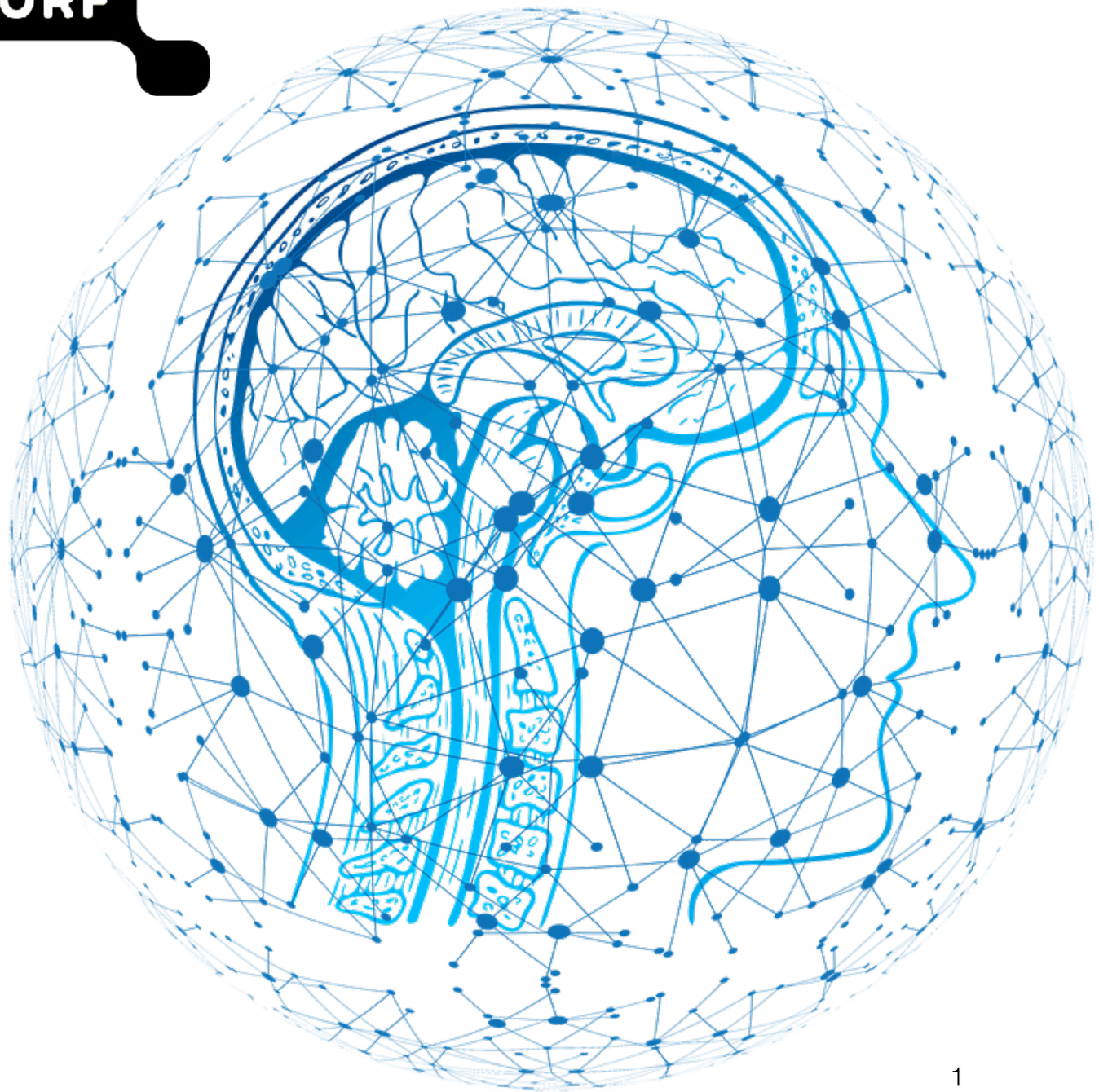


Introduction to Deep Learning

Maxwell Cai, Joris Mollinga
surf.nl

Objectives and Outline

Part I: Fundamentals, ML→DL, Common practices

- Theoretical introduction: 9:00 - 10:00
- Short break: 10:00 - 10:15
- Hands-on: 10:15 - 11:15

Part II: CNN

- Theoretical introduction: 11:15 - 11:45
- Hands-on: 11:45 - 12:30
- Lunch break: 12:30 - 13:30

Part III: RNN, sequential modelling

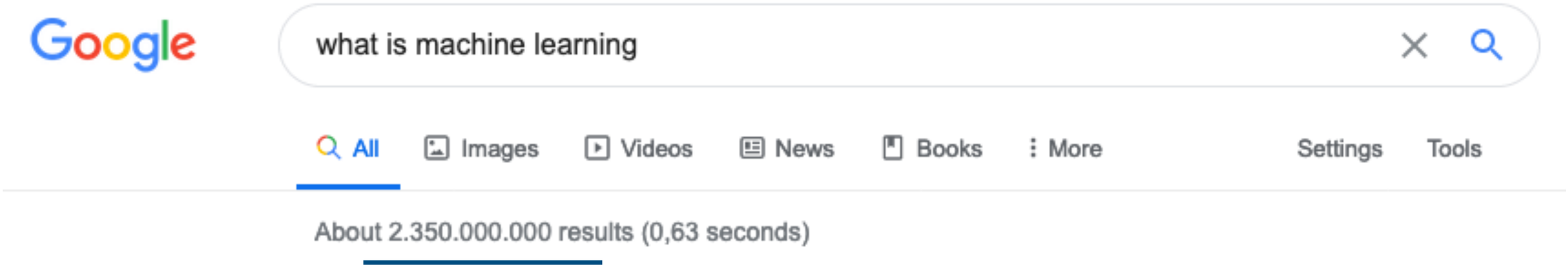
- Theoretical introduction: 13:30-14:15
- Hands-on: 14:15-15:15
- Short break: 15:15 - 15:30

Part IV: High Performance Machine Learning (optional if time permits)

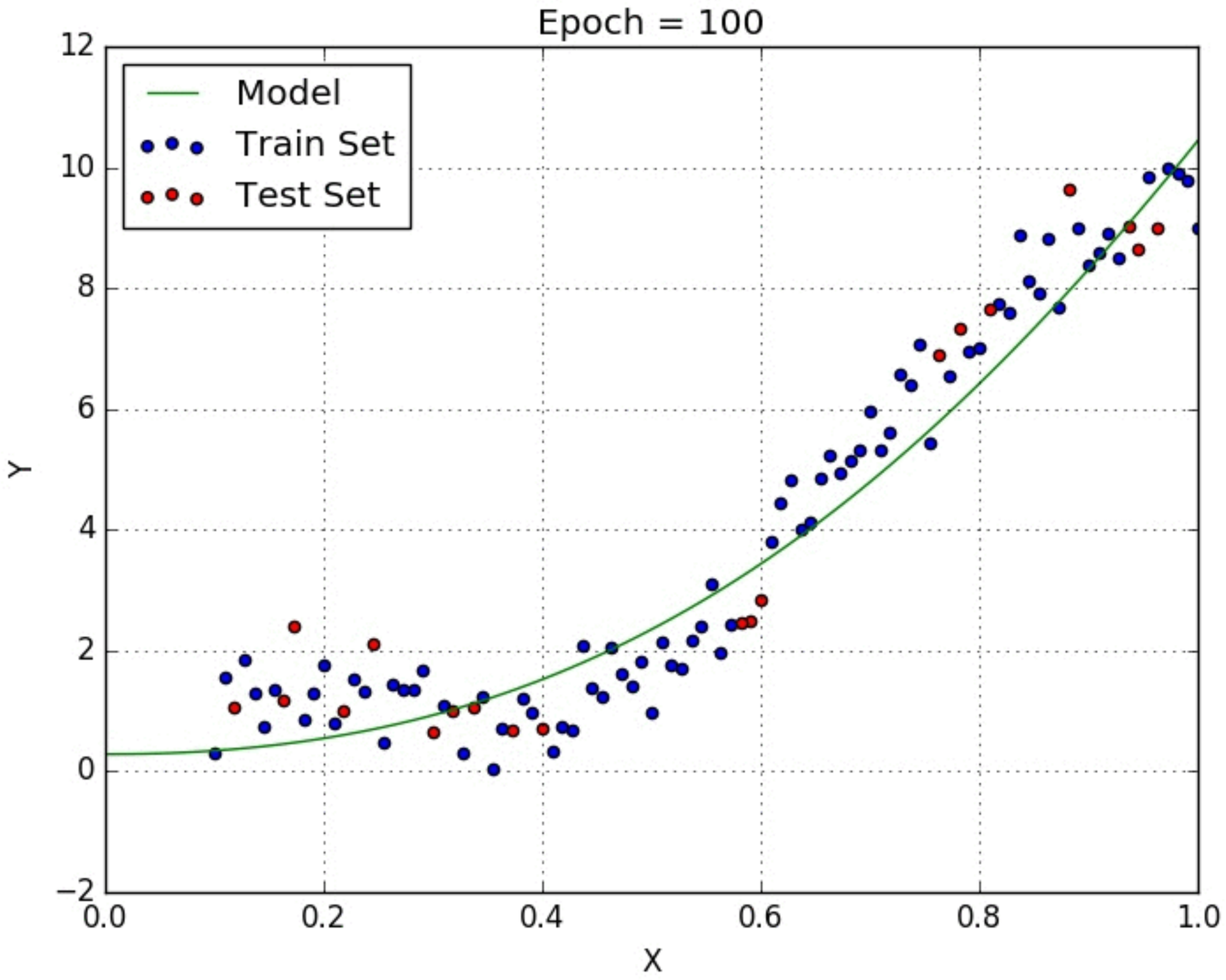
- Theoretical introduction: 15:30 - 16:00
- Hands-on: 16:00 - 17:00

What *is* machine learning?

Search: “what is machine learning”



What *is* machine learning?



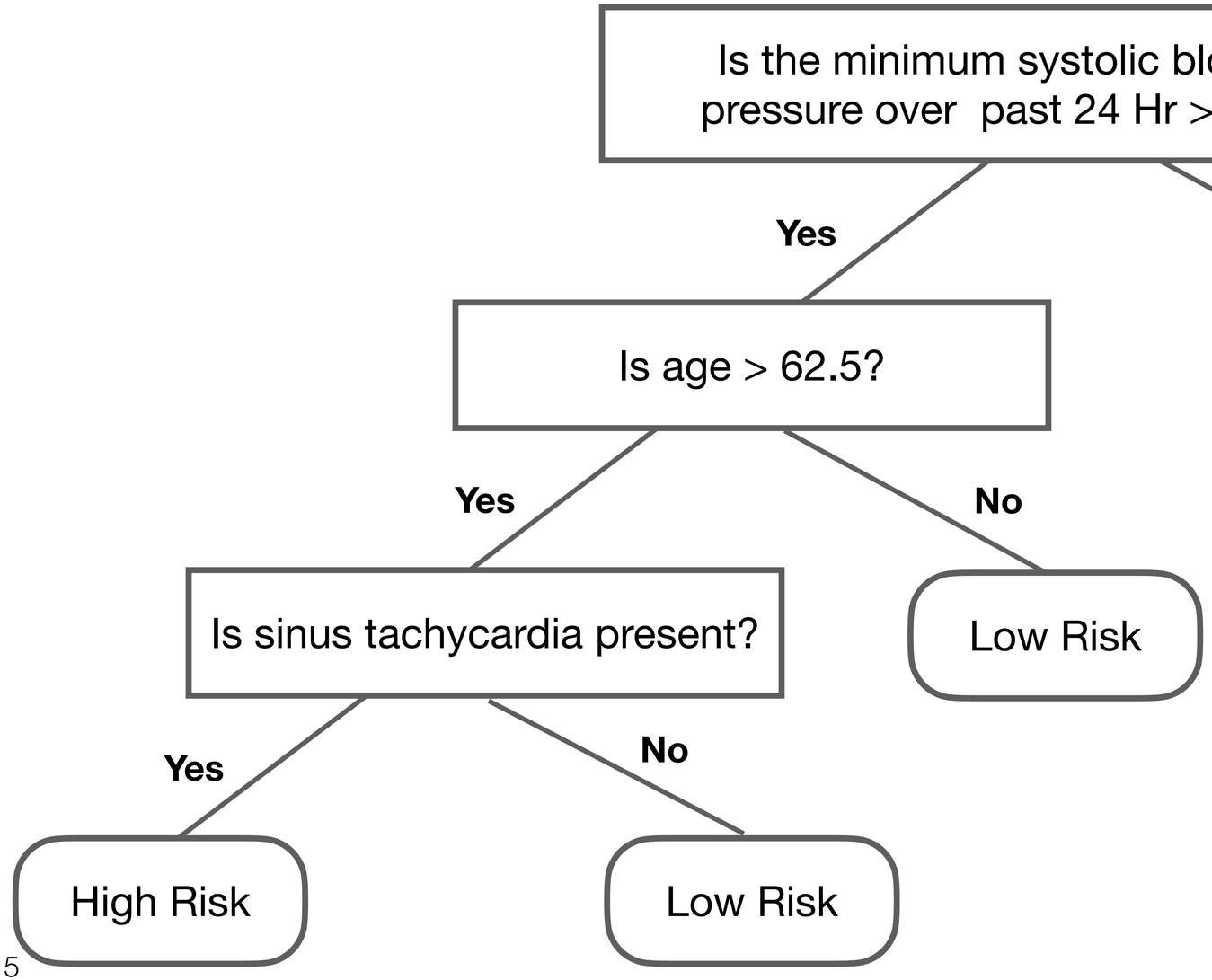
Could be as simple as curve fitting!

Typical applications in general:

- Classification
- Regression
- Dimensionality reduction
- Control

Why Machine Learning?

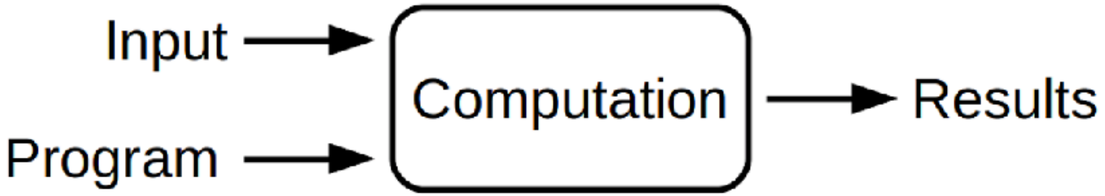
Think about a simple **decision tree**:




```
if blood_pressure > 91:
    if age > 62.5:
        if sinus_tach:
            ...
        else:
            ...
else:
    ...
```

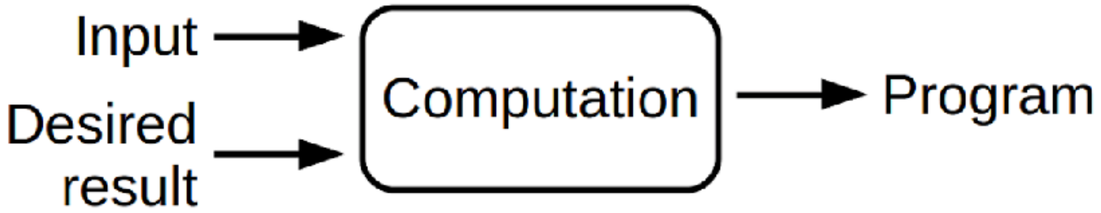
What *is* machine learning?

Traditional programming



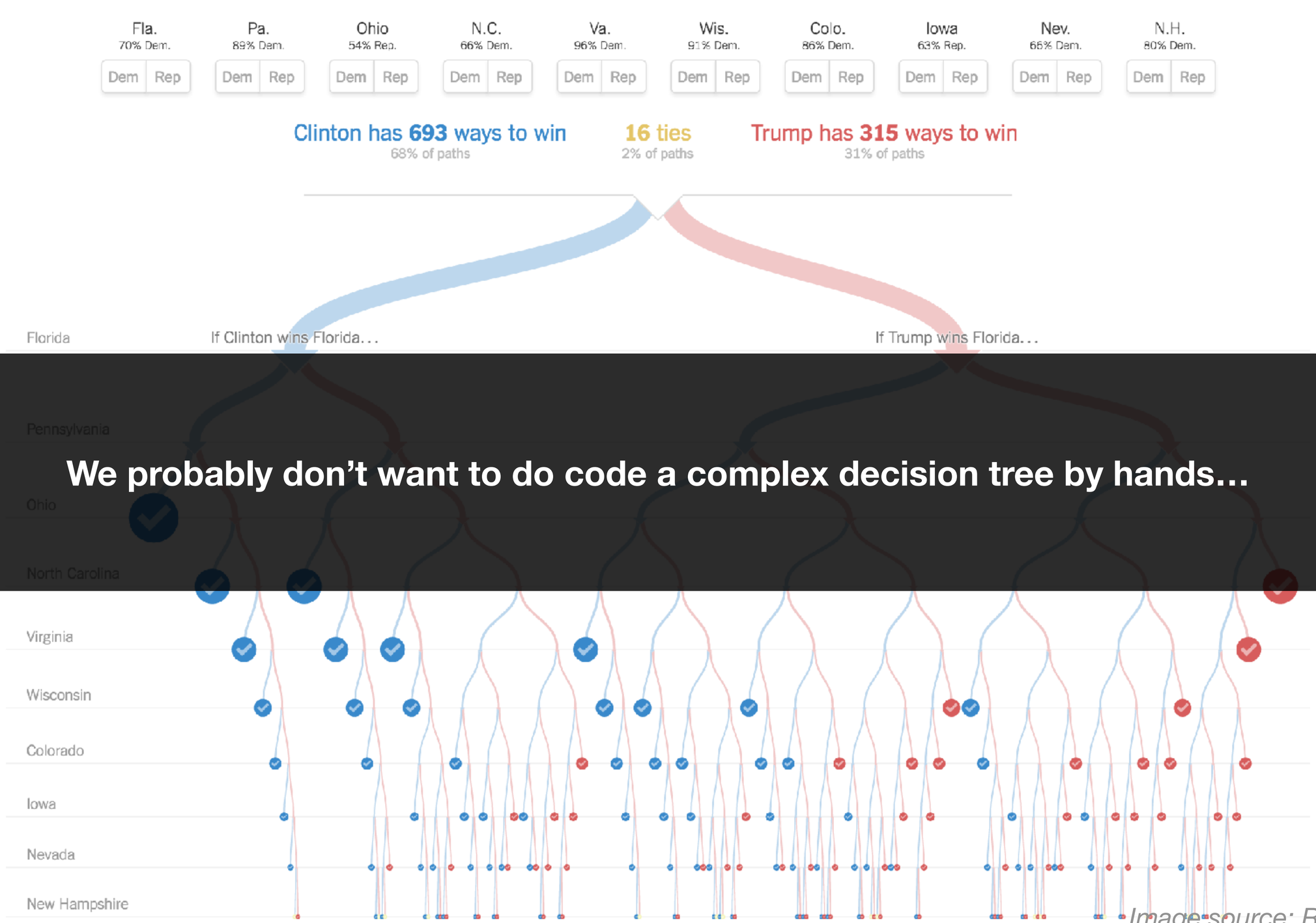
Knowledge base & rules
Expert systems
Human input

Machine learning



Features and result
'Decision' system
Limited human input

Source: Machine Learning for an Expert System to Predict Preterm Birth Risk , Woolery et al. (1994)



We probably don’t want to do code a complex decision tree by hands…

Image source: Rahul/Medium

AI vs ML vs DL

Artificial Intelligence

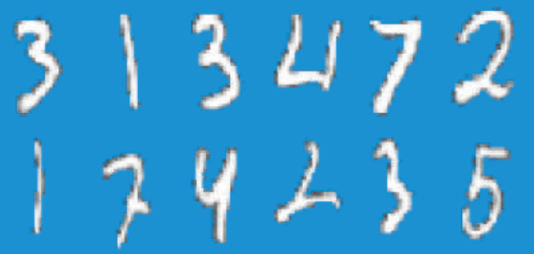
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

Extract patterns from data using neural networks

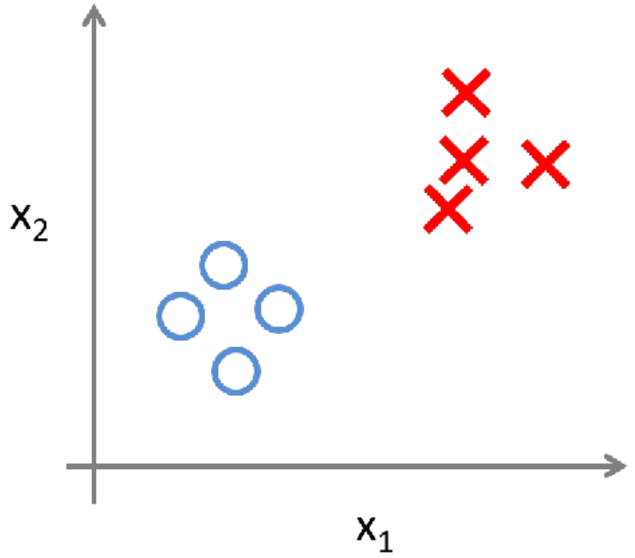


Credit: Alexander Amini/MIT

Categories of machine learning

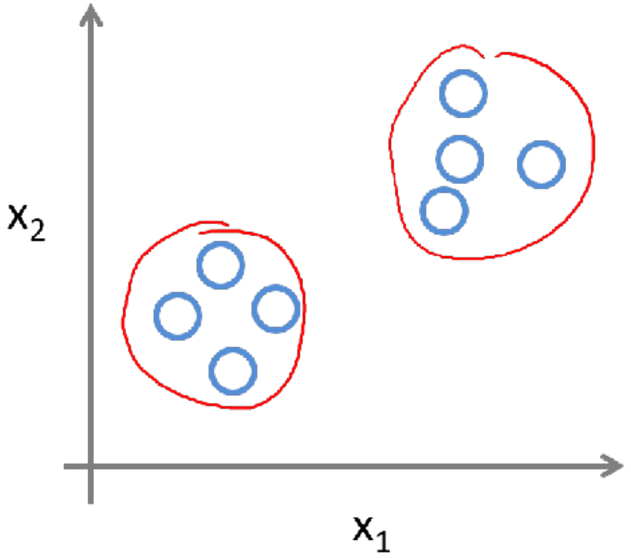
Supervised

Learn from the labels



Unsupervised

Detect patterns in the data



Reinforcement

Learn from mistakes

Categories of machine learning

Supervised

Classification
- Naive bayes
- Support vector machine
- Decision tree
- Random forest
- K-Nearest Neighbor
- Logistic regression

Regression
- Linear
- Generalized

Unsupervised

Clustering
- K-means
- K-medoids

Dimensionality reduction
- PCA
- SVD

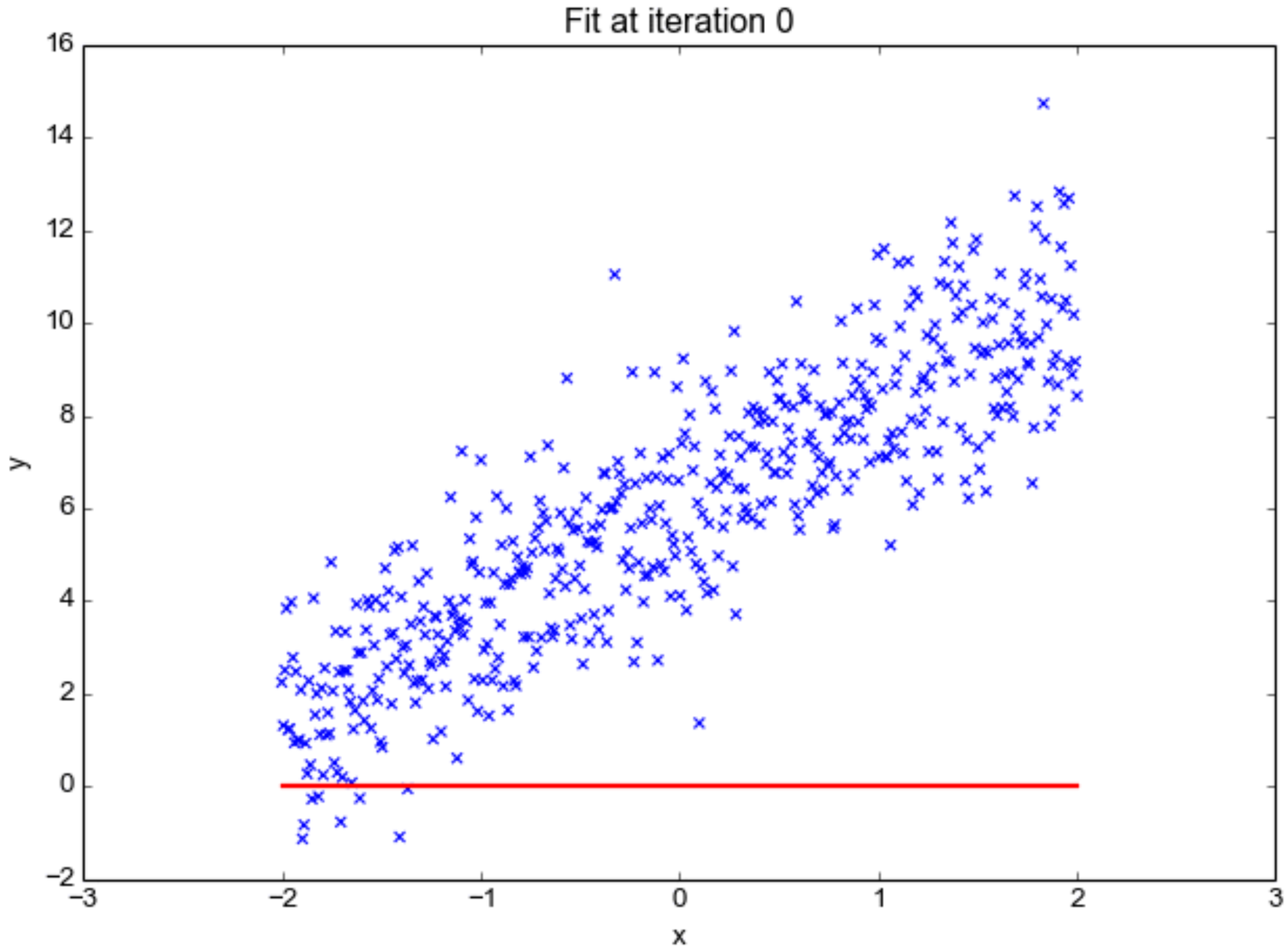
Reinforcement

Discrete
- Markov decision process
- Deep Q Network
- A2C
- A3C

Continuous
- DDPG
- NAF

… and many more…This list is incomplete!

Linear Regression



Iteratively Optimizing the parameters in a linear function

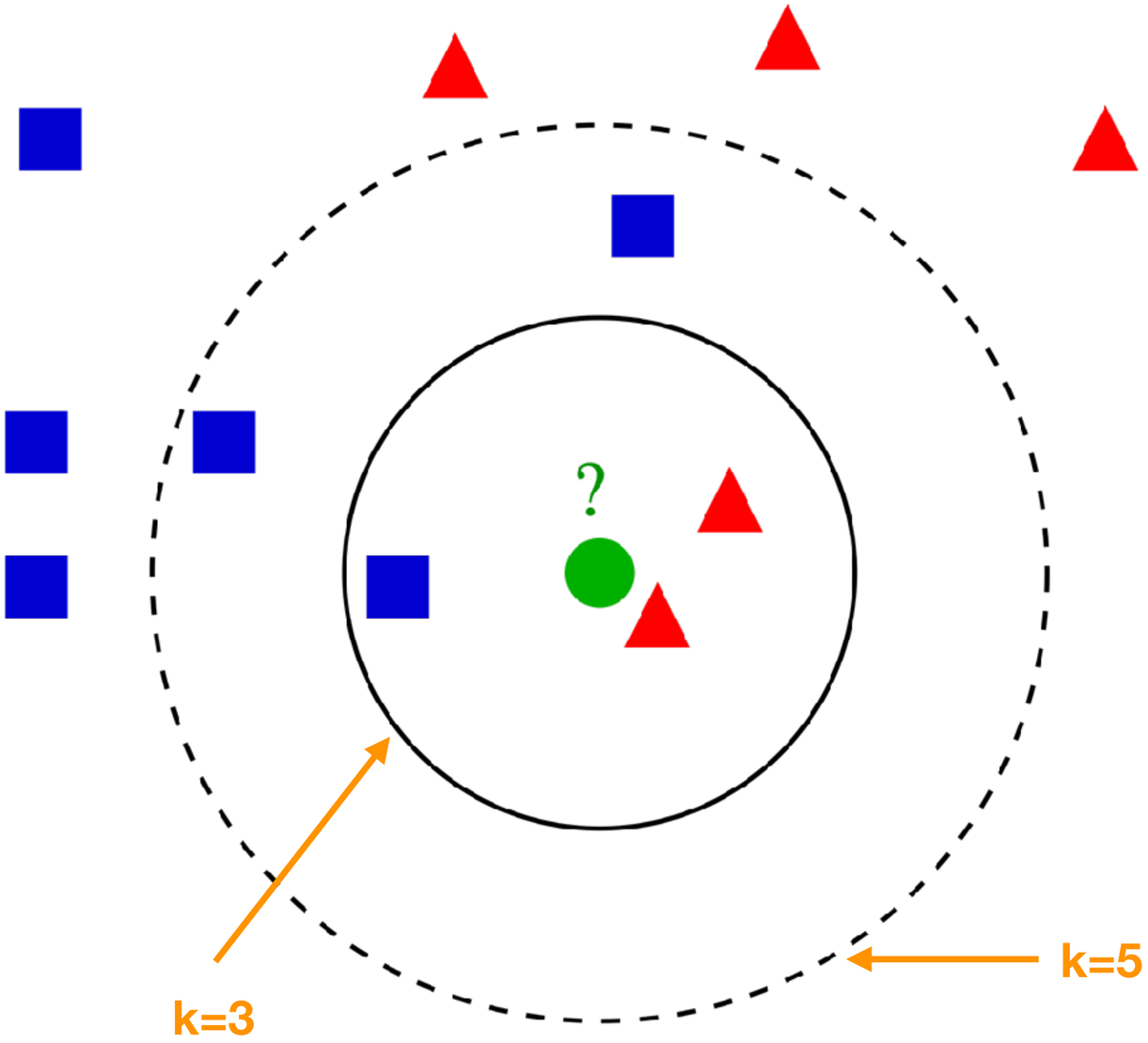
$$\hat{y} \equiv f(x) = wx + b$$

Such that

$$L(\hat{y}, y)$$

Reaches a minimum value.

Classification: *k*-Nearest Neighbor

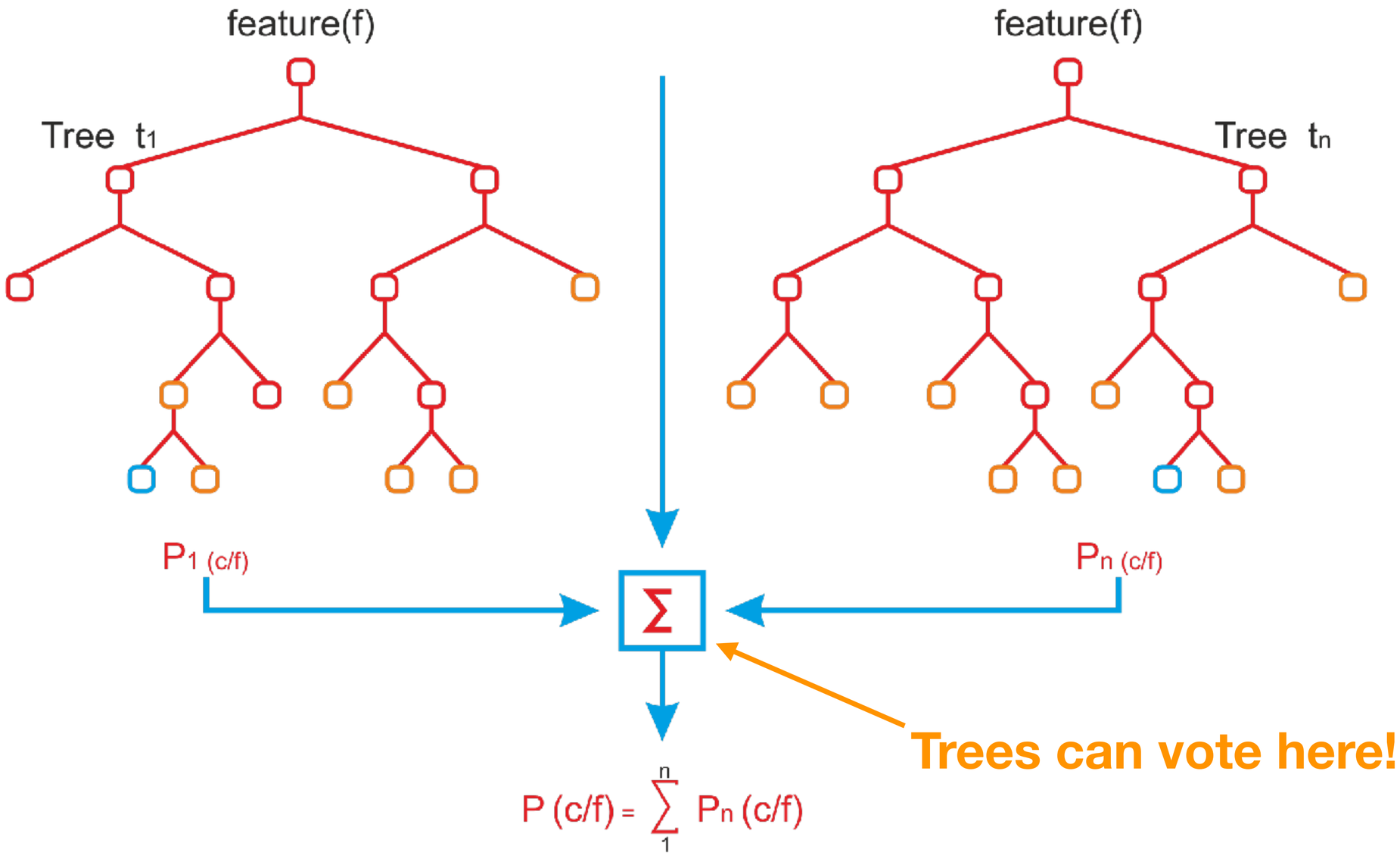


… is a **classification** algorithm

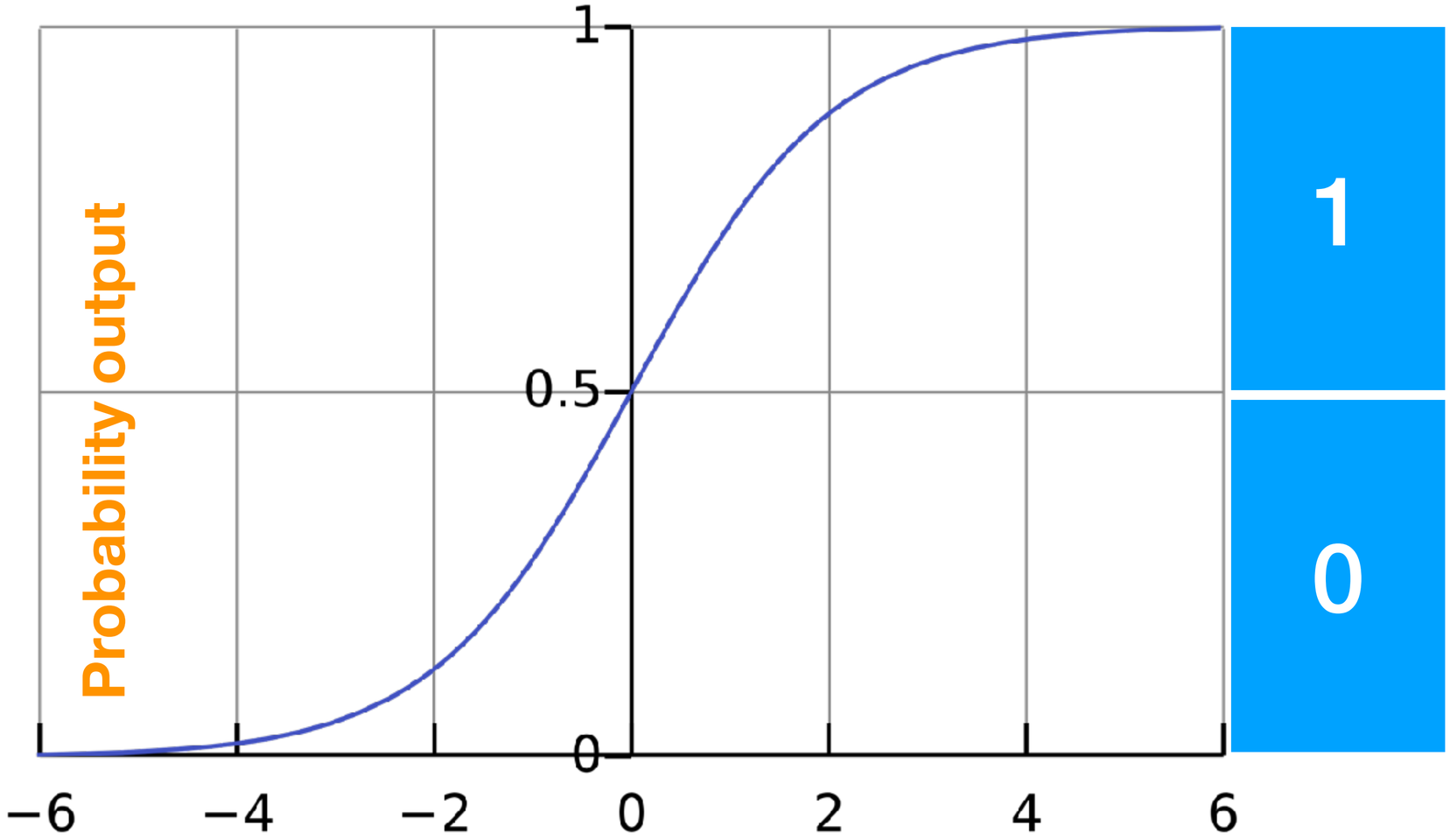
… based on a **voting** process

Image source: Wikipedia

Classification: Decision Tree → Random Forest



Classification: Logistic Regression



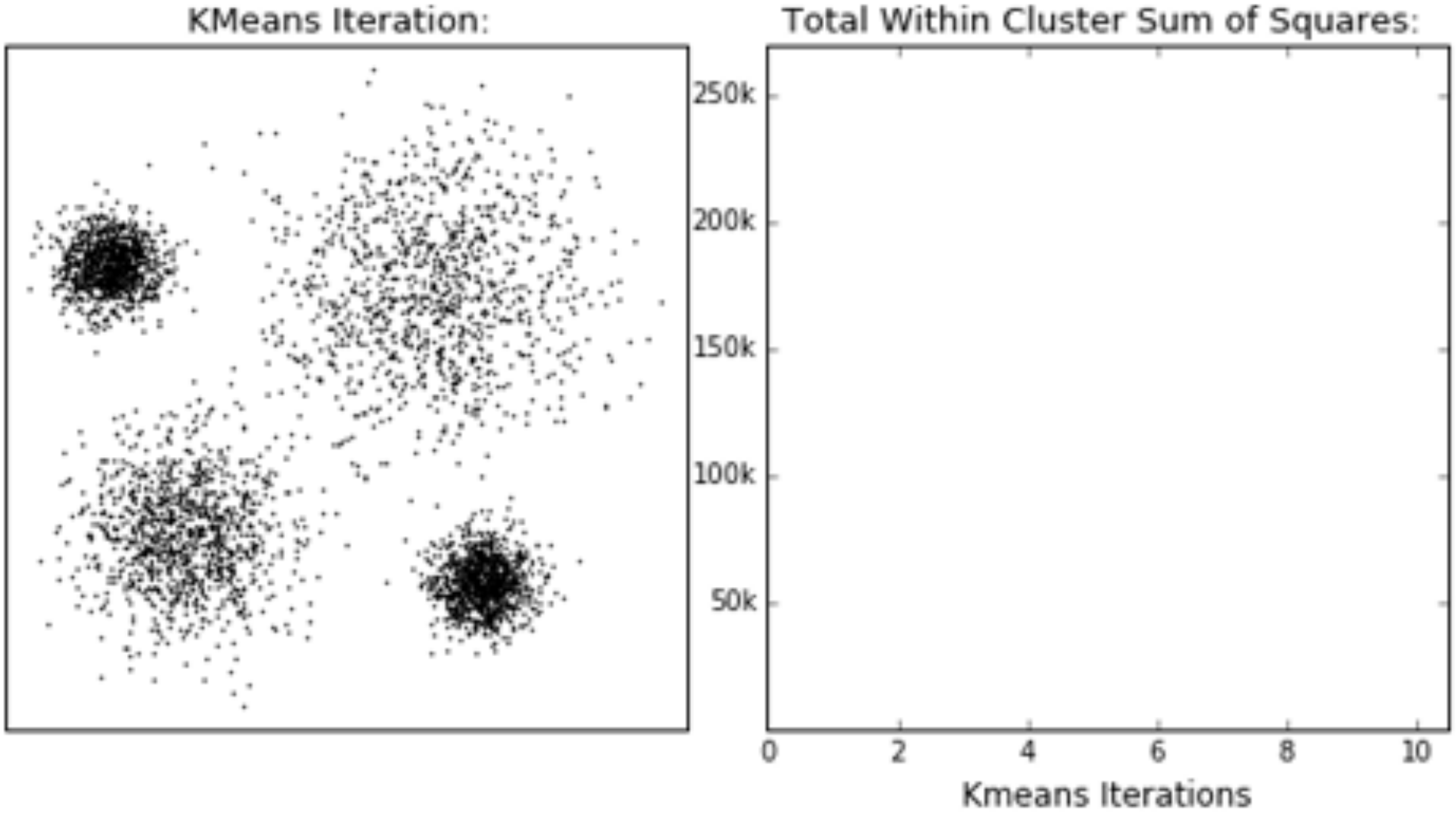
Logistic sigmoid function

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

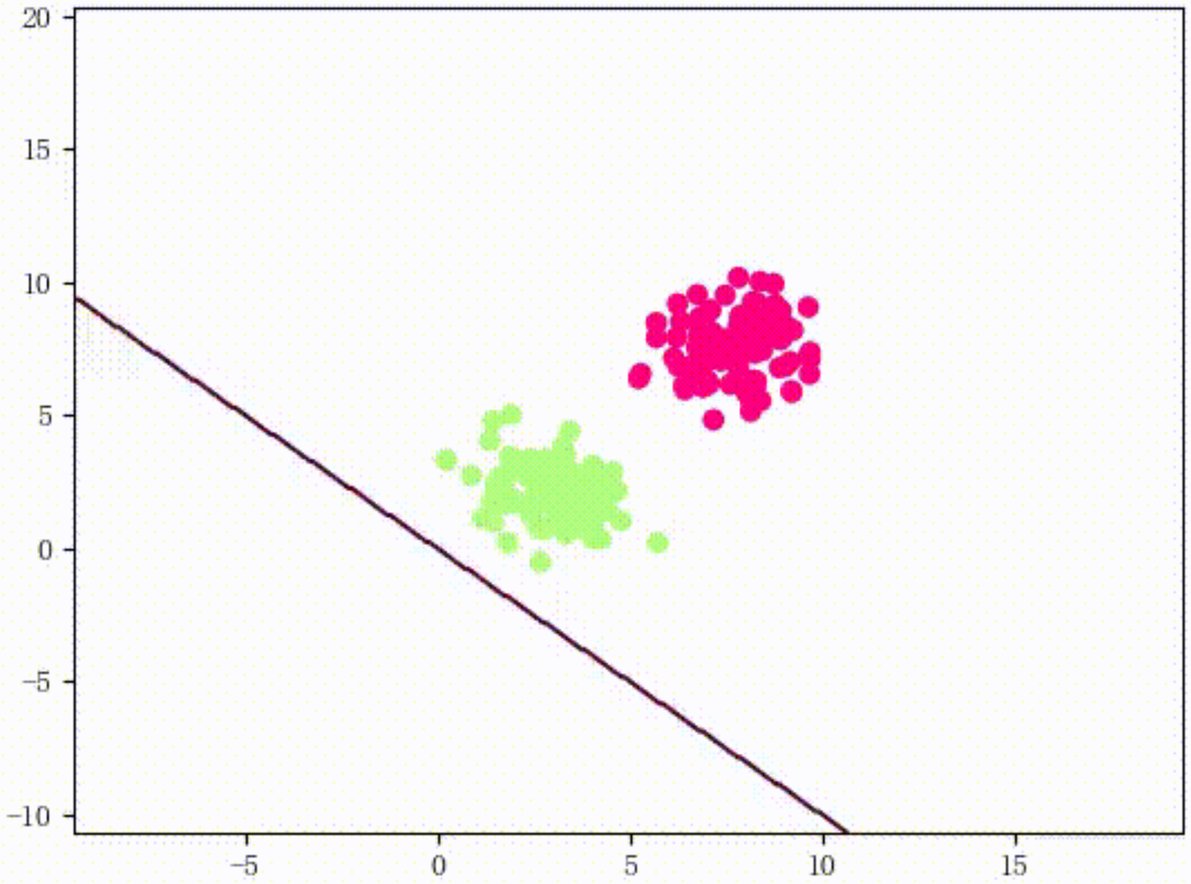
… is a binary **classification** algorithm, not a regression algorithm

… Gives the **probability** of each class

K-means clustering



Support Vector Machine



… is a **classification** algorithm

… does not provide probabilities, but only output class identities.

… aims to find an optimal way (perpendicular to the **support vector**) to separate different classes

Animation source: GitHub.com/carefree0910

Boser et al. 1992; Cortes & Vapnik 1995

What is the machine actually *learning*?

Machine learning is an optimisation process

Learning: Optimizing the loss function

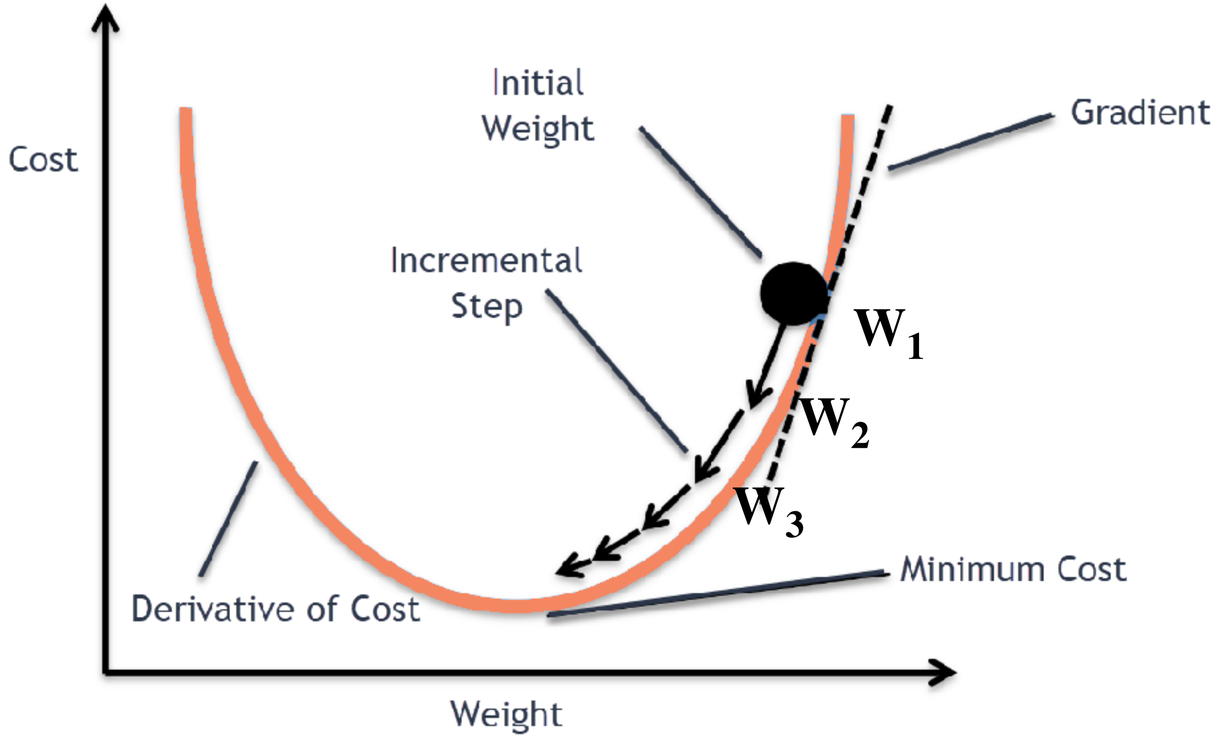
Objective: minimize the **difference** between predicted value the actual value

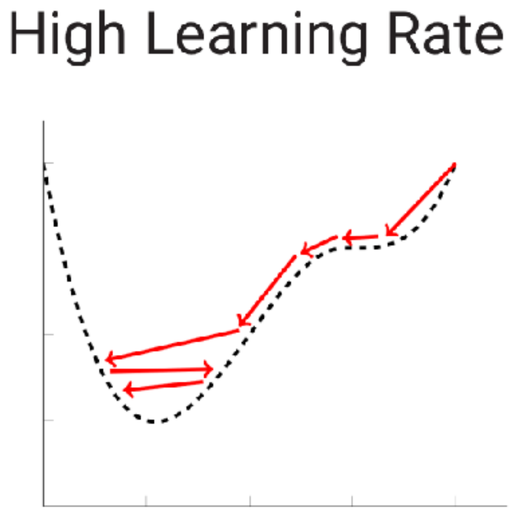
$$\mathscr{L}(\mathbf{W}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

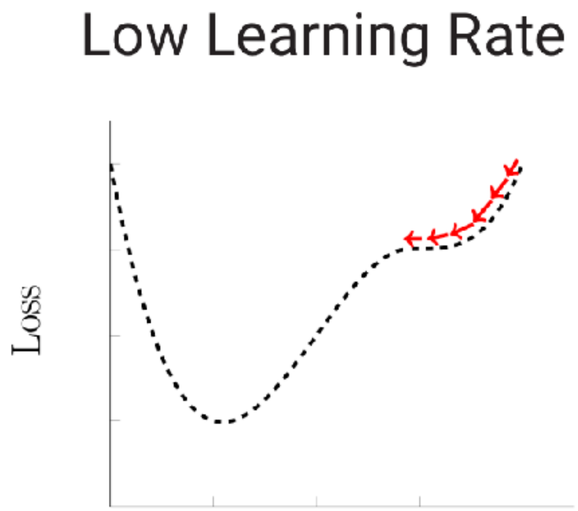
$$\nabla\mathscr{L}(\mathbf{W}, b)$$

$$\mathbf{W}_{j+1} = W_j - \alpha\nabla\mathscr{L}(\mathbf{W}_j, b)$$

Learning rate







A few more terms…

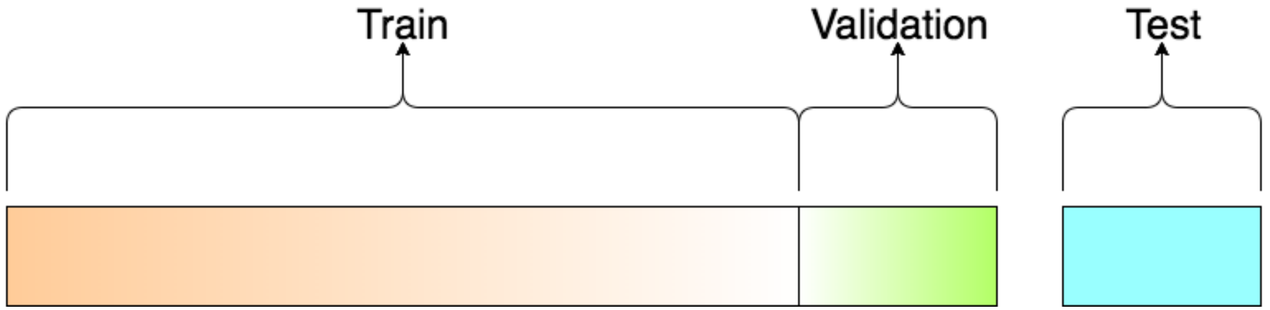
Weights

Bias

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

Estimate

Input features



Evaluation with the **mean squared error**

$$\text{MSE}(\mathbf{W}, b) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i^{(\text{test})} - \hat{y}_i^{(\text{test})} \right)^2$$

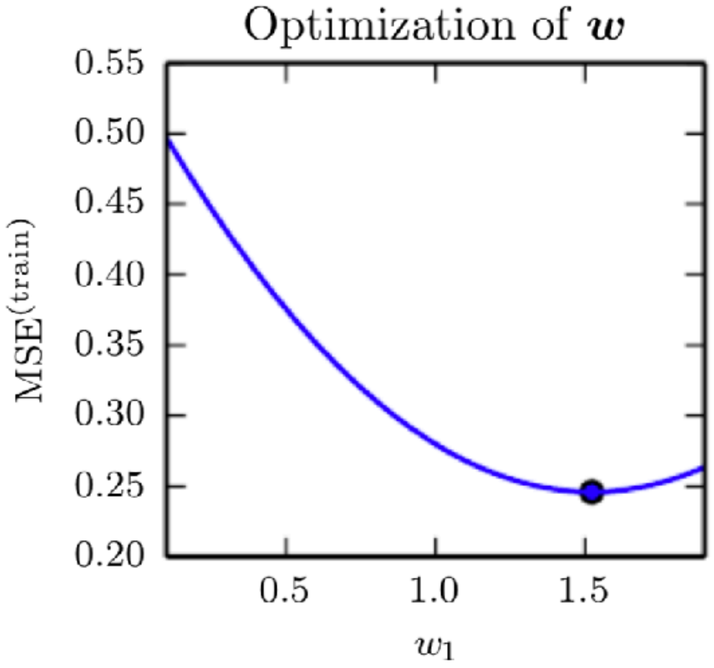
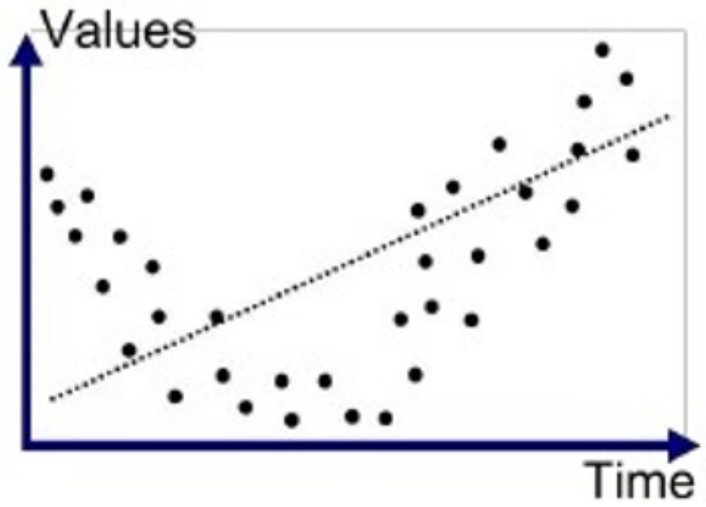
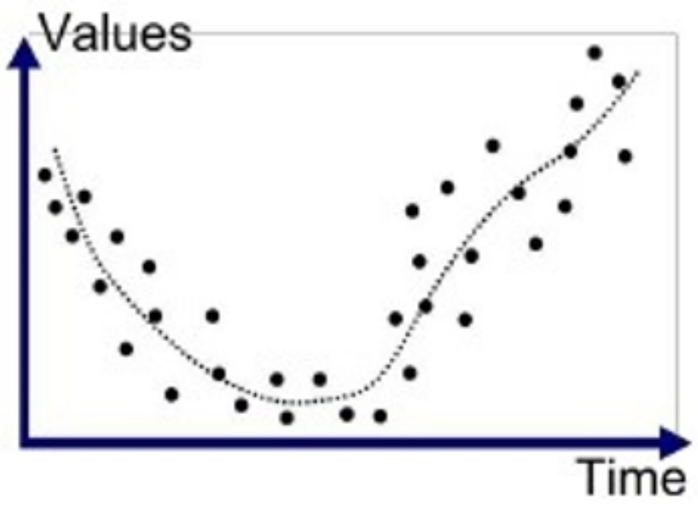


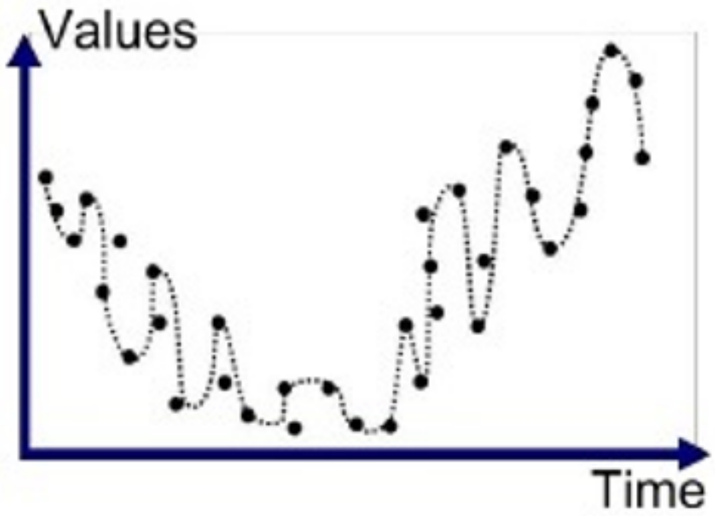
Image credits: Deep Learning (Goodfellow, Bengio & Courville); Tarang Shah

The bias-variance tradeoff

The lower the losses, the better? **Not really.**

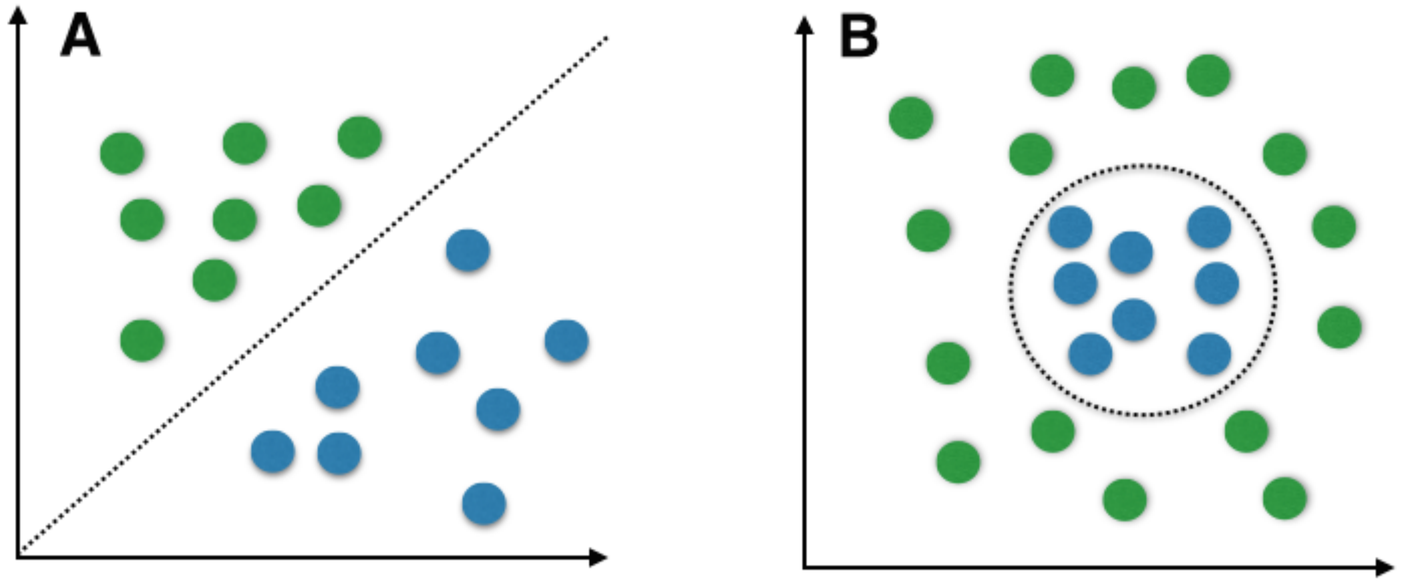


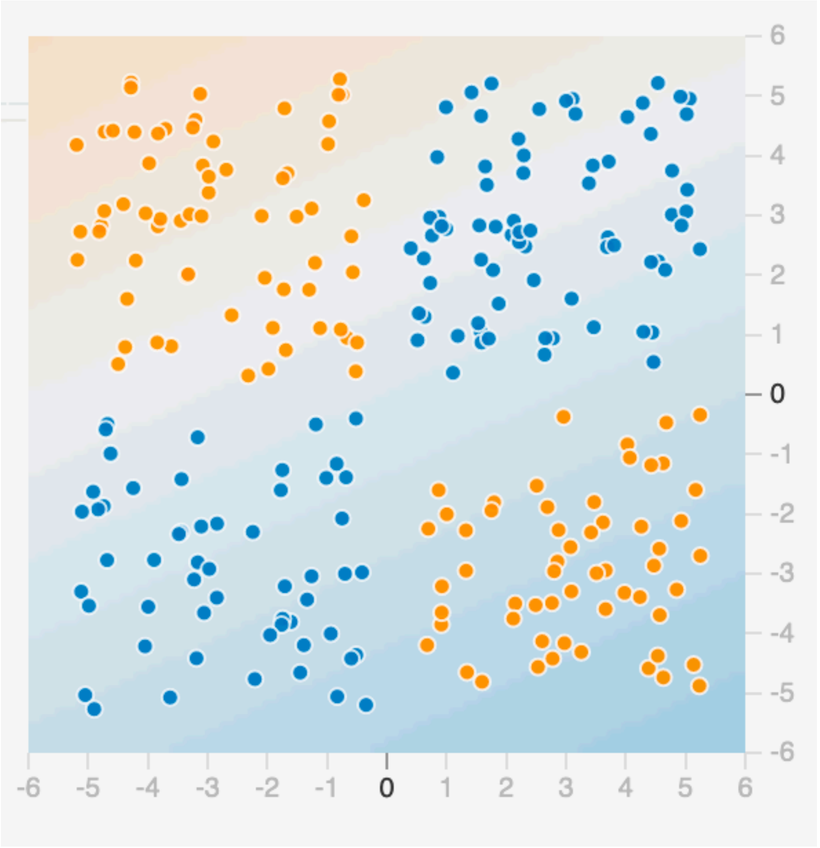




What if the problem is more complicated?

What if the problem is not so simple?

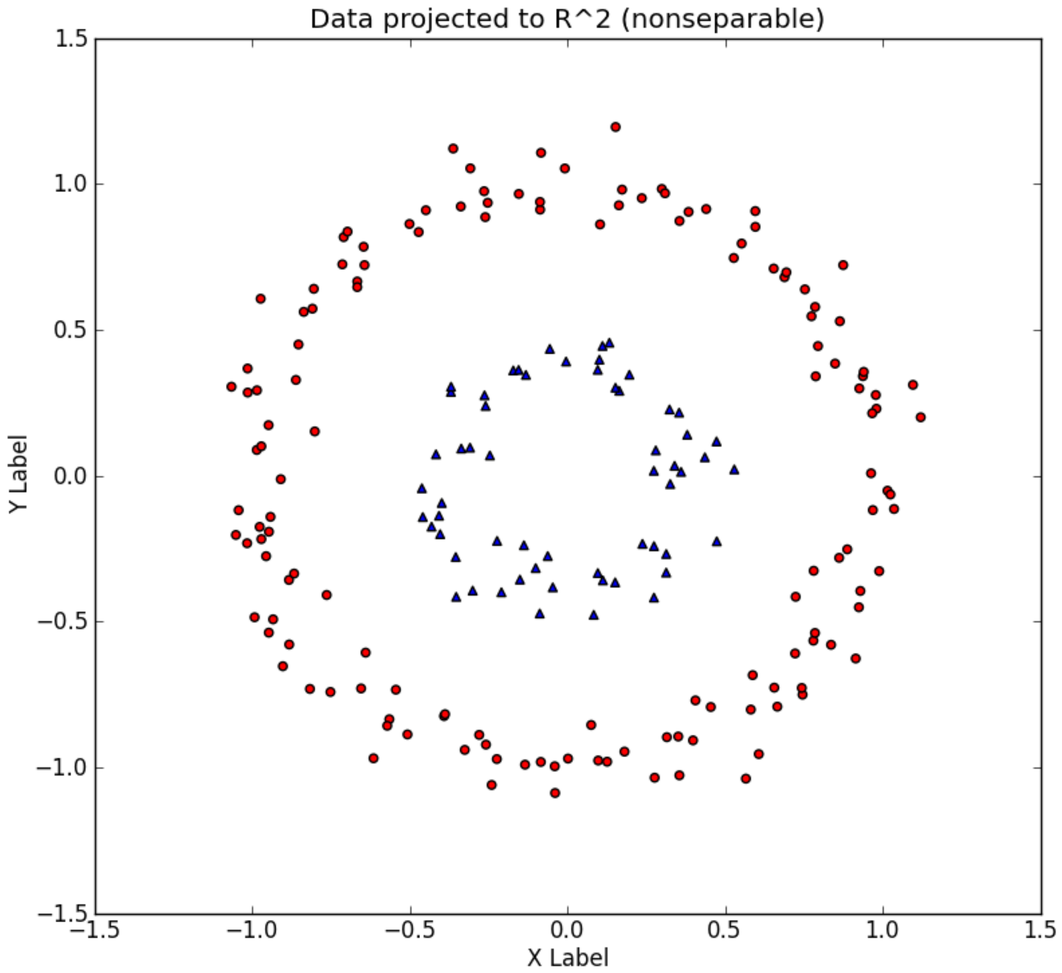


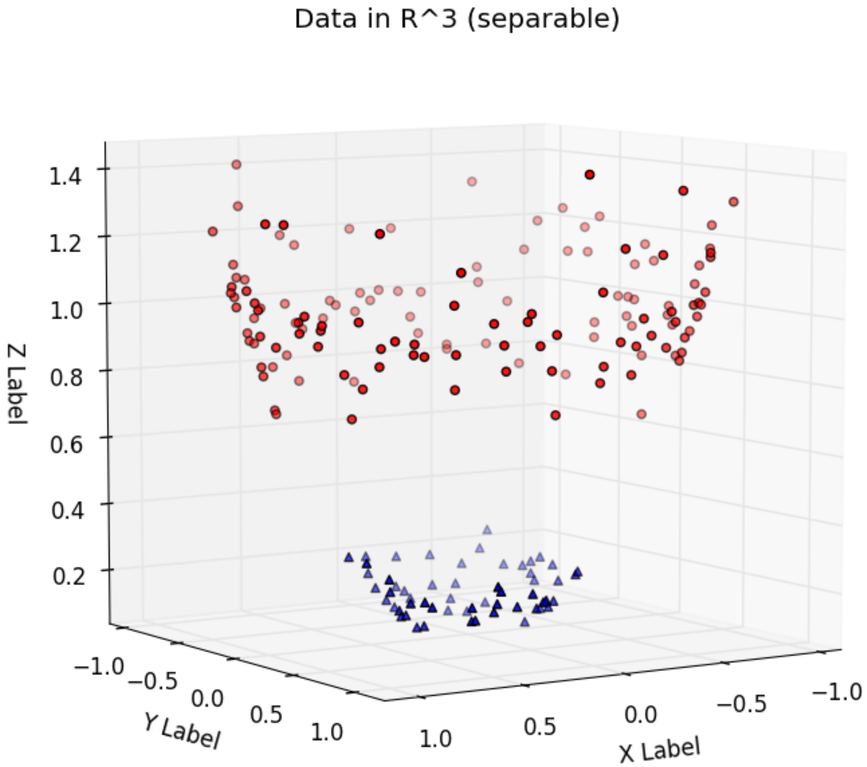


Possible solutions:

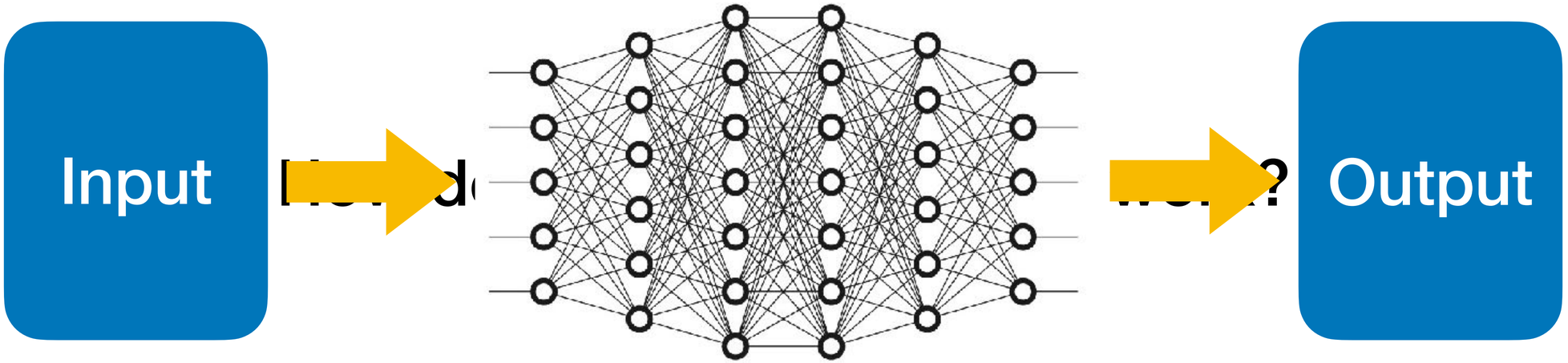
- map into another space (kernel trick)
- Add more layers (deep learning)

Kernel Trick



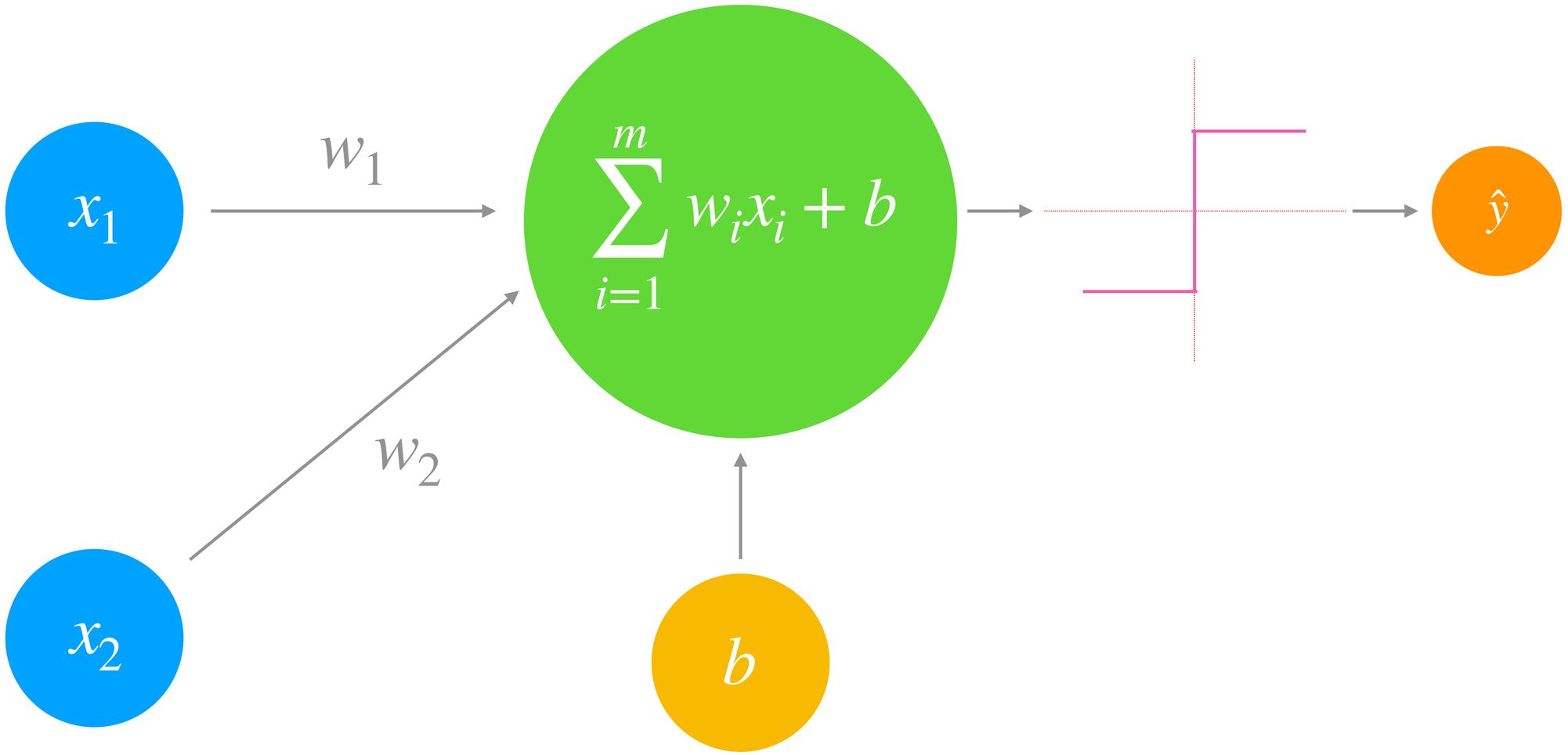


From (traditional) ML to DL

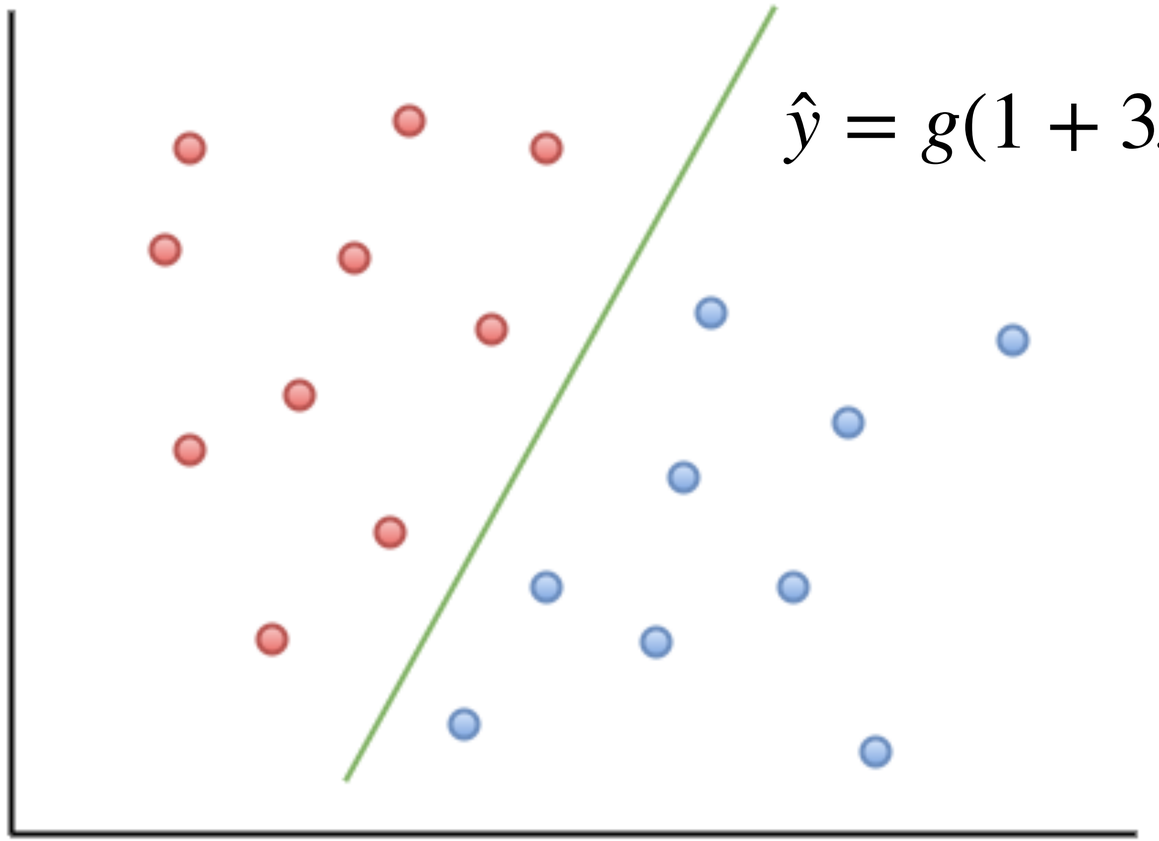


$$\hat{y} = f_{\mathrm{NN}}(x_1, x_2, \ldots, x_n)$$

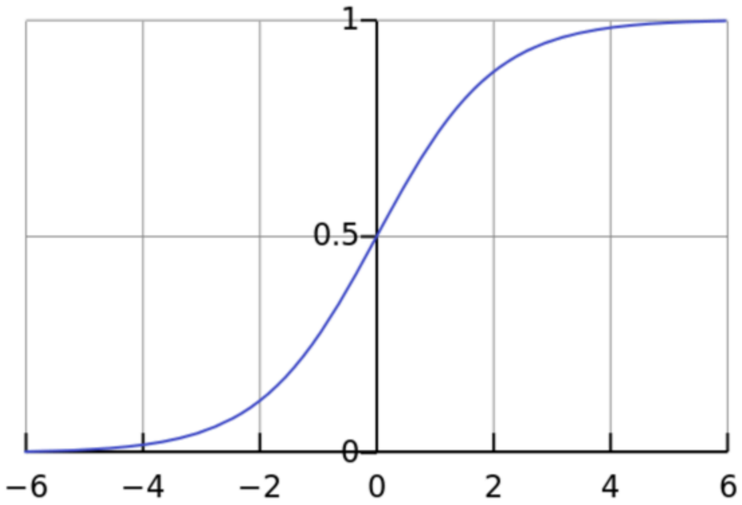
Prediction

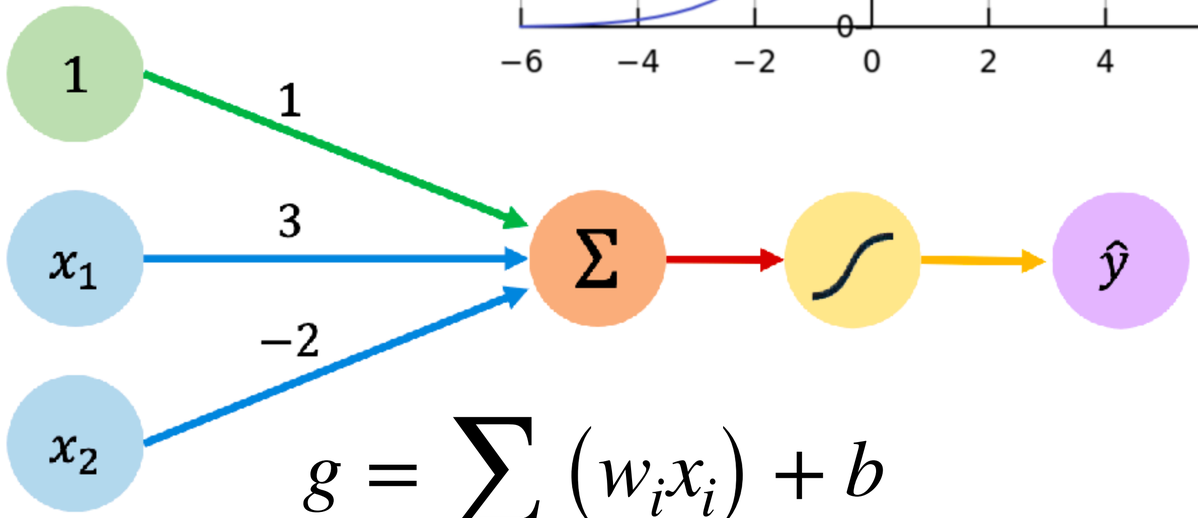

x_1
w_1
$\sum_{i=1}^{m} w_i x_i + b$
$\hat{y}$
w_2
x_2
b
inputs
weights
summation + bias
activation
output

Neurons/Perceptrons & Activation Functions



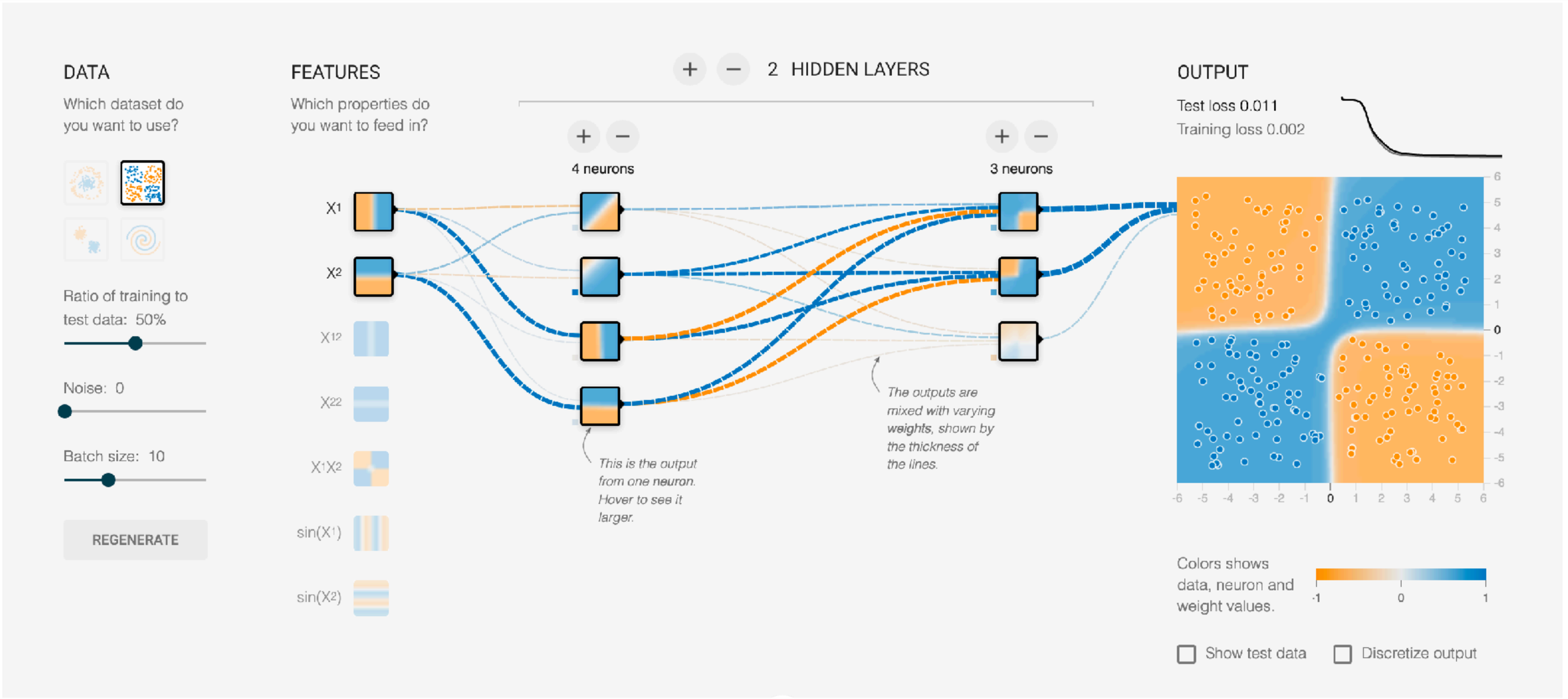
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$





$$g = \sum (w_i x_i) + b$$

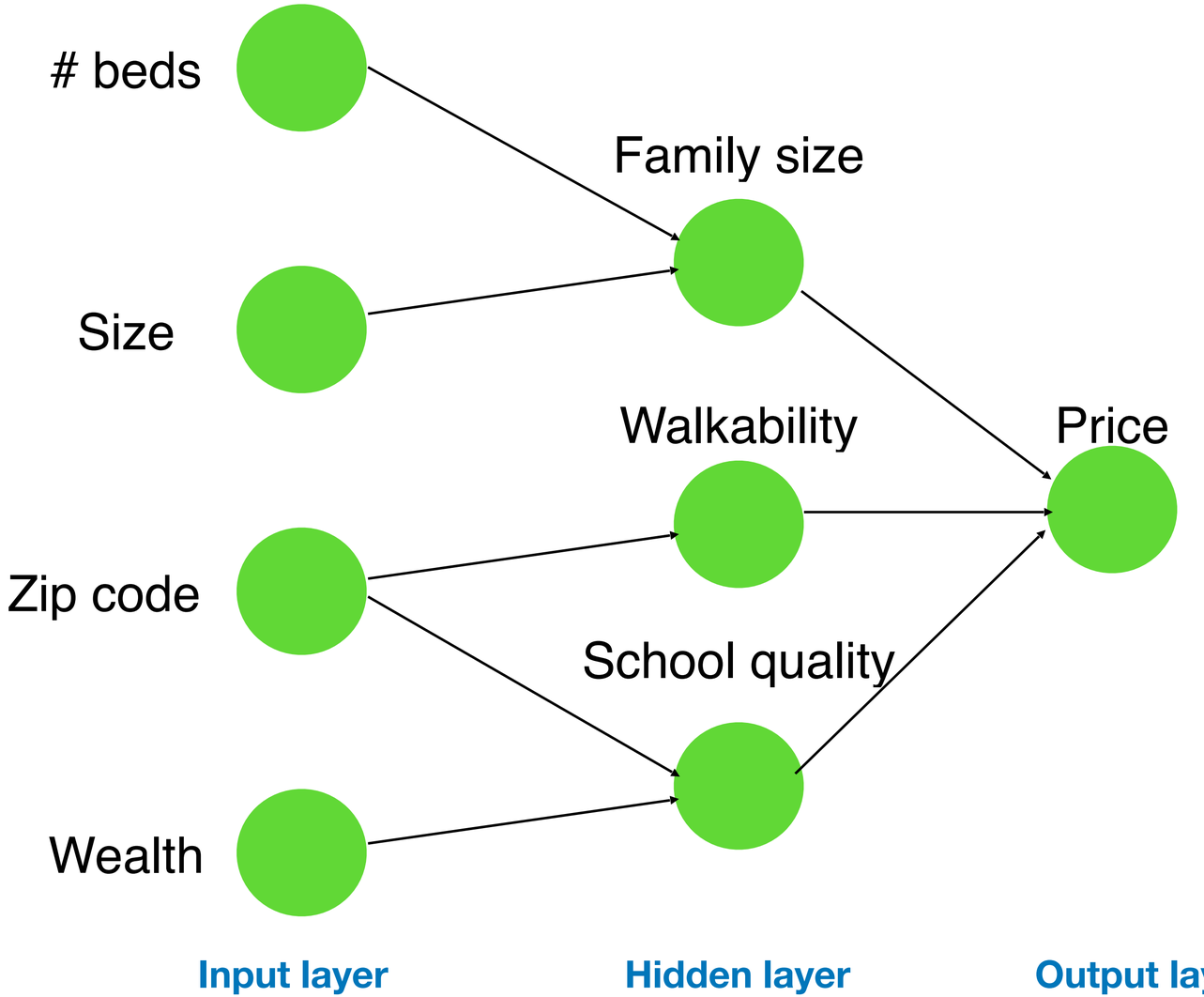
Live demo: Multi-layer Perceptrons



https://playground.tensorflow.org

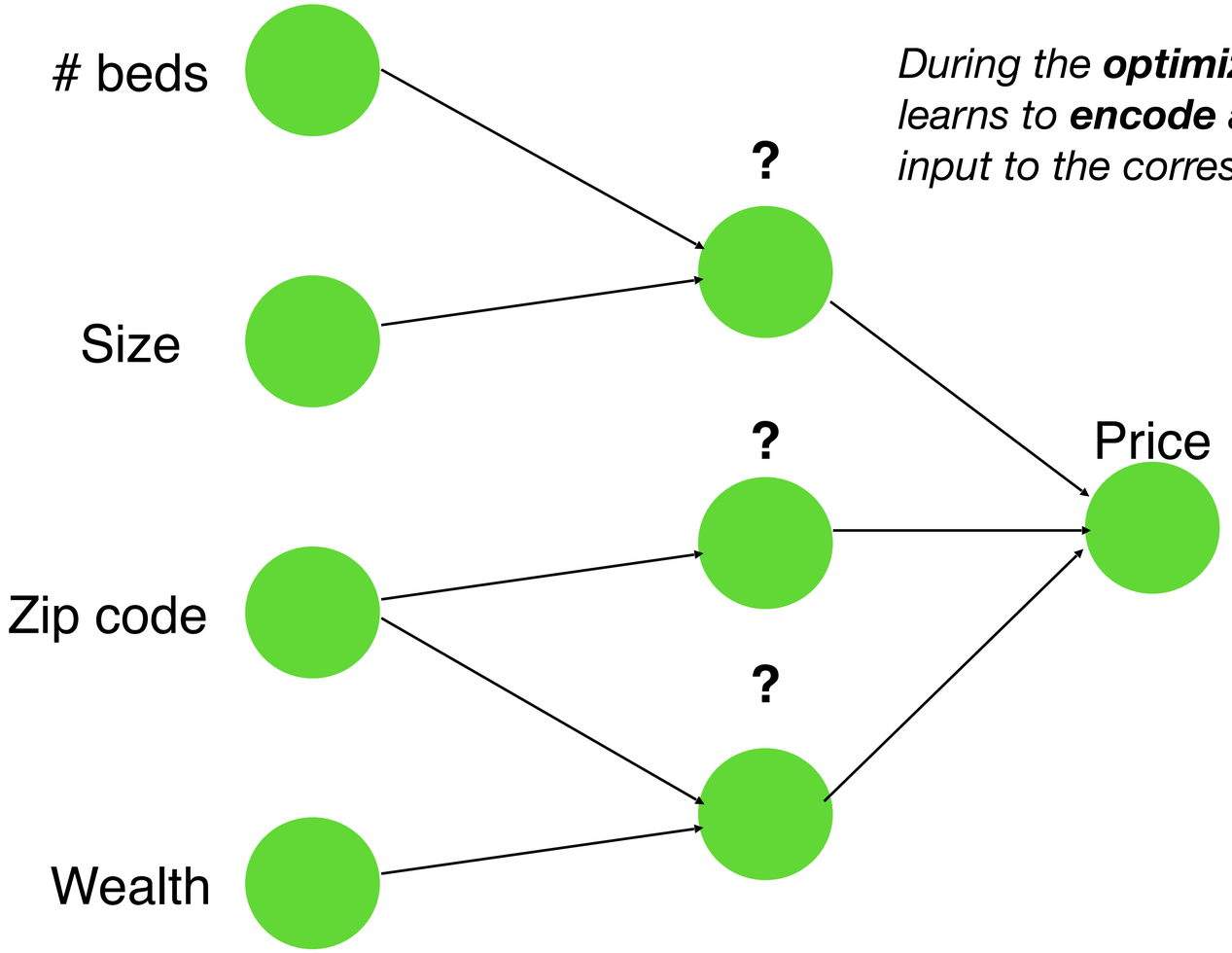
Why multiple layers?

Example: house price prediction model (designed by humans)



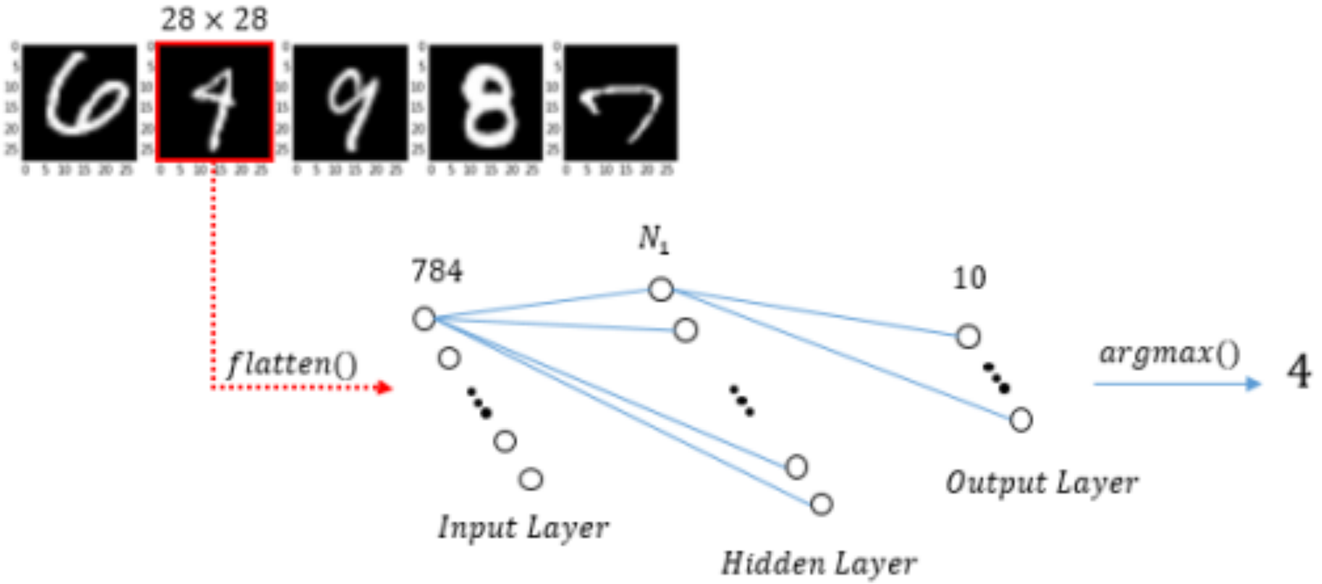
Why multiple layers?

Example: house price prediction model (designed by machines)



*During the **optimization** process, the machine learns to **encode** a presentation that **maps** the input to the corresponding output.*

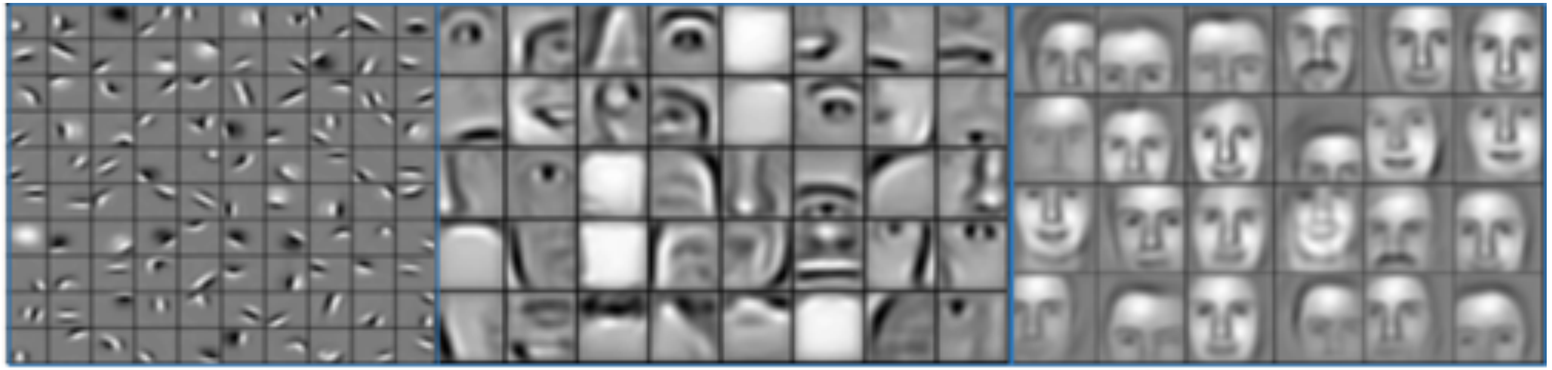
Multi-Layer Perception

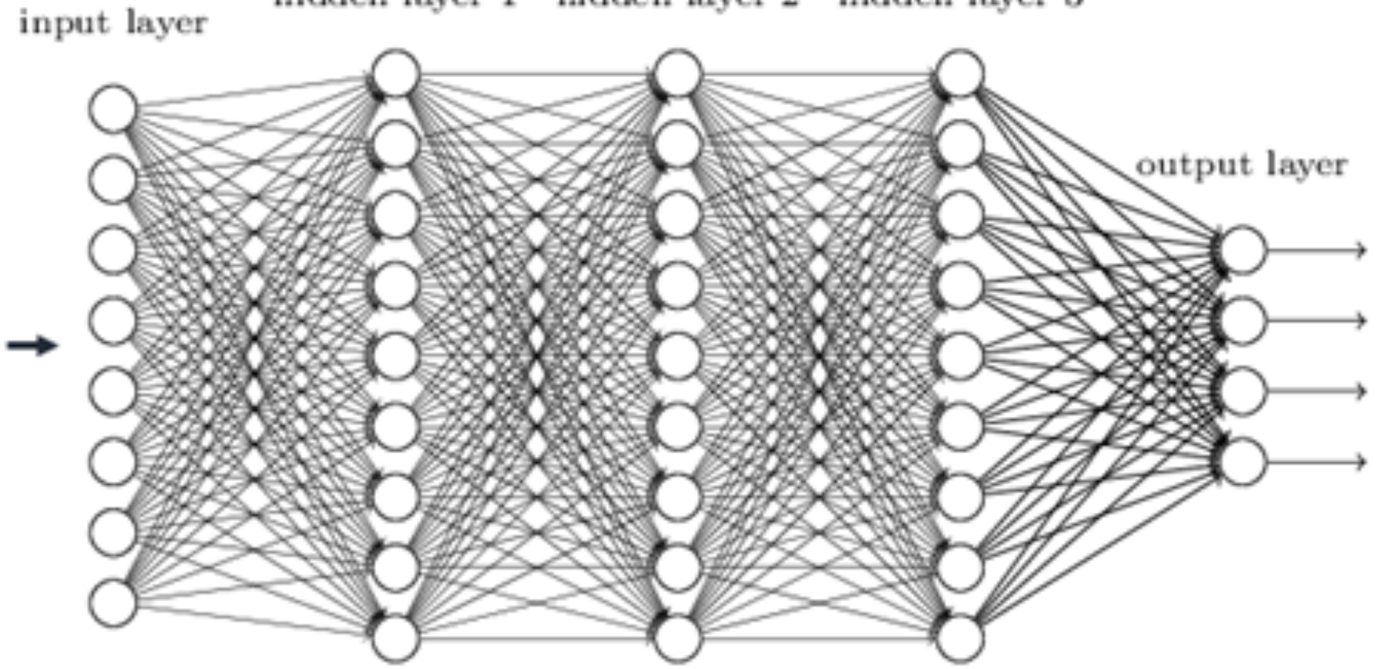


MNIST dataset
(**M**odified **N**ational **I**nstitute for **S**tandards and **T**echnology)

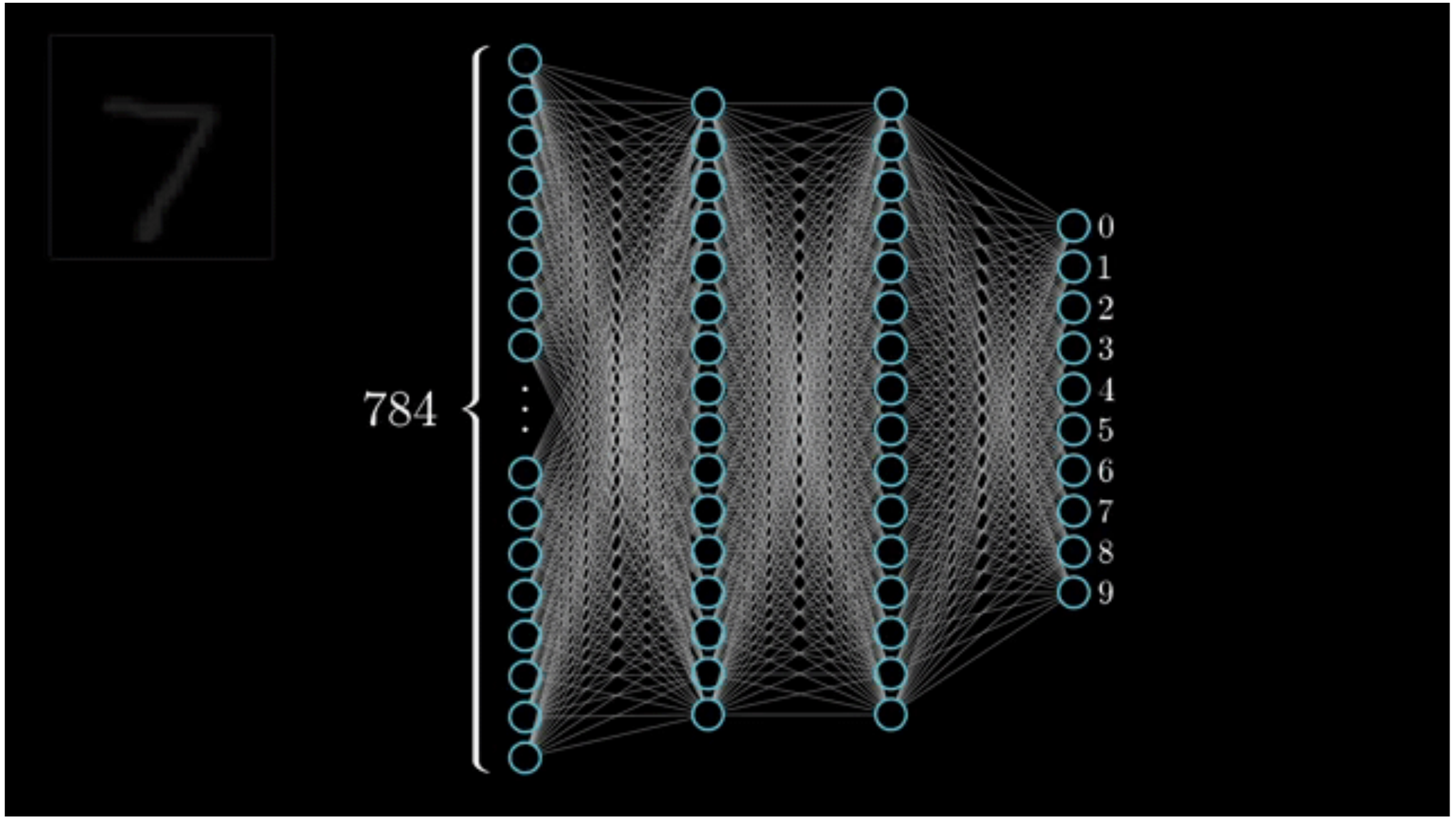
A DNN encodes the representation hierarchically

Deep neural networks learn hierarchical feature representations

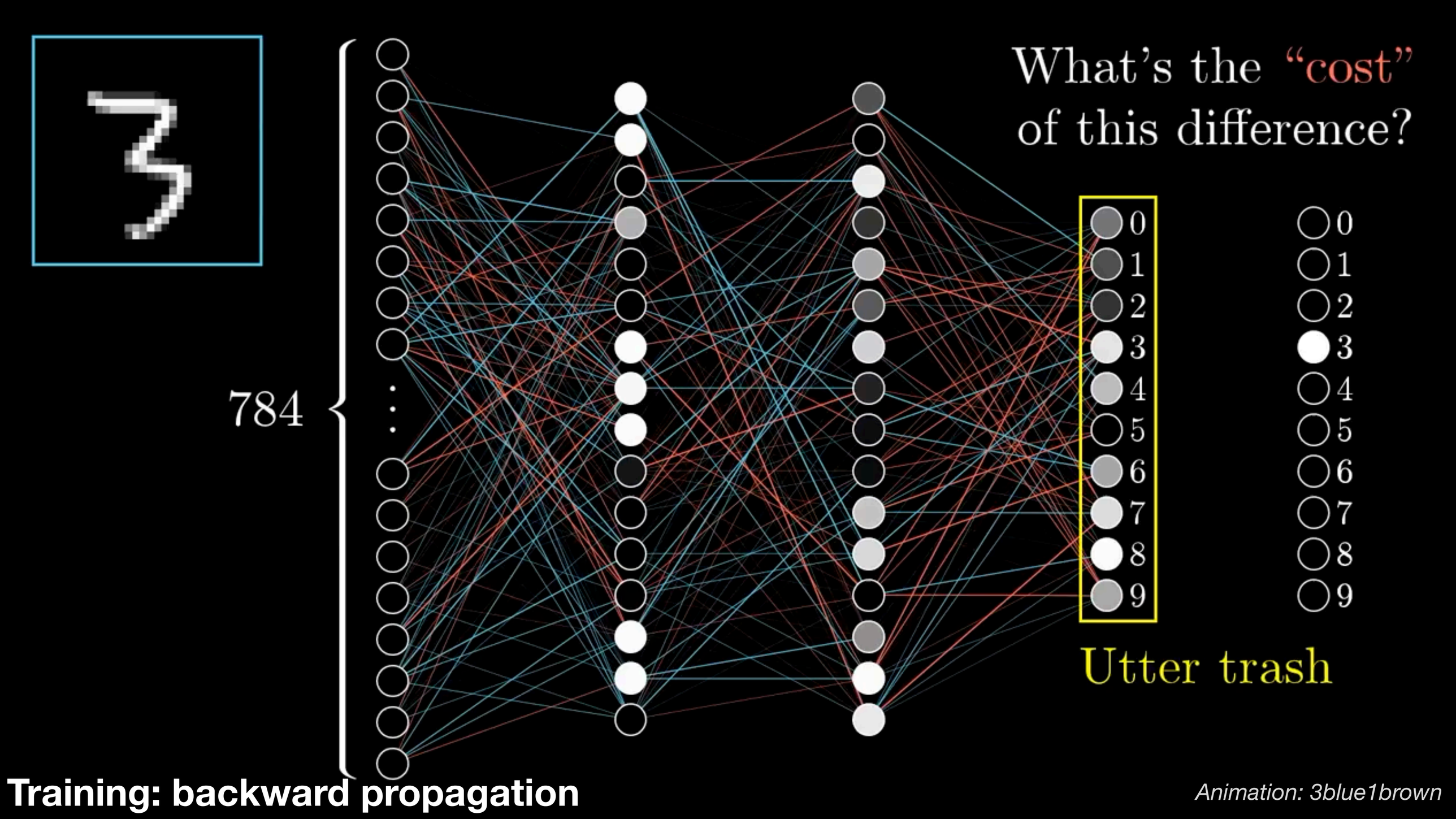




Prediction: forward propagation

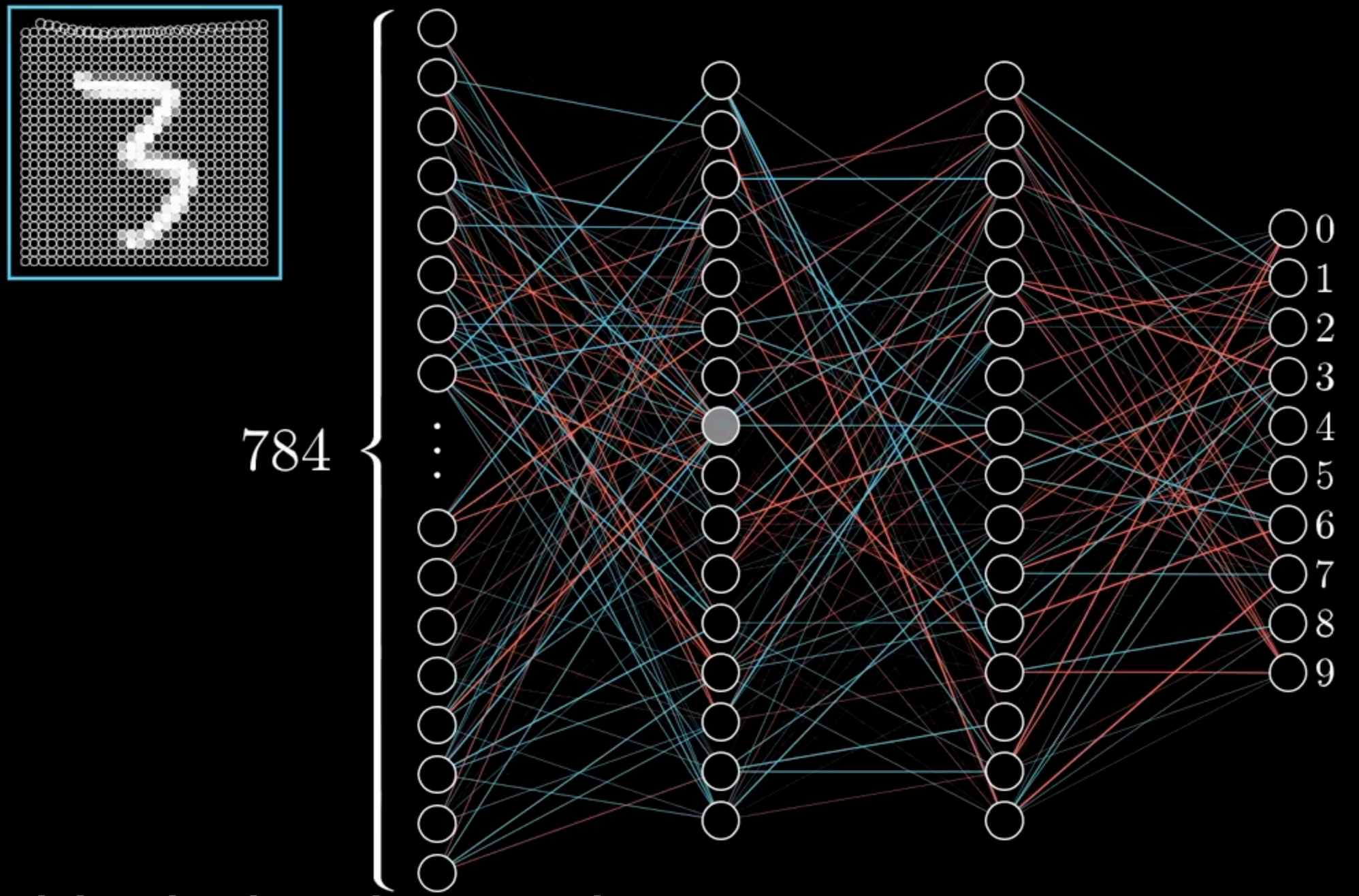


Source: 3blue1brown



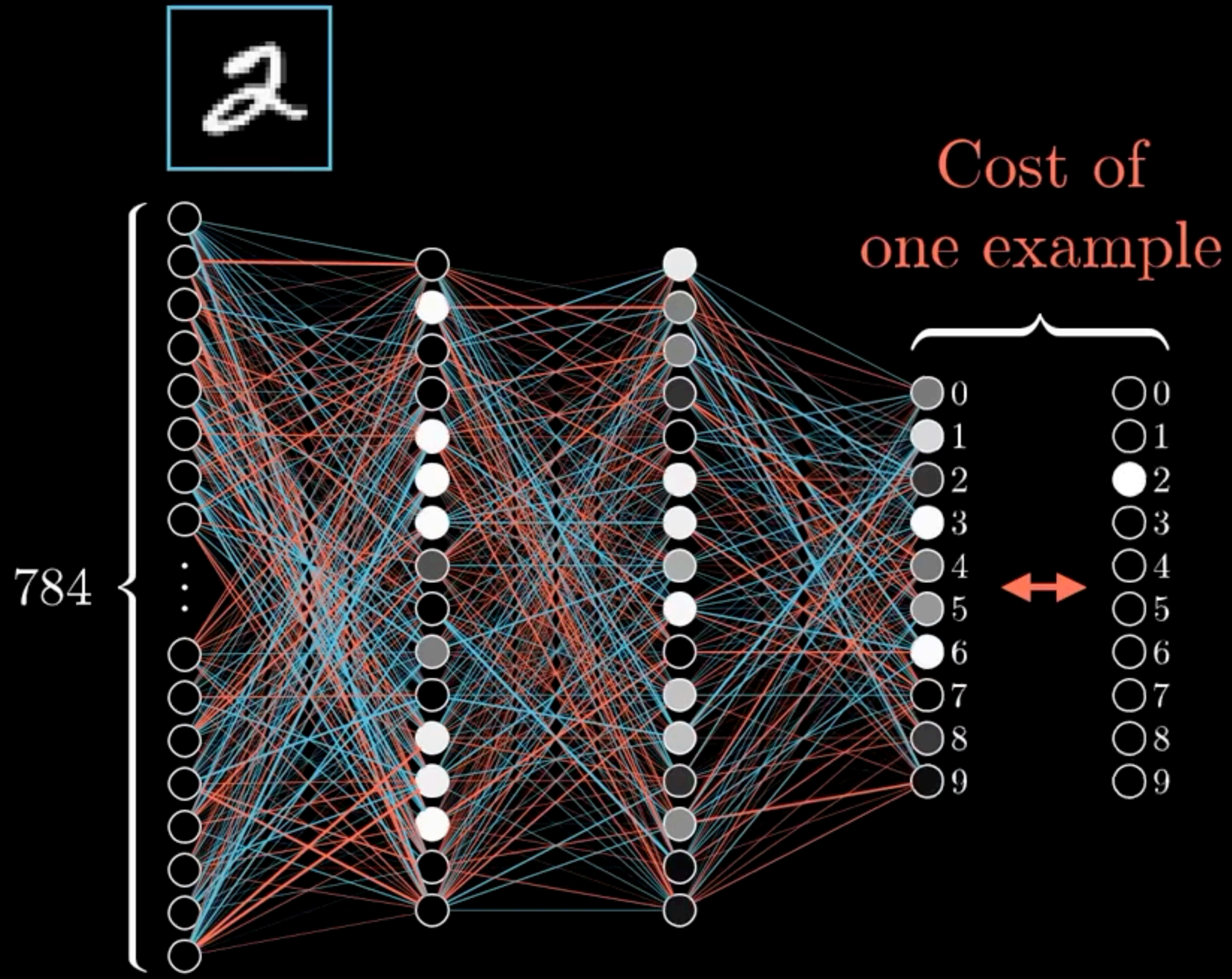
Training: backward propagation

Animation: 3blue1brown



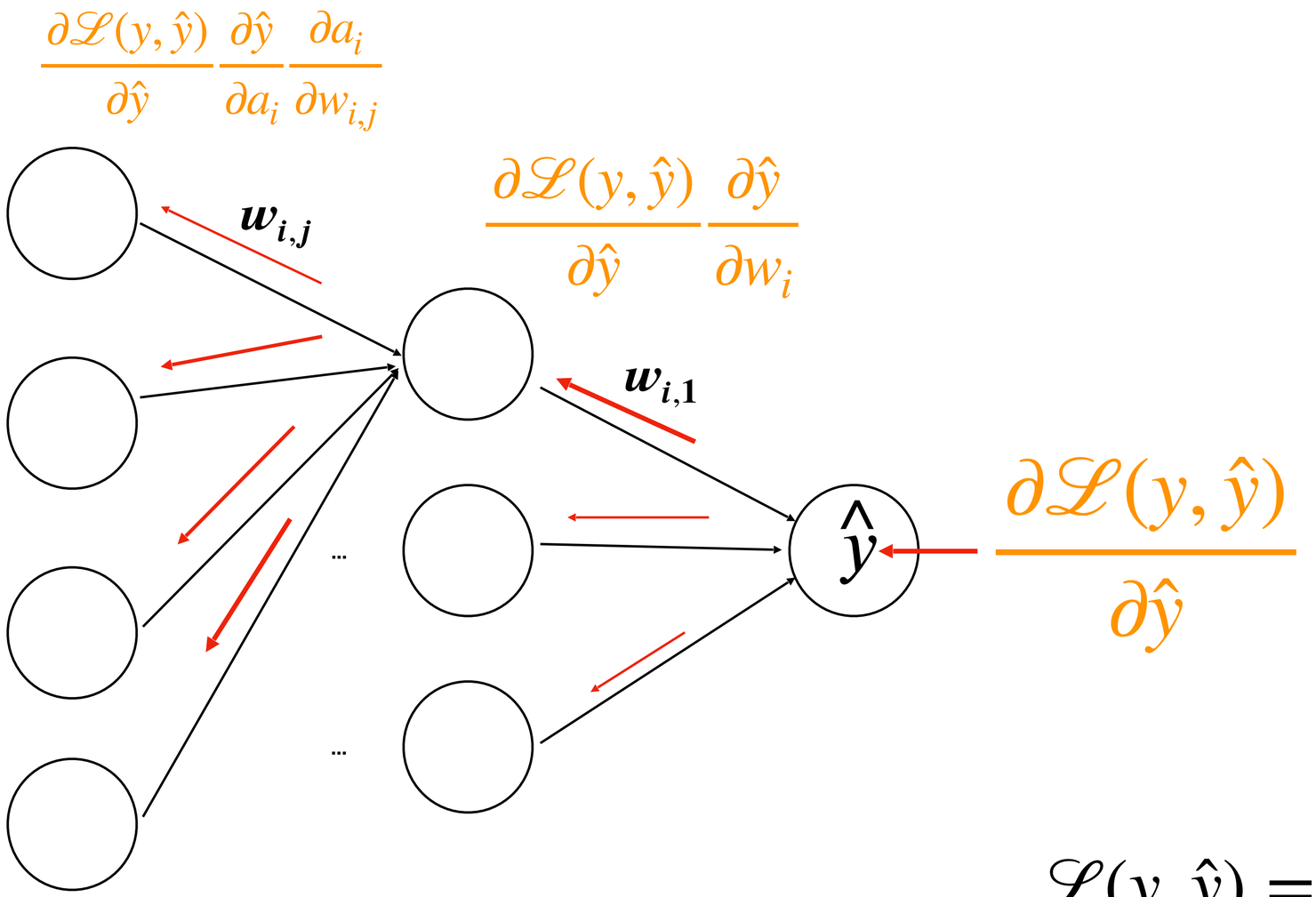
Training: backward propagation

Animation: 3blue1brown



Training: backward propagation

Training: backward propagation



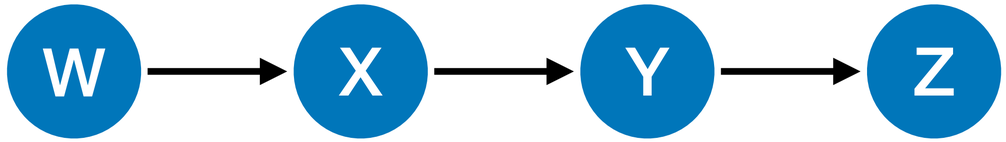
- In the context of DL we need to compute the gradient for each layer.
- We do this by applying the **chain rule** of derivatives.
- This algorithm is known as **backpropagation**.

$$\mathscr{L}(y,\hat{y}) = L(\mathbf{W},b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

Neural Network: the deeper, the better?

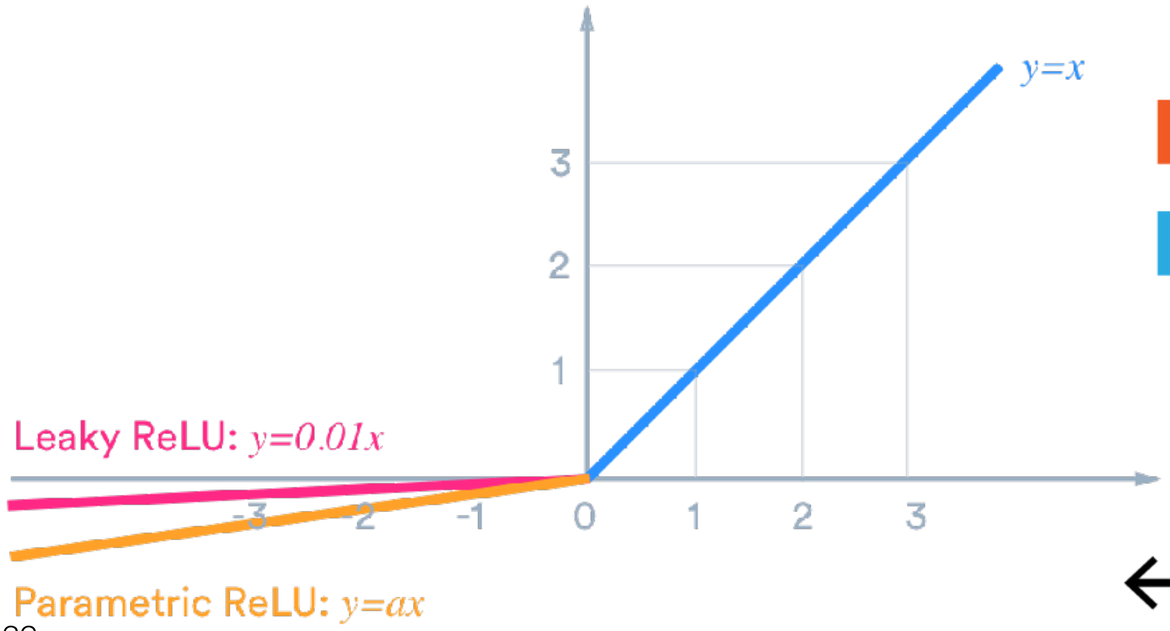
Not really.

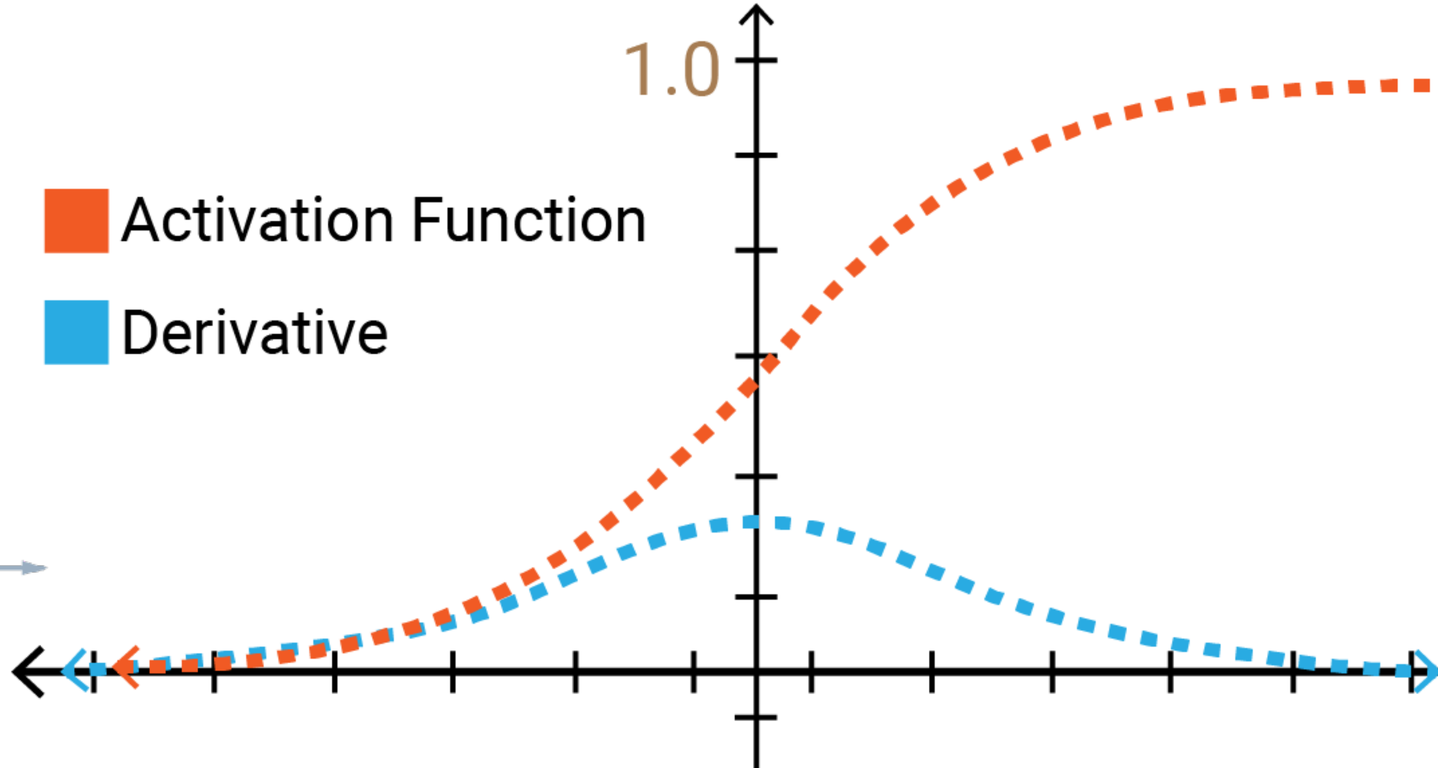
The vanishing gradient problem



Chain rule

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

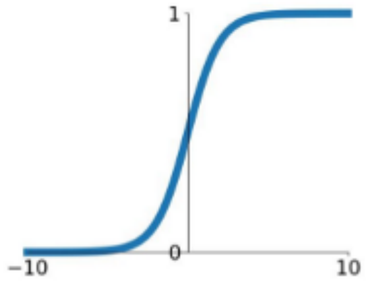




Activation Functions

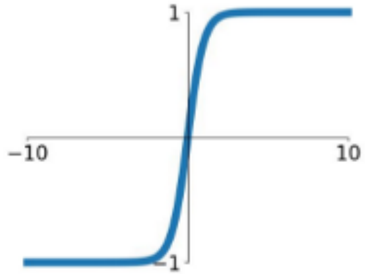
Sigmoid

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



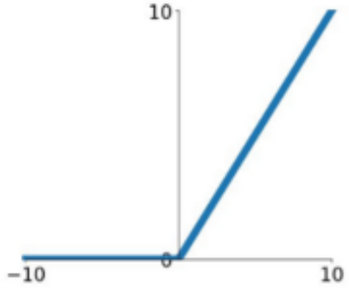
tanh

$$\tanh(x)$$



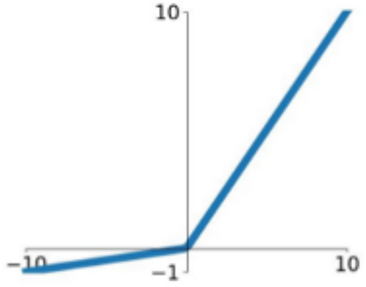
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

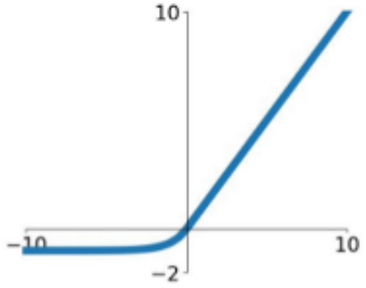


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

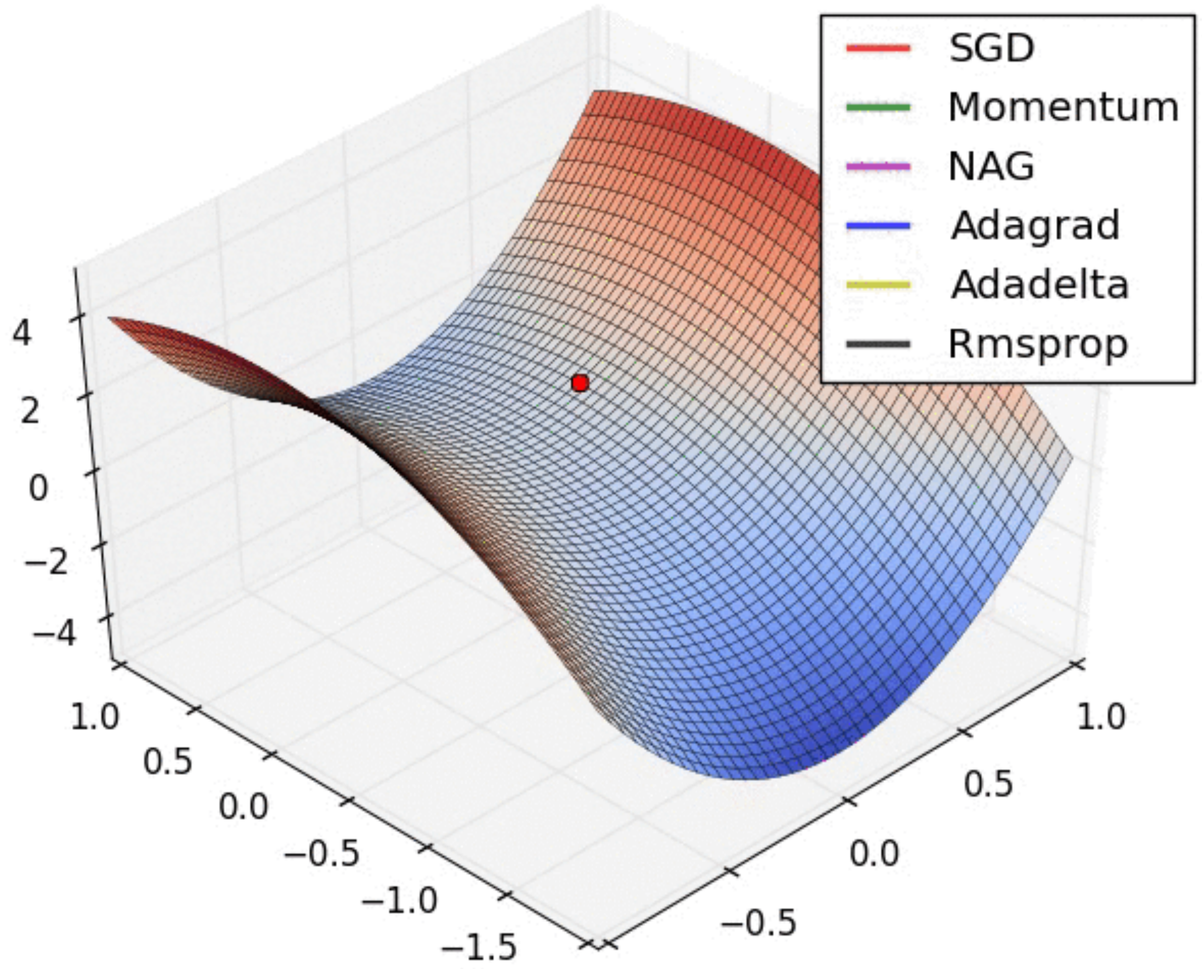
ELU

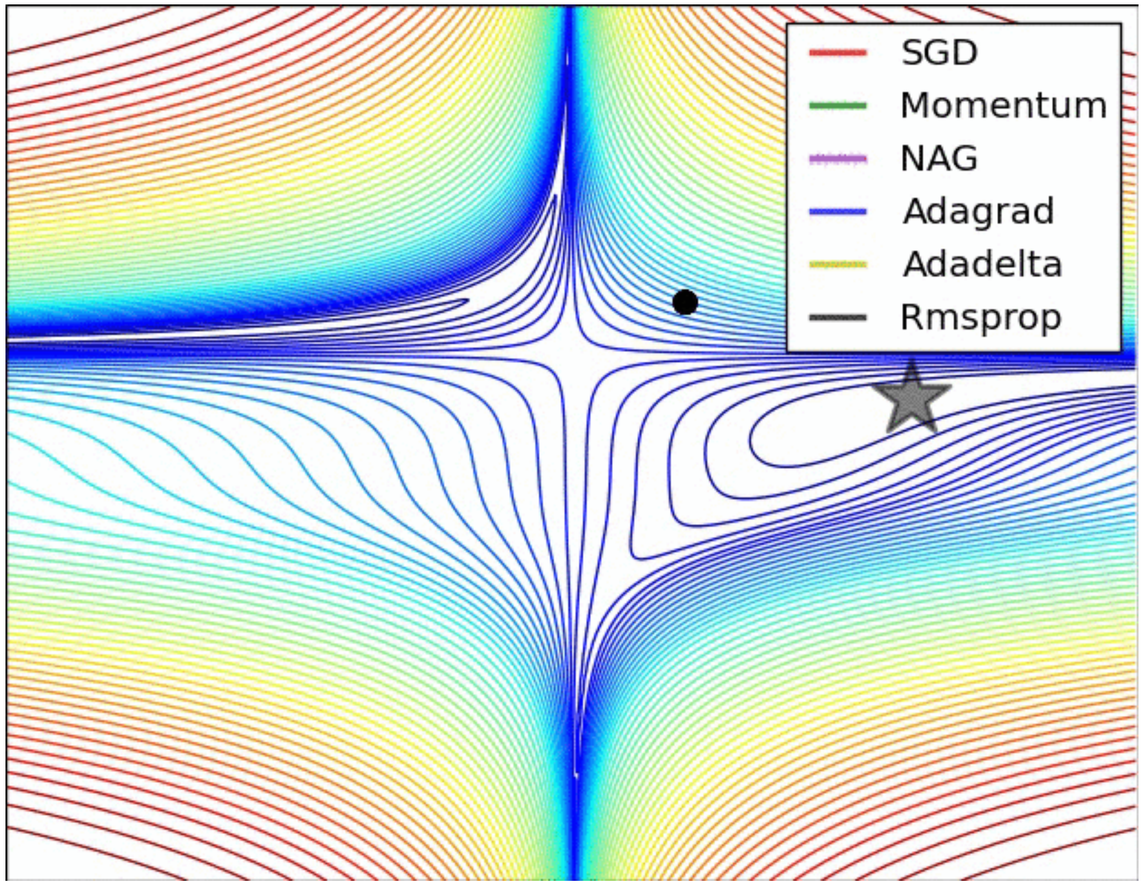
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



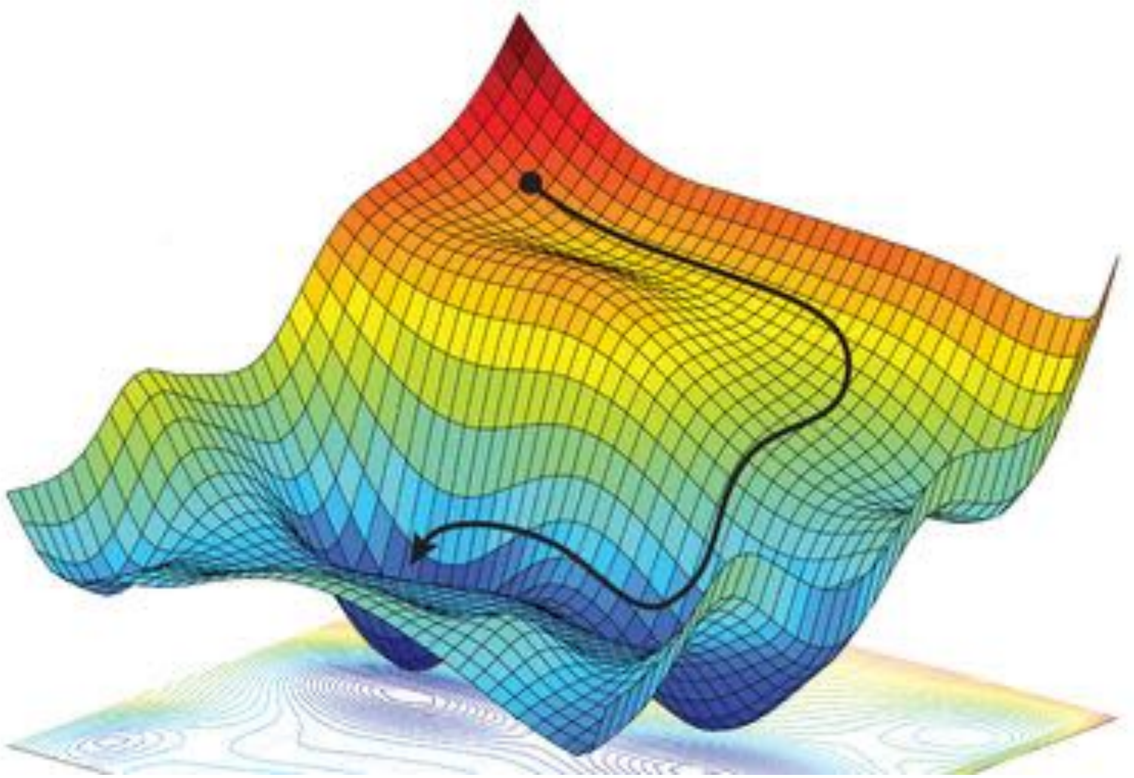
But one can design their own activation functions!

Optimizers





Source: medium.com/analytics-vidhya

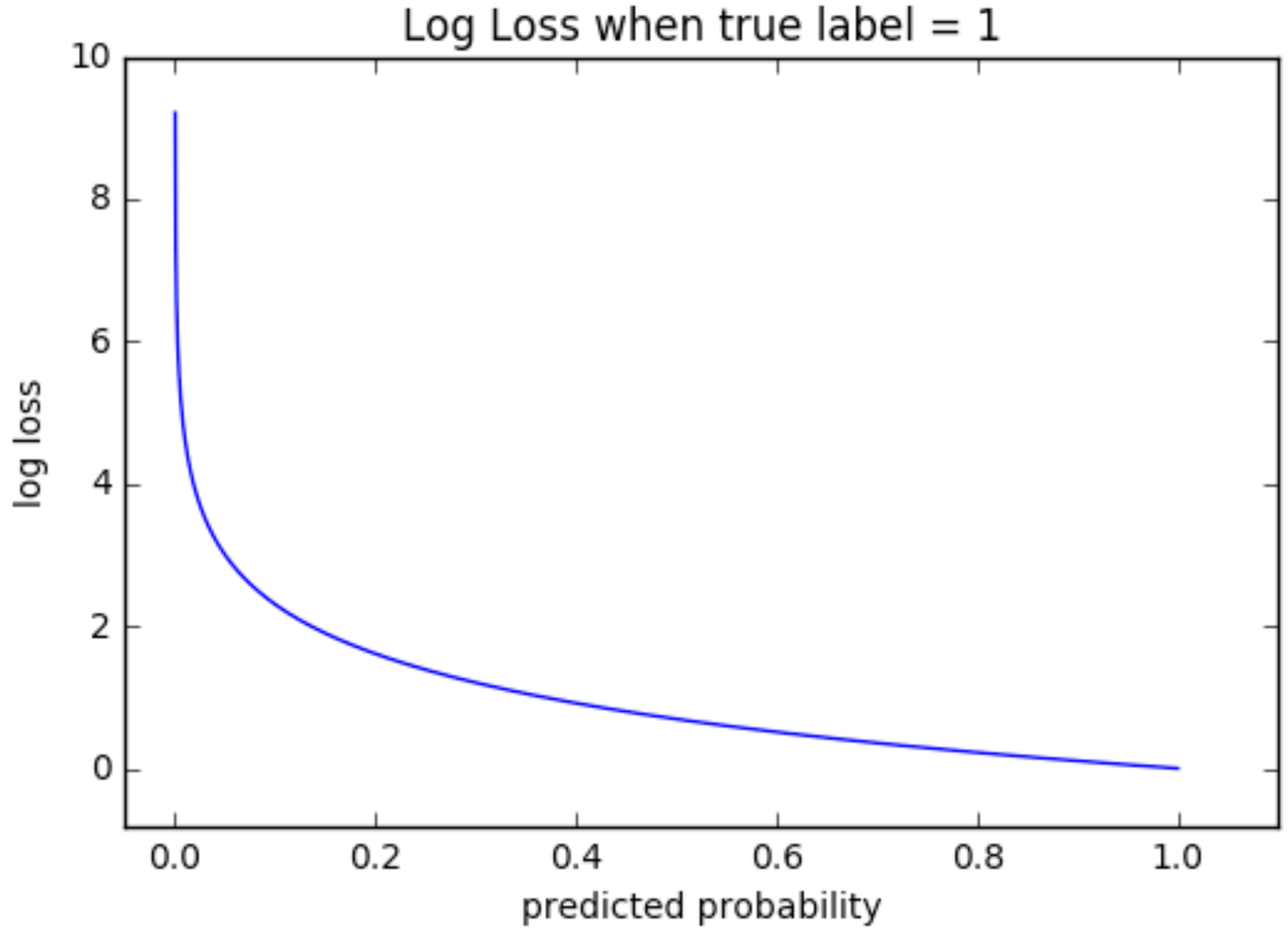


Common practice for loss functions

Regression
- Mean squared error
- Mean squared logarithmic error
- Mean absolute error

Binary Classification
- Binary cross-entropy
- Hinge loss
- Squared hinge loss

Multi-Class Classification
- Multi-class cross-entropy
- Sparse multi-class cross-entropy
- Kullback-Leibler divergence

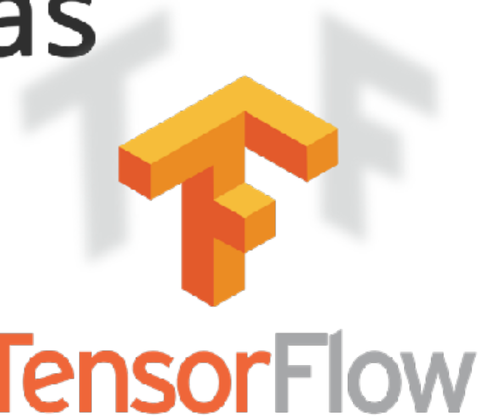


Cross-entropy loss outputs a log probability

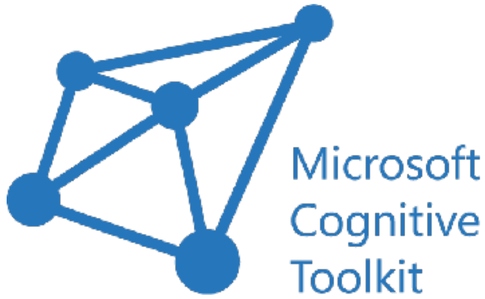
DL frameworks

In DL, you need to

- Define neurons and layers
- Define loss function
- Calculate losses
- Calculate gradient (multivariate calculus)
- Backward propagation
- Update weights

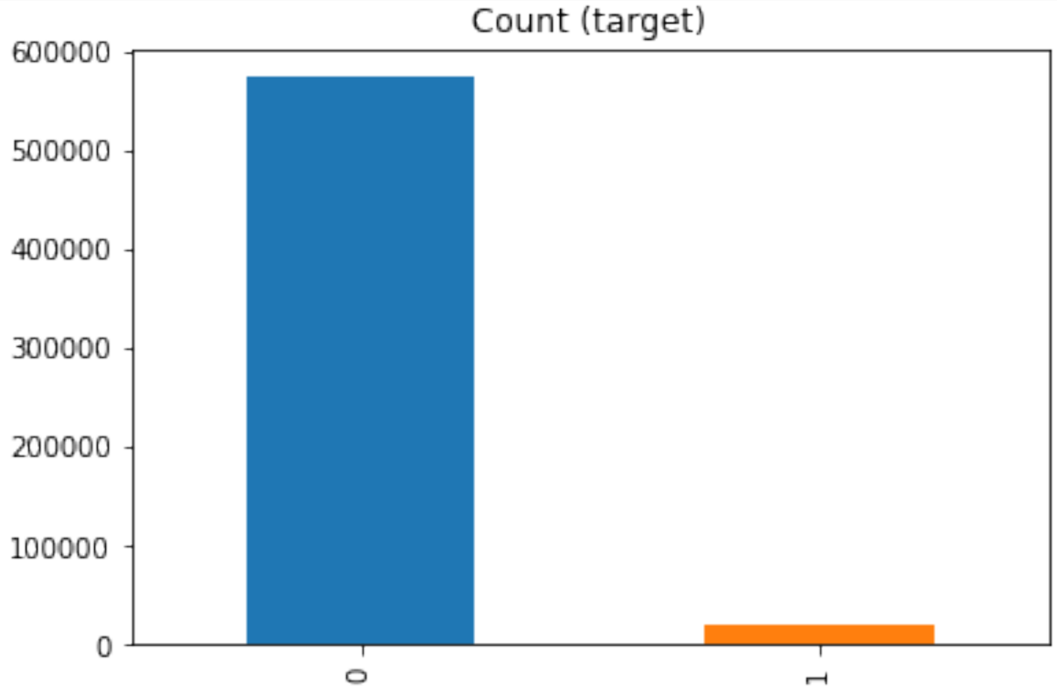


- Many frameworks exist; **TensorFlow**, **CNTK**, **Torch**, **Keras**, **Theano**, **Caffe**, ...
- We will use **TensorFlow/Keras**
- Keras used to call TensorFlow as a *backend*, but is now fully integrated in TensorFlow.

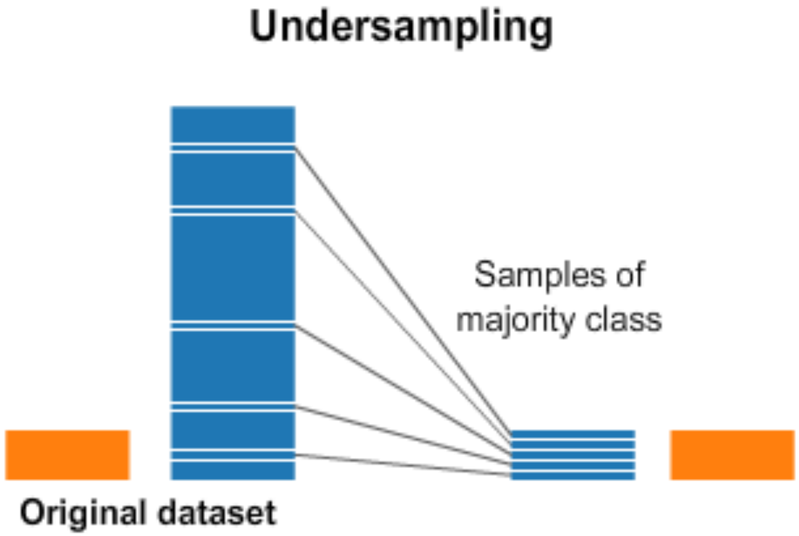


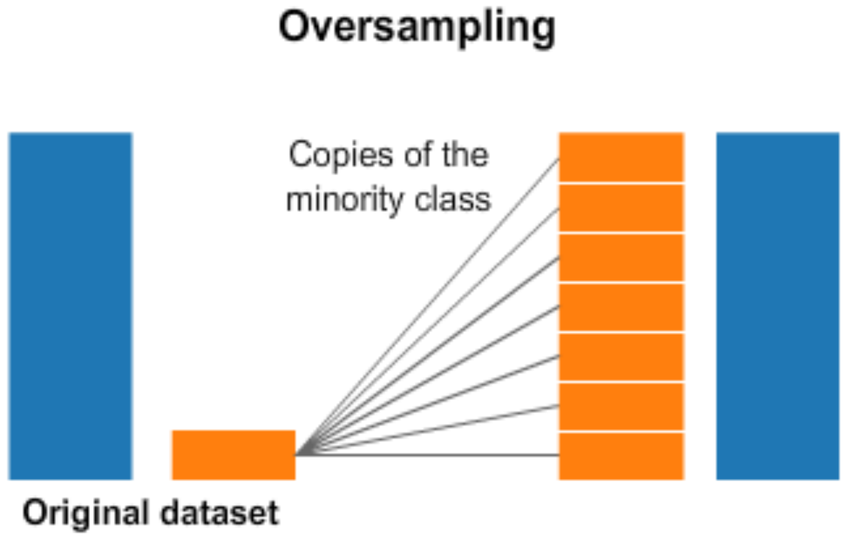
Model Evaluation

Balanced/Imbalanced training set



Pay attention to your data: they may fool your model.

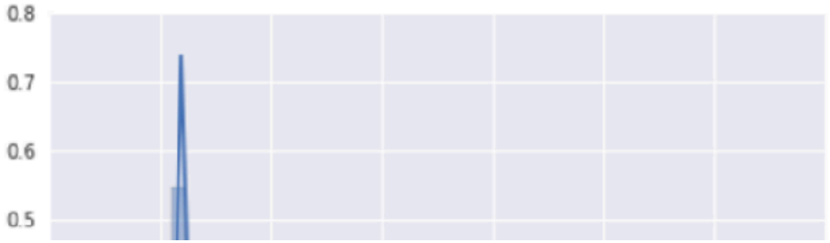


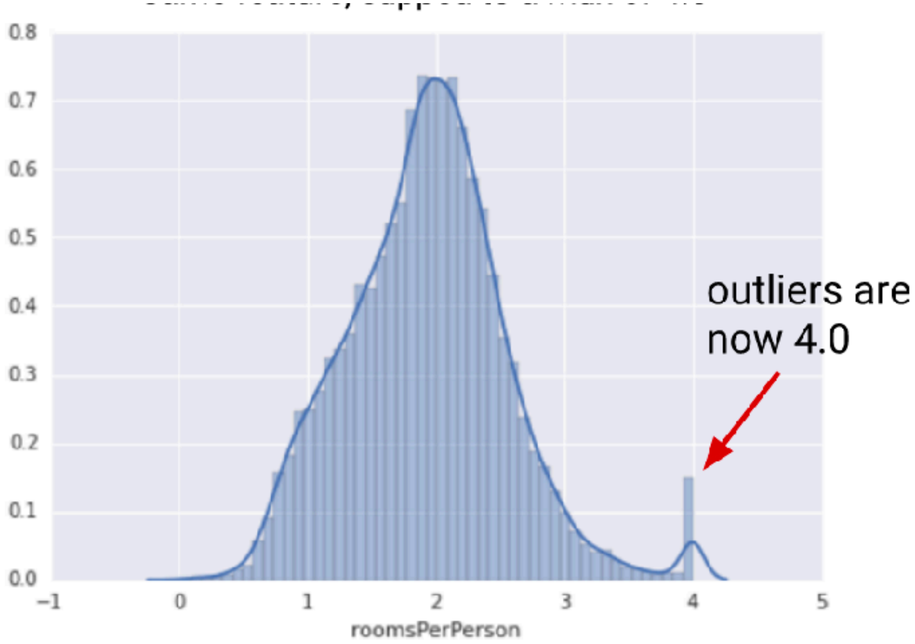


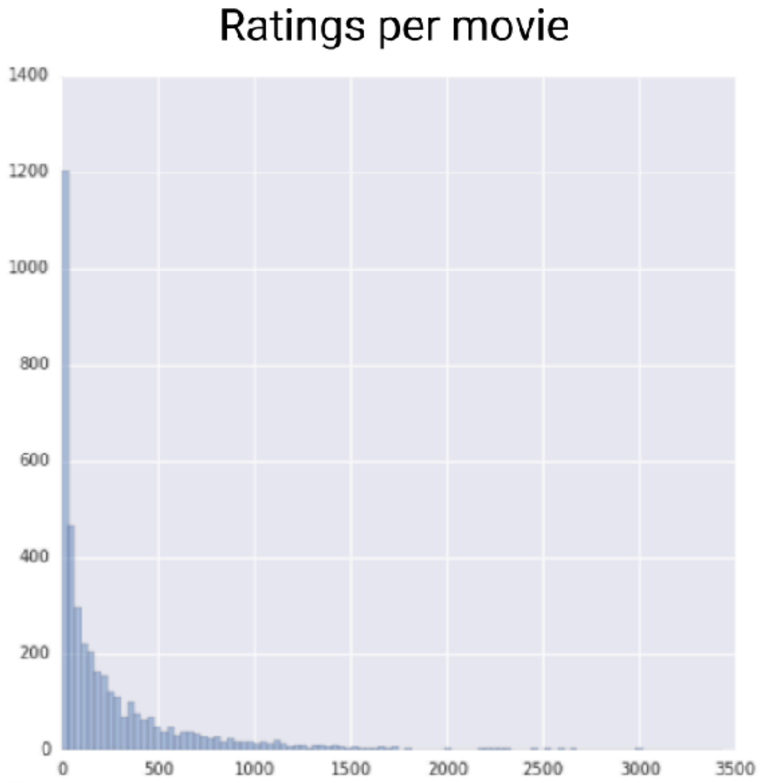
Source: kaggle.com

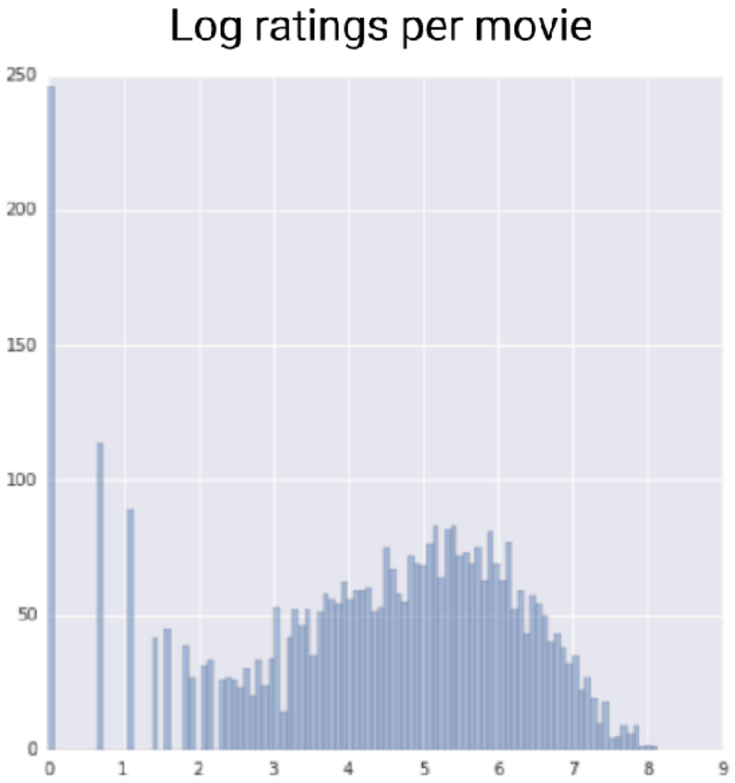
Data Normalization

A process to transform the input **data** in a **well-behaved** form.









Further reading: sklearn scalers

Source: developers.google.com

Dataset splitting

Training set

Model training

Best model parameters
Weights

70%

Validation set

Model selection

Best hyperparameters
Learning rate
#neurons
#layers

20%

Test set

Model testing

Final model
Accuracy
Sensitivity/specificity

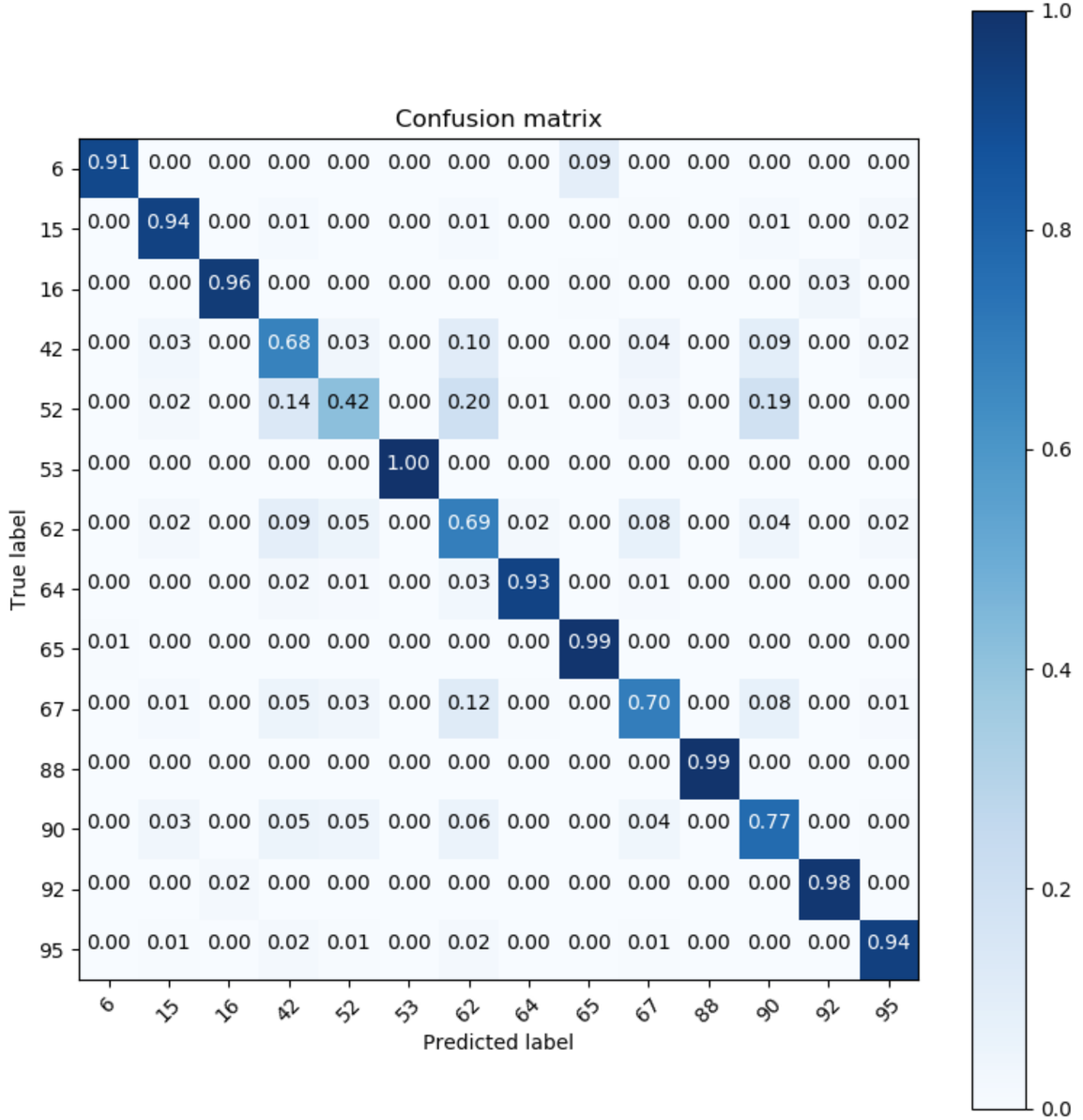
10%

Confusion Matrix

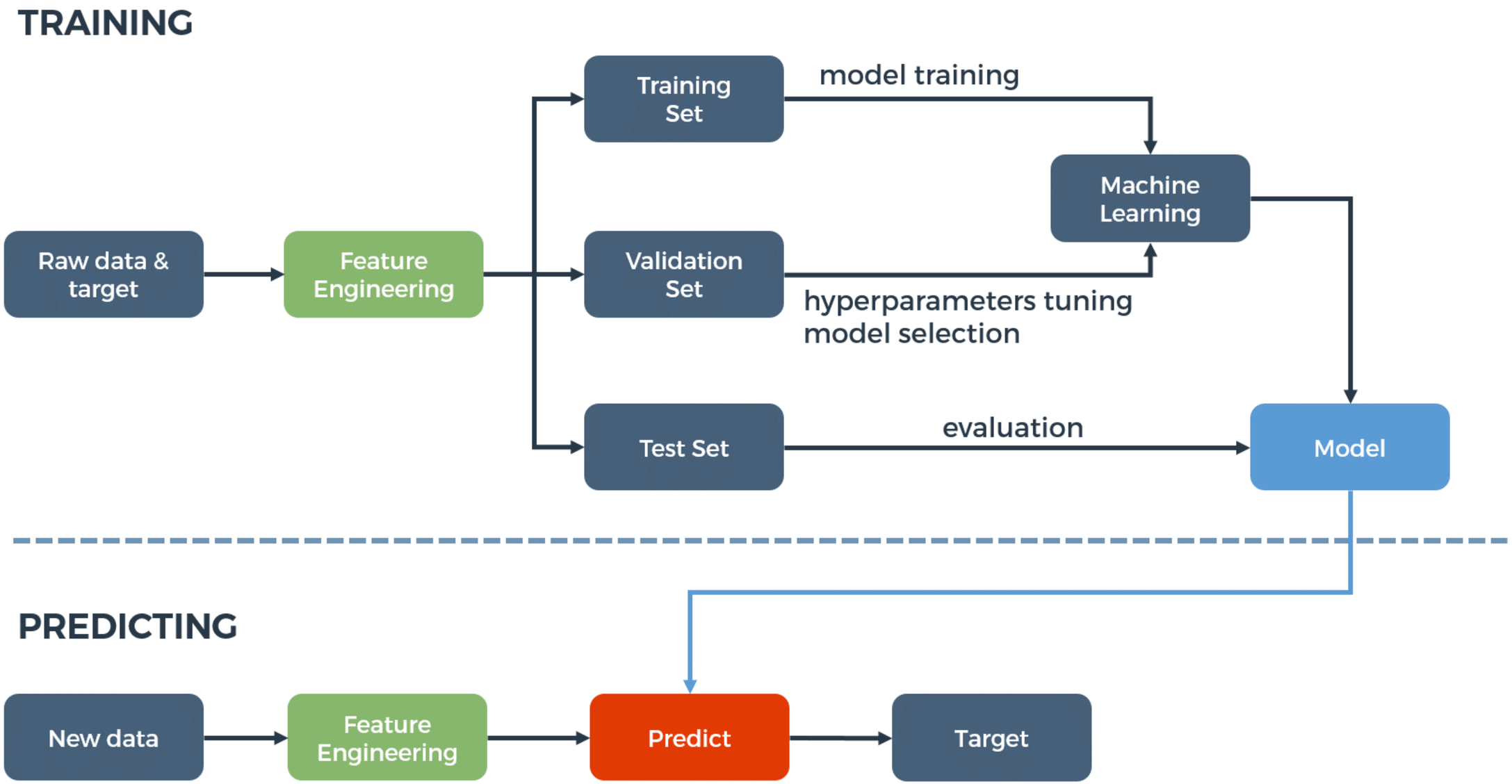
		Prediction outcome p	Prediction outcome n	total
actual value	p′	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Accuracy = (TP + TN) / (TP + FN + FP + TN)
Precision (p) = TP / (TP + FP)
Recall (r) = TP / (TP + FN)

$$F_1 = \frac{2}{r^{-1} + p^{-1}}$$



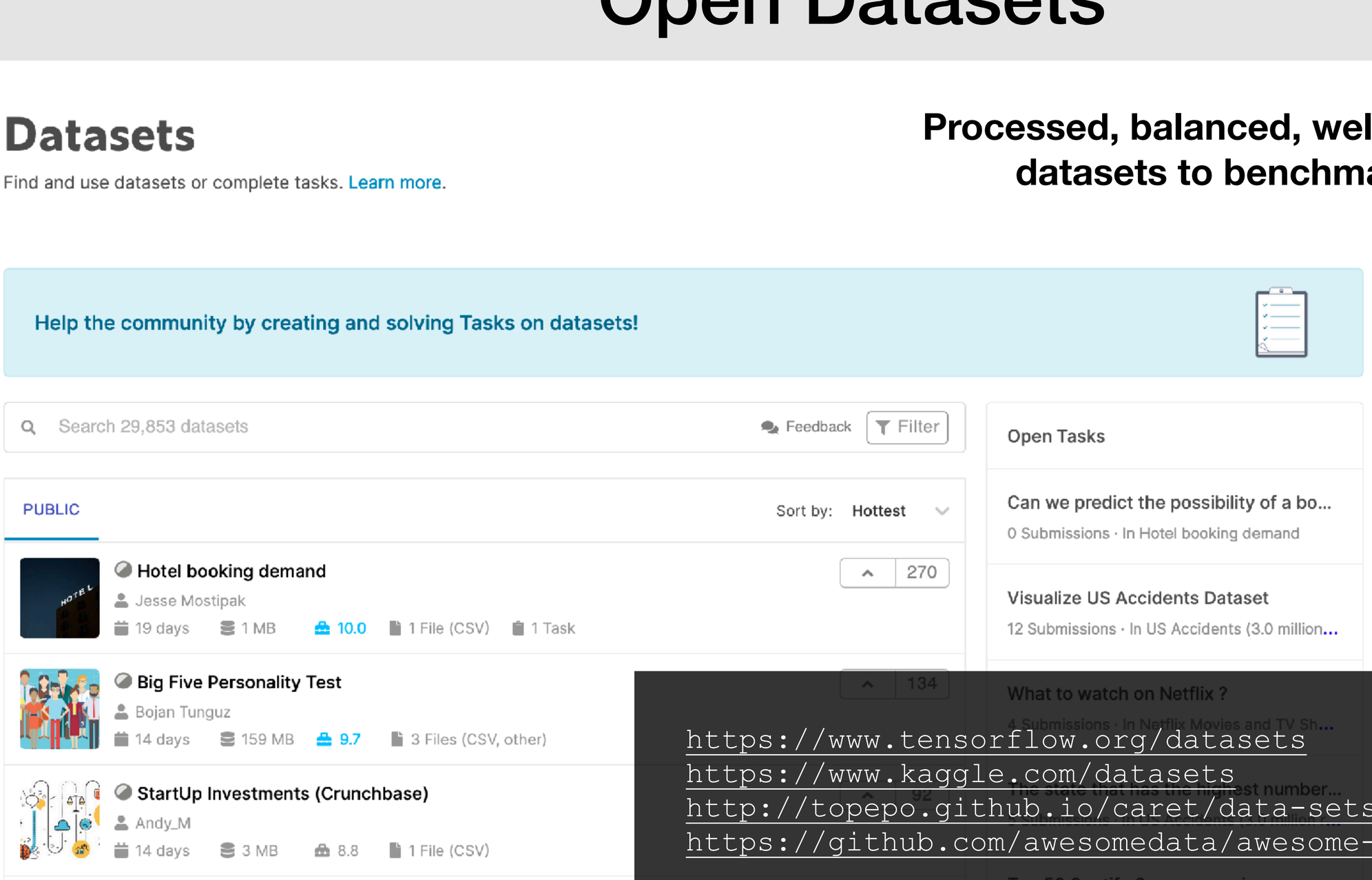
General Workflow of ML/DL



Source: hackernoon.com

Open Datasets

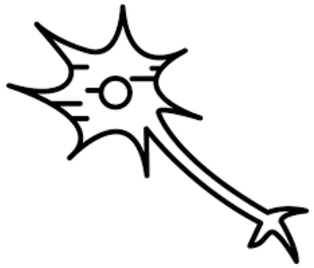
Processed, balanced, well-behaved, labeled datasets to benchmark your networks!

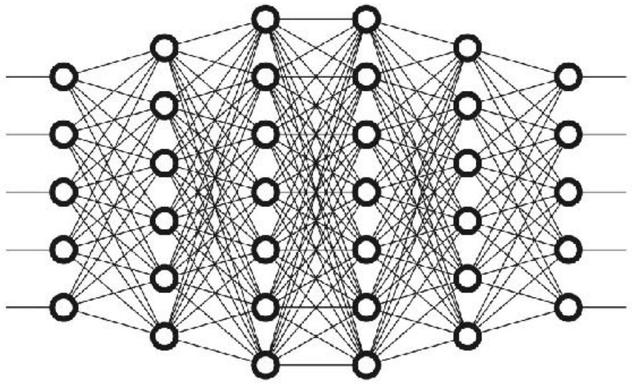


https://www.tensorflow.org/datasets
https://www.kaggle.com/datasets
http://topepo.github.io/caret/data-sets.html
https://github.com/awesomedata/awesome-public-datasets

Take-home messages

In a neuron:
… the main job is to calculate a **weighted average**
… the **decision** is made through the **activation** function





In a neural network:
… losses are calculated using the loss function
… losses are calculated by **comparing** the truths and the prediction
… **predictions** are made through **forward** propagation
… weights are **updated** through the **backward** propagation process
… **optimizers** are used to decide the weights updating **strategies**

In a deep learning workflow:
… the heavy lifting is mostly done by **DL frameworks**
… open datasets are crucial for benchmarking and bootstrapping DNNs

Go to https://jupyter.lisa.surfsara.nl/jhlsrf003

Select “SURF jupyterhub - course hours” profile

Notebook: notebook_1_mnist.ipynb

Hacking until 11:15