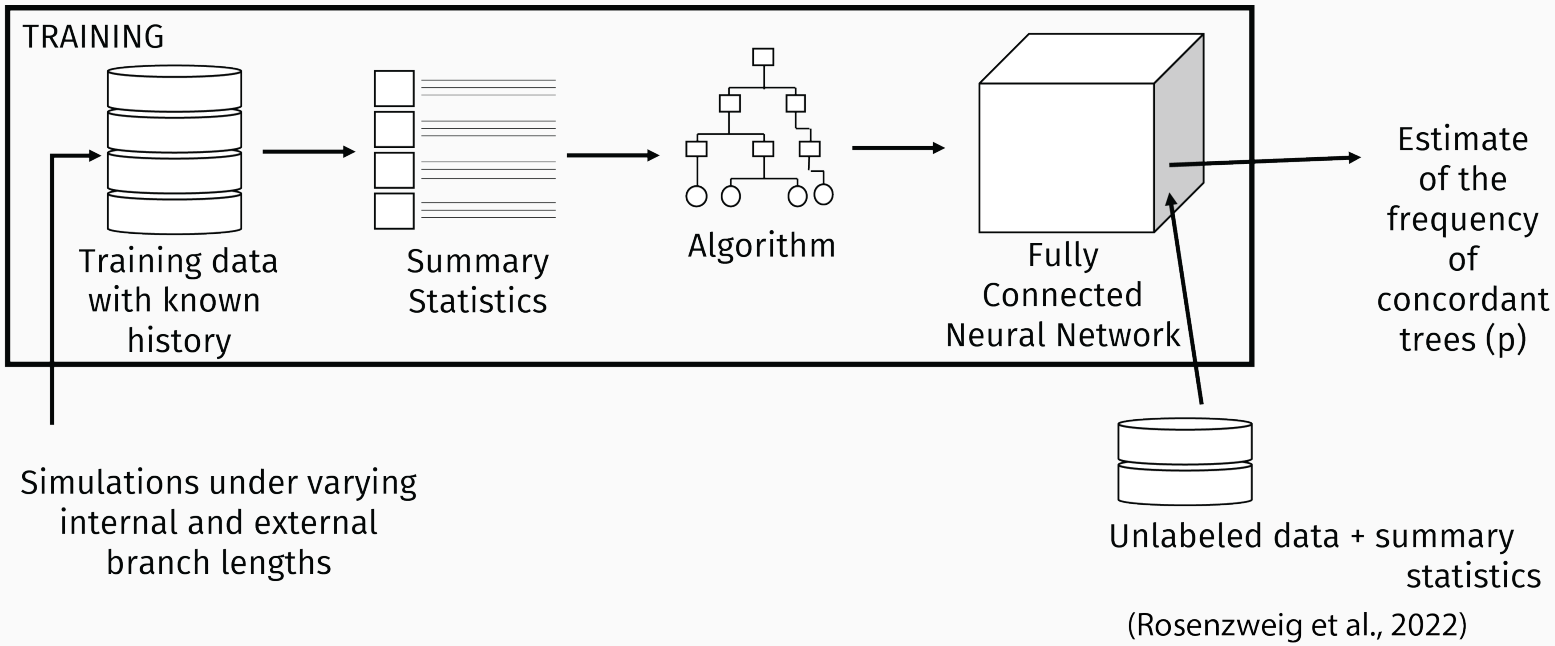
Examples: ml4ils

Goal: to estimate the frequency of concordant topologies from a set of alignments and gene trees.



Examples: ml4ils

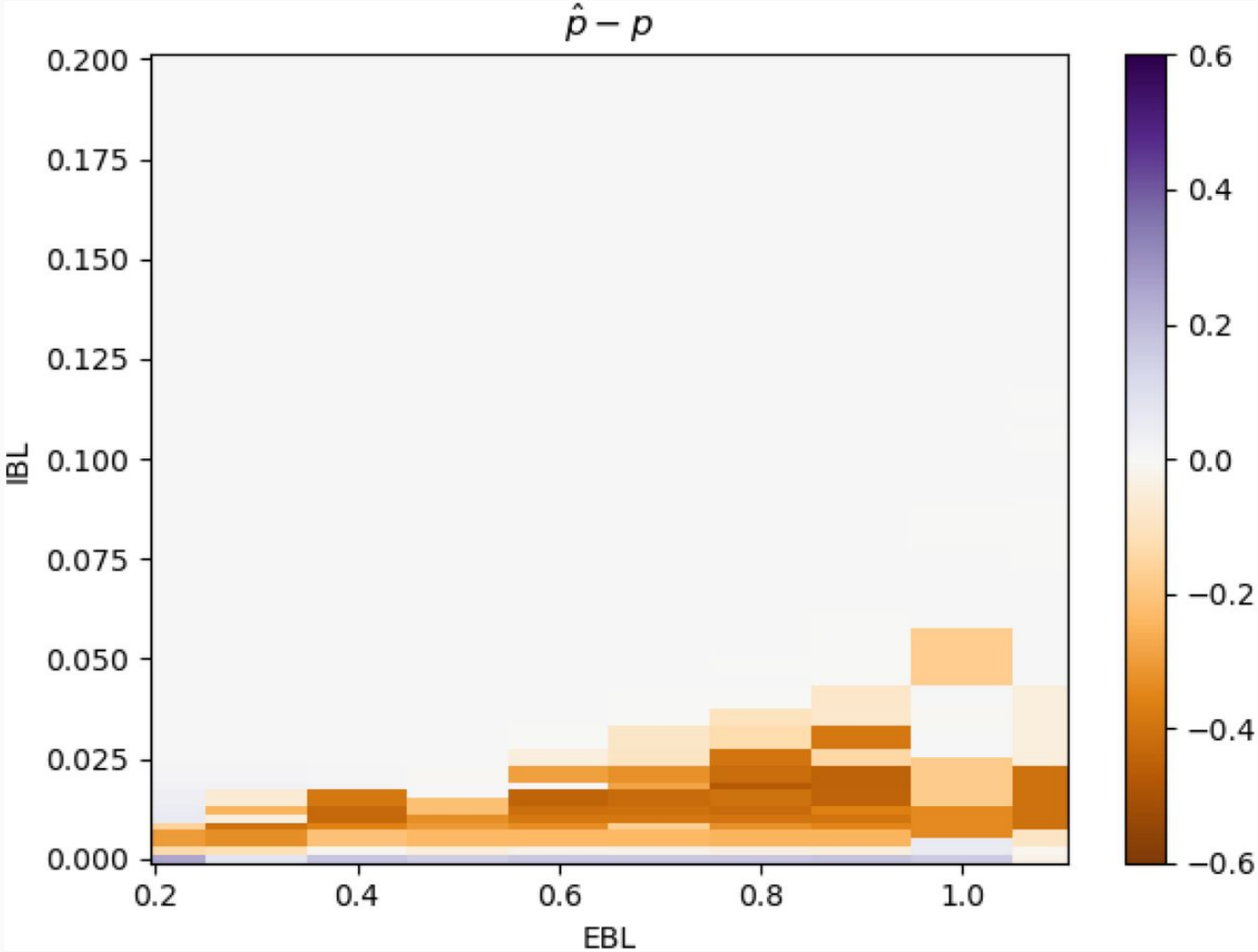
Goal: to estimate the frequency of concordant topologies from a set of alignments and gene trees.



Summary statistics: sCFs, patristic distances, etc.

Examples: ml4ils

Goal: to estimate the frequency of concordant topologies.

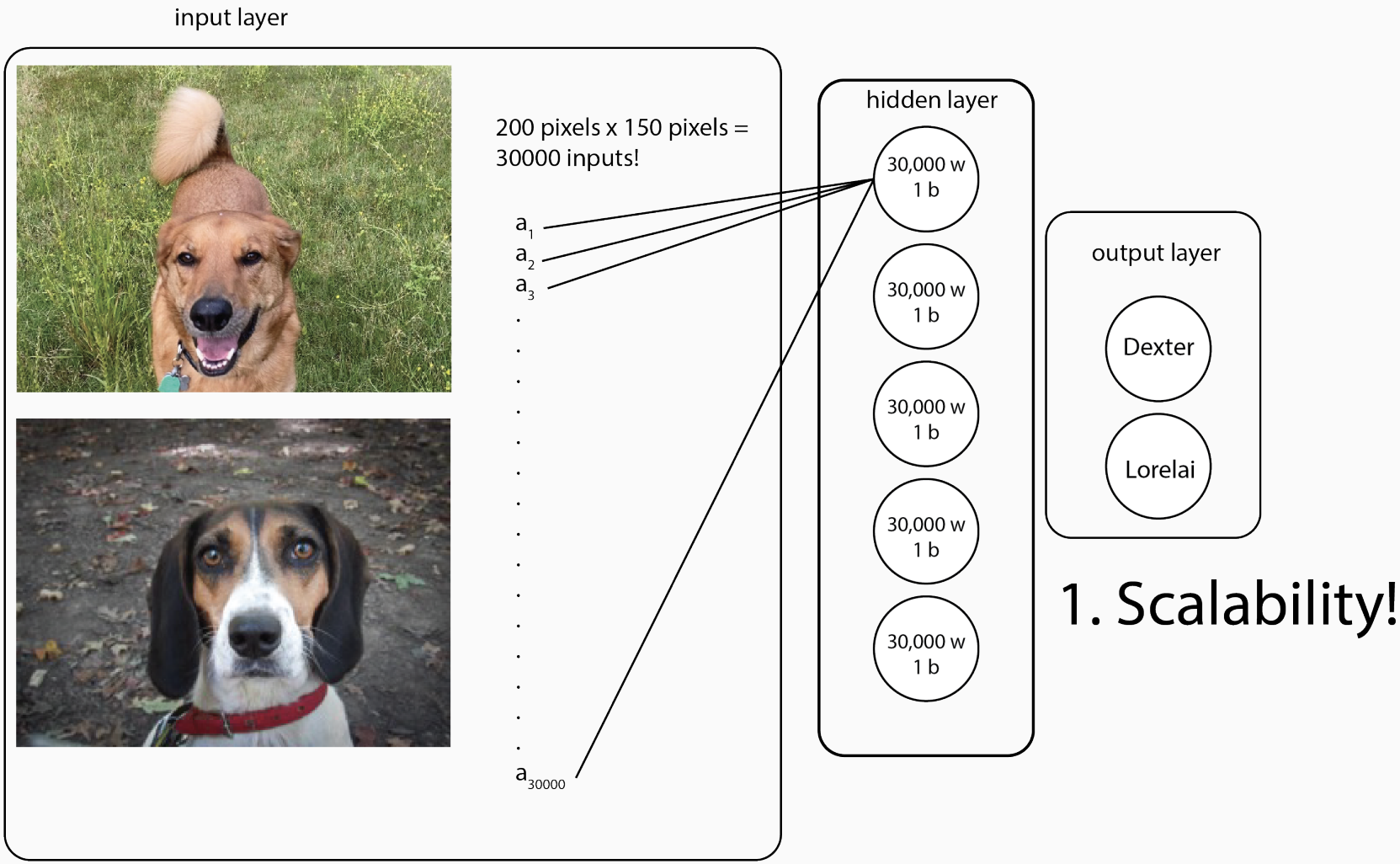


Fully Connected Neural Networks

- FCNNs are composed of a set of dependent non-linear functions.
- FCNNs learn weights and biases that minimize prediction error.
- We can adjust hyperparameters to build FCNNs that perform well.
- FCNNs rely on some set of features (i.e., summary statistics) to learn the connection between input and output.
- We saw examples using an FCNN to detect selection, estimate demographic parameters, and estimate the probability of concordance! (And there are many more!!)

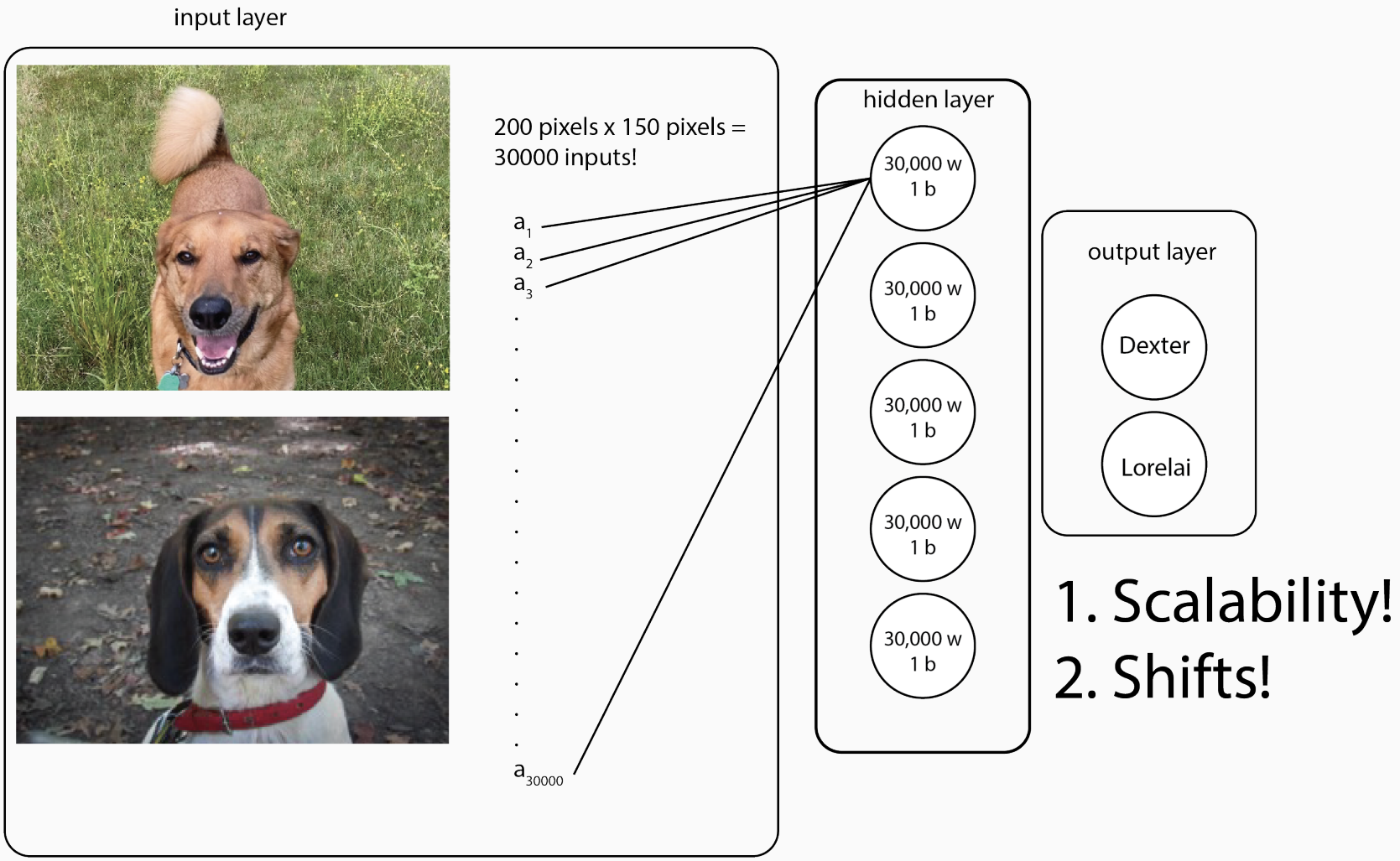
Outline

What if I want to classify images?



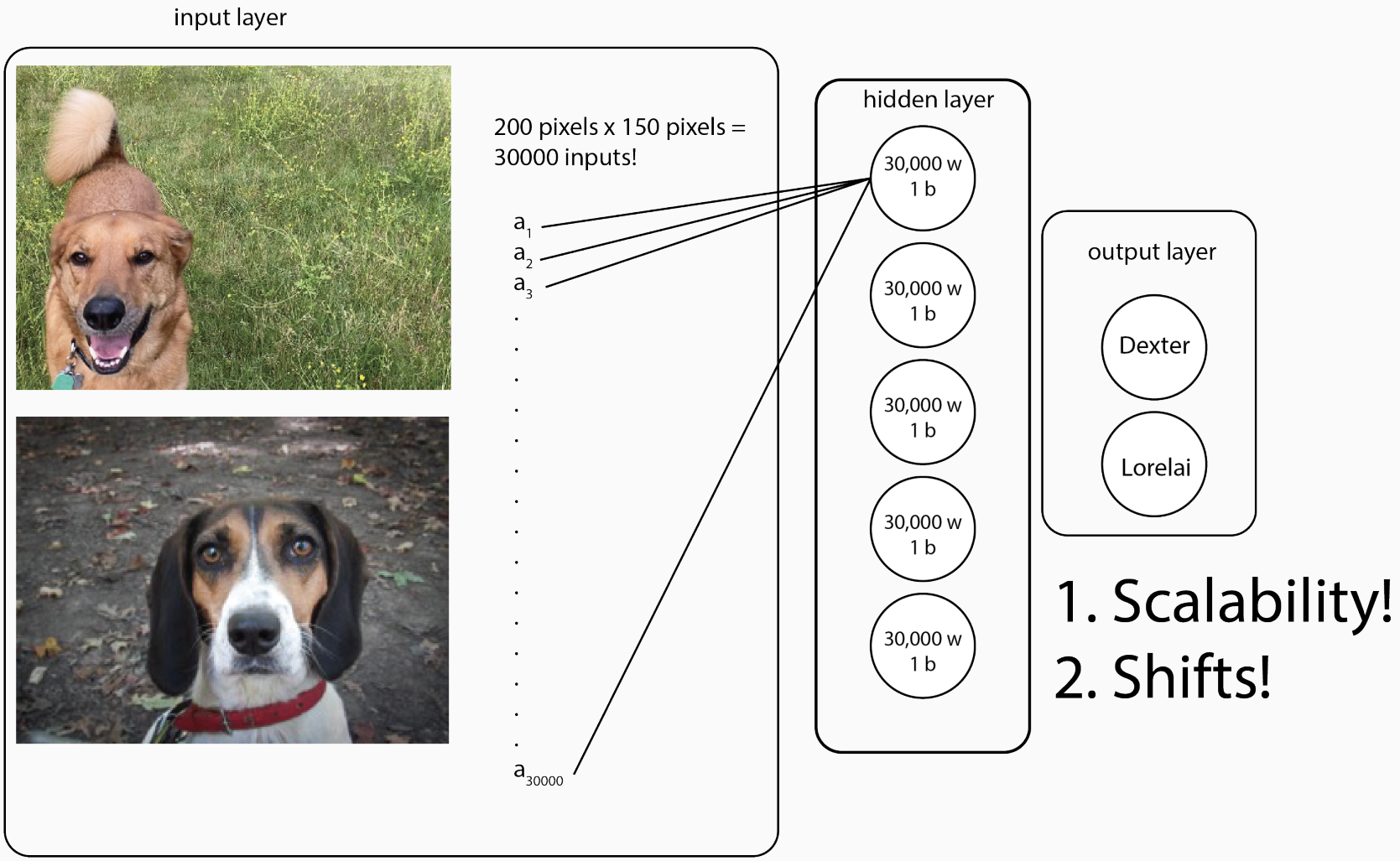
1. Scalability!

What if I want to classify images?



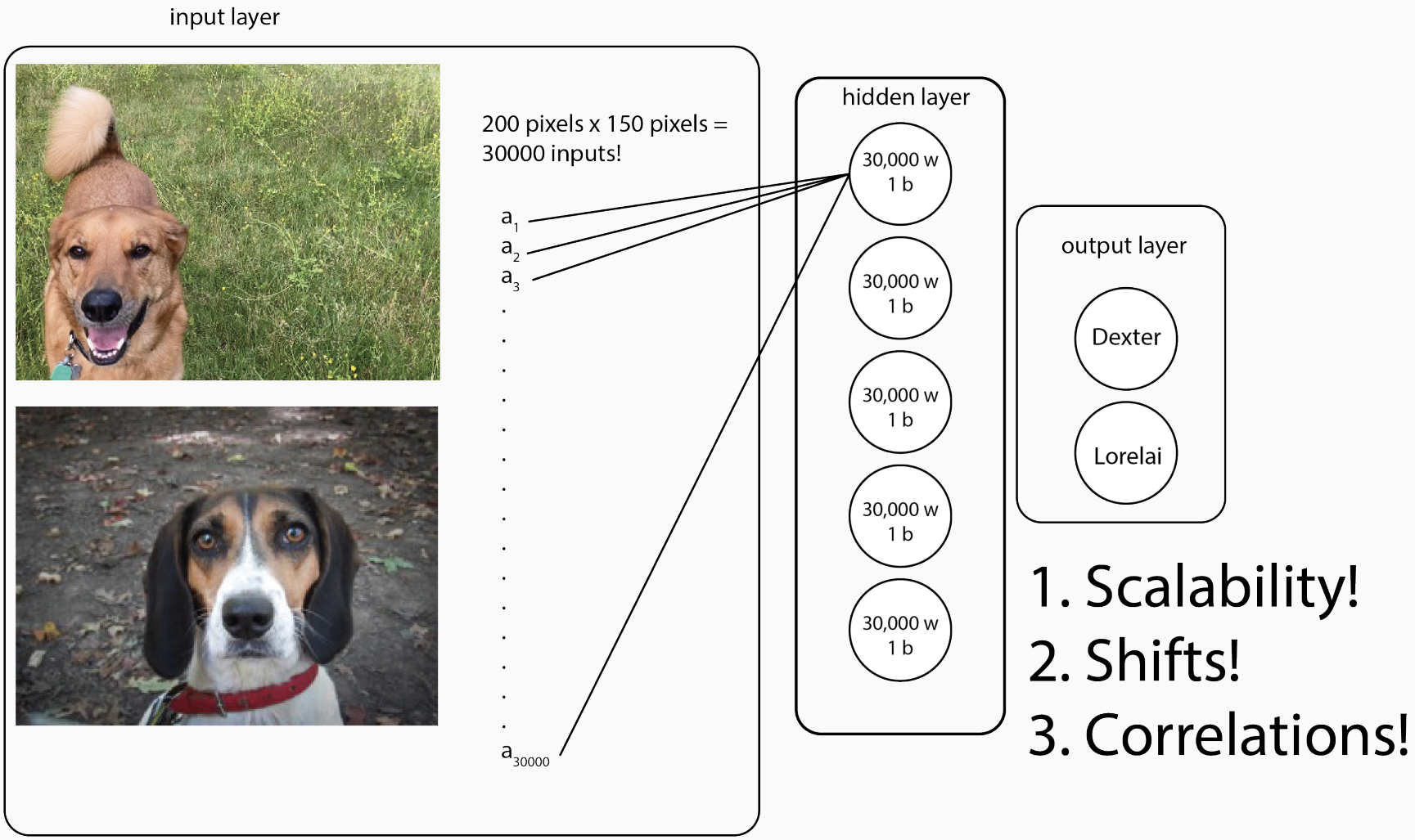
1. Scalability!
2. Shifts!

What if I want to classify images?



1. Scalability!
2. Shifts!

What if I want to classify images?

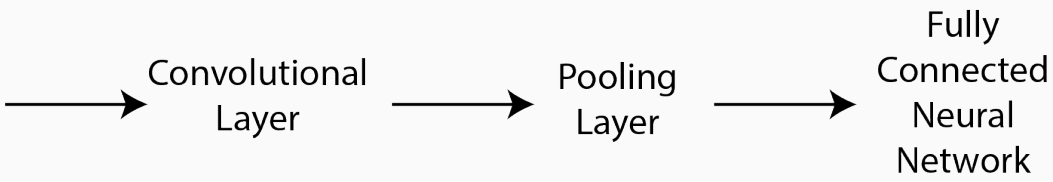


What if I want to classify images?

Convolutional Neural Networks (CNNs) are designed to perform well on input images by improving on fully connected layers in several ways, including:

1. Scalable
2. Robust to shifts
3. Take advantages of correlations between pixels

How do CNNs work?



How do CNNs work?

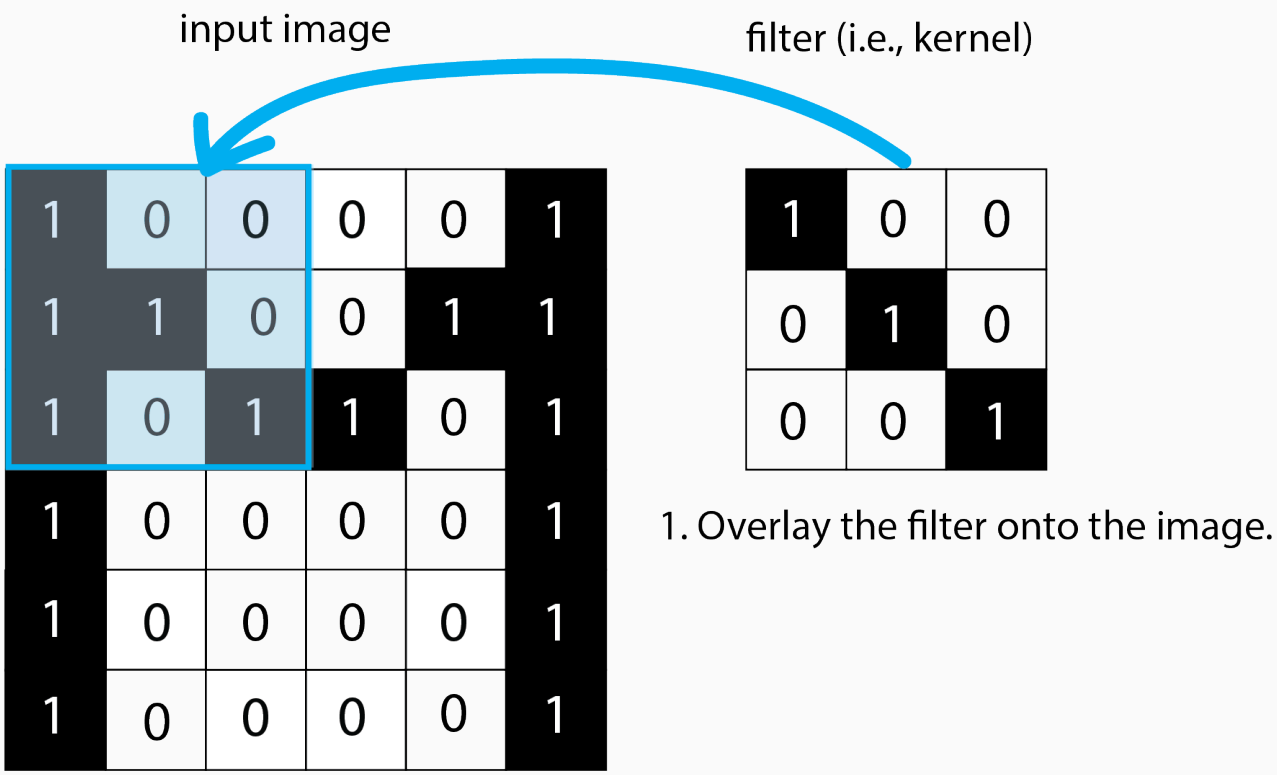
input image

1	0	0	0	0	1
1	1	0	0	1	1
1	0	1	1	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1

filter (i.e., kernel)

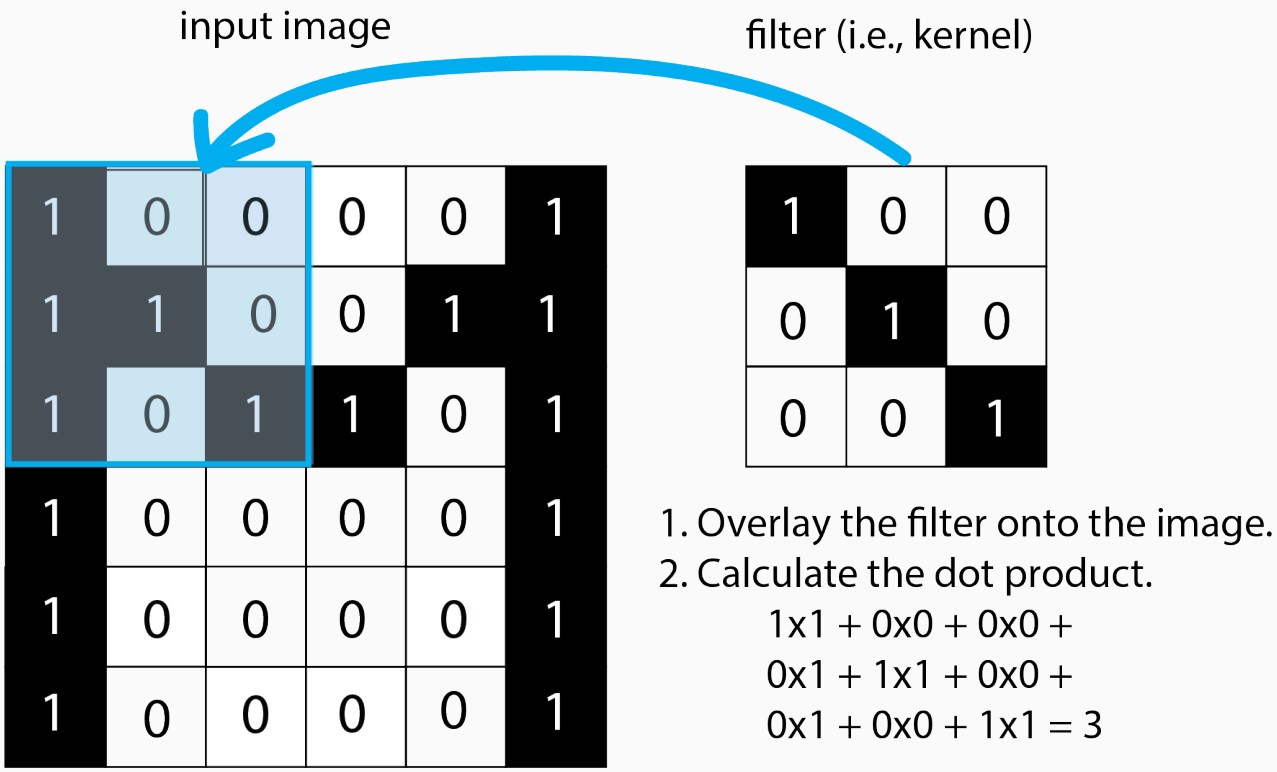
1	0	0
0	1	0
0	0	1

How do CNNs work?



1. Overlay the filter onto the image.

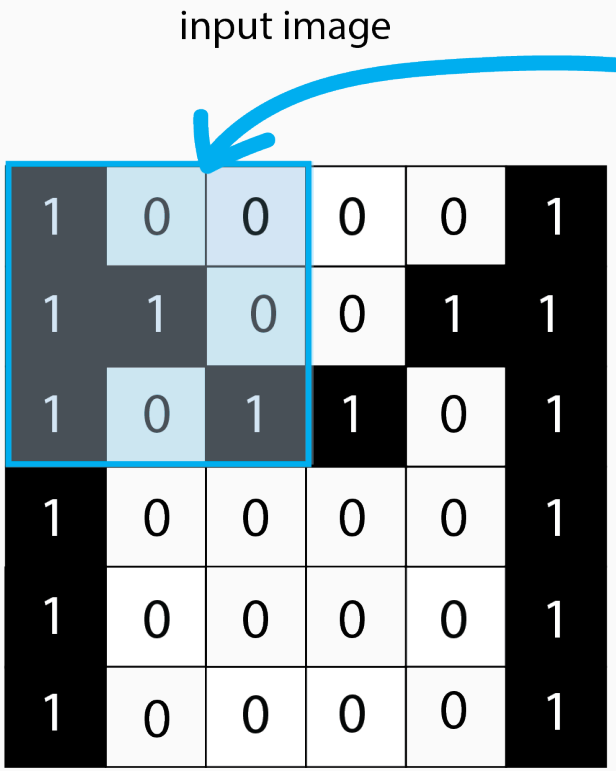
How do CNNs work?

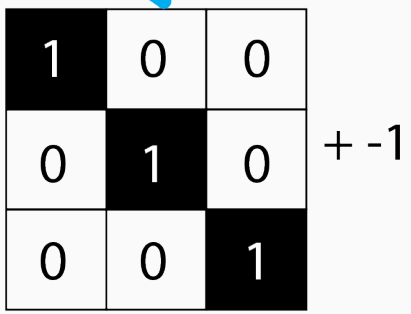


1. Overlay the filter onto the image.
2. Calculate the dot product.

$$1x1 + 0x0 + 0x0 +$$
$$0x1 + 1x1 + 0x0 +$$
$$0x1 + 0x0 + 1x1 = 3$$

How do CNNs work?

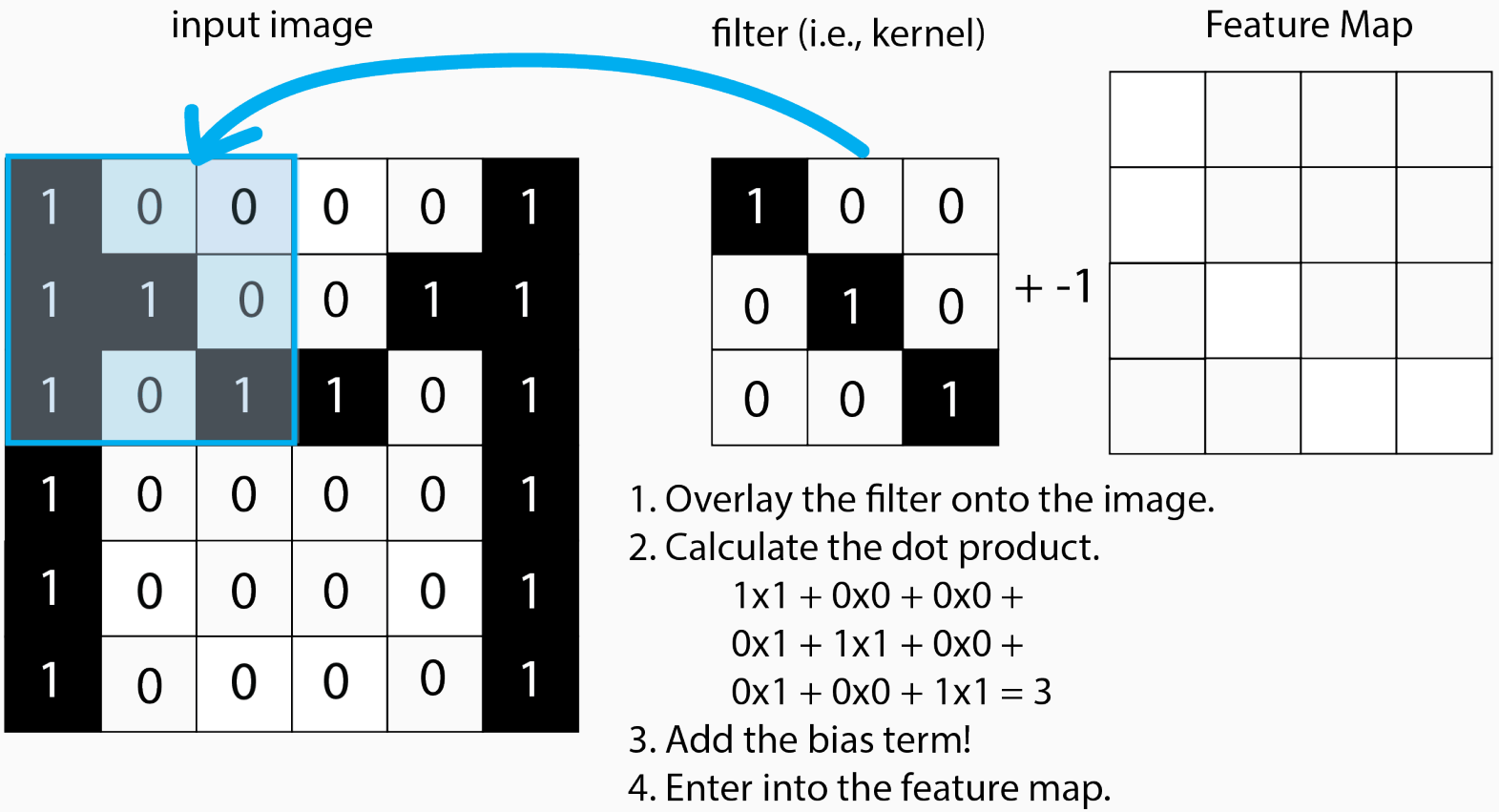




1. Overlay the filter onto the image.
2. Calculate the dot product.

   1x1 + 0x0 + 0x0 +
   0x1 + 1x1 + 0x0 +
   0x1 + 0x0 + 1x1 = 3
3. Add the bias term!

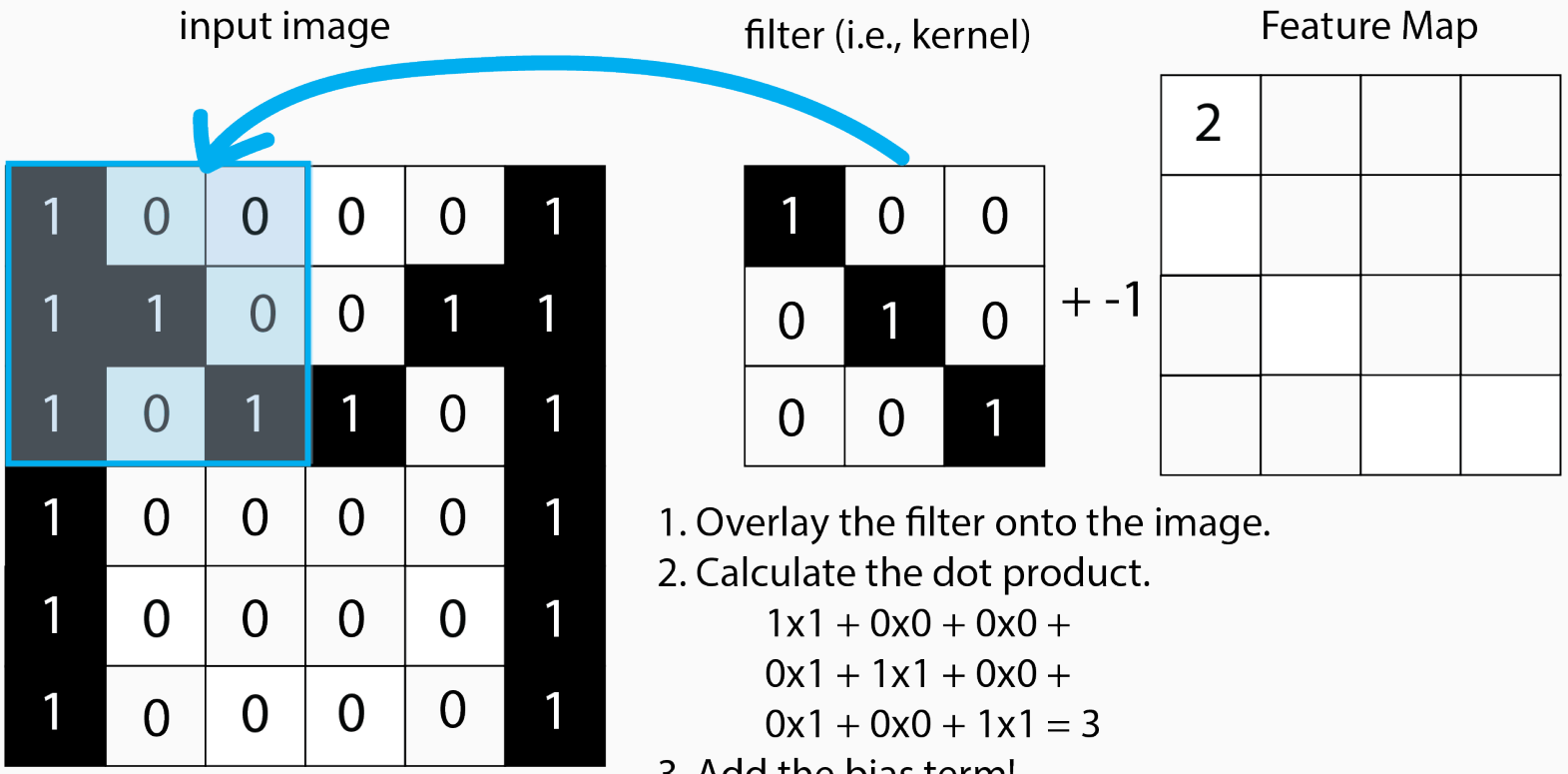
How do CNNs work?



1. Overlay the filter onto the image.
2. Calculate the dot product.

   1x1 + 0x0 + 0x0 +
   0x1 + 1x1 + 0x0 +
   0x1 + 0x0 + 1x1 = 3
3. Add the bias term!
4. Enter into the feature map.

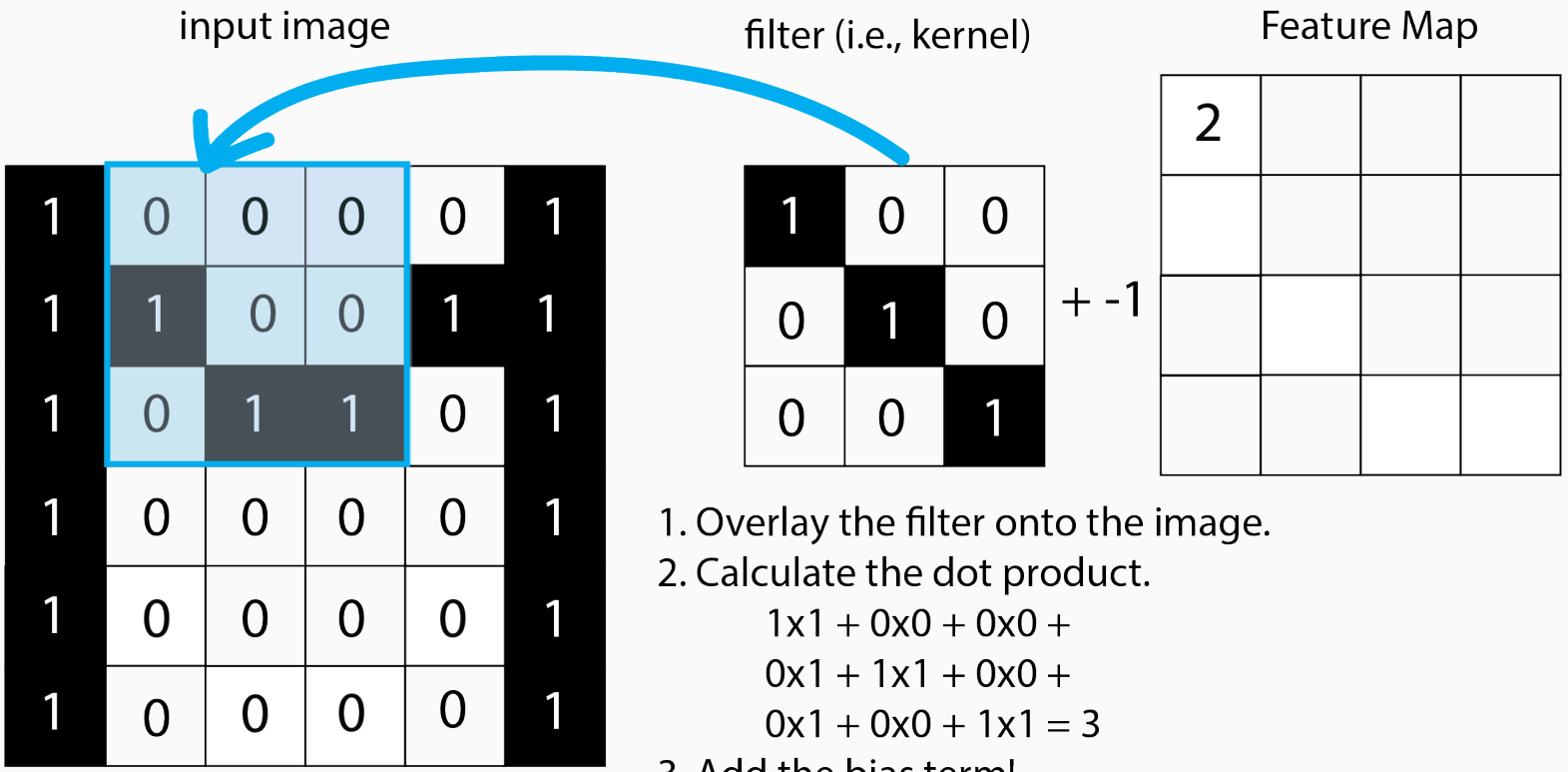
How do CNNs work?



1. Overlay the filter onto the image.
2. Calculate the dot product.

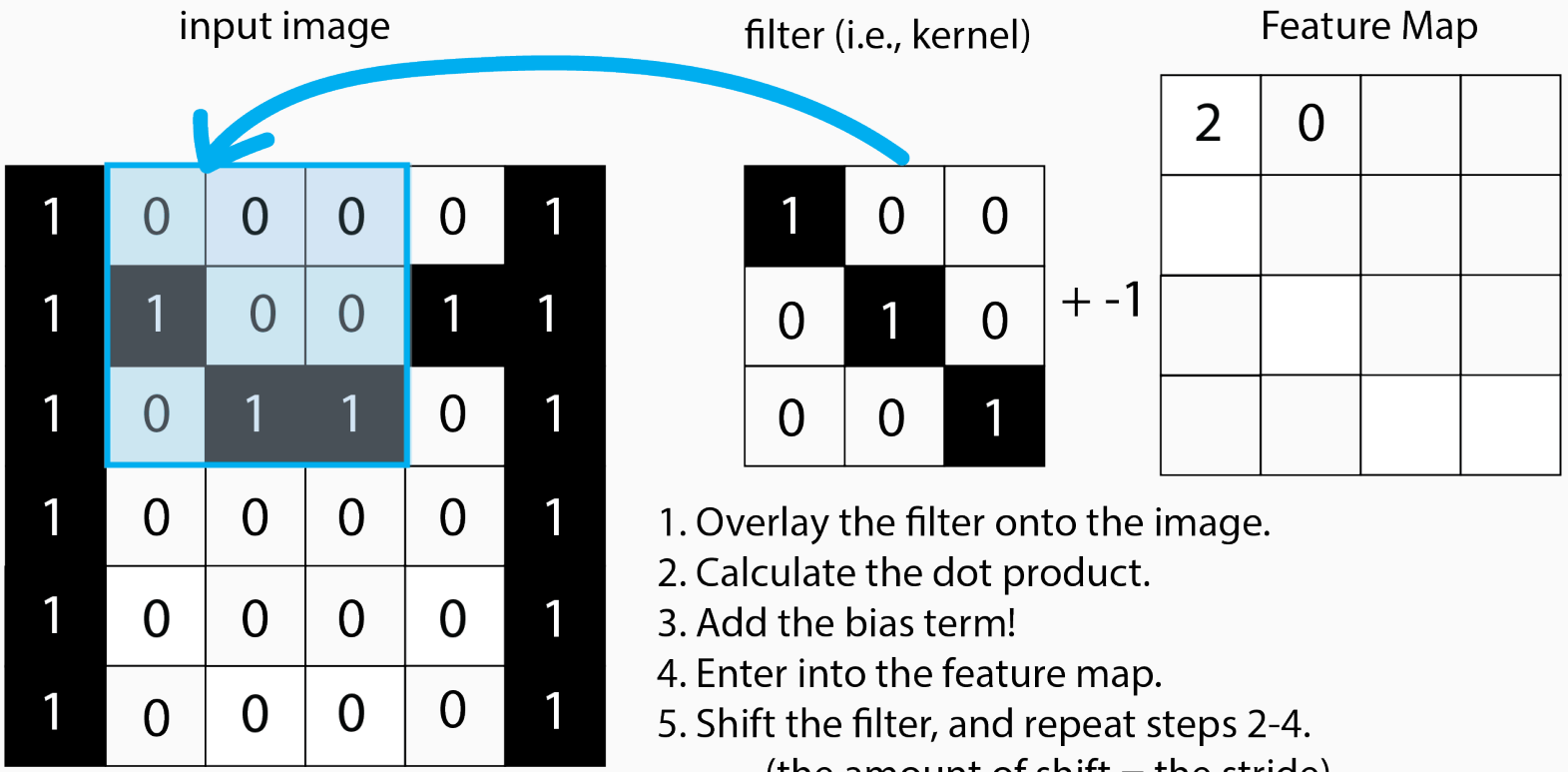
   1x1 + 0x0 + 0x0 +
   0x1 + 1x1 + 0x0 +
   0x1 + 0x0 + 1x1 = 3
3. Add the bias term!
4. Enter into the feature map.

How do CNNs work?



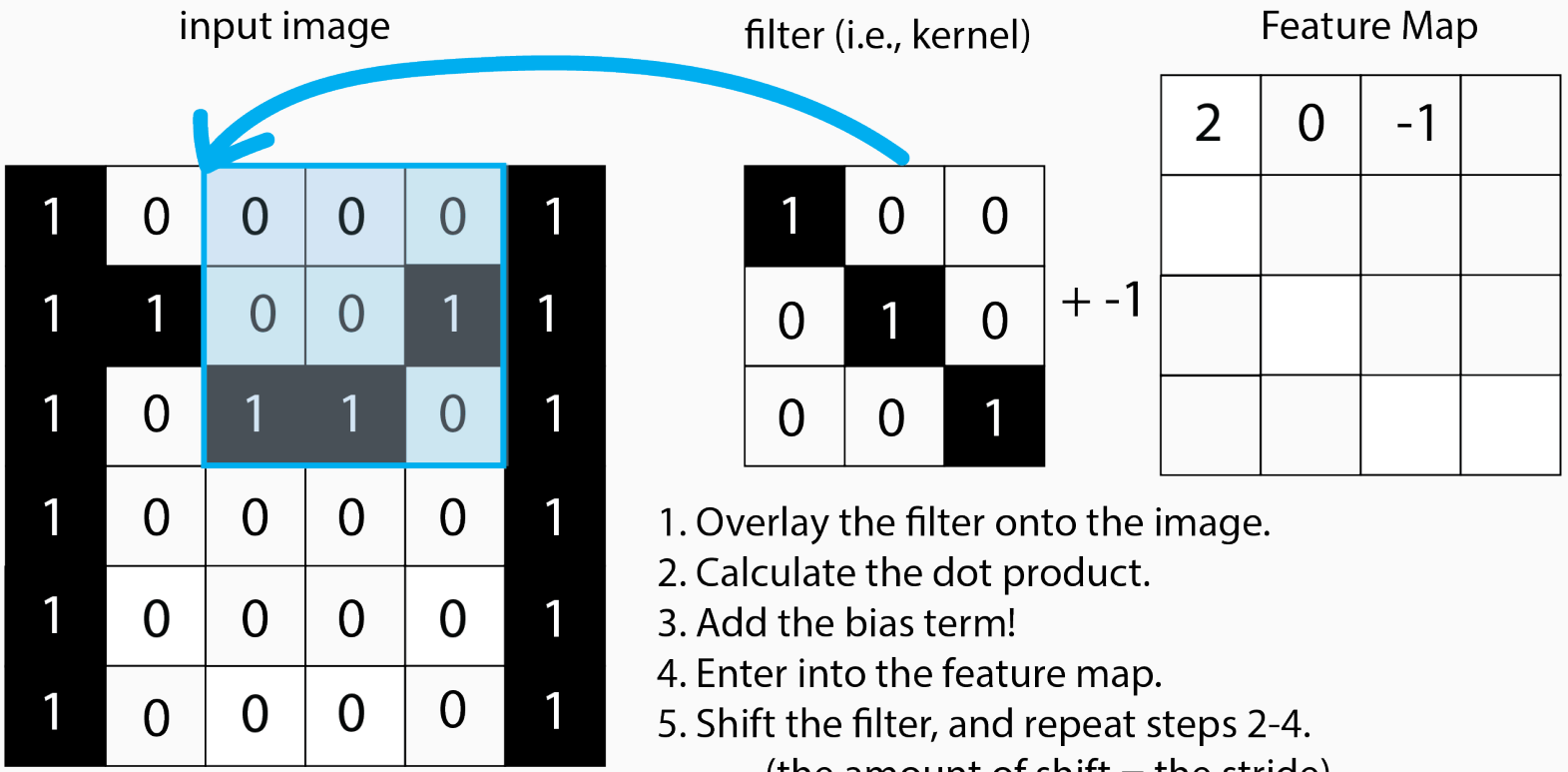
1. Overlay the filter onto the image.
2. Calculate the dot product.
   1x1 + 0x0 + 0x0 +
   0x1 + 1x1 + 0x0 +
   0x1 + 0x0 + 1x1 = 3
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?



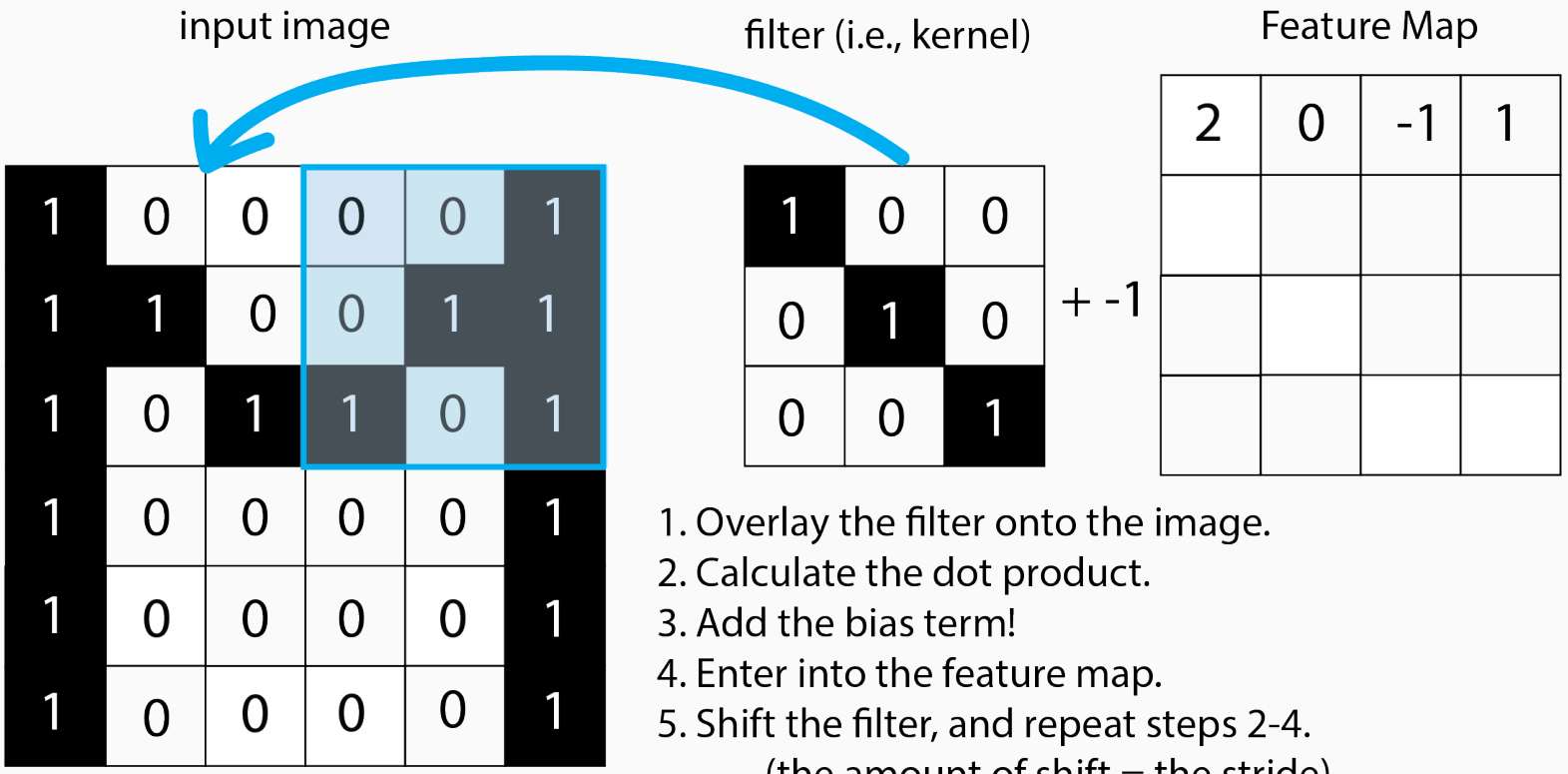
1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?



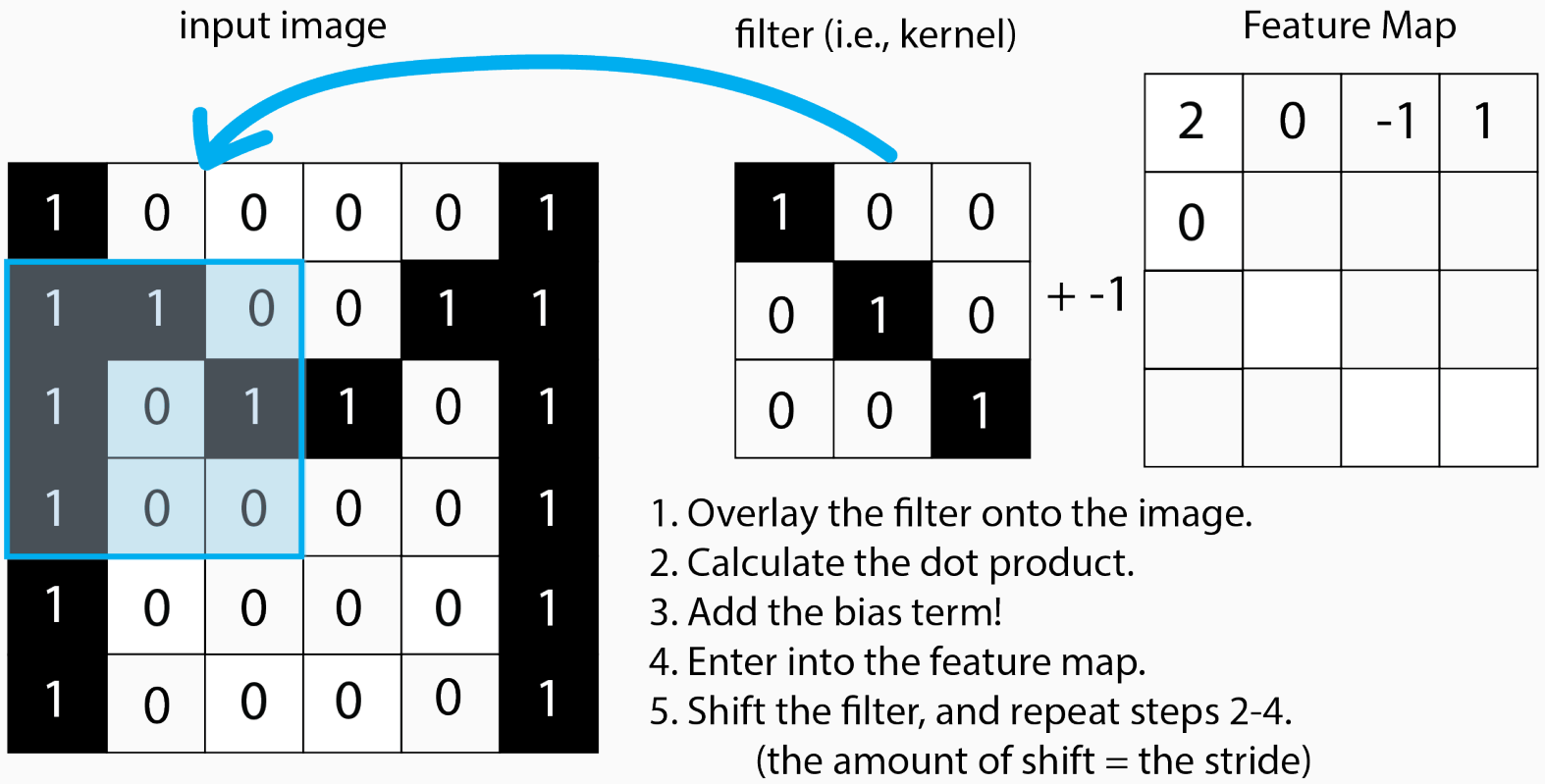
1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?



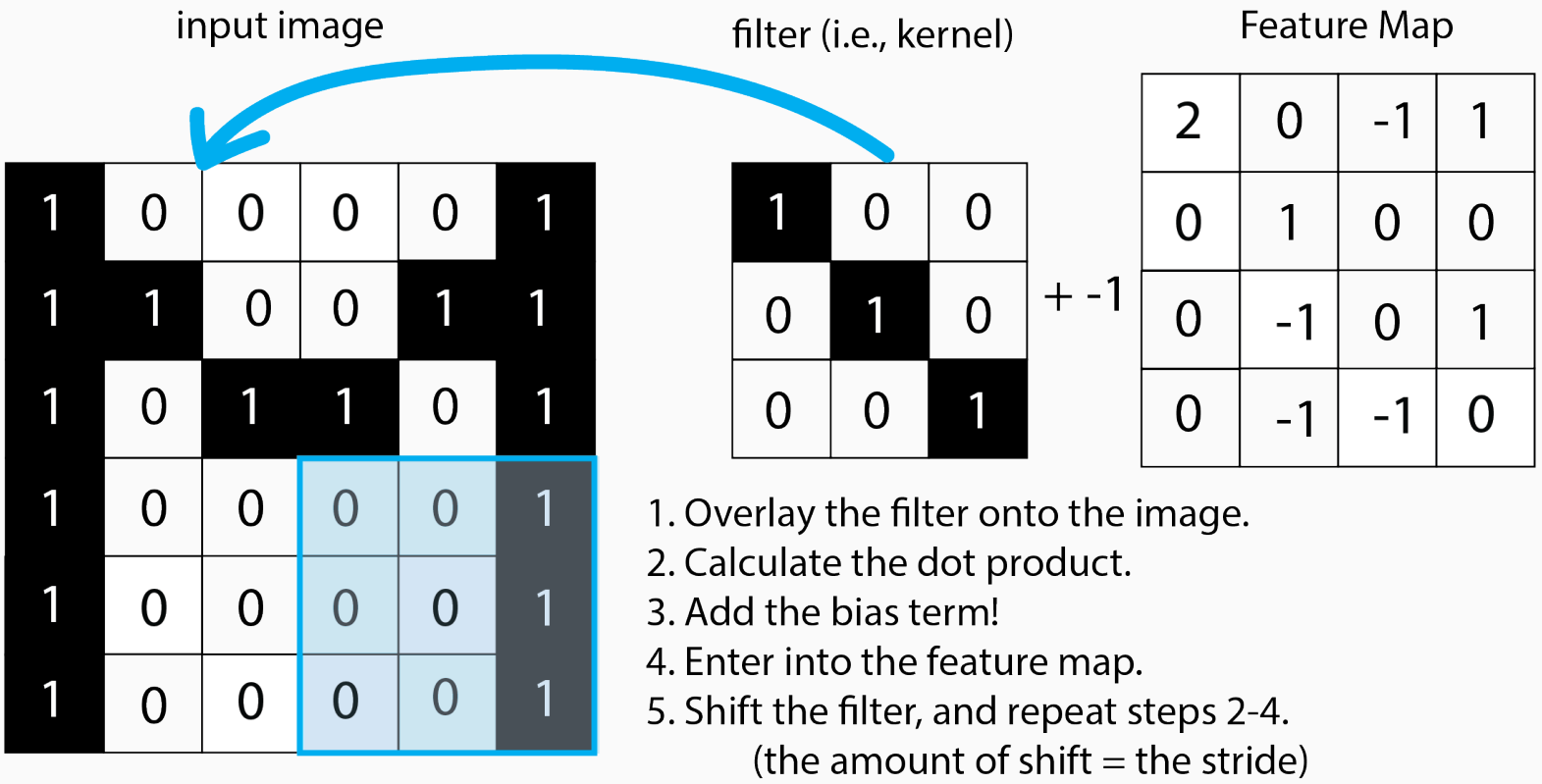
1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?



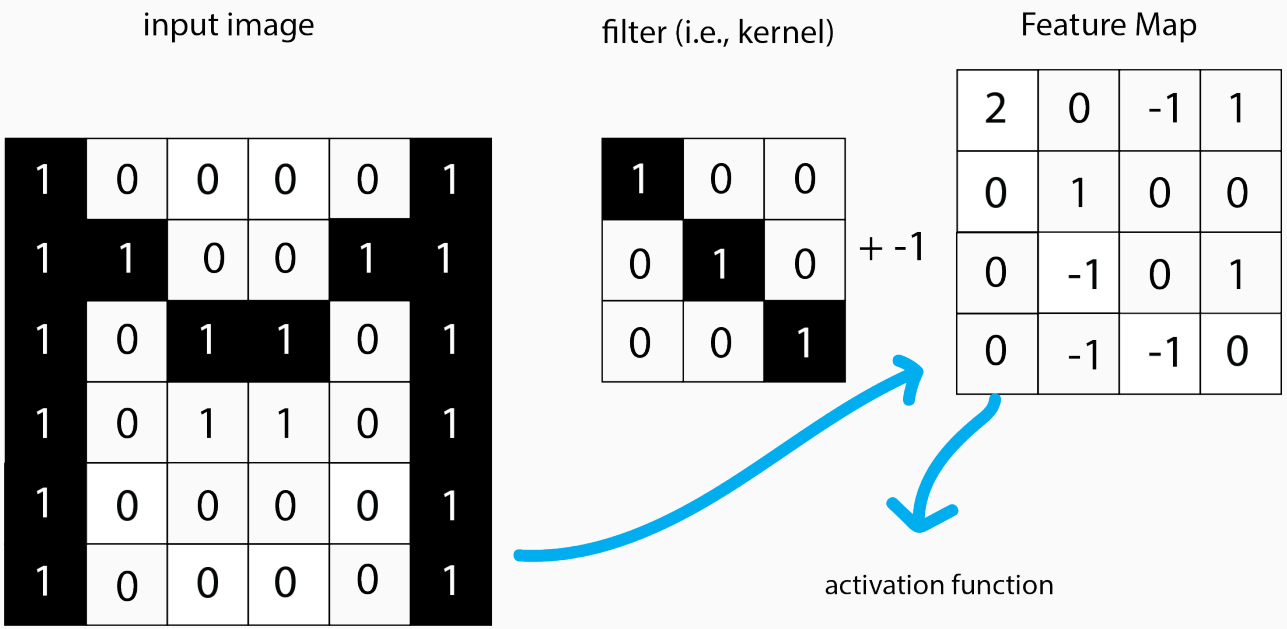
1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?

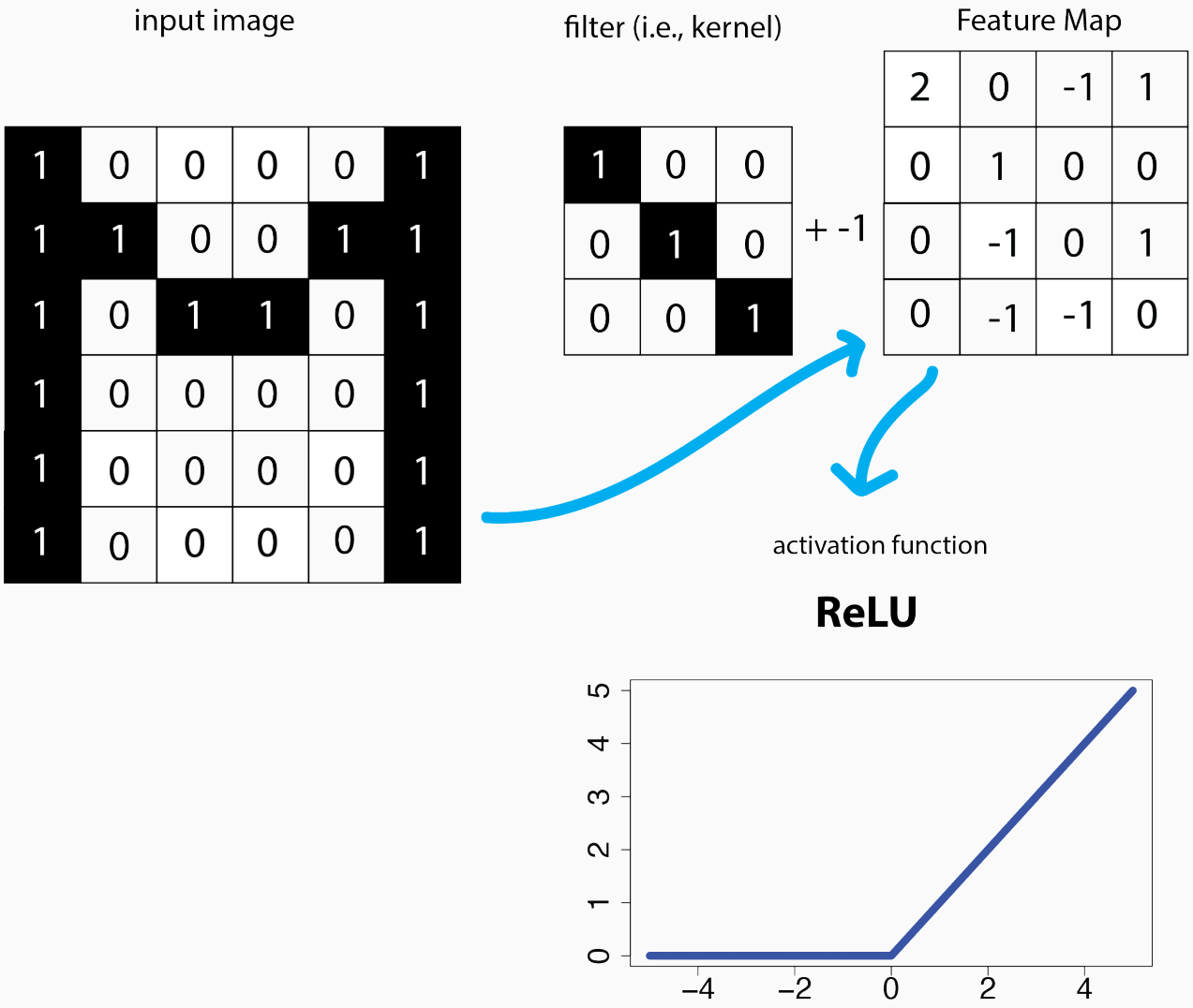


1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
   (the amount of shift = the stride)

How do CNNs work?



How do CNNs work?



How do CNNs work?

Activated Feature Map

2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

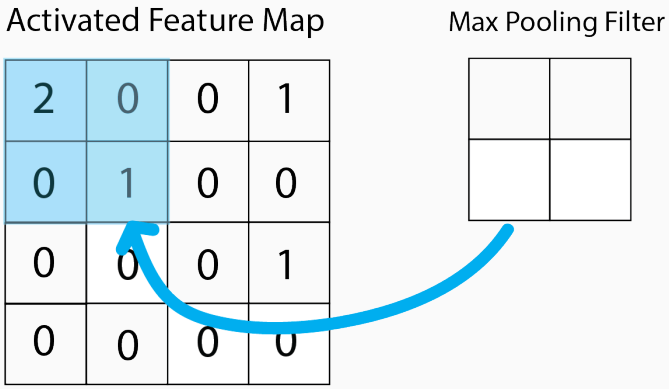
How do CNNs work?

Activated Feature Map

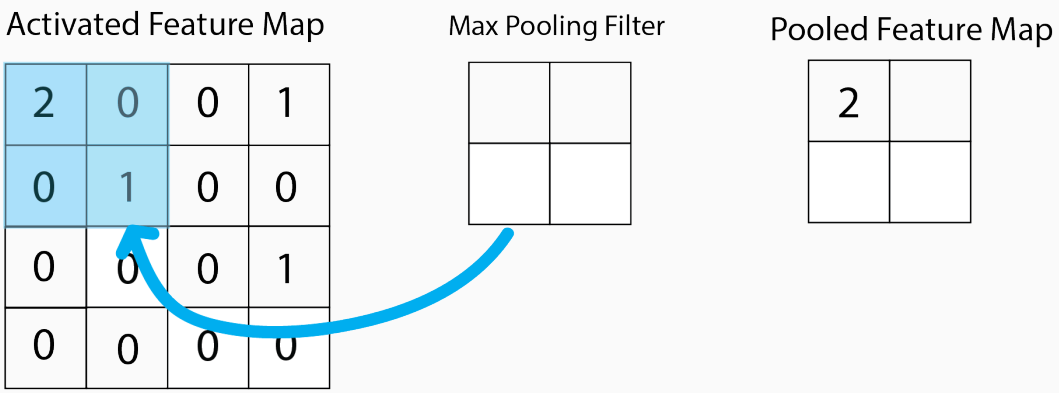
2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

Max Pooling Filter

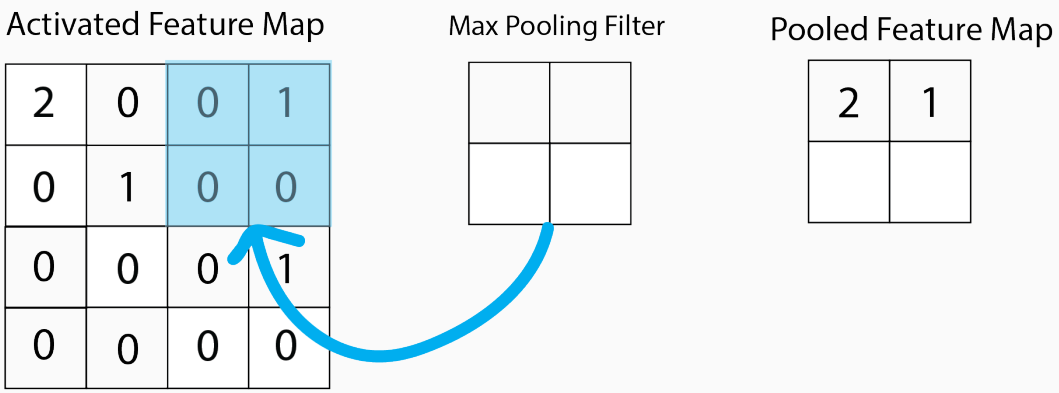
How do CNNs work?



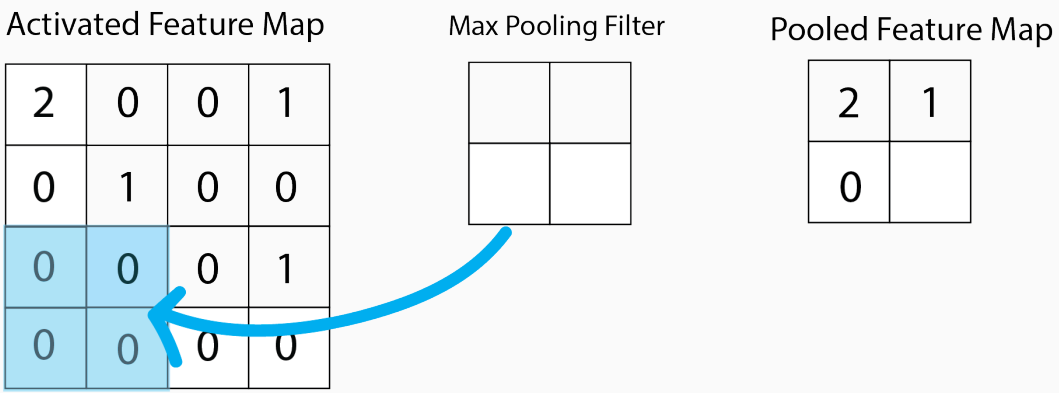
How do CNNs work?



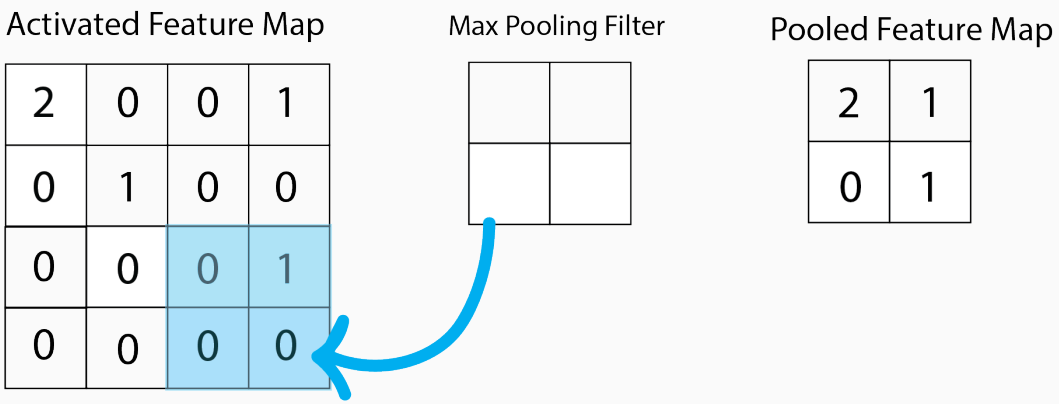
How do CNNs work?



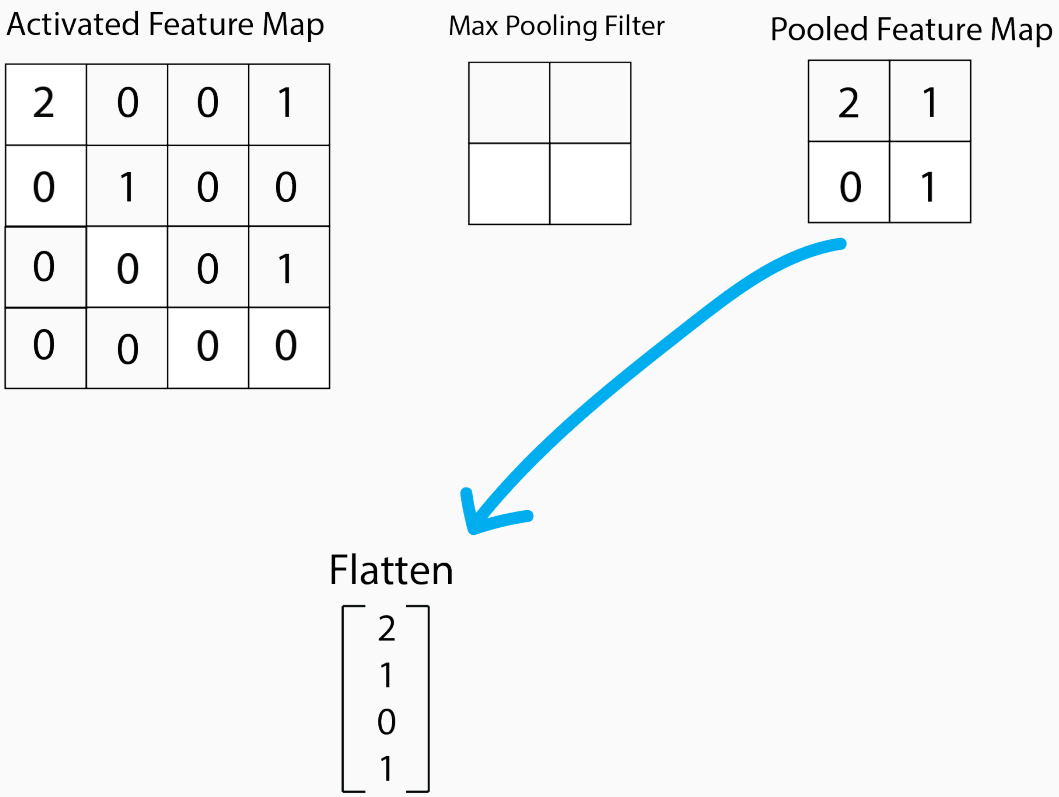
How do CNNs work?



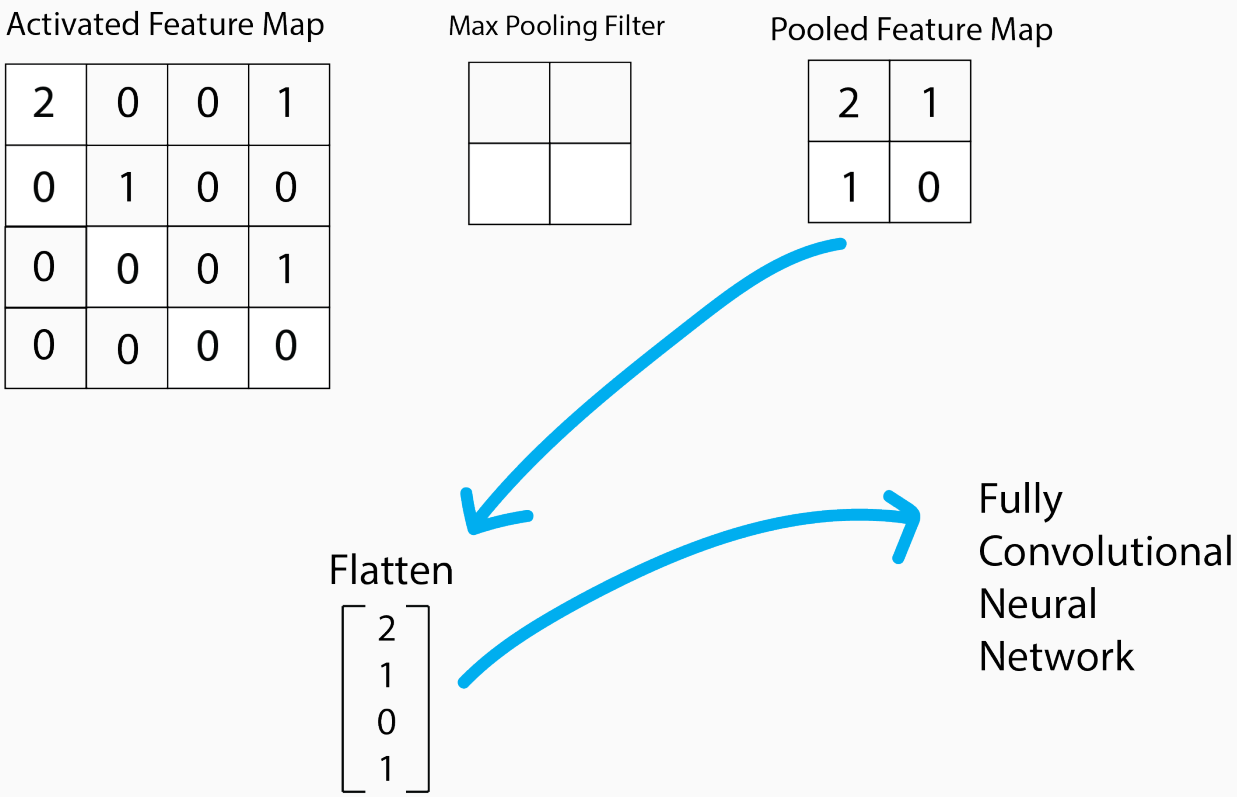
How do CNNs work?



How do CNNs work?



How do CNNs work?



How do CNNs work?

- Input: Image
- Apply Convolutional Filter
    1. Overlay filter on input image.
    2. Calculate dot product.
    3. Add bias term.
    4. Populate feature map.
    5. Slide filter, and repeat.
- Activate Feature Map.
- Perform Max Pooling.
- Flatten into vector.
- Feed into fully connected neural network.

How do CNNs work?

- **learnable parameters:** weights and biases (weights for each cell of each convolutional filter)
- **hyperparameters:** the number of convolutional layers, the activaiton functions, the size of the filters, the stride size, the type of pooling, the FCNN hyperparameters!

How do we use CNNs in population genetics?

The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference

Lex Flagel,[1,2] Yaniv Brandvain,[2] and Daniel R. Schrider*,[3]

- We can treat alignments as images!
- This allows us to avoid *a priori* summarization of our data using features.
- Instead, our network learns the features directly.

How do we use CNNs in population genetics?

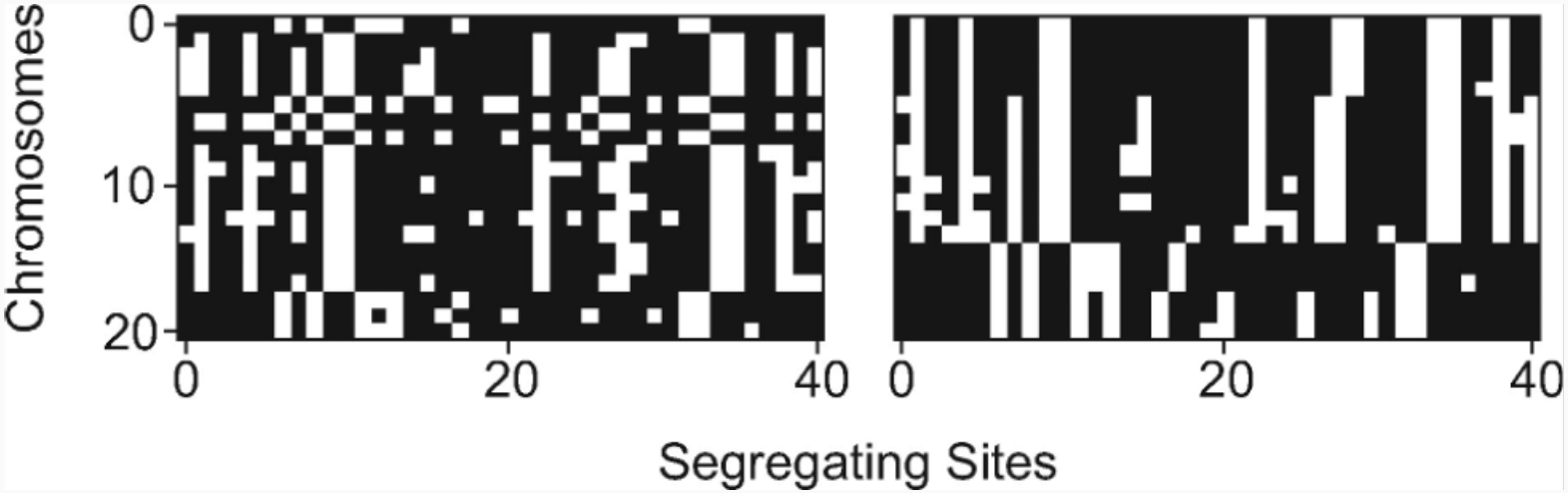
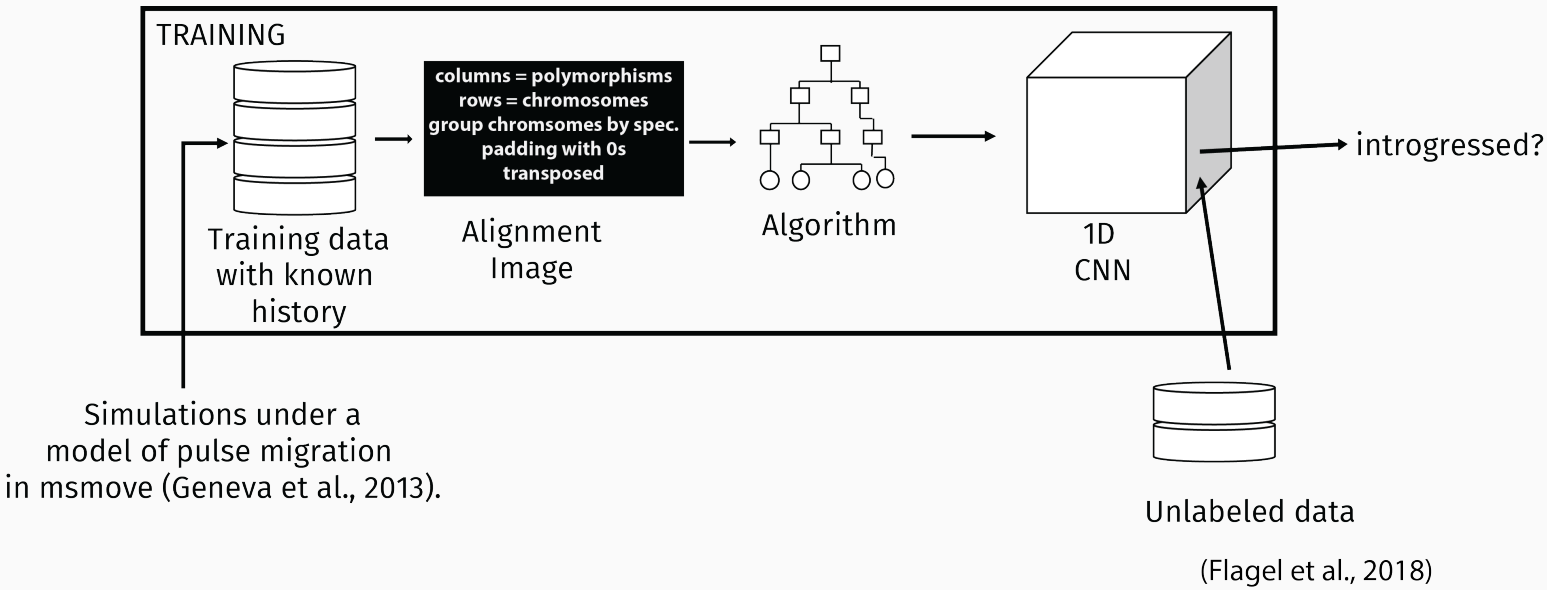


Figure 2 from Flagel et al., 2019

Examples: Flagel

Goal: to predict whether a genomic window is introgressed.



(Flagel et al., 2018)

Examples: Flagel

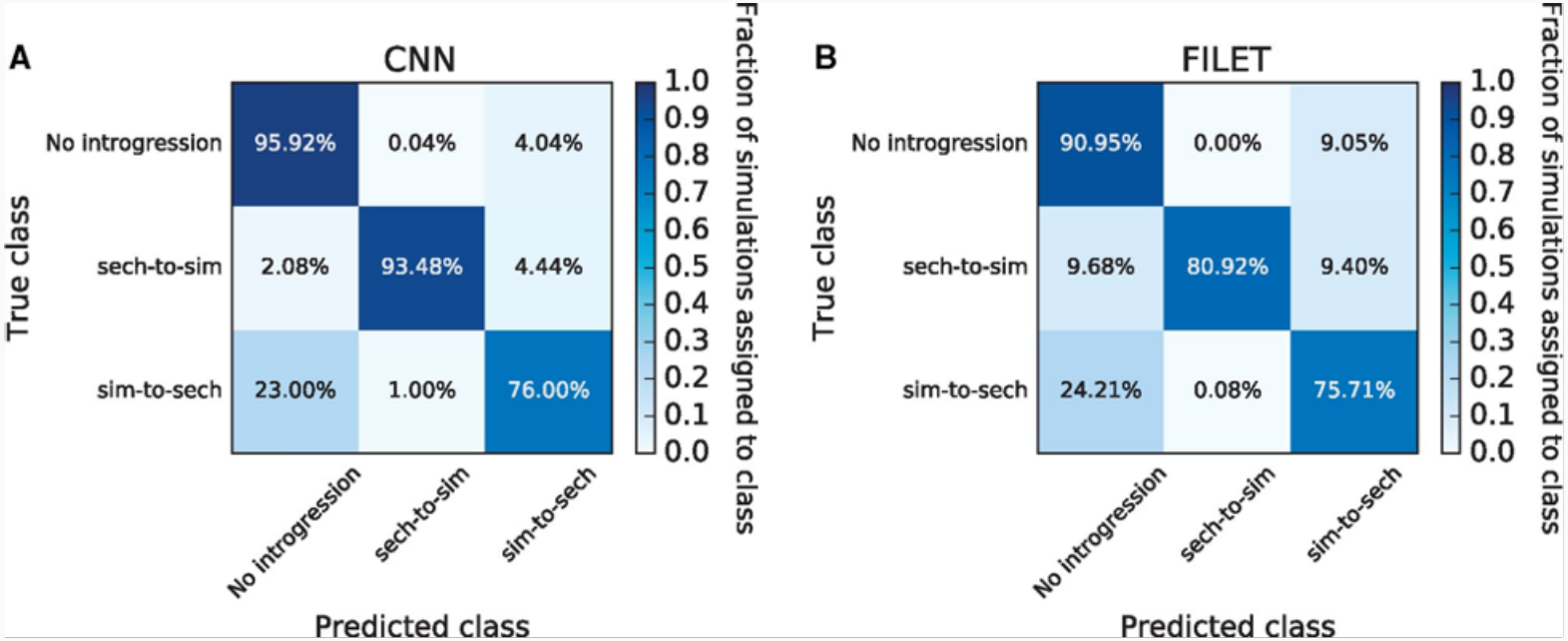
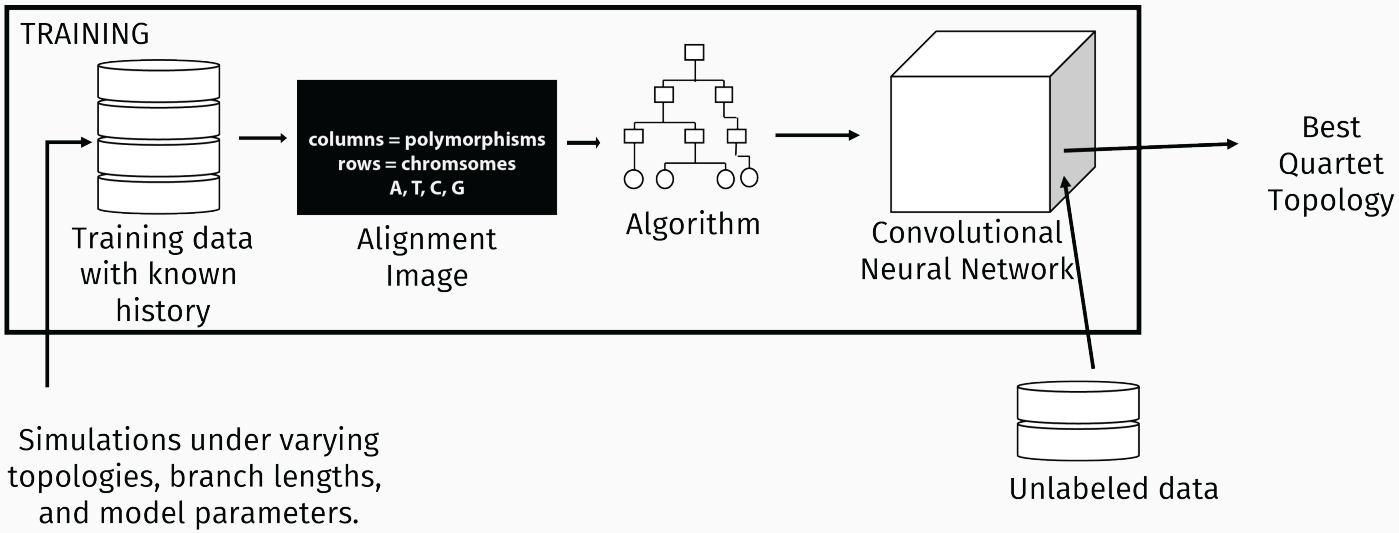


Figure 4 from Flagel et al., 2019

Examples: Suvurov

Goal: to infer the unrooted quartet topology.



(Suvorov et al., 2020)

Examples: Suvurov

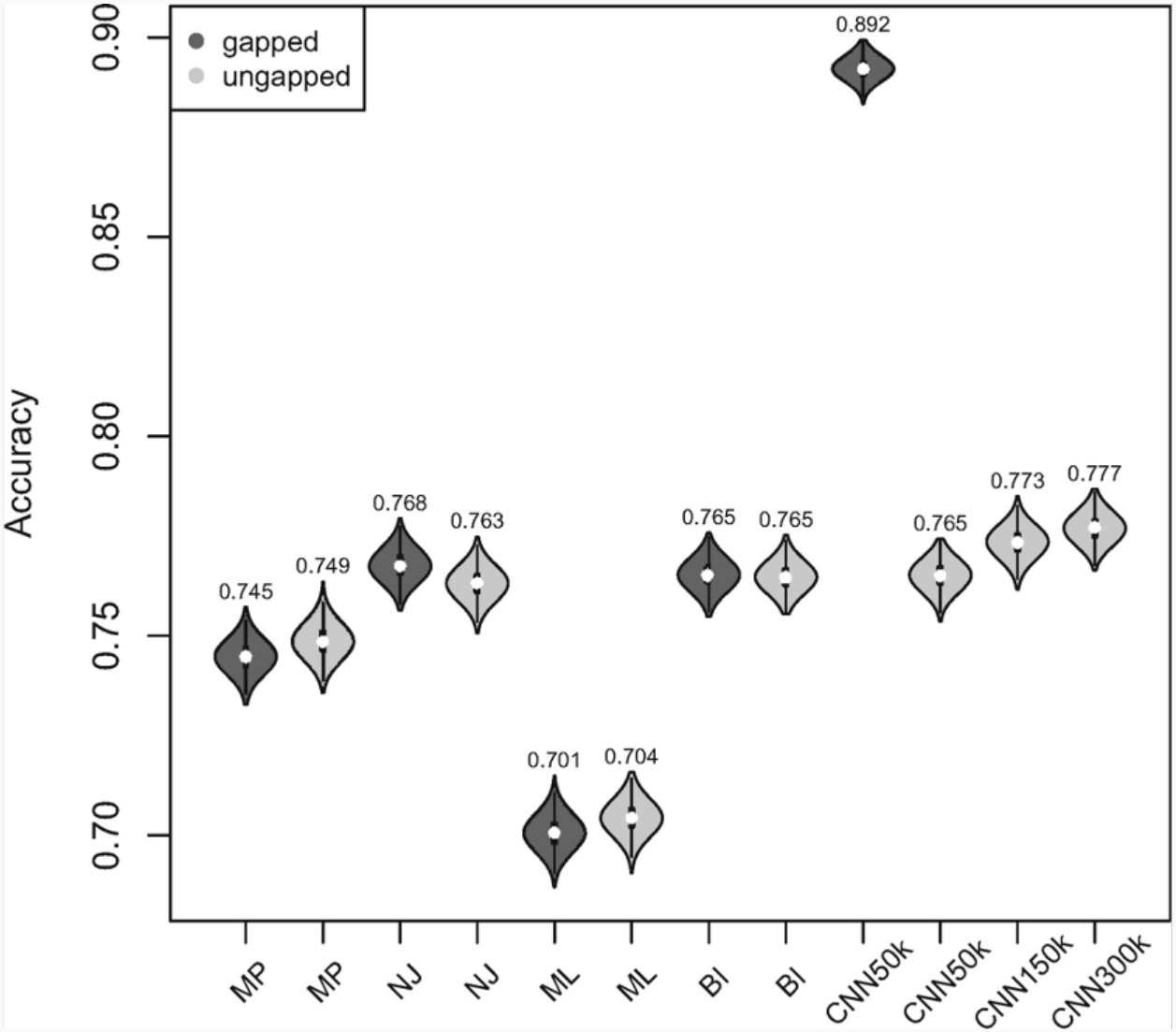


Figure 2 from Suvorov et al., 2020

Examples: Suvurov

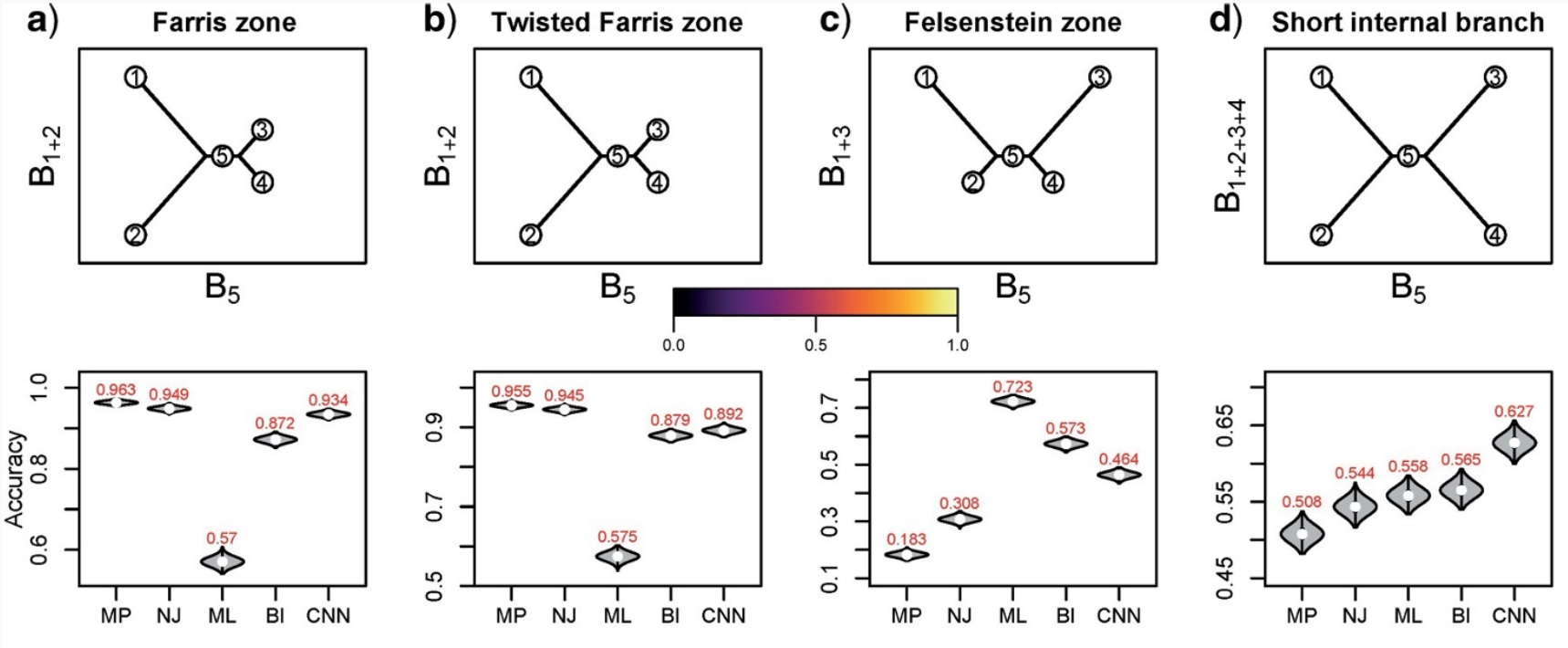


Figure 4 from Suvorov et al., 2020

Convolutional Neural Networks

- Take images directly as input!
- Bypass the need to summarize the data!
- Have even more hyperparameters than a FCNN!