Queen Mary
University of London

Machine learning and deep learning for population genetic inferences

A gentle and brief introduction

Matteo Fumagalli, Manolo Perez, Flora Jay

Agenda

Morning session: introduction to

- basic concepts in supervised ML
- neural networks
- deep learning

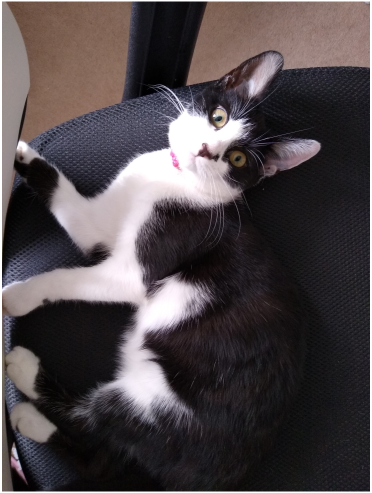
with examples from the literature.

Agenda

Late morning and afternoon sessions:

- guided example on simple neural networks with `python`
- group challenge

with applications on detecting signals of natural selection



Intended Learning Outcomes

By the end of this very first part, you **will** be able to:

- Provide a basic definition of machine learning
- Illustrate the concepts of data, labels, and task
- Describe the difference between unsupervised and supervised learning

What is machine learning?

A typical example

TASK:
predict y from x

Angermueller et al Mol Syst Biol. (2016) 12: 878

What is machine learning?

A typical example

TASK:
predict y from x

Angermueller et al Mol Syst Biol. (2016) 12: 878

Data + Task: ?

slide from Flora

What is the data?

- Learning something from **data**
  data = multidimensional object with e.g lots of samples (rows) and lots of variables/predictors/factors/features/markers ...
  (one vector/one matrix/several matrix per sample)

	loc1	loc2	loc3	...
ind1	A/A	C/C	C/G	
ind2	T/A	C/C	G/G	
...				

	Age	Gender	Work	Salary
ind1	55	F	baker	35k
ind2	43	M	..	..
...				

Quantitative and qualitative variables

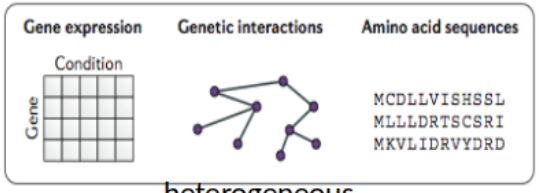
	loc1	loc2	...	Sport activity	Hours of free time	...	Disease X ?
ind1	A/A	C/C					
ind2	T/A	C/C					
...							

multidimensional and heterogeneous data

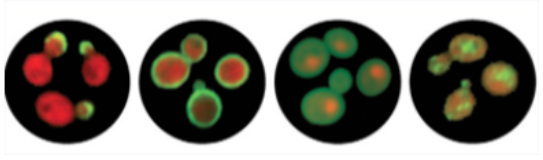
slide from Flora

What is the data?

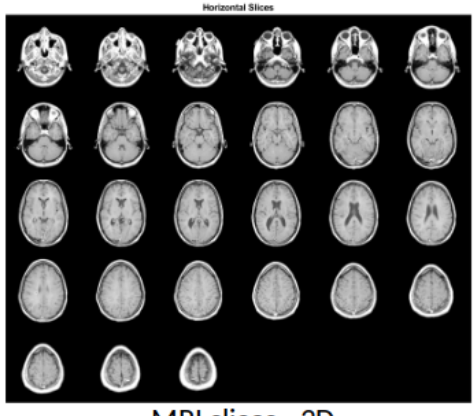
- Learning something from **data**
  data = multidimensional object with e.g lots of samples (rows) and lots of variables/ predictors/factors/features/markers ... (one vector/one matrix/several matrix per sample)



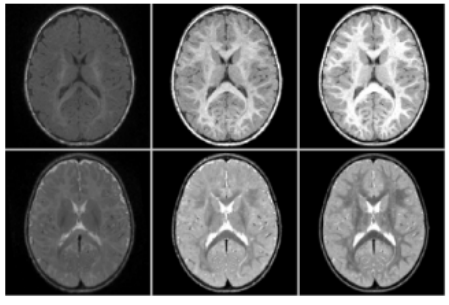
heterogeneous



Images with colors - micrographs of yeast cells expressing GFP-tagged proteins



MRI slices ~3D



temporal

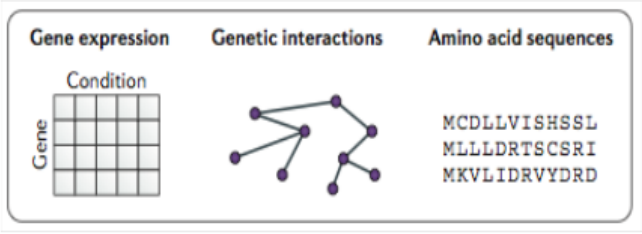
slide from Flora

What is the learning **task**?

- **Data with or without label**
- What's a label ? a target class or a target value observed for each sample
  Data are not always labeled. They can also have multiclass labels
  ex : pic of dog/person/car..., price of house, level of cholesterol
- **Task/objective ?**

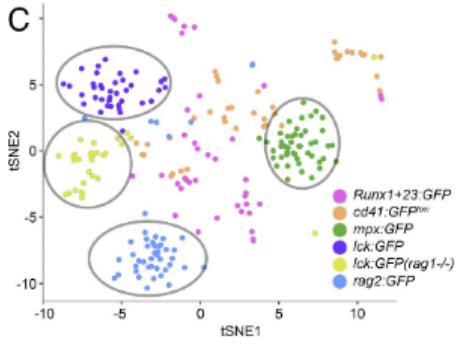
What is the learning **task**?

- **Data with or without label**
- What's a label ? a target class or a target value observed for each sample
  Data are not always labeled. They can also have multiclass labels
  ex : pic of dog/person/car..., price of house, level of cholesterol
- **Task/objective ?**
- **Task/objective ?**



Task = predicting gene function labels

Task = predicting protein 3D structure/contact map from DNA sequences and secondary structure, ... (blue=truth, red=pred)



Task = identifying groups (clusters) of eg single-cell (T cells, NK cells ,..) with similar pattern of gene expression

Tang et al
JEM 2017

Unsupervised vs. Supervised Tasks

- Learning something from **data**
- Either **unsupervised** (no labels) or **supervised** (discrete or continuous labels)

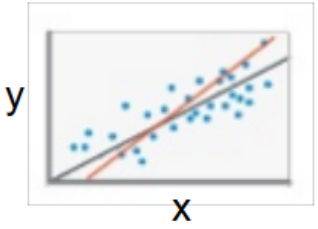
Can you think of examples of unsupervised and supervised tasks in evolutionary genomics?

Supervised learning

- Supervised = Learn a relationship (a general model) linking input data (or features) to observed labels

Classification (predict a class)

Regression (predict a variable)



What for:
- Predict labels of new unlabeled samples (eg what's on an image?),
- Understand better the relationship between features and the label (eg understand which set of genes allow to predict a disease risk),
- ...

slide from Flora

Intended Learning Outcomes

At the end of this very first part, you are **now** able to:

- Provide a basic definition of machine learning
- Illustrate the concepts of data, labels, and task
- Describe the difference between unsupervised and supervised learning

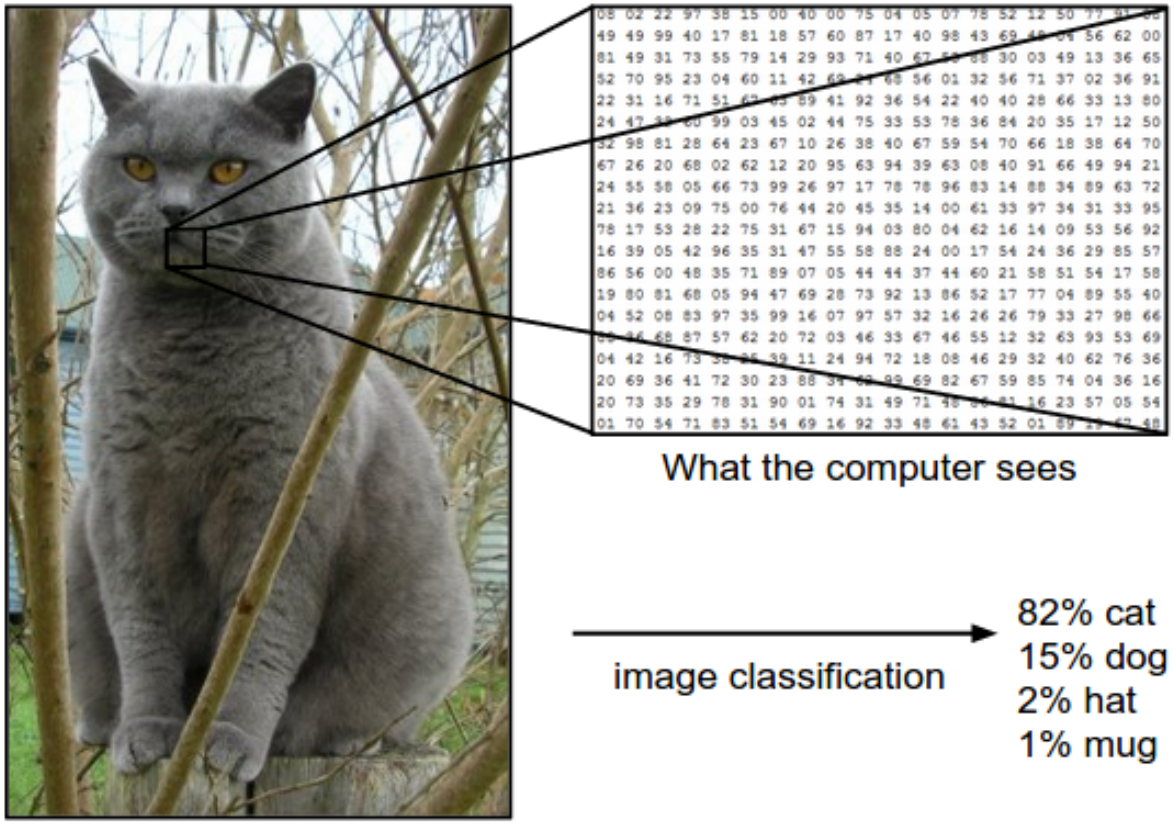
Intended Learning Outcomes

By the end of this session, you will be able to:

- Describe the three key components of a classifier: score function, loss function, optimisation
- Identify the elements of a neural networks, including neurons and hyper-parameters
- Illustrate the layers in a neural network
- Demonstrate how to implement, train and evaluate neural networks in `python`

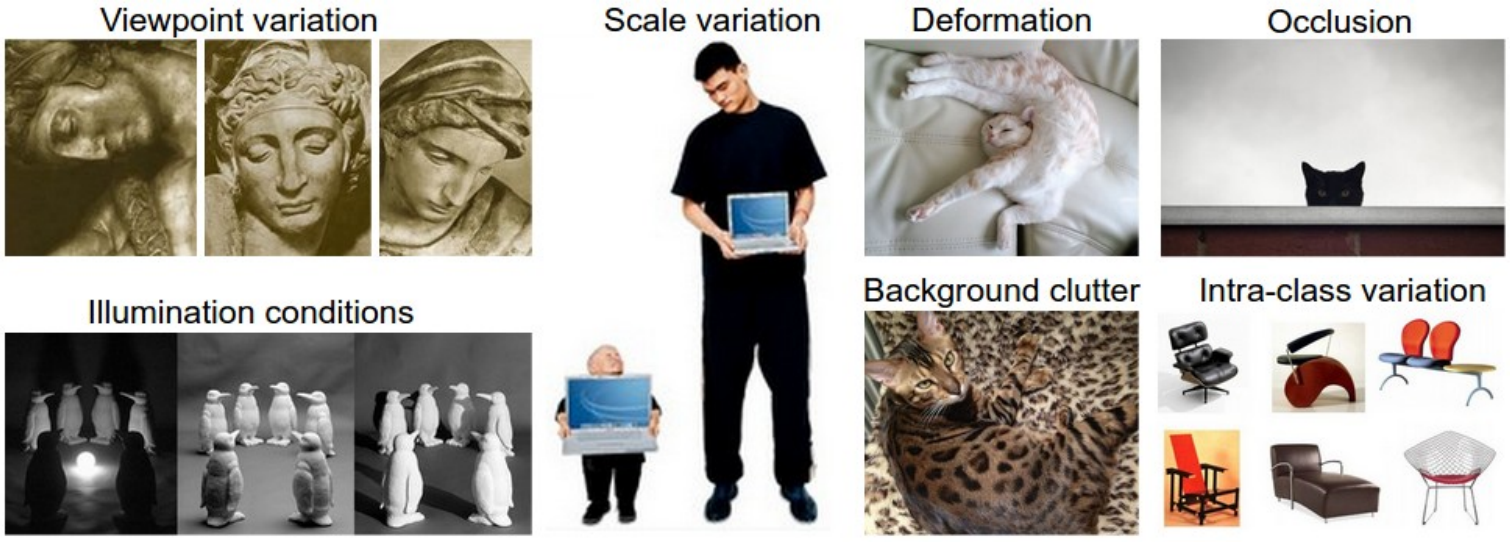
What do you see?

What does the computer see?



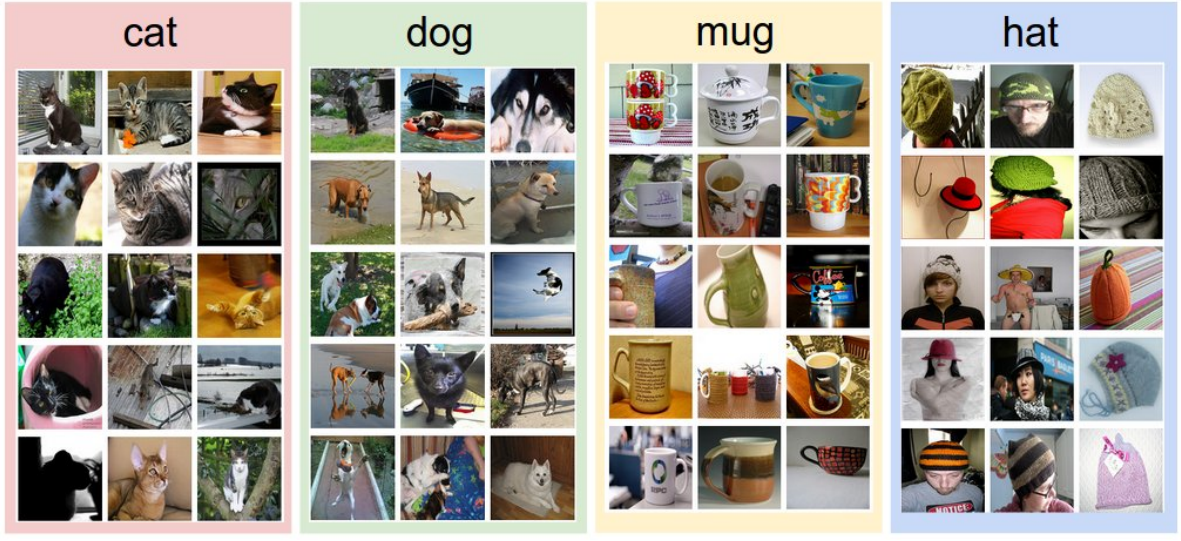
Is it THAT difficult?

Challenges



- invariant to the cross product of all these variations
- retaining sensitivity to the inter-class variations

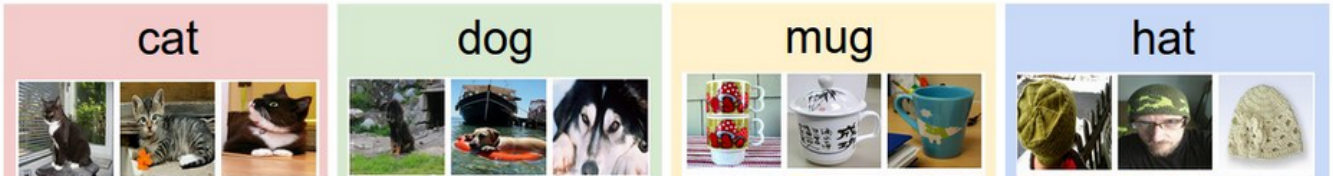
Data-driven approach



We need a (large) training dataset of labeled images.

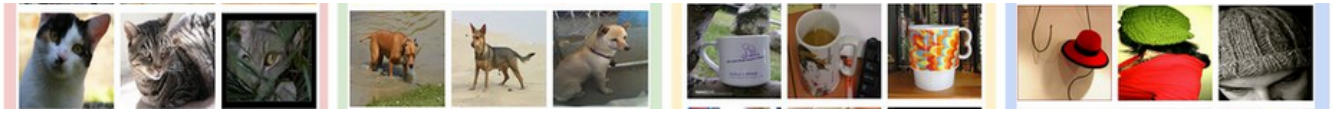
Pipeline for classification

1. Training set: *N* images of *K* classes



2. Learning: training a classifier

3. Evaluation: against the *ground truth*



Nearest Neighbour Classifier

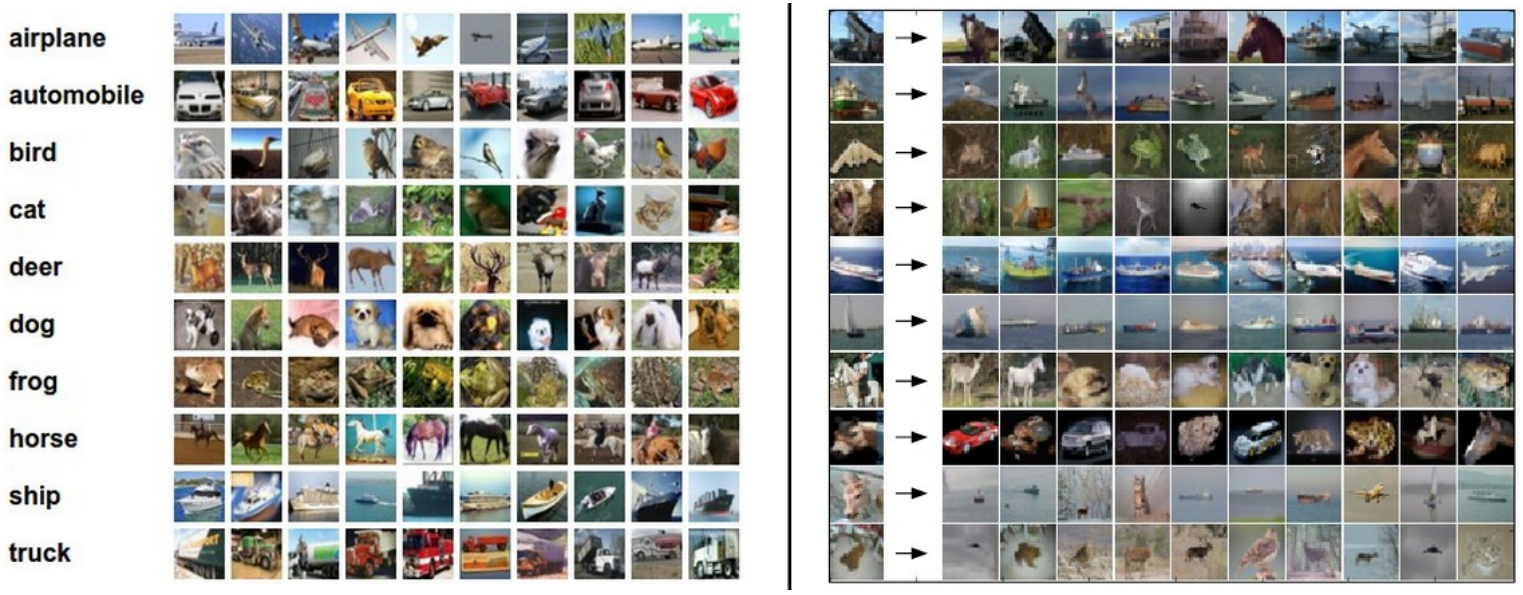


Figure 1: CIFAR-10 dataset: 60k tiny images of 10 classes.

The nearest neighbour classifier will take a test image, **compare** it to every single one of the training images, and predict the label of the closest training image.

Nearest Neighbour Classifier

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

Nearest Neighbour Classifier

test image

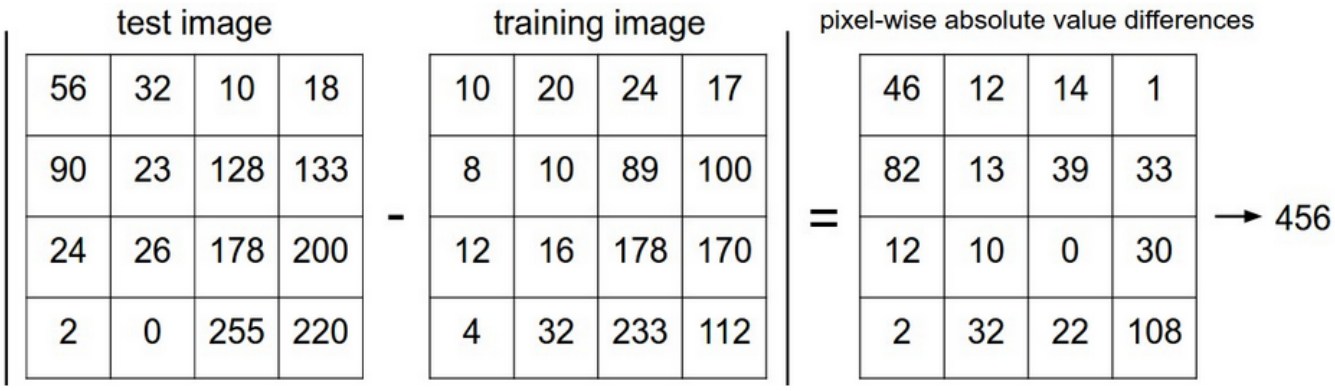
56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

Nearest Neighbour Classifier



Nearest Neighbour Classifier

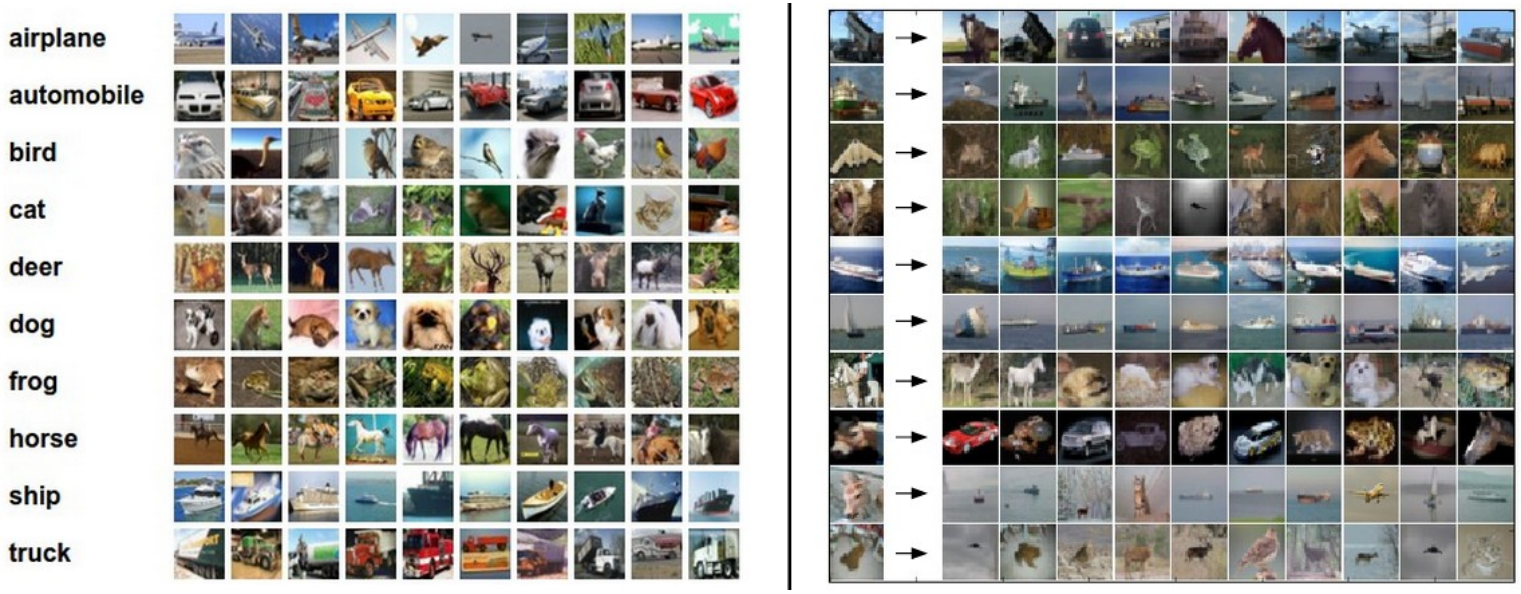


Figure 2: CIFAR-10 dataset: 60k tiny images of 10 classes.

Do you think it works well? If not, why? Can you think of solutions?

k-Nearest Neighbour Classifier

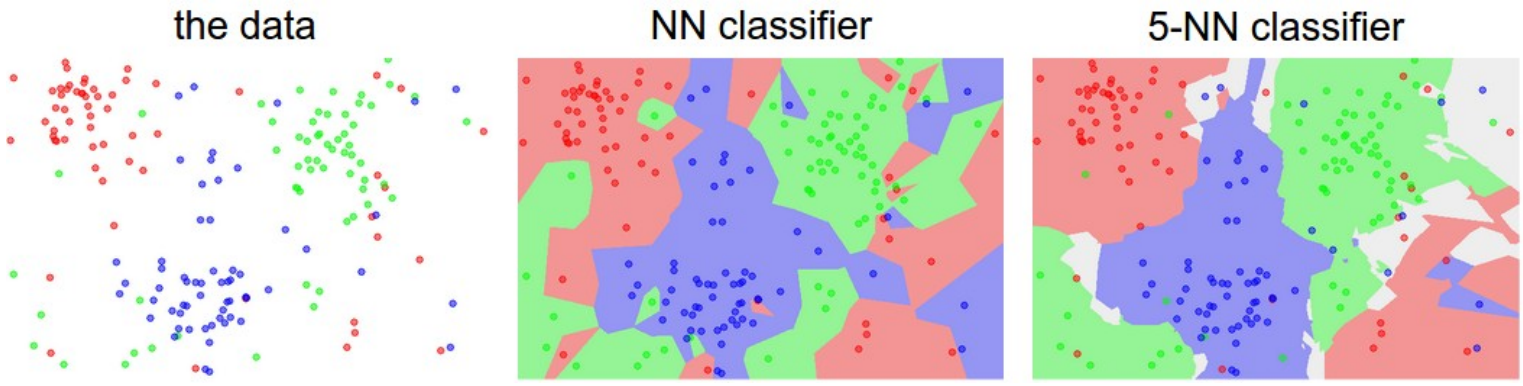


Figure 3: An example of the difference between Nearest Neighbor and a 5-Nearest Neighbor classifier, using 2-dimensional points and 3 classes (red, blue, green).

What value of k should we use? Which distance?

Hyperparameter tuning

The engineer says: "We should try out many different values and see what works best."

Agree or disagree?

Validation test

The good engineer says:
"Evaluate on the test set only a single time, at the very end."

- Split your training set into training set and a validation set.
- Use validation set to tune all hyperparameters.
- At the end run a single time on the test set and report performance.

Data splits

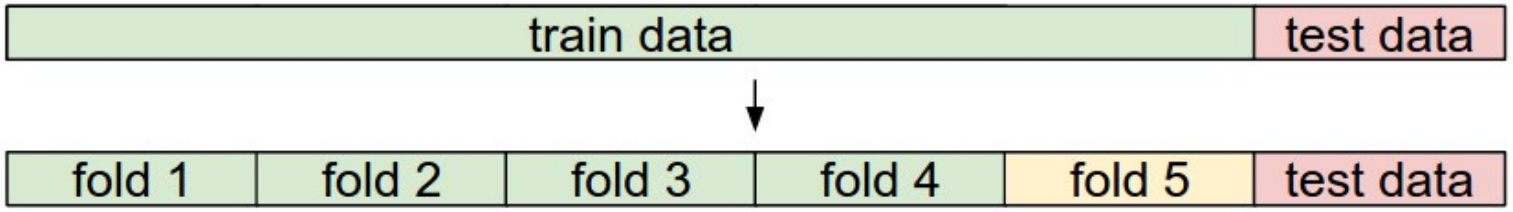


Figure 4: The training set is split into folds: 1-4 become the training set while 5 is the validation set used to tune the hyperparameters.

Where is the Nearest Neighbour classifier spending most of its (computational) time?

Wrap up

- the problem of classification: predicting labels for novel test entries
- training set vs testing set
- a simple Nearest Neighbour classifier requires hyperparameters
- validation set to tune hyperparameters
- Nearest Neighbour classifier has low accuracy (distances based on raw pixel values!) and is expensive at testing

Our aim: a solution which gives very high accuracy, discards the training set once learning is complete, and evaluates a test data point in less than a millisecond!

Linear classification

New approach based on:

- **score function** to map raw data to class scores
- **loss function** to quantify the agreement between predicted and true labels

Parameterised mapping from data points to label scores

Our aim is to define the score function that maps the pixel values of an image (or features) to confidence scores for each class.

Assuming that:

N images, each with dimensionality D, and K distinct classes

$x_i \in R^D$ is image i-th with dimensions D and label y_i, with $i = 1...N$ and $y_i \in 1...K$

then we define a **score function**: $f : R^D \rightarrow R^K$

Linear classifier

Linear mapping: $f(x_i; W, b) = Wx_i + b$

W are called **weights** and b is the **bias** vector.

What are the dimensions of x_i, W and b?

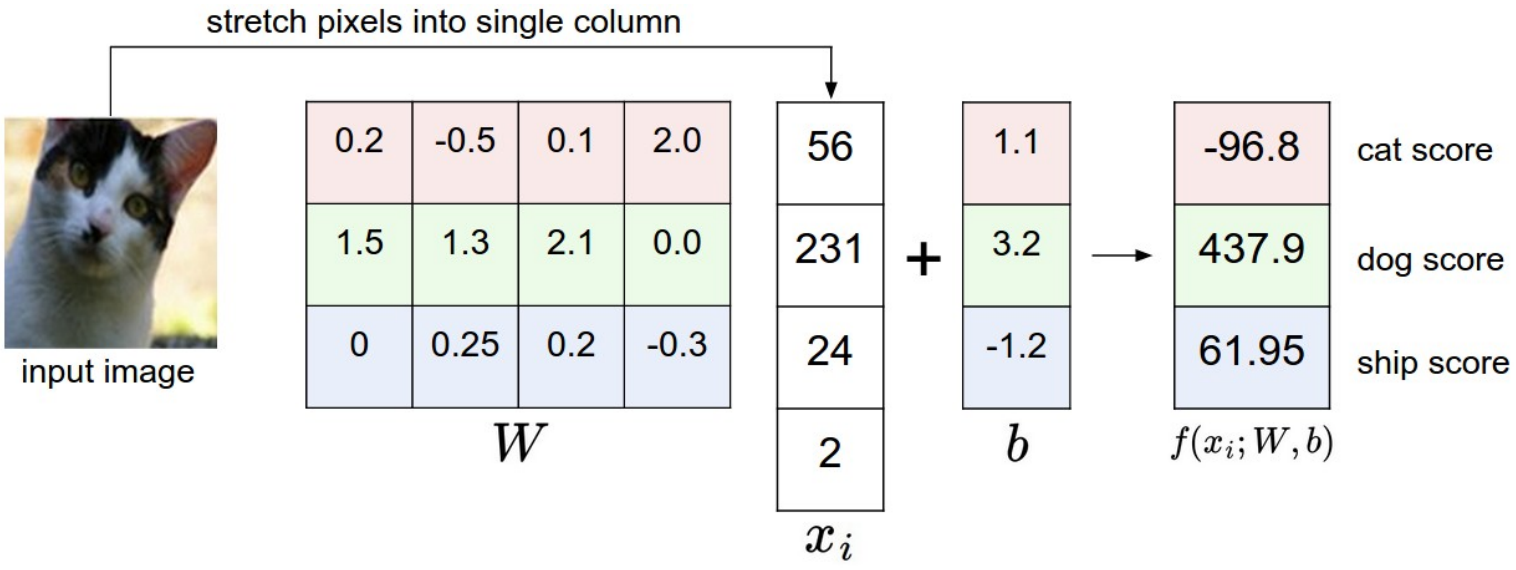
Linear classifier

Linear mapping: $f(x_i; W, b) = Wx_i + b$

W are called **weights** and b is the **bias** vector.

What are the dimensions of x_i, W and b?
W has size [K x D]
x_i has size [D x 1]
b has size [K x 1]

Linear classifier

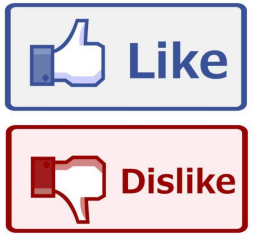


https://en.wikipedia.org/wiki/Matrix_multiplication

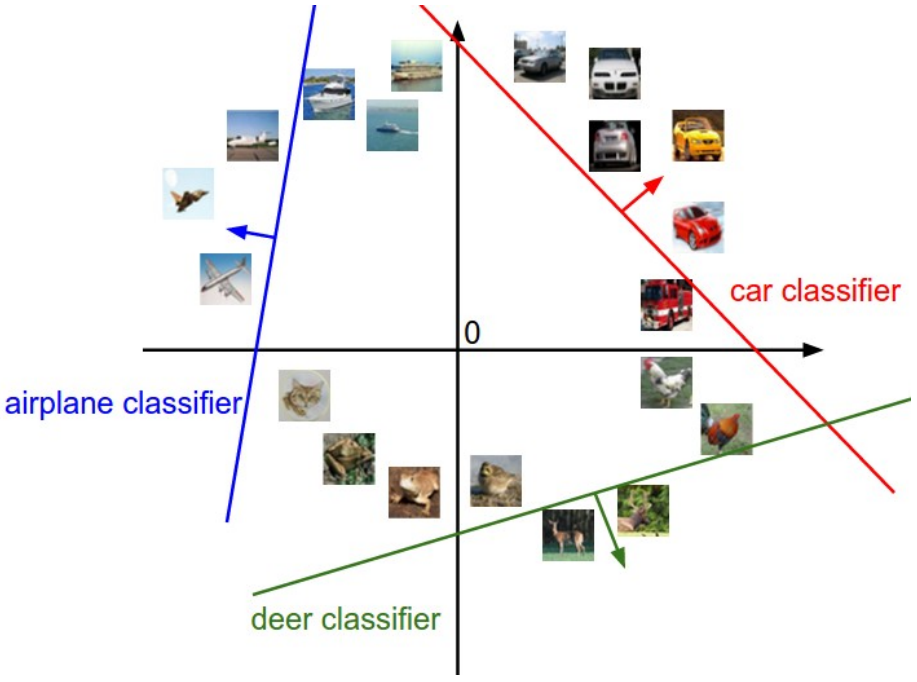
Interpreting a linear classifier (i)

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W



Interpreting a linear classifier (ii)



$$W = \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \quad x_i = \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} \quad + \quad b = \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}$$

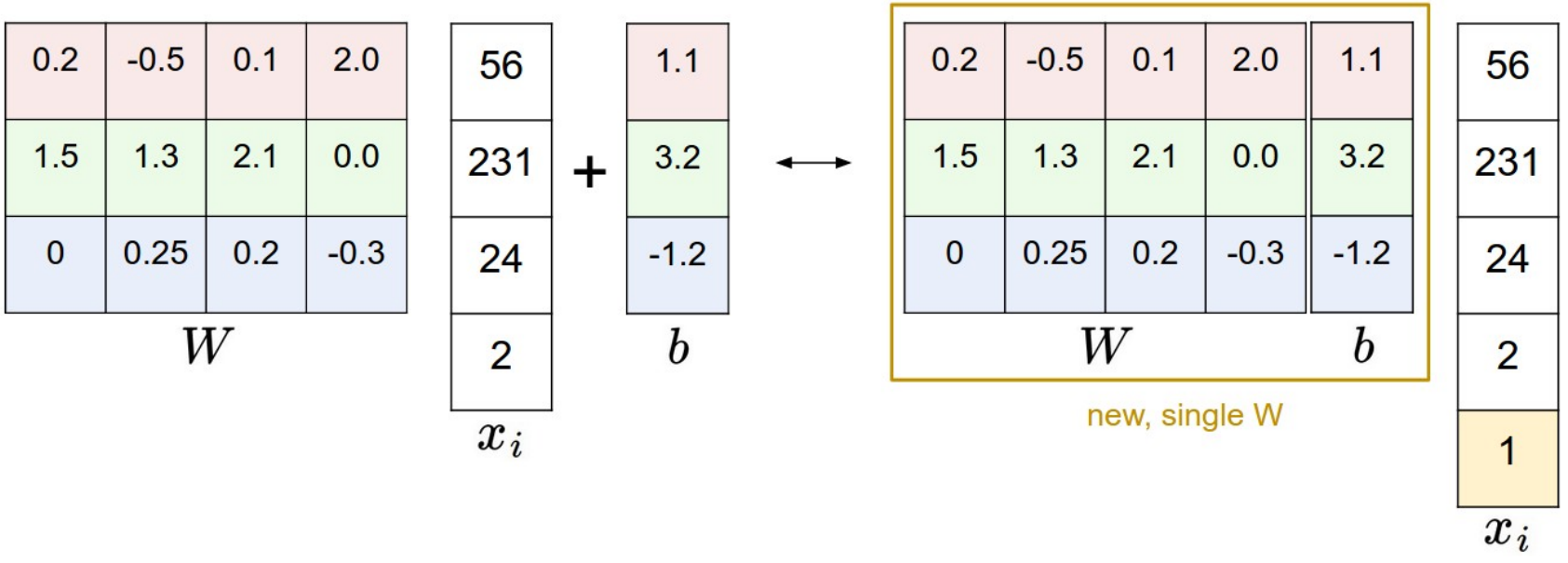
Interpreting a linear classifier (iii)

Template (or prototype) matching.

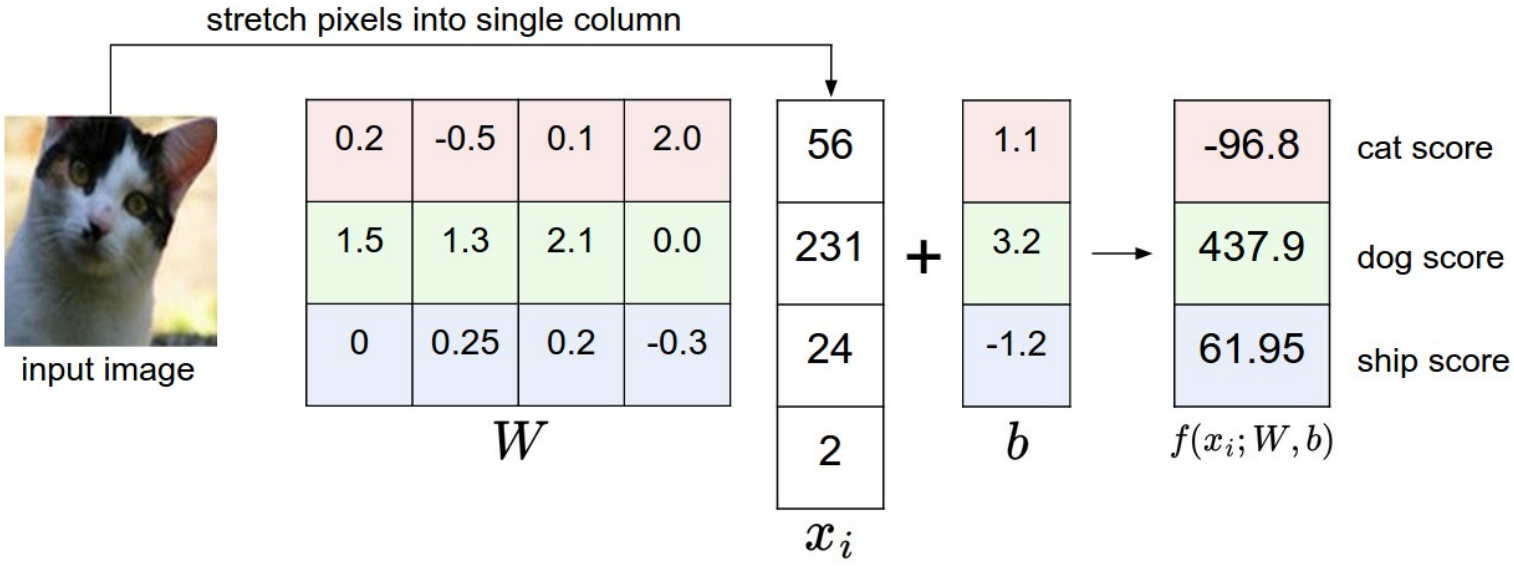
Bias trick

Our new **score function**:

$f(x_i; W) = Wx_i$



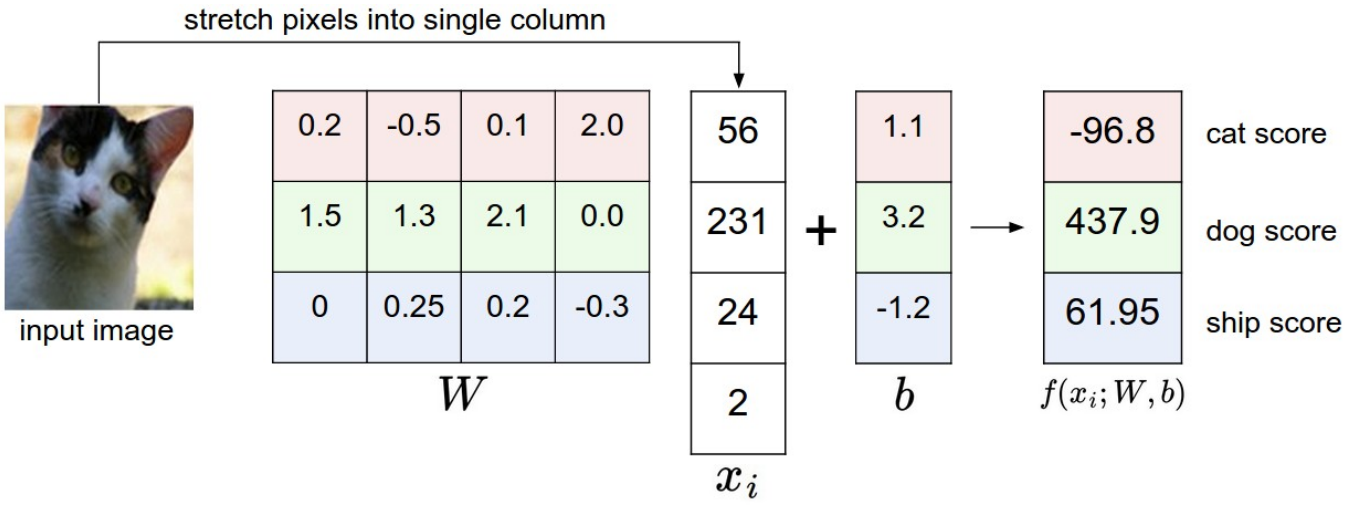
Score function



$f : R^D \rightarrow R^K$ to map raw data to class scores.

Loss function*

To measure our "unhappiness" with predicted outcomes.



* sometimes called cost function or objective

Multiclass Support Vector Machine (SVM) loss

The SVM loss is set so that the SVM "wants" the correct class for each image (y_i) to a have a higher score (s_{y_i}) than the incorrect ones (s_j) by some fixed margin (δ).

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \delta)$$

Example:
$s = [13, -7, 11], y_i = 0, \delta = 10$
$L_i =$?

Hinge loss

$max(0, -)$ or $max(0, -)^2$

Regularisation

If W correctly classifies each sample, then all λW with $\lambda > 1$ will have zero loss.

Which W should we choose?

Regularisation

$W_1 = [1, 0; 0, 1]$

vs.

$W_2 = [0.8, 0.2; 0.2, 0.8]$

Regularisation

$W_1 = [1, 0; 0, 1]$

vs.

$W_2 = [0.8, 0.2; 0.2, 0.8]$

additional cost function is $\lambda \sum_k \sum_l W_{k,l}^2$

Regularisation

$W_1 = [1, 0; 0, 1]$
vs.
$W_2 = [0.8, 0.2; 0.2, 0.8]$

additional cost function is $\lambda \sum_k \sum_l W_{k,l}^2$

if $\lambda = 1$
additional cost for $W1$ is 2
additional cost for $W2$ is $(0.8)^2 + (0.2)^2 + (0.8)^2 + (0.2)^2 = 1.36$

Regularisation

Our new multiclass SVM loss function is:

$$L = \frac{1}{N} \sum_{i} \sum_{j \neq y_i} [max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \delta)] + \lambda \sum_{k} \sum_{l} W_{k,l}^2$$

including one data loss and one regularisation loss term $\lambda R(W)$, specifically L2 penalty.

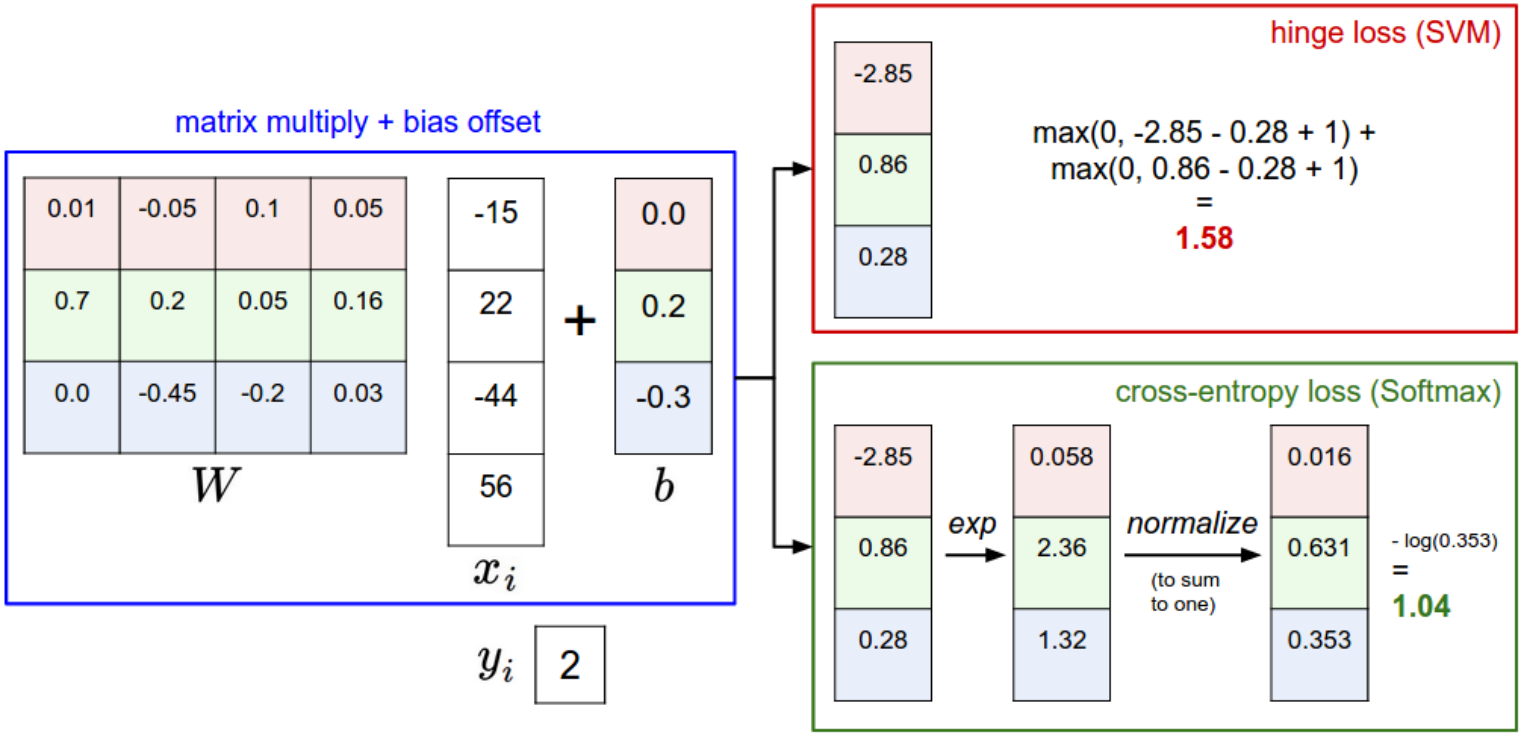
Softmax classifier

Generalisation of the binary logistic regression classifier to multiple classes.

Cross-entropy loss function:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

SVM vs. Softmax classifier



Wrap up

- A **score function** maps image pixels to class scores (using a linear function that depends on W and b).
- Once we learning is done, we can discard the training data and prediction is fast.
- A **loss function** (e.g. SVM and Softmax) measures how compatible a given set of parameters is with respect to the ground truth labels in the training dataset.

Wrap up

- A **score function** maps image pixels to class scores (using a linear function that depends on W and b).
- Once we learning is done, we can discard the training data and prediction is fast.
- A **loss function** (e.g. SVM and Softmax) measures how compatible a given set of parameters is with respect to the ground truth labels in the training dataset.

How do we find the parameters (weights) that give the lowest loss?

Examples of using SVM to detect natural selection

DOI: 10.1534/genetics.110.116459

Searching for Footprints of Positive Selection in Whole-Genome SNP Data From Nonequilibrium Populations

Pavlos Pavlidis,*,[1] Jeffrey D. Jensen† and Wolfgang Stephan*

**Department of Biology II, Ludwig-Maximilians-University Munich, 82152 Planegg, Germany and †Program in Bioinformatics and Integrative Biology, University of Massachusetts Medical School, Worcester, Massachusetts*

Manuscript received March 9, 2010
Accepted for publication April 7, 2010

Examples of using SVM to detect natural selection

Learning Natural Selection from the Site Frequency Spectrum

Roy Ronen,*,[1] Nitin Udpa,* Eran Halperin,[†] and Vineet Bafna[‡]

*Bioinformatics and Systems Biology Program, University of California, San Diego, California 92093, [†]The Blavatnik School of Computer Science and Department of Molecular Microbiology and Biotechnology, Tel-Aviv University, Tel-Aviv 69978, Israel, International Computer Science Institute, Berkeley, California 94704, and [‡]Department of Computer Science and Engineering, University of California, San Diego, California 92093

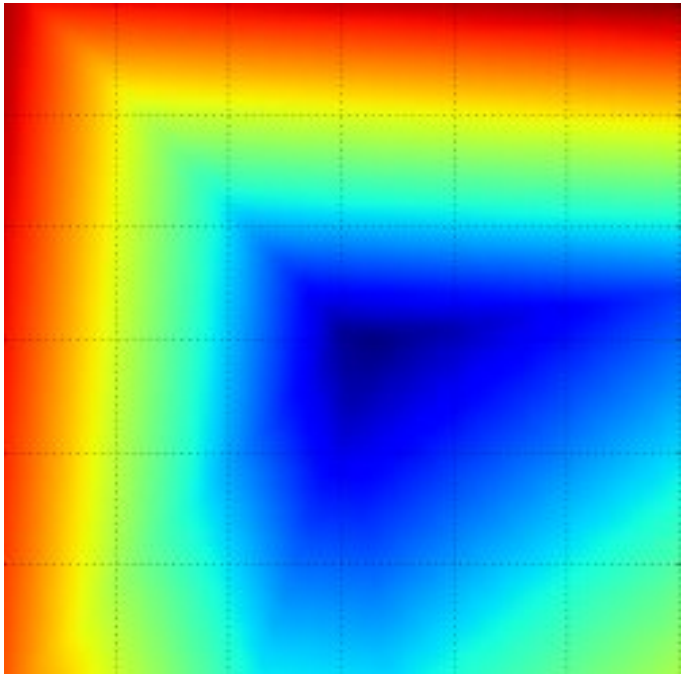
Key components for classification tasks

1. score function
2. loss function
3. optimisation

Optimisation is the process of finding the set of parameters W that minimise the loss function L.

Visualising the loss function

If W_0 random starting point, W_1 random direction, then compute $L(W_0 + aW_1)$ for different values of a.



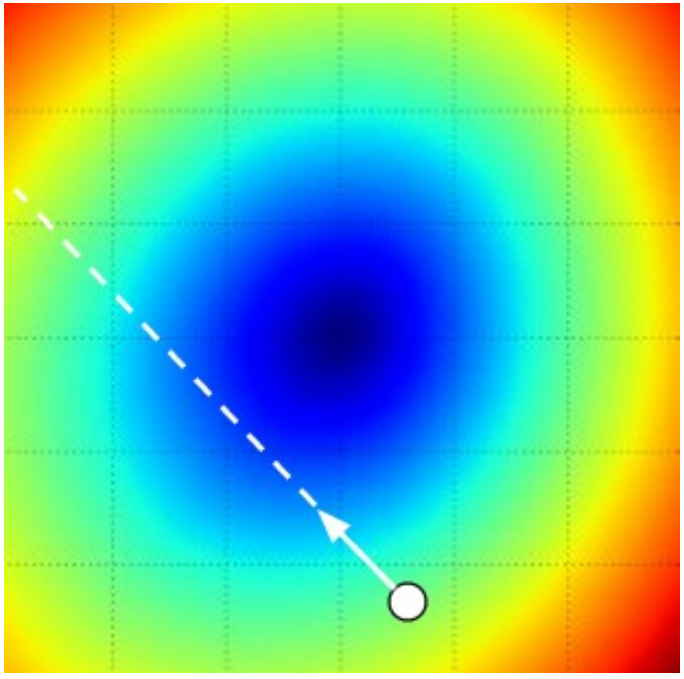
(averaged across all images, x_i)

Optimisation

- Random search
- Random local search
- Gradient descent (numerical or analytical)

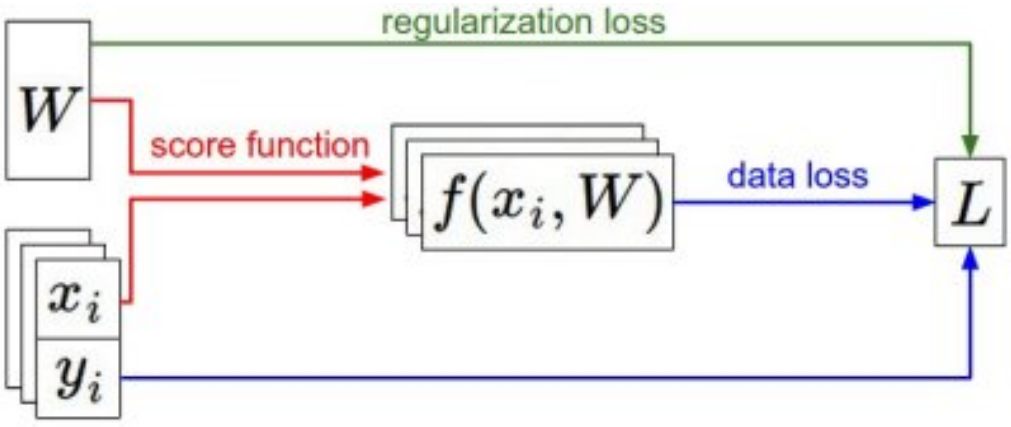
Hyperparameters

Step size or learning rate



Batch size:
Compute the gradient over batches (e.g. 32, 64, 128...) of the training data.

Wrap up



The 3 elements: score function, loss function, optimisation.

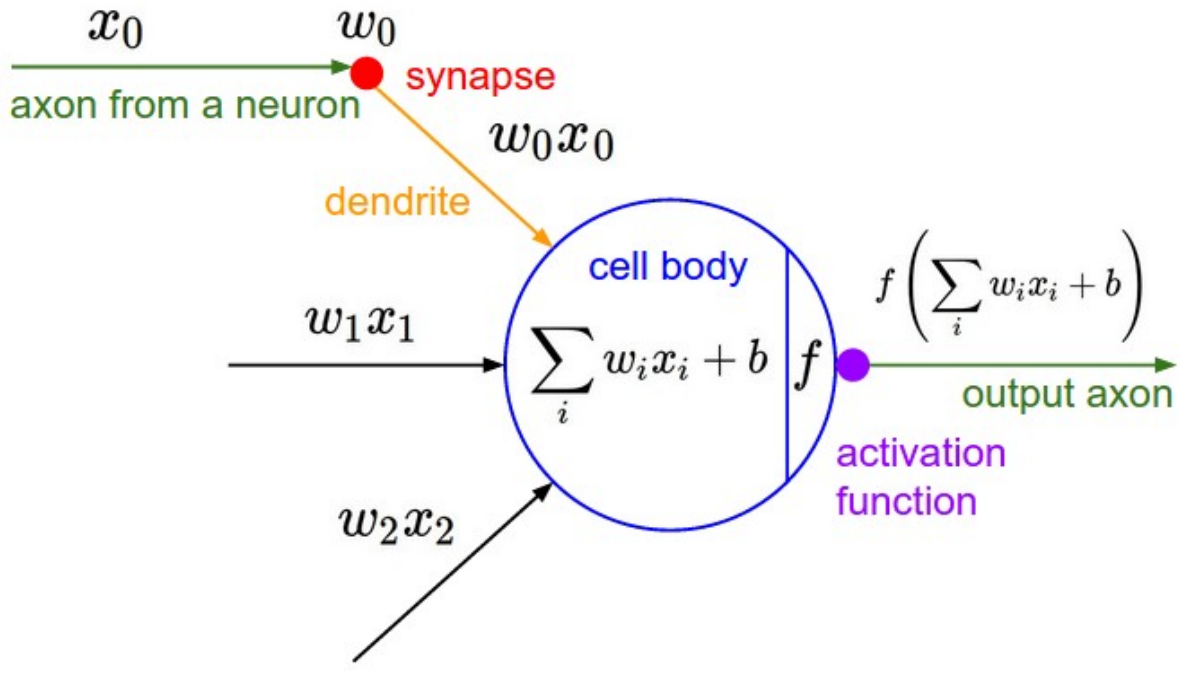
Next: let's put them all together in a neural network.

Intended Learning Outcomes

You are now able to:

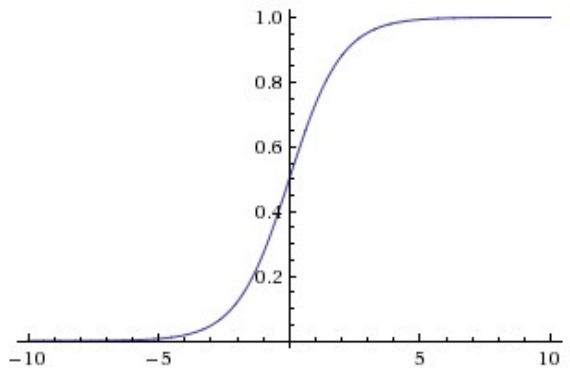
- Describe the three key components of a classifier: score function, loss function, optimisation

Neurons

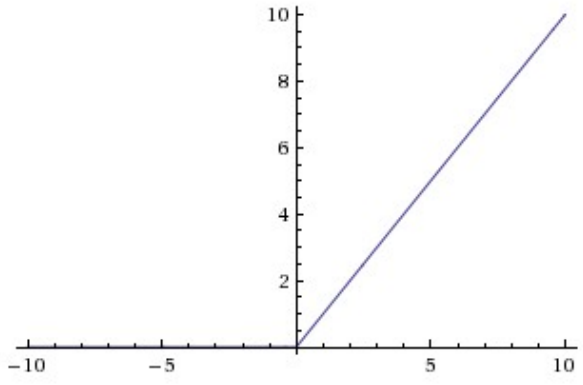

x_0
w_0
axon from a neuron
synapse
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$

Activation functions

It defines the *firing rate*



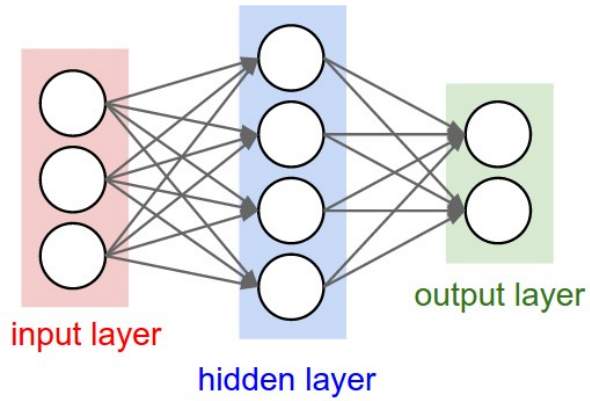
Sigmoid non-linearity squashes real numbers to range between $[0, 1]$



Rectified Linear Unit (ReLU):
$f(x) = max(0, x)$

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.

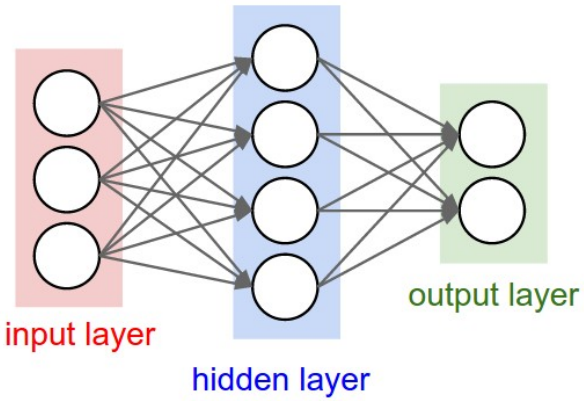


A 2-layer Neural Network

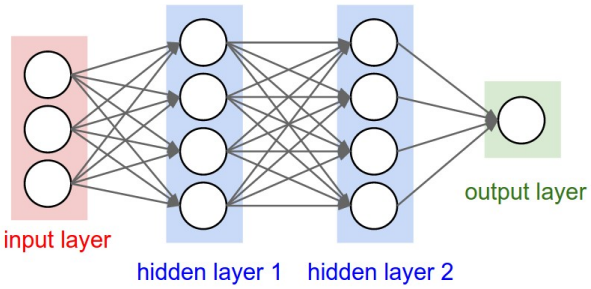
Size:

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.



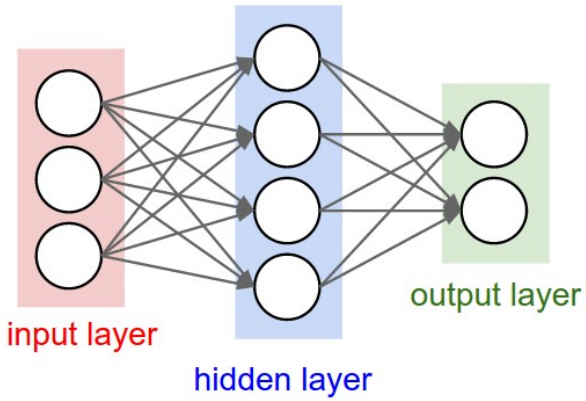
A 2-layer Neural Network
Size: $4 + 2 = 6$ neurons, $[3 \times 4] + [4 \times 2] = 20$ weights and $4 + 2 = 6$ biases, for a total of 26 learnable parameters.



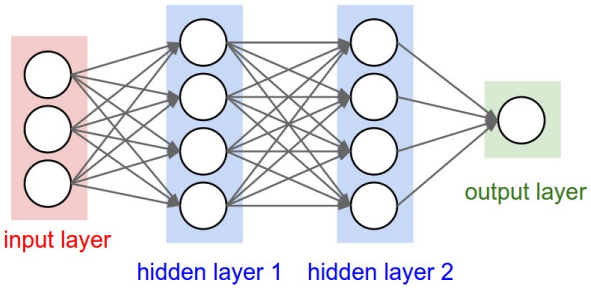
A 3-layer Neural Network
Size:

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.



A 2-layer Neural Network
Size: $4 + 2 = 6$ neurons, $[3 \times 4] + [4 \times 2] = 20$ weights and $4 + 2 = 6$ biases, for a total of 26 learnable parameters.



A 3-layer Neural Network
Size: $4 + 4 + 1 = 9$ neurons, $[3 \times 4] + [4 \times 4] + [4 \times 1] = 12 + 16 + 4 = 32$ weights and $4 + 4 + 1 = 9$ biases, for a total of 41 learnable parameters.

Representational power

Given any continuous function $f(x)$ and some $\epsilon > 0$, there exists a Neural Network $g(x; W)$ with one hidden layer (with a reasonable choice of non-linearity, e.g. sigmoid) such that for all x, $| f(x) - g(x) | < \epsilon$.

In other words, the neural network can approximate any continuous function.

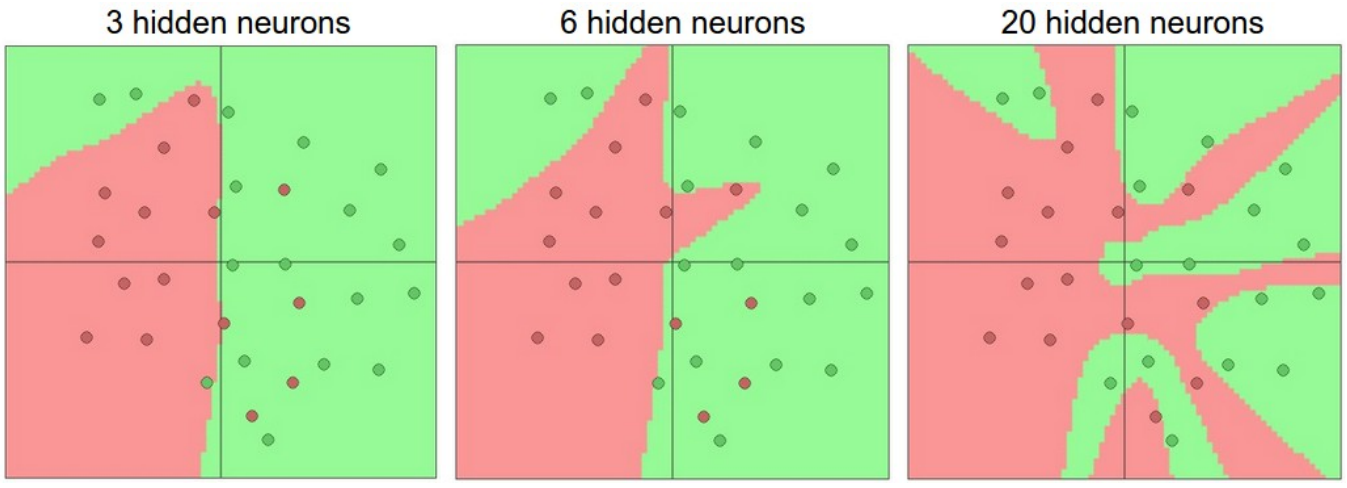
In practice, more layers work better...

How to build a neural network

Setting up the:

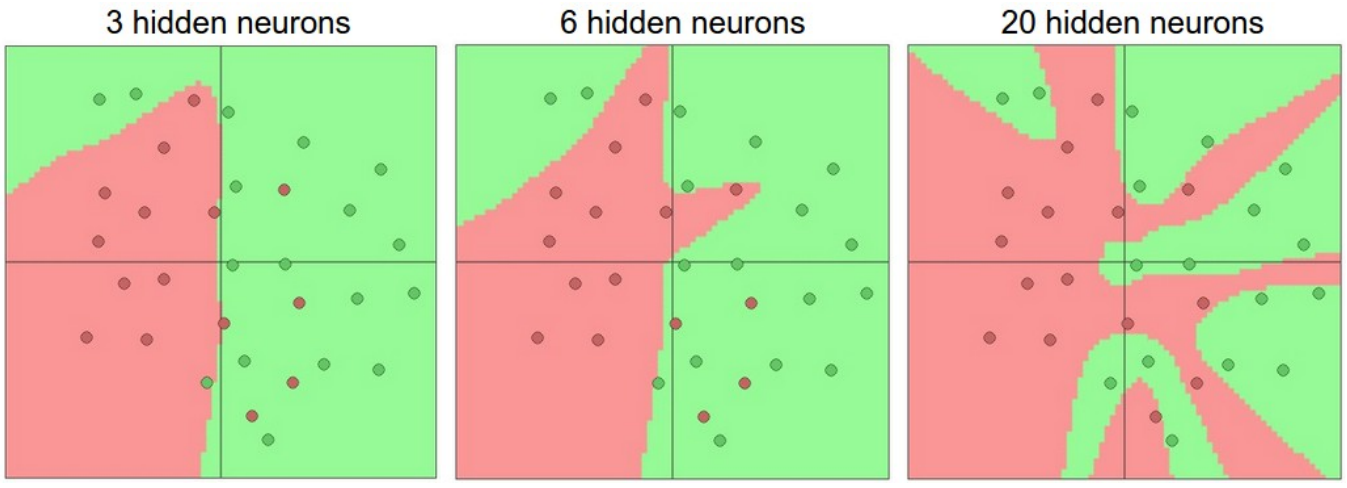
1. architecture
2. data
3. model
4. learning

Setting up the architecture



Capacity vs. ?

Setting up the architecture



Capacity vs. ? Overfitting

We aim at a better **generalisation**.

Setting up the data

Data preprocessing:

- mean subtraction
- normalisation
- PCA and Whitening

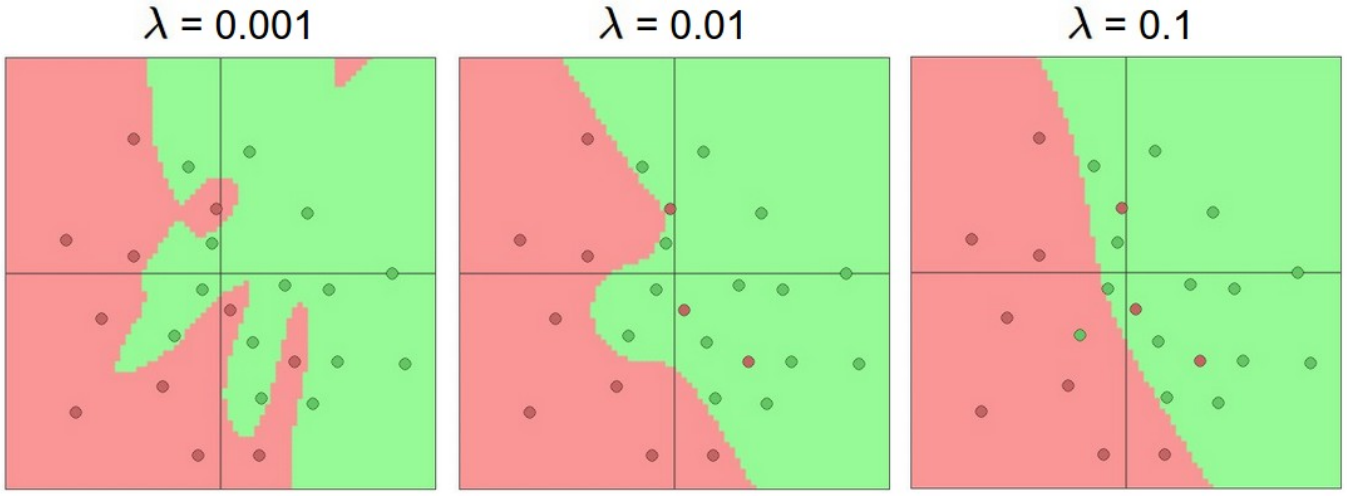
Setting up the model

Weight initialisation:

- all zero
- small random numbers
- calibrate the variances
- sparse

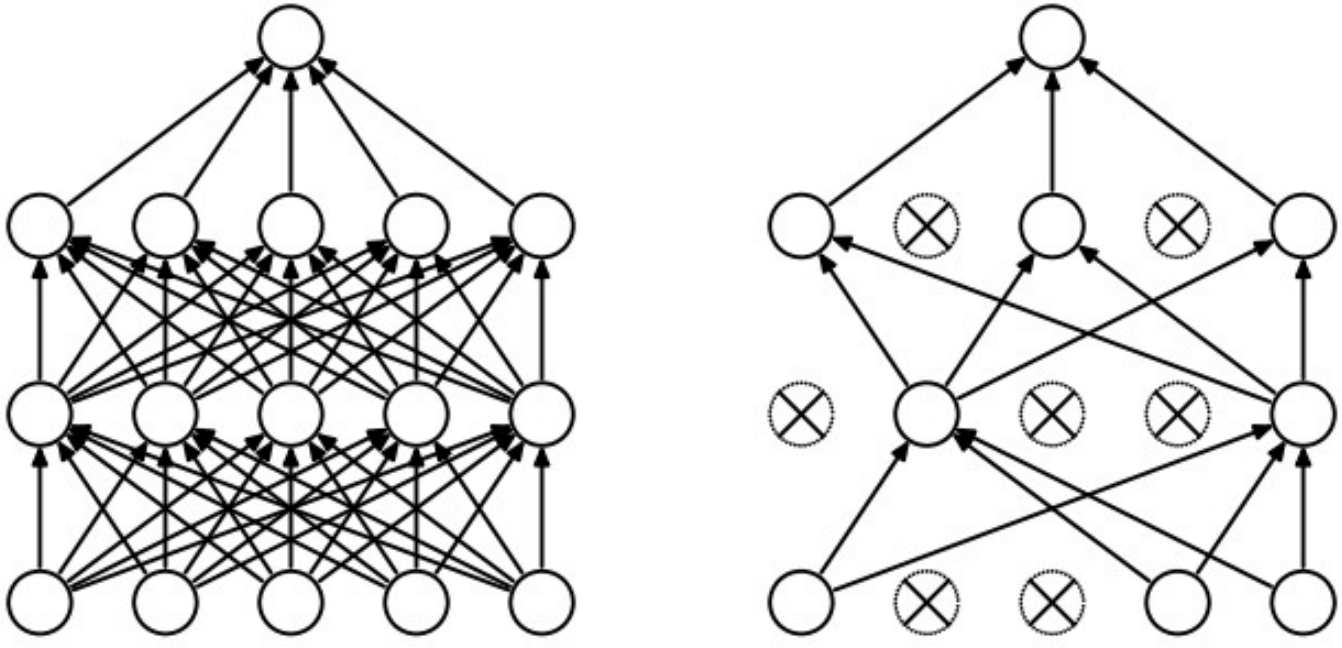
Setting up the model

Regularization



Options: L2, L1, maxnorm and dropout.

Dropout



(a) Standard Neural Net

(b) After applying dropout.

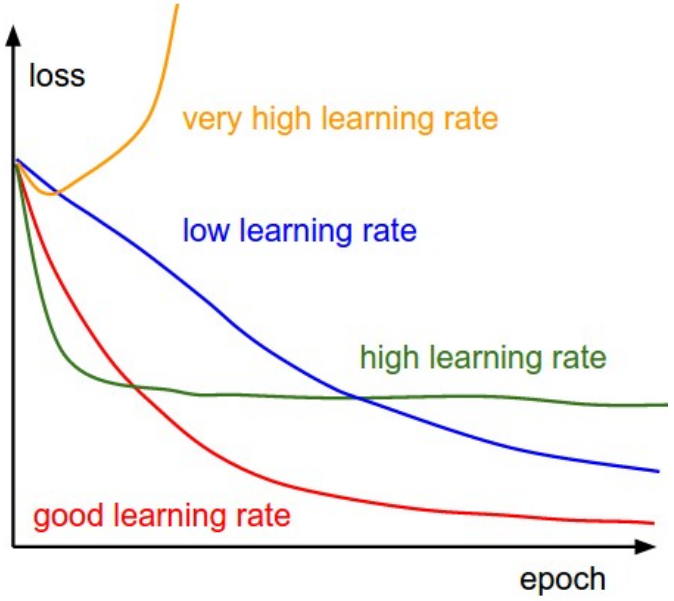
Dropout can be interpreted as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data.

Setting up the model

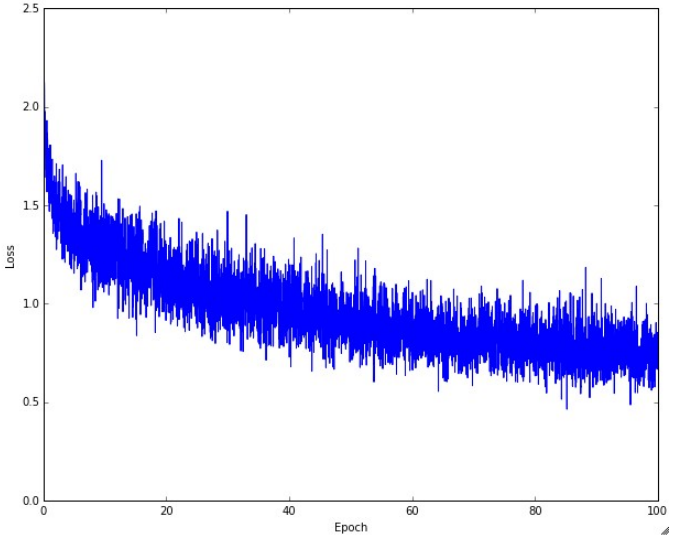
Loss functions:

- SVM (hinge loss)
- cross-entropy
- hierarchical softmax
- attribute classification
- regression (?)

Setting up the learning



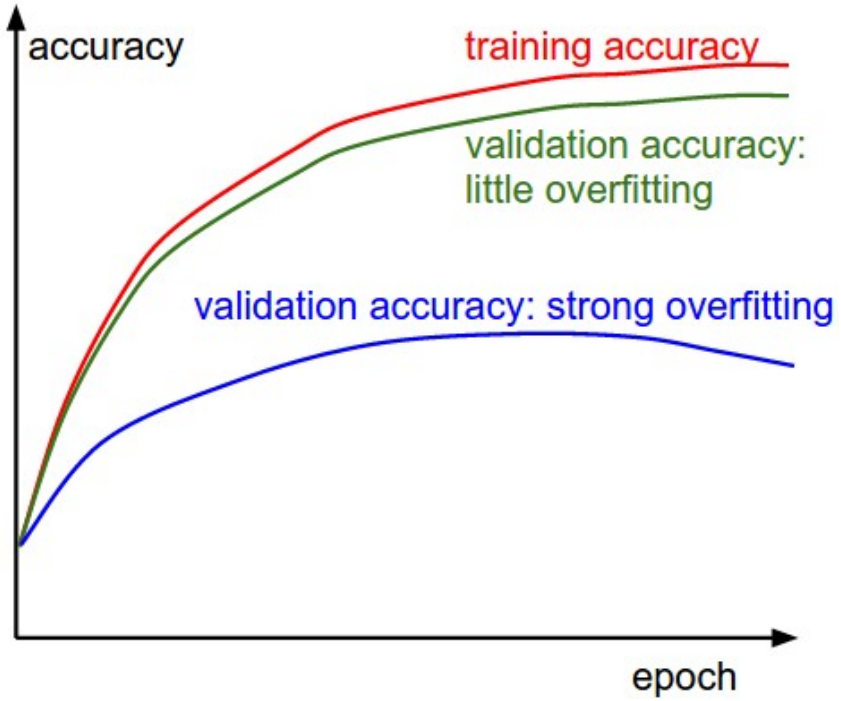
effects of different learning rates



loss decay

Setting up the learning

Training vs. validation accuracy



Wrap up

- Neural Networs are made of layers of neurons/units with activation functions
- Choice of the architecture: capacity vs overfitting
- Preprocessing of the data and choice of hyperparameters for the model and learning

ANNs to detect natural selection

PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Deep Learning for Population Genetic Inference

Sara Sheehan[1,2]*, **Yun S. Song**[2,3,4,5,6]*

1 Department of Computer Science, Smith College, Northampton, Massachusetts, United States of America, **2** Computer Science Division, UC Berkeley, Berkeley, California, United States of America, **3** Department of Statistics, UC Berkeley, Berkeley, California, United States of America, **4** Department of Integrative Biology, UC Berkeley, Berkeley, California, United States of America, **5** Department of Mathematics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America, **6** Department of Biology, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America

Matteo: open the paper

ANNs to detect natural selection

Deciphering signatures of natural selection via deep learning

Xinghu Qin[1*], Charleston W. K. Chiang[2], Oscar E. Gaggiotti[1*]

[1] Centre for Biological Diversity, Sir Harold Mitchell Building, University of St Andrews, Fife, KY16 9TF, UK

[2] Center for Genetic Epidemiology, Keck School of Medicine & Department of Quantitative and Computational Biology, University of Southern California, USA

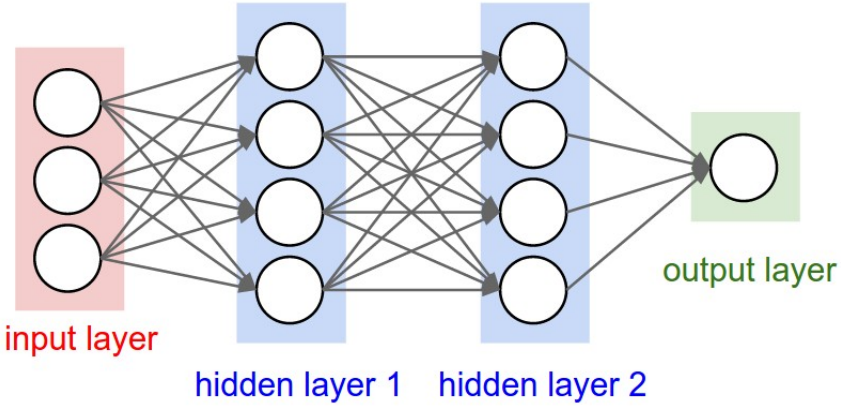
Intended Learning Outcomes

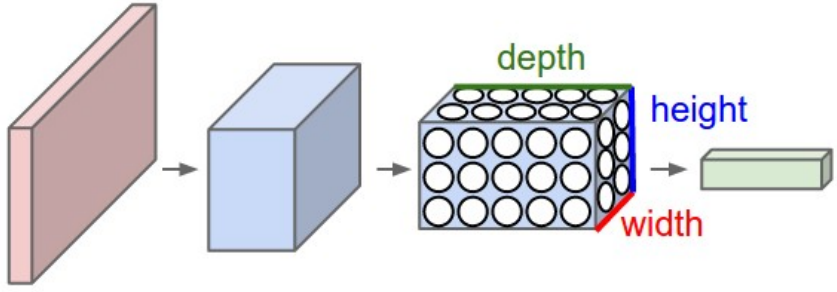
By the end of this session, you will be able to:

- Describe the three key components of a classifier: score function, loss function, optimisation
- Identify the elements of a neural networks, including neurons and hyper-parameters
- Illustrate the layers in a neural network
- Demonstrate how to implement, train and evaluate neural networks in `python`

What about images or highly-dimensional data (like images)? Can we use neural networks straight from individual data points? What's the issue?

Convolutional Neural Networks (CNN)





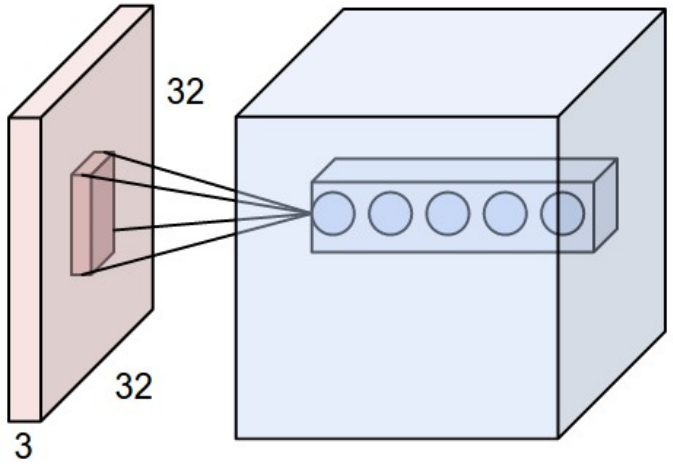
A CNN arranges its neurons in three dimensions (width, height, depth). Every layer of a CNN transforms the 3D input volume to a 3D output volume of neuron activations. Neurons in a layer are connected only to a small region of the layer before it.

CNN architecture

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer

We will stack these layers to form a full CNN architecture.

Convolutional layer



Set of learnable filters which *slides* across the width and height of the input volume.
3 hyper-parameters: depth (nr of filters), stride, zero-padding.

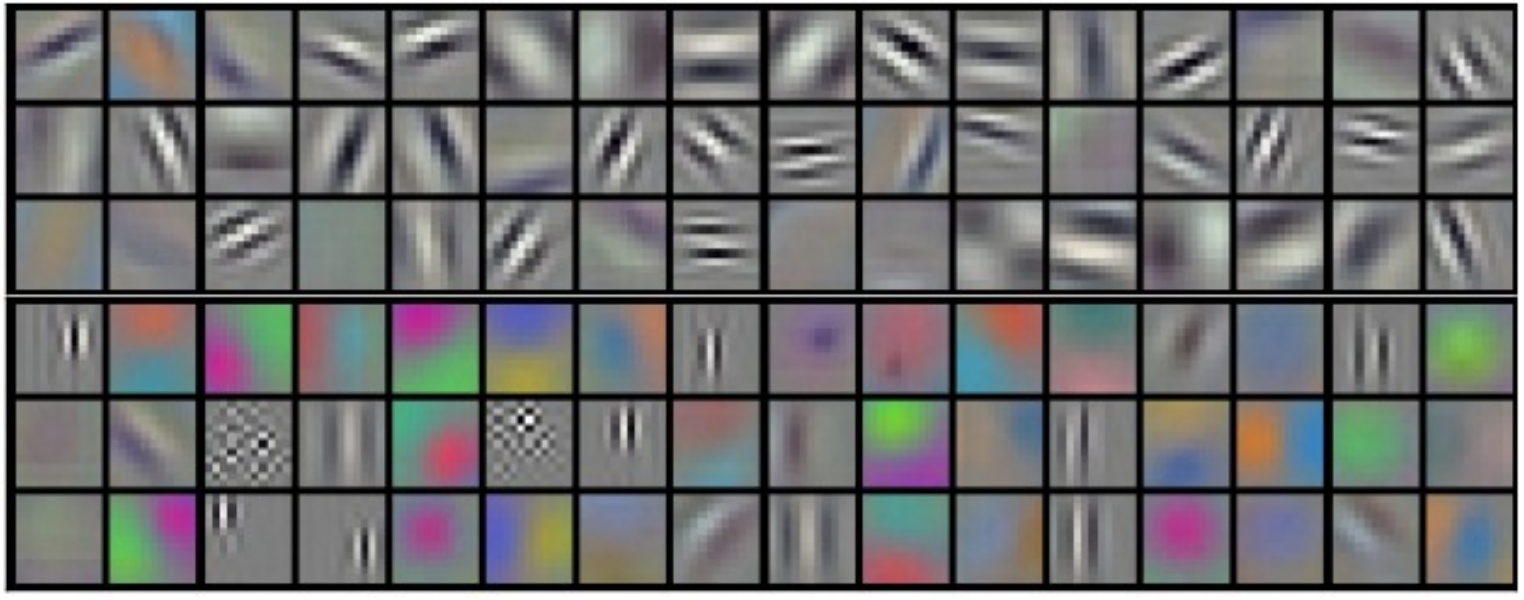
Convolutional layer

- Accepts a volume of size $W_1 x H_1 x D_1$
- Requires 4 parameters: number of filters K, their size F, the stride S, the amount of zero padding P
- Produces a volume of size $W_2 x H_2 x D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$

Usually: $F = 3, S = 1, P = 1$.

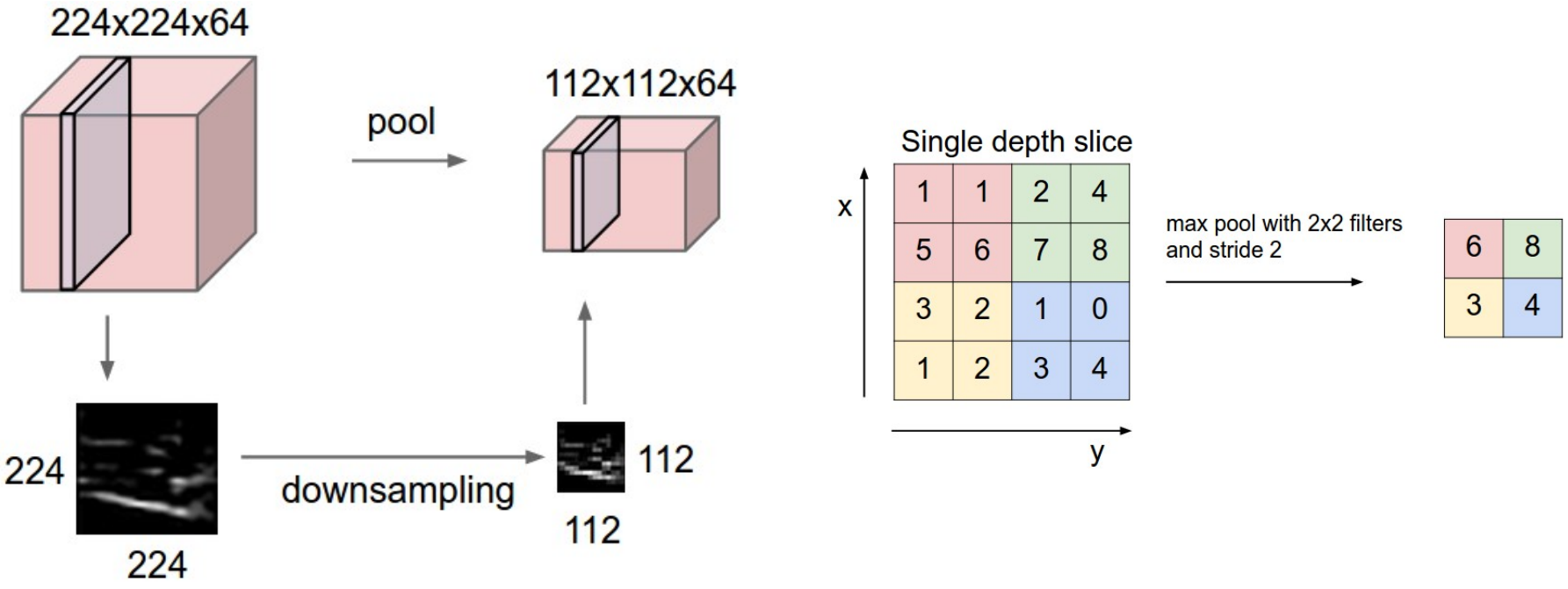
Demo: `http://cs231n.github.io/convolutional-networks/`

Weights in filters



Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice

Pooling layer



Size will influence the proportion of weights retained.

Layer patterns

INPUT $\rightarrow$ [[CONV $\rightarrow$ RELU]*N $\rightarrow$ POOL?]*M $\rightarrow$ [FC $\rightarrow$ RELU]*K $\rightarrow$ FC

More examples:
`http://cs231n.github.io/convolutional-networks/`

Applications

Now you know the theory behind deep learning. How would you

use this technology in population genetics?
Which tasks do you think you can perform?
What is your training data?

Applications

ML/DL algorithms in population genetics are typically trained with synthetic data sets.
They can be considered part of the likelihood-free simulation-based techniques, such as Approximate Bayesian Computation (ABC).

What are the advantages of DL over ABC?

Applications

ML/DL algorithms in population genetics are typically trained with synthetic data sets.
They can be considered part of the likelihood-free simulation-based techniques, such as Approximate Bayesian Computation (ABC).

What are the advantages of DL over ABC?

no curse of dimensionality; capacity to handle any feature; less sensitive to poorly crafted summary statistics; eural networks are universal approximators

Which algorithms have been used as first applications of deep learning to detect selection?

Applications

To detect natural selection

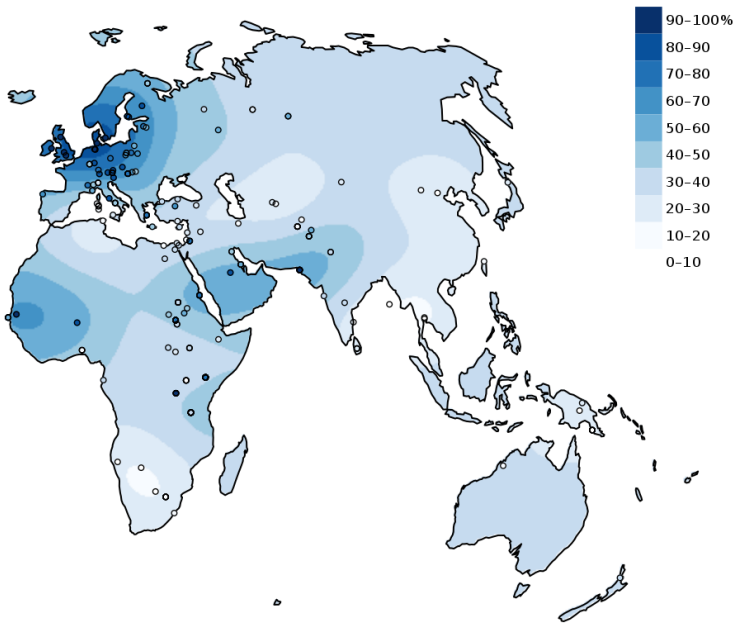
- CNN on summary statistics
- CNN in full genomic data
- Short-Term Memory (LSTM) architecture, a particular type of a Recurrent Neural Network (RNN)
- Graph Neural Networks using ARGs as input

What are the ongoing issues using DL in population genetics in your opinion? And which solutions?

What are the ongoing issues using DL in population genetics in your opinion? And which solutions?
biological data is uncertain, training by synthetic data, lack of interpretability

Practical

The case of LCT gene and lactase persistance (`https://en.wikipedia.org/wiki/Lactase_persistence`)



Task: to predict positive selection at LCT locus in European populations using deep learning in python.