

# Machine learning and deep learning for demographic and selection inference

Flora Jay, CNRS, LISN  
Matteo Fumagalli, QMUL  
Jean Cury, Erik Madison Bray, LISN

+ credits for some slides: T Sanchez





EvoGenomics.AI

sign up to the mailing list for seminars and training opportunities

# ImaGene: a convolutional neural network to quantify natural selection from genomic data

Luis Torada, Lucrezia Lorenzon, Alice Beddis, Ulas Isildak, Linda Pattini, Sara Mathieson & Matteo Fumagalli 

*BMC Bioinformatics* 20, Article number: 337 (2019) | [Cite this article](#)

SPECIAL ISSUE |  Open Access |  

## Distinguishing between recent balancing selection and incomplete sweep using deep neural networks

Ulas Isildak, Alessandro Stella, Matteo Fumagalli 

First published: 22 March 2021 | <https://doi.org/10.1111/1755-0998.13379>

## Detecting adaptive introgression in human evolution using convolutional neural networks



Graham Gower , Pablo Iñález Picazo, Matteo Fumagalli, Fernando Racimo  
University of Copenhagen, Denmark; Imperial College London, United Kingdom

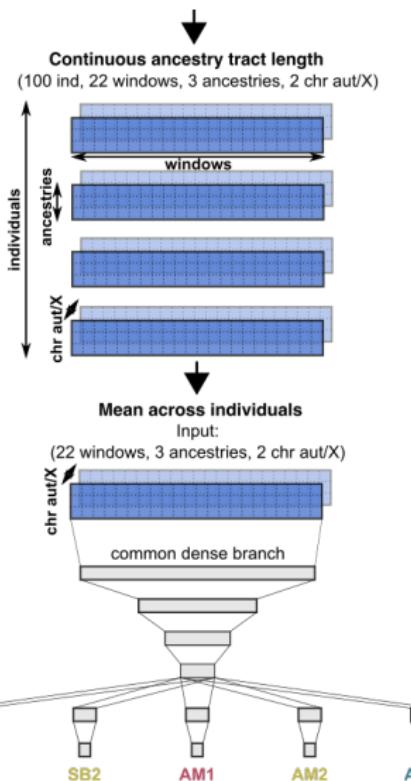
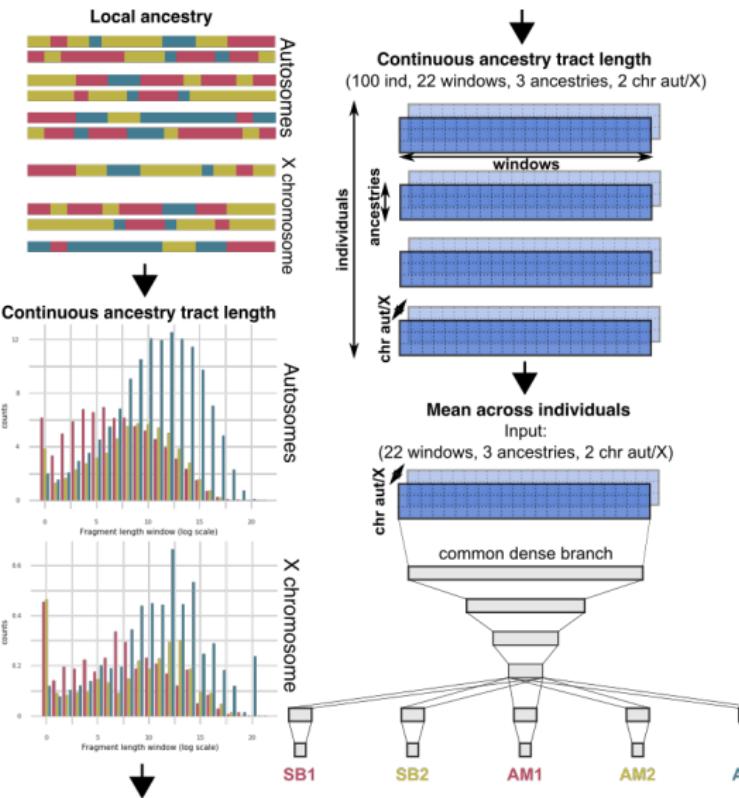
Tools and Resources · May 25, 2021

## Artificial intelligence can help spot traces of natural selection

by [Hayley Dunning](#)

22 March 2021

# Admixture and assortative mating



# Machine learning (ML) in evolutionary genomics

Morning session: introduction to

- basic concepts in supervised ML (Matteo)
- neural networks (Matteo)
- deep learning (Flora)
- unsupervised learning (Flora)

with examples from the literature.

# Machine learning (ML) in evolutionary genomics

Afternoon and evening sessions:

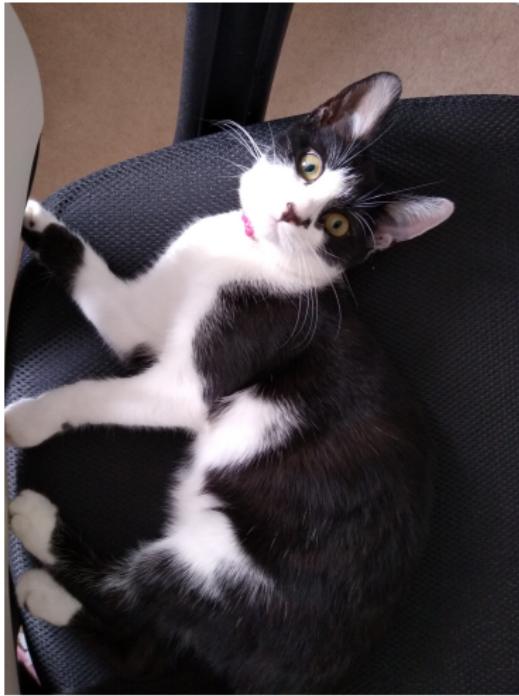
- simple neural networks with keras (Matteo)
- advanced architectures with pytorch (Flora)
- scalable deep learning with dnadna (Flora)

with applications on detecting selection and inferring demographic histories.

# Machine learning for population genetics

A (gentle and brief) introduction

Matteo Fumagalli



# Intended Learning Outcomes

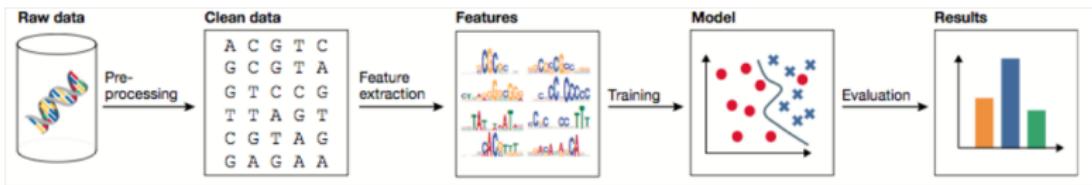
By the end of this very first part, you **will** be able to:

- 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

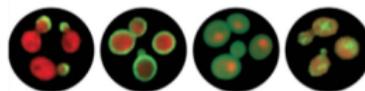
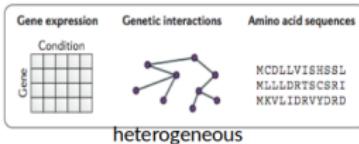
	loc1	loc2	...	Sport activity	Hours of free time	...	Disease X ?
ind1	A/A	C/C					
ind2	T/A	C/C					
...							

multidimensional and heterogeneous data

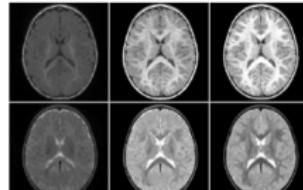
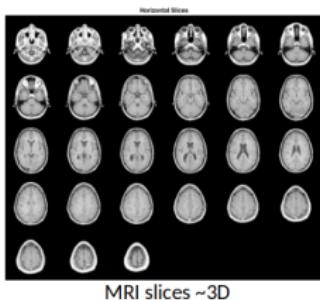
# What is the data?

- 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)



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

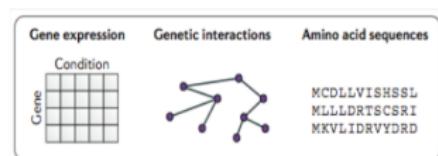


# What is the learning task?

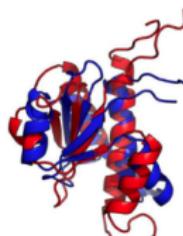
- Data with or without label
- What's a label ? a target class or a target value observed for each sample  
*Data are not always labeled. They can also have multiclass labels*  
ex : pic of dog/person/car..., price of house, level of cholesterol
- Task/objective ?

# What is the learning task?

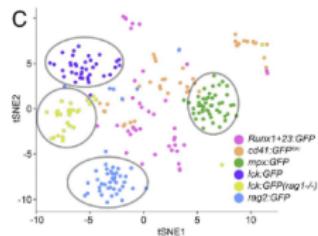
- Data with or without label
- What's a label ? a target class or a target value observed for each sample  
*Data are not always labeled. They can also have multiclass labels*  
ex : pic of dog/person/car..., price of house, level of cholesterol
- Task/objective ?
- Task/objective ?



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)

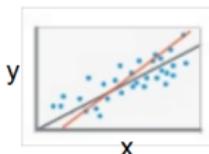
# Supervised learning

- 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),
- ...

(Flora will cover unsupervised learning later today)

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?



```
08 02 22 97 88 15 00 40 00 75 04 05 07 78 52 12 80 97 91  
49 49 99 40 17 81 18 57 60 87 17 40 99 43 69 33 41 54 62 00  
81 49 31 75 55 79 14 29 93 71 40 47 05 08 30 03 49 13 36 65  
52 70 95 28 04 69 11 42 05 05 56 01 32 54 73 37 02 36 93  
22 31 16 74 51 03 05 89 41 92 36 54 22 40 40 26 66 33 13 80  
24 47 30 00 39 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50  
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 66 70  
67 26 20 02 62 12 20 95 63 94 39 63 08 40 91 66 49 95 21  
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72  
21 36 23 09 75 00 76 44 20 45 35 14 00 81 33 97 34 31 33 95  
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92  
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57  
06 56 00 45 35 71 89 07 05 44 44 37 44 60 21 54 51 54 17 58  
19 80 81 63 05 94 47 69 28 73 92 13 86 82 17 77 04 89 55 40  
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66  
04 44 65 57 57 62 20 72 05 46 33 67 66 55 12 32 63 93 53 69  
04 42 16 73 35 10 10 11 24 94 72 18 08 46 29 32 40 62 76 36  
20 69 36 41 72 30 23 88 34 00 00 69 02 67 59 85 74 04 36 16  
20 73 35 29 78 31 90 01 74 31 49 71 49 00 00 16 23 57 05 54  
01 70 54 71 03 51 54 69 16 92 33 48 61 43 52 01 59 00 00 48
```

What the computer sees

→ 82% cat  
image classification 15% dog  
2% hat  
1% mug

Is it THAT difficult?

# Challenges



Scale variation



Deformation



Occlusion



- 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

$$\begin{array}{c|c|c} \text{test image} & \text{training image} & \text{pixel-wise absolute value differences} \\ \hline \begin{matrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{matrix} & - & \begin{matrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{matrix} \\ \hline & = & \begin{matrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{matrix} \end{array} \rightarrow 456$$

# The choice of distance

L1 distance:  $d_1(l_1, l_2) = \sum_{pixel} |l_1^p - l_2^p|$

L2 distance:  $d_1(l_1, l_2) = \sqrt{\sum_{pixel} (l_1^p - l_2^p)^2}$

What's their accuracy?

What's human accuracy?

What's state-of-the-art neural networks' accuracy?

Let's give it a try!

# IUCN Red List of Threatened Species

LC: least concern



EN: endangered



VU: vulnerable



CR: critically endangered



The challenge: predict whether a species is endangered, vulnerable or of least concern from genomic data.

Let's try it!

Matteo: open notebook



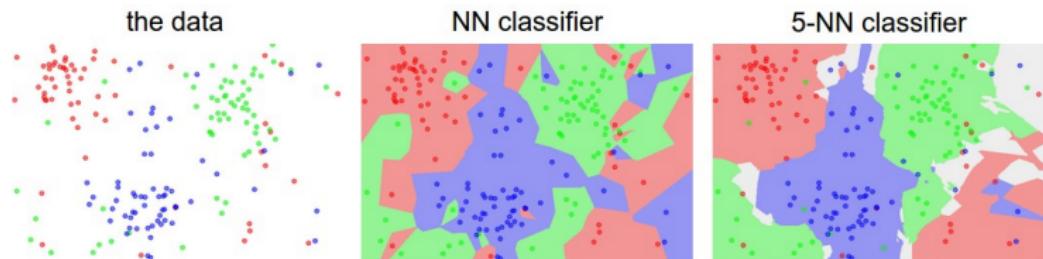
*Ursus arctos marsicanus*

# Nearest Neighbour Classifier

What went wrong when using this algorithm to predict the **label** "conservation status" from population "genotype" **data**? Any undesired behaviours? Any suggestions on how we can improve our prediction accuracy?

Matteo: mentimeter [www.menti.com](http://www.menti.com) with code 2032 9375

# k-Nearest Neighbour Classifier



**Figure 2:** 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?

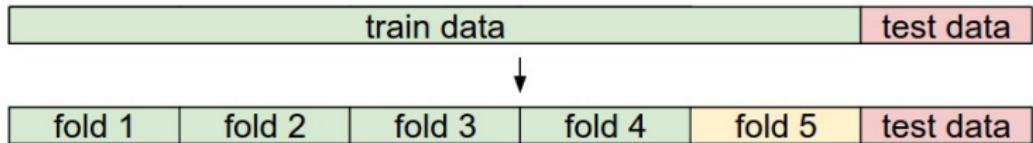
# Validation test



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 3:** 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  
 $i = 1 \dots N$  and  $y_i \in 1 \dots K$

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.

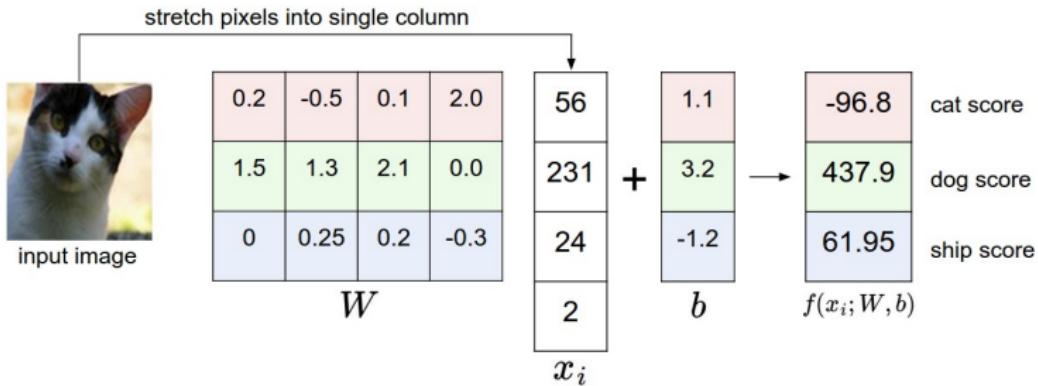
What are the dimensions of  $x_i$ ,  $W$  and  $b$ ?

$W$  has size  $[K \times D]$

$x_i$  has size  $[D \times 1]$

$b$  has size  $[K \times 1]$

# Linear classifier



[https://en.wikipedia.org/wiki/Matrix\\_multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication)

# Interpreting a linear classifier (i)

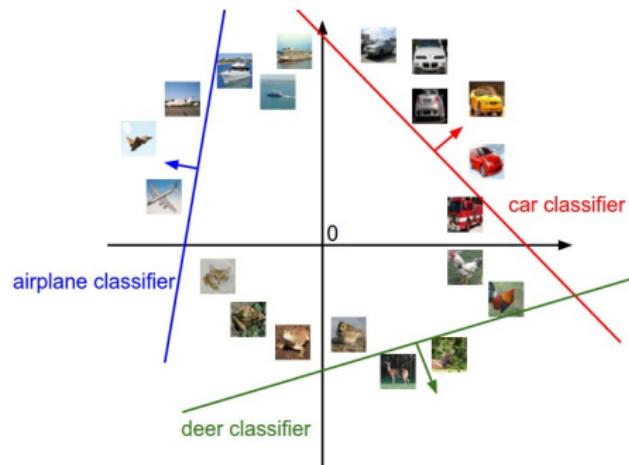


0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

$W$



## Interpreting a linear classifier (ii)



$$\begin{array}{c} \downarrow \\ \begin{matrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{matrix} \\ W \end{array} + \begin{array}{c} \downarrow \\ \begin{matrix} 56 \\ 231 \\ 24 \\ 2 \end{matrix} \\ x_i \end{array} \quad b$$

# Interpreting a linear classifier (iii)

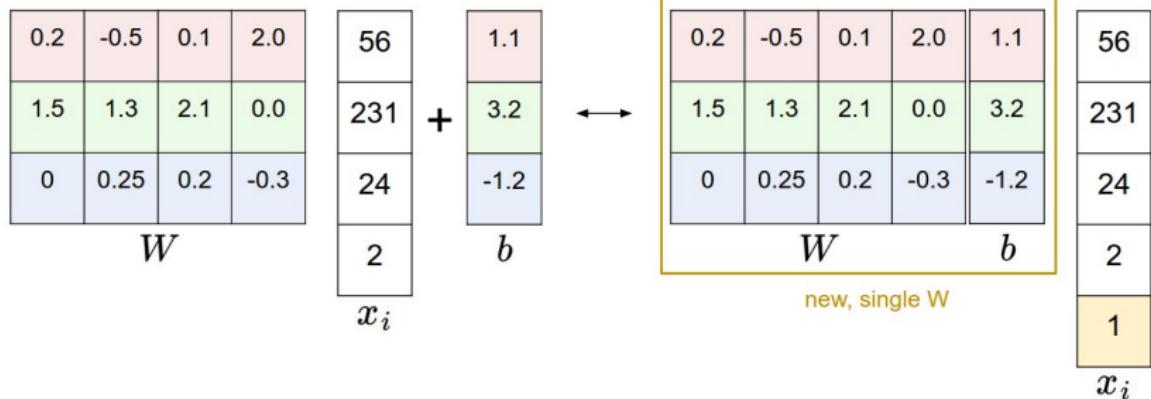
Template (or prototype) matching.



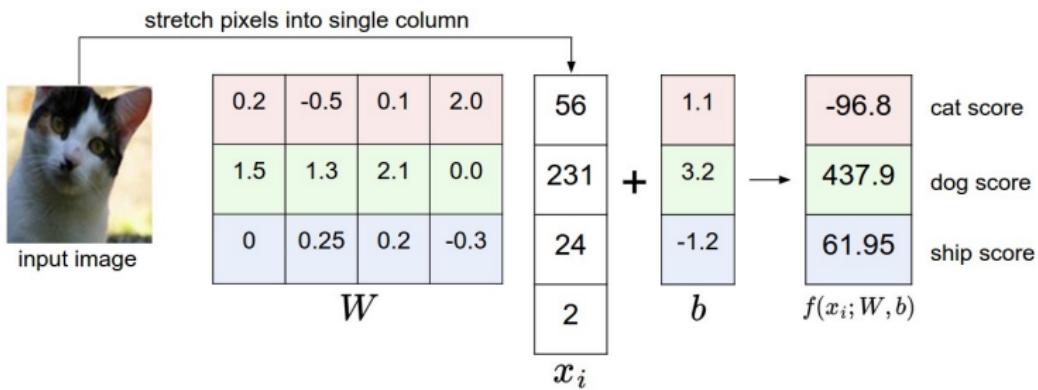
# Bias trick

Our new **score function**:

$$f(x_i; W) = Wx_i$$



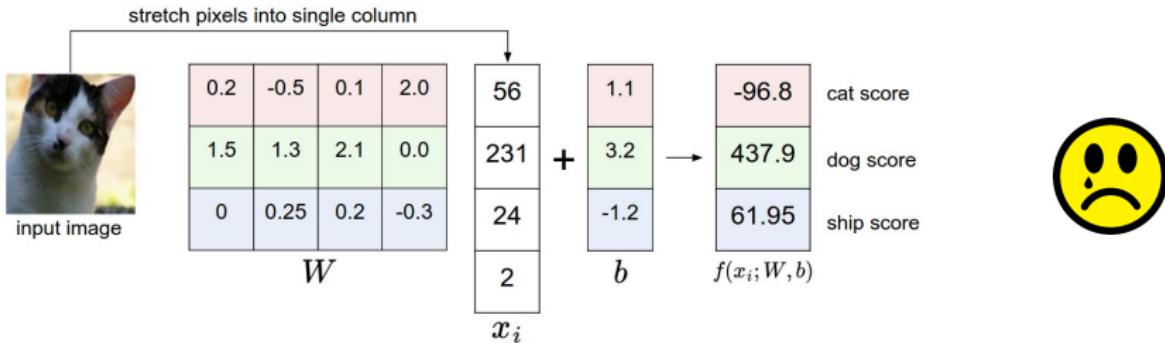
# Score function



$f : R^D \rightarrow R^K$  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 have a higher score ( $s_{y_i}$ ) than the incorrect ones ( $s_j$ ) by some fixed margin ( $\delta$ ).

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

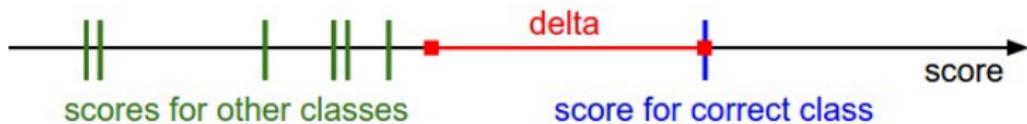
Example:

$$s = [13, -7, 11], y_i = 0, \delta = 10$$

$$L_i = ?$$

# Hinge loss

$$\max(0, -) \text{ or } \max(0, -)^2$$



# Regularisation

If  $W$  correctly classifies each sample, then all  $\lambda W$  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  $W_1$  is 2

additional cost for  $W_2$  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.

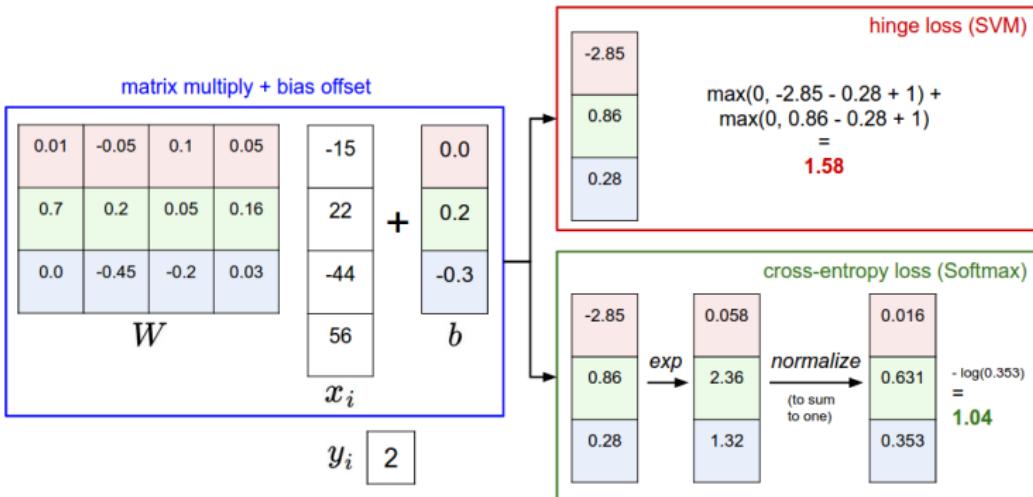
## Softmax classifier

Generalisation of the binary logistic regression classifier to multiple classes.

Cross-entropy loss function:

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

# 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 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 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

Copyright © 2010 by the Genetics Society of America  
DOI: 10.1534/genetics.110.116459

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

Pavlos Pavlidis,<sup>\*†</sup> Jeffrey D. Jensen<sup>†</sup> and Wolfgang Stephan<sup>\*</sup>

<sup>\*</sup>Department of Biology II, Ludwig-Maximilians-University Munich, 82152 Planegg, Germany and <sup>†</sup>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,<sup>\*†</sup> Nitin Udpa,<sup>\*</sup> Eran Halperin,<sup>†</sup> and Vineet Bafna<sup>‡</sup>**

<sup>\*</sup>Bioinformatics and Systems Biology Program, University of California, San Diego, California 92093, <sup>†</sup>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 <sup>‡</sup>Department of Computer Science and Engineering, University of California, San Diego, California 92093

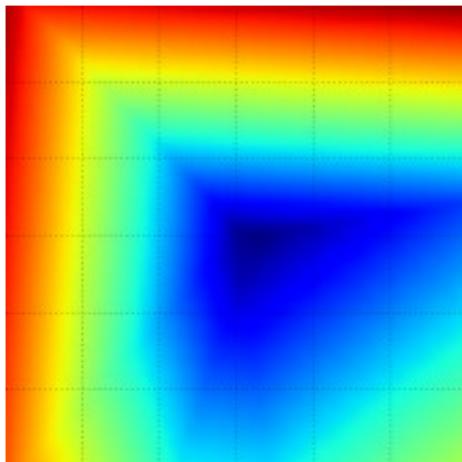
# Key components for classification tasks

- ➊ score function
- ➋ loss function
- ➌ 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,  $x_i$ )

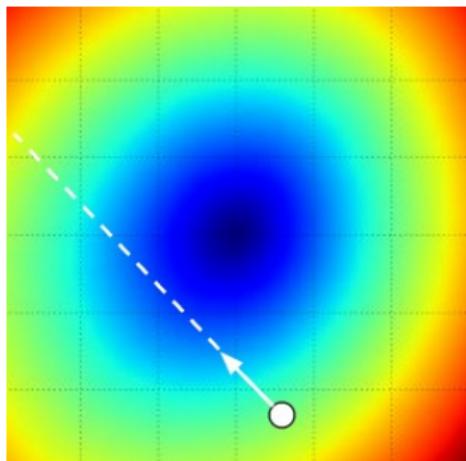
# Optimisation



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

# Hyperparameters

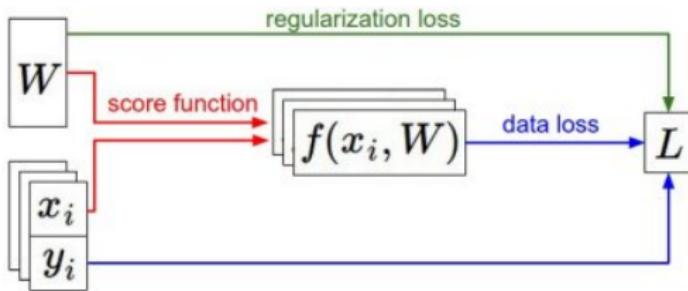
**Step size** or learning rate



**Batch size:**

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

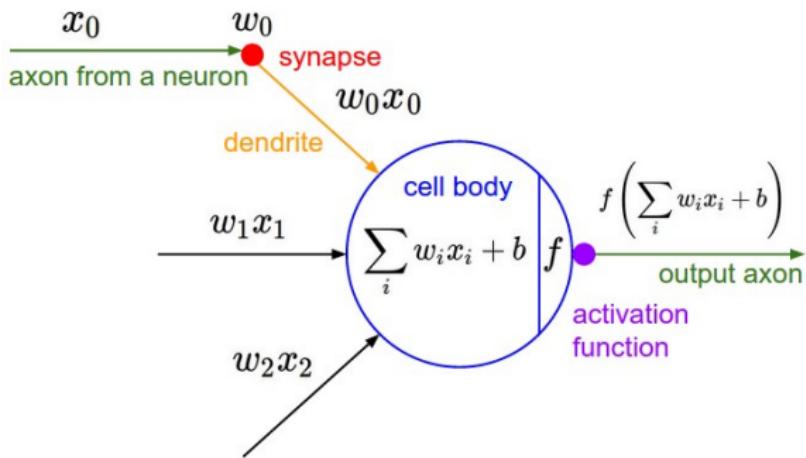
## Wrap up



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

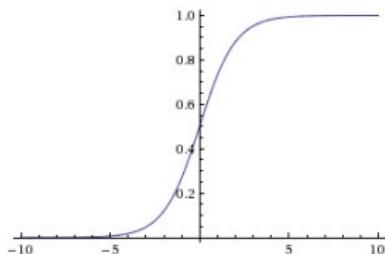
Next: let's put them all together in a neural network.

# Neurons

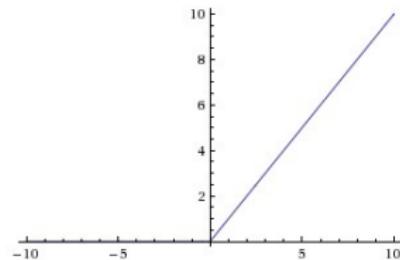


# Activation functions

It defines the *firing rate*



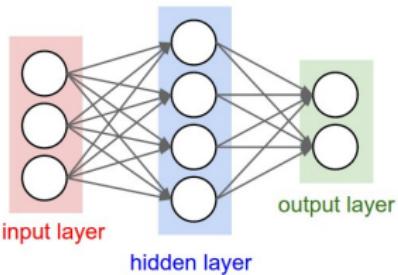
Sigmoid non-linearity squashes real numbers to range between  $[0, 1]$



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.

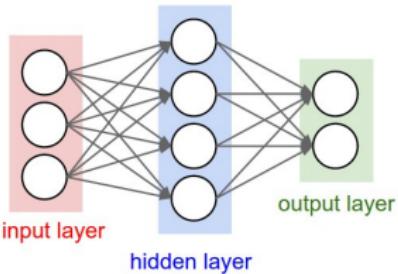


A 2-layer Neural Network

Size:

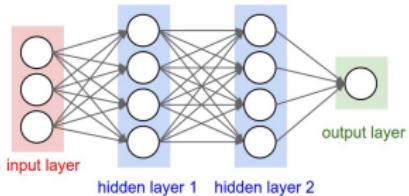
# Neural network architecture

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.

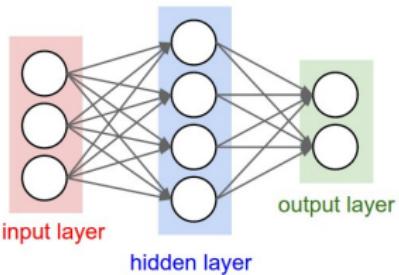


A 3-layer Neural Network

Size:

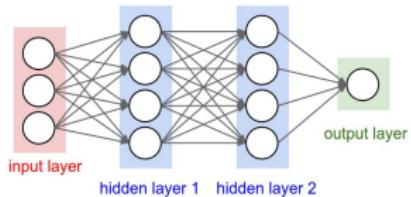
# Neural network architecture

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:  $4 + 2 = 9$  neurons,  $[3 \times 4] + [4 \times 4] + [4 \times 1] = 12 + 16 + 4 = 32$  weights and  $4 + 2 + 1 = 7$  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.

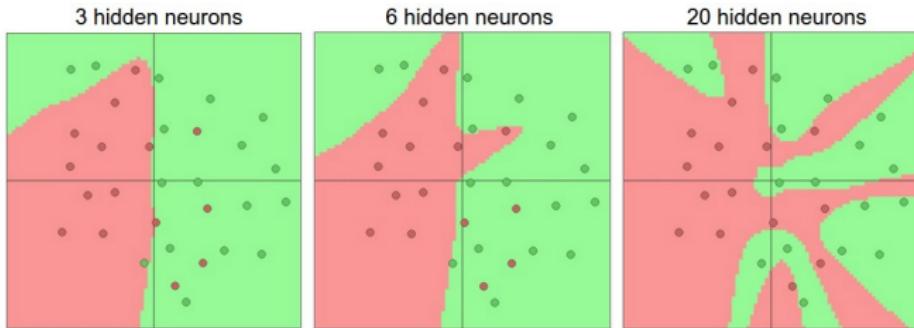
In practice, more layers work better...

# How to build a neural network

Setting up the:

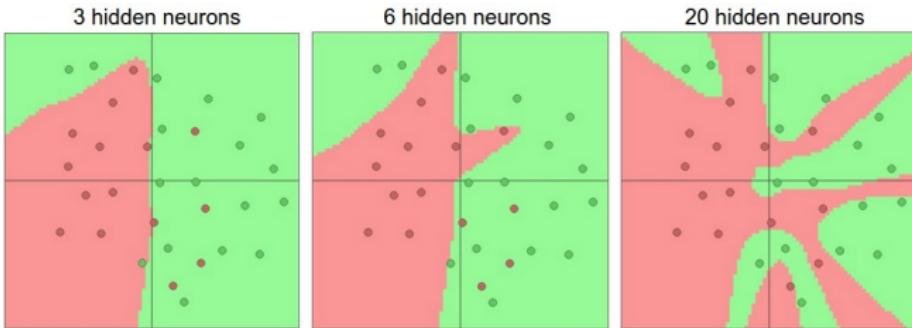
- ① architecture
- ② data
- ③ model
- ④ learning

# 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

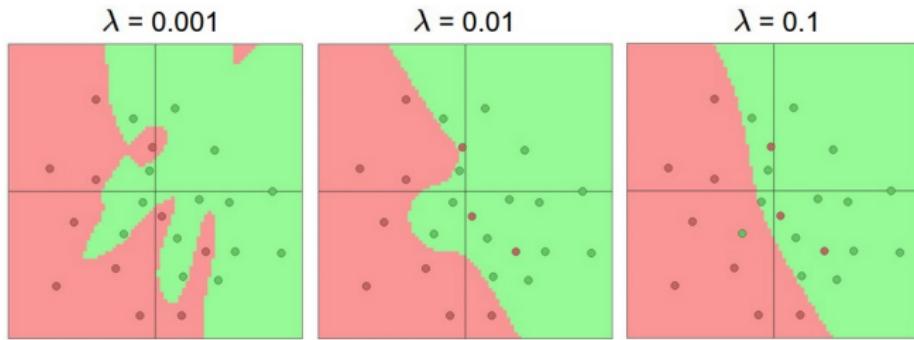
# Setting up the model

Weight initialisation:

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

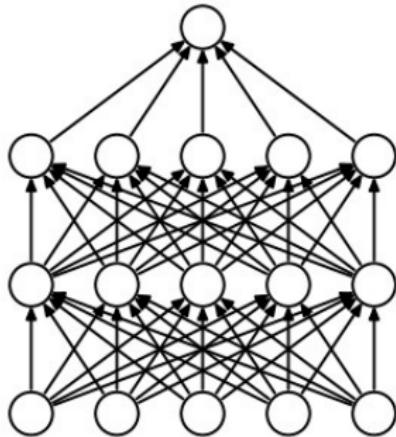
# Setting up the model

## Regularization

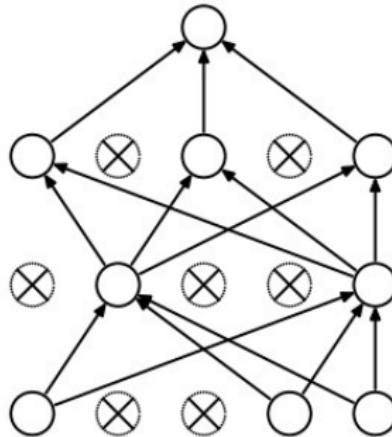


Options: L2, L1, maxnorm and dropout.

# Dropout



(a) Standard Neural Net



(b) After applying dropout.

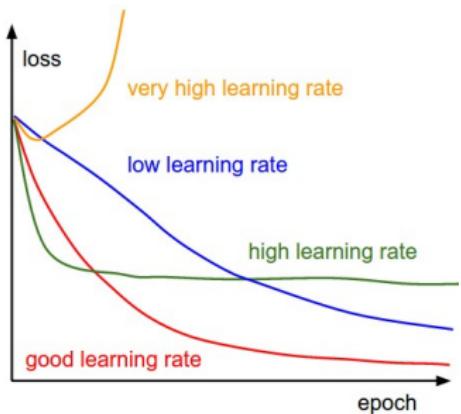
Dropout can be interpreted as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data.

# Setting up the model

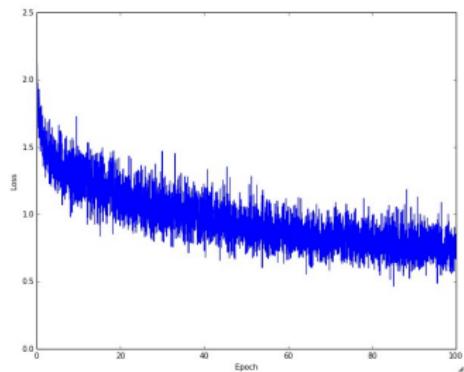
Loss functions:

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

# Setting up the learning



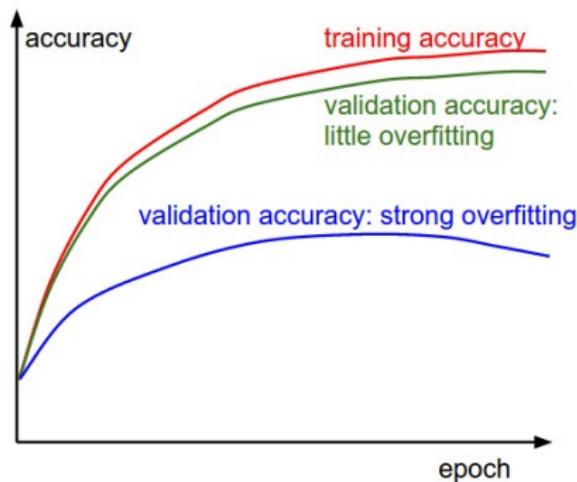
effects of different learning rates



loss decay

# Setting up the learning

Training vs. validation accuracy



## Wrap up

- Neural Networks 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



RESEARCH ARTICLE

## Deep Learning for Population Genetic Inference

Sara Sheehan<sup>1,2\*</sup>, Yun S. Song<sup>2,3,4,5,6\*</sup>

**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<sup>1\*</sup>, Charleston W. K. Chiang<sup>2</sup>, Oscar E. Gaggiotti<sup>1\*</sup>

<sup>1</sup> Centre for Biological Diversity, Sir Harold Mitchell Building, University of St Andrews,  
Fife, KY16 9TF, UK

<sup>2</sup> Center for Genetic Epidemiology, Keck School of Medicine & Department of Quantitative  
and Computational Biology, University of Southern California, USA

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

# Intended Learning Outcomes

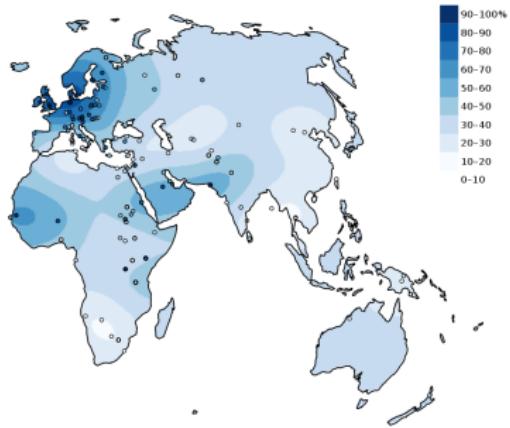
At the end of this session, you are now able to:

- 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

# Practical

The case of LCT gene and lactase persistence

([https://en.wikipedia.org/wiki/Lactase\\_persistence](https://en.wikipedia.org/wiki/Lactase_persistence))



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