Bauman Moscow State Technical University

Lecture

Data Science and Machine Learning

Kanev Anton

NVIDIA Deep Learning Institute

This NVIDIA DLI Certificate has been awarded to

Anton Kanev

for the successful completion of

Fundamentals of Deep Learning

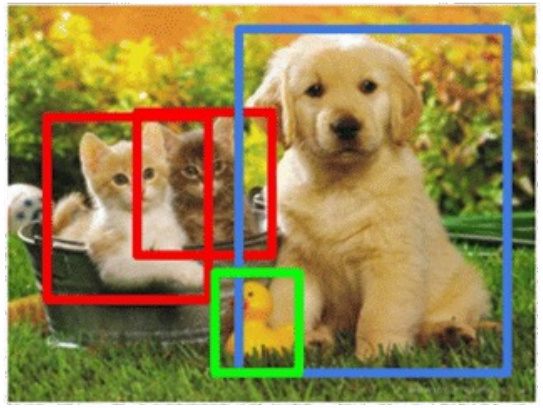
Machine Learning

- Machine learning is the ability to train a computer without detailed programming.
- To train a computer, sets of examples are used to solve problems that are difficult to represent in program code

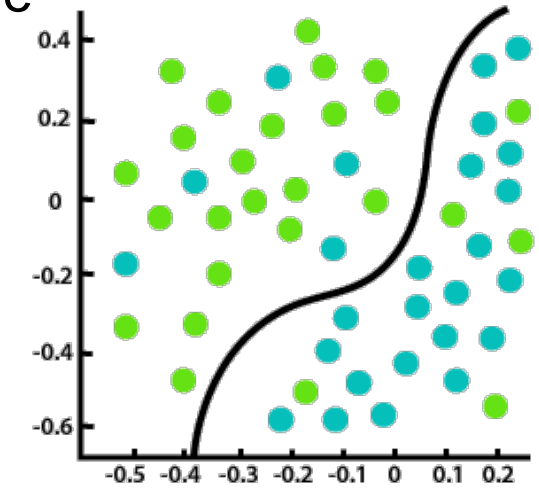
Classification

CAT

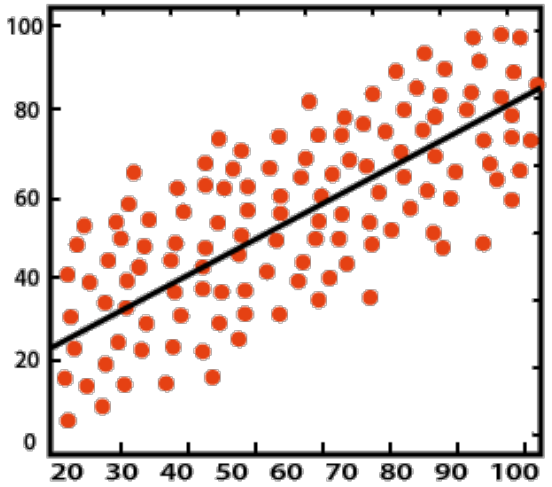
Object Detection



CAT, DOG, DUCK



Classification



Regression

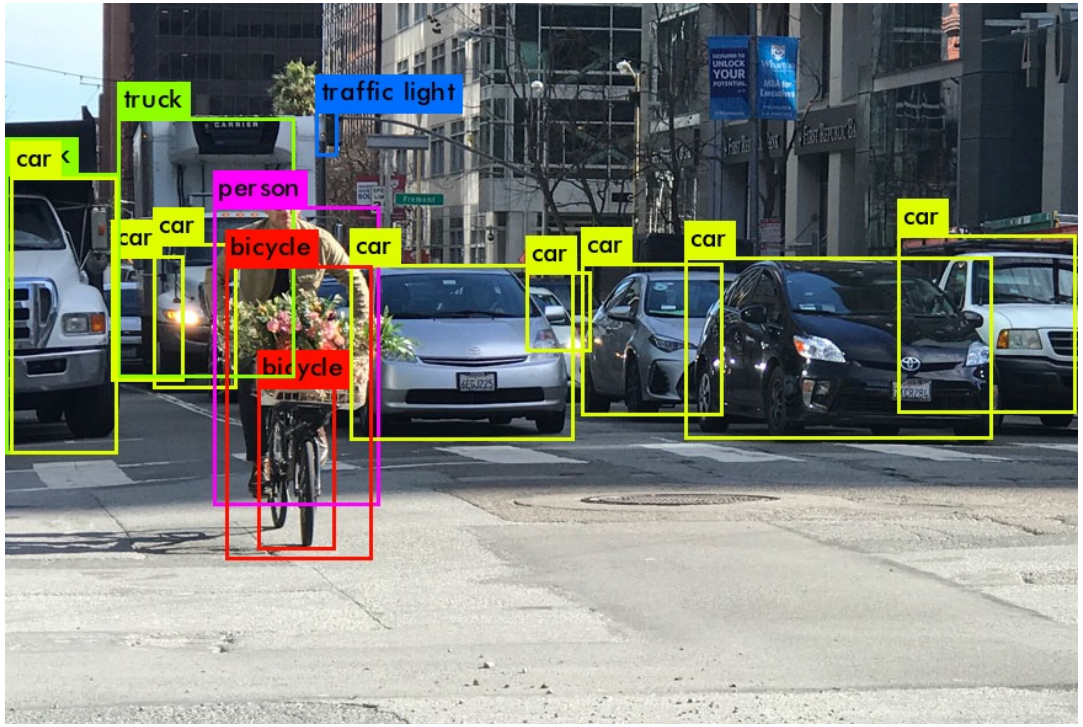
Types of machine learning

Supervised training
- Training data is labeled
- The task is to correctly determine the class of the object

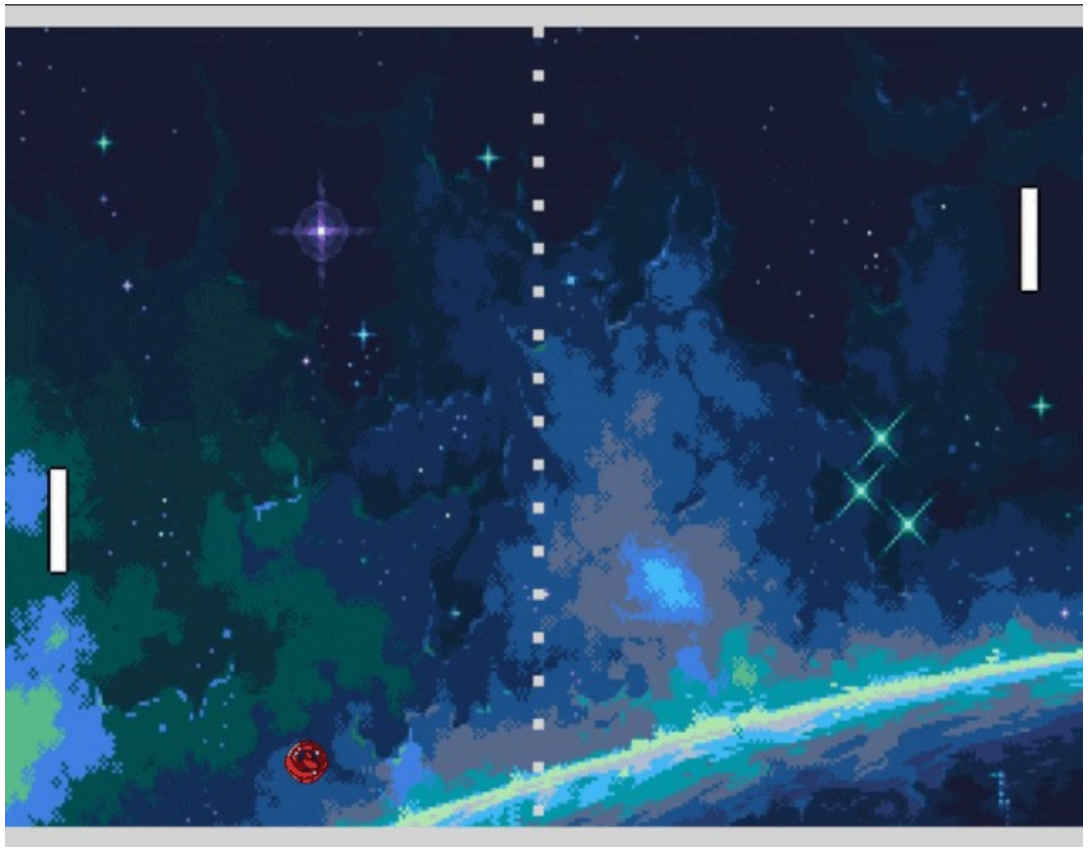
Reinforcement learning
- Training data is not labeled
- The system receives feedback from its actions
- The challenge is choosing the right actions

Unsupervised learning
- Training data is not labeled
- The task is to correctly determine the category

- Linear regression
- Decision tree
- K-nearest neighbors
- Neural networks



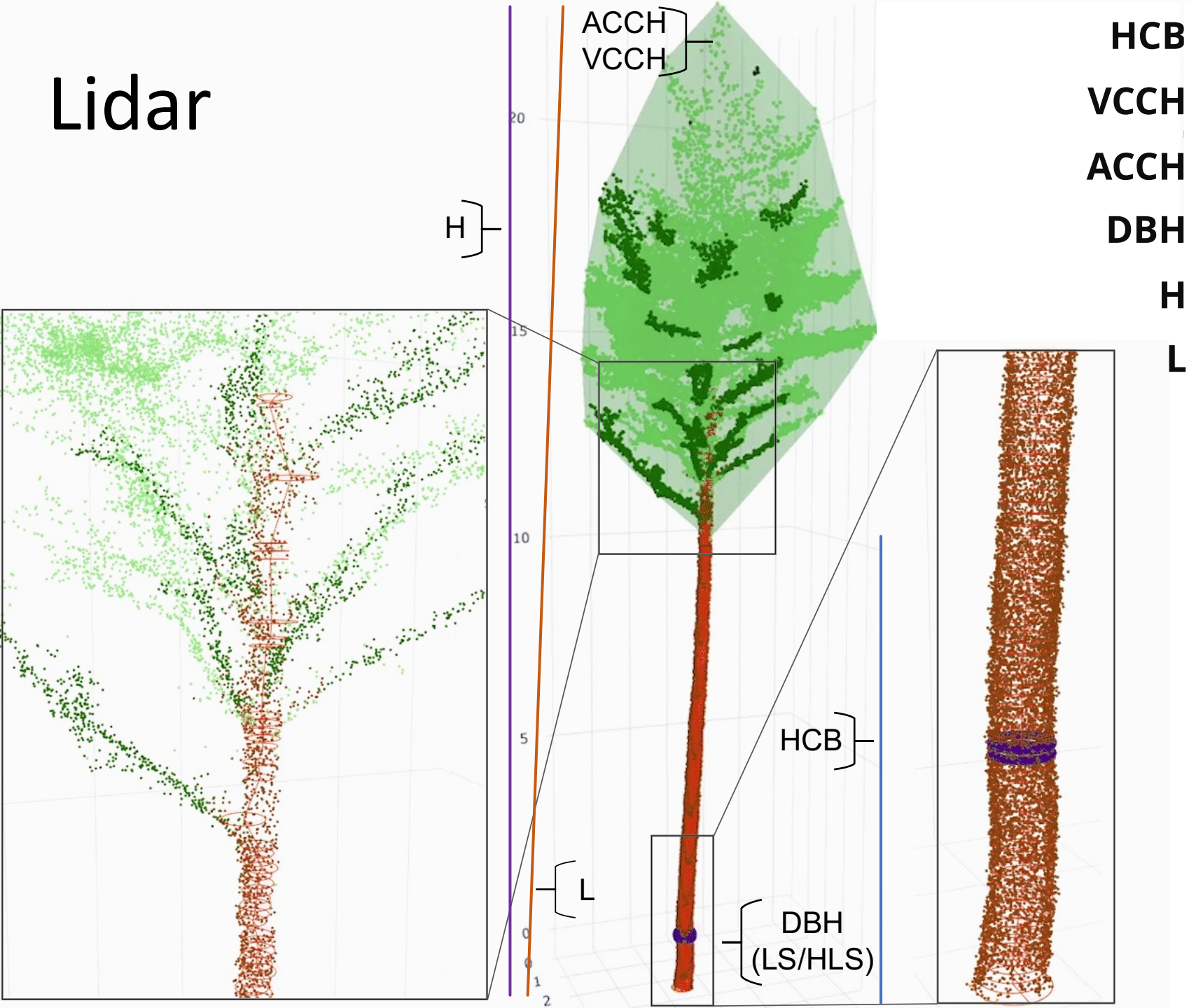
Our computer game for school students



https://github.com/iu5git/ai-bot-games-in-js

- An example of an intelligent ping-pong agent with neural network training
- Supervised learning with labeled data of gamer actions

Lidar



HCB Height to crown base = 10.331 m

VCCH Crown volume = 149.632 cubic m

ACCH Crown area = 154.790 sq m

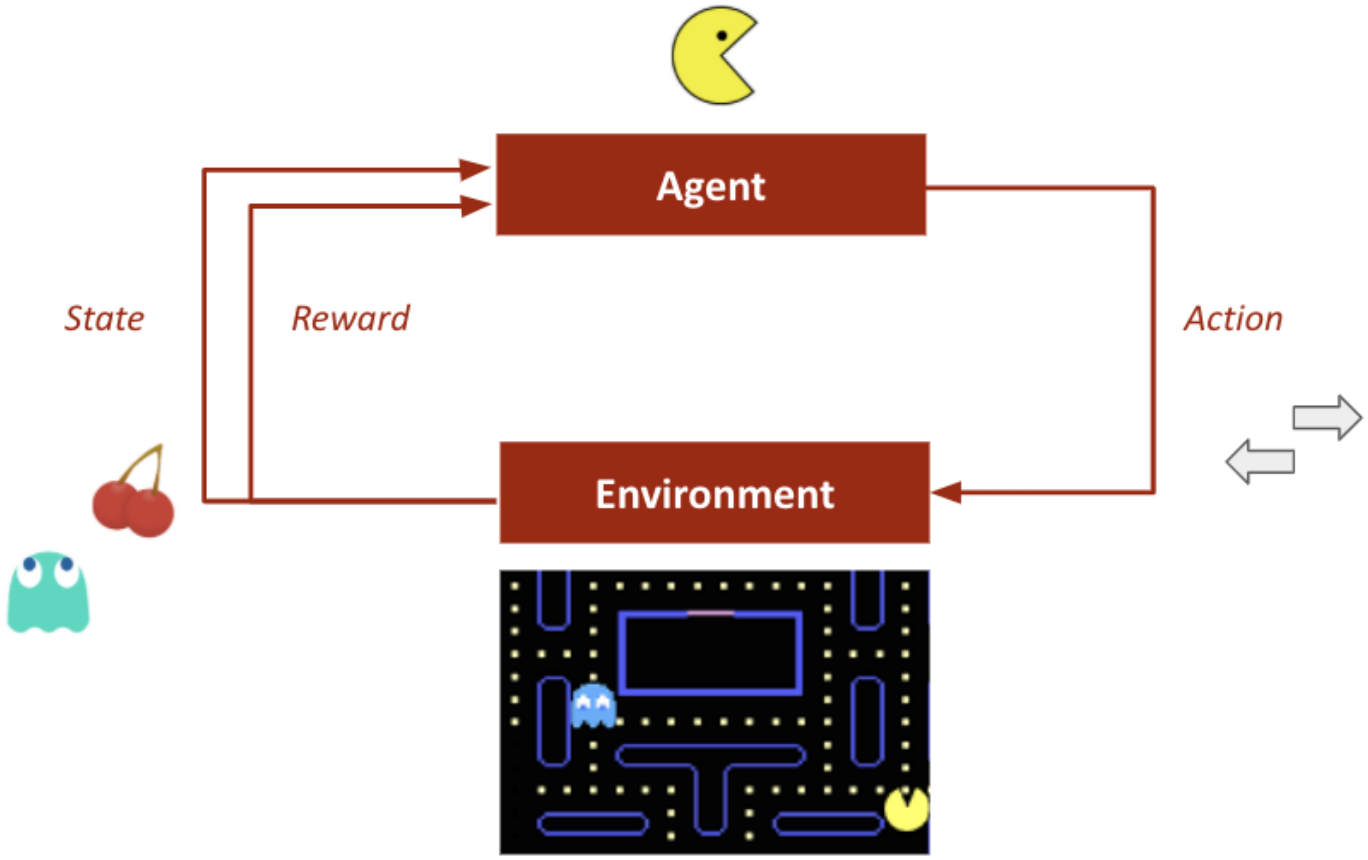
DBH Diameter = 28.76 sm / 28.85 sm

H Height = 21.734 m

L Length – 21.754 m

- LiDAR segmentation is an example of unsupervised learning

Reinforcement learning



Our ONNX practice

Step 3. Select class labels and get predictions

Выбрать файл cifar100_CNN.onnx Select ONNX file

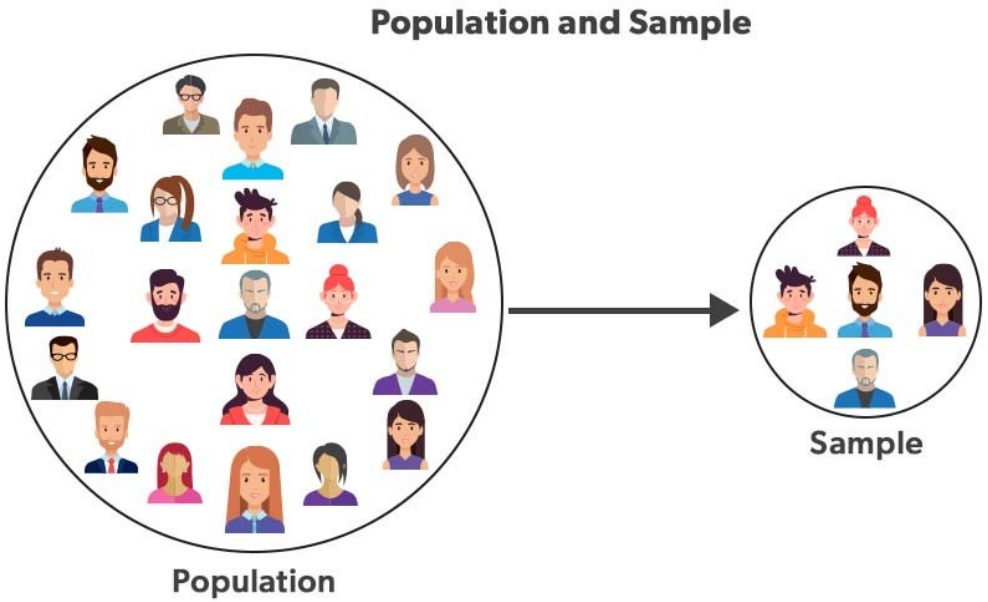
Выбрать файл c25c94fe96_1000.jpg

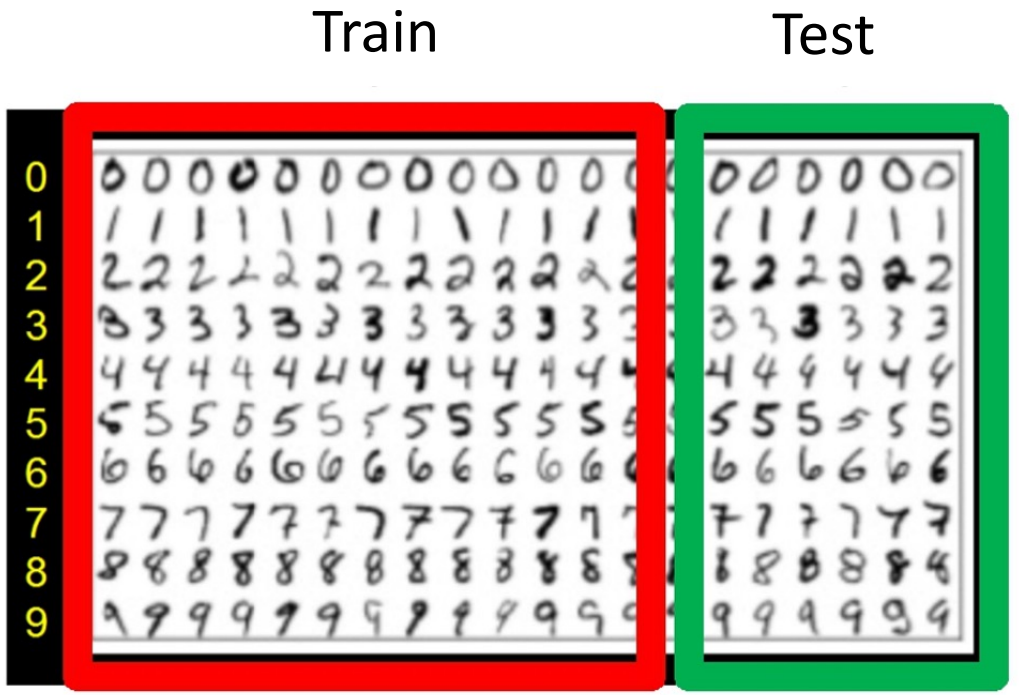
Class label 54 Add 10 Random Undo Reset

0,50,54

Dataset

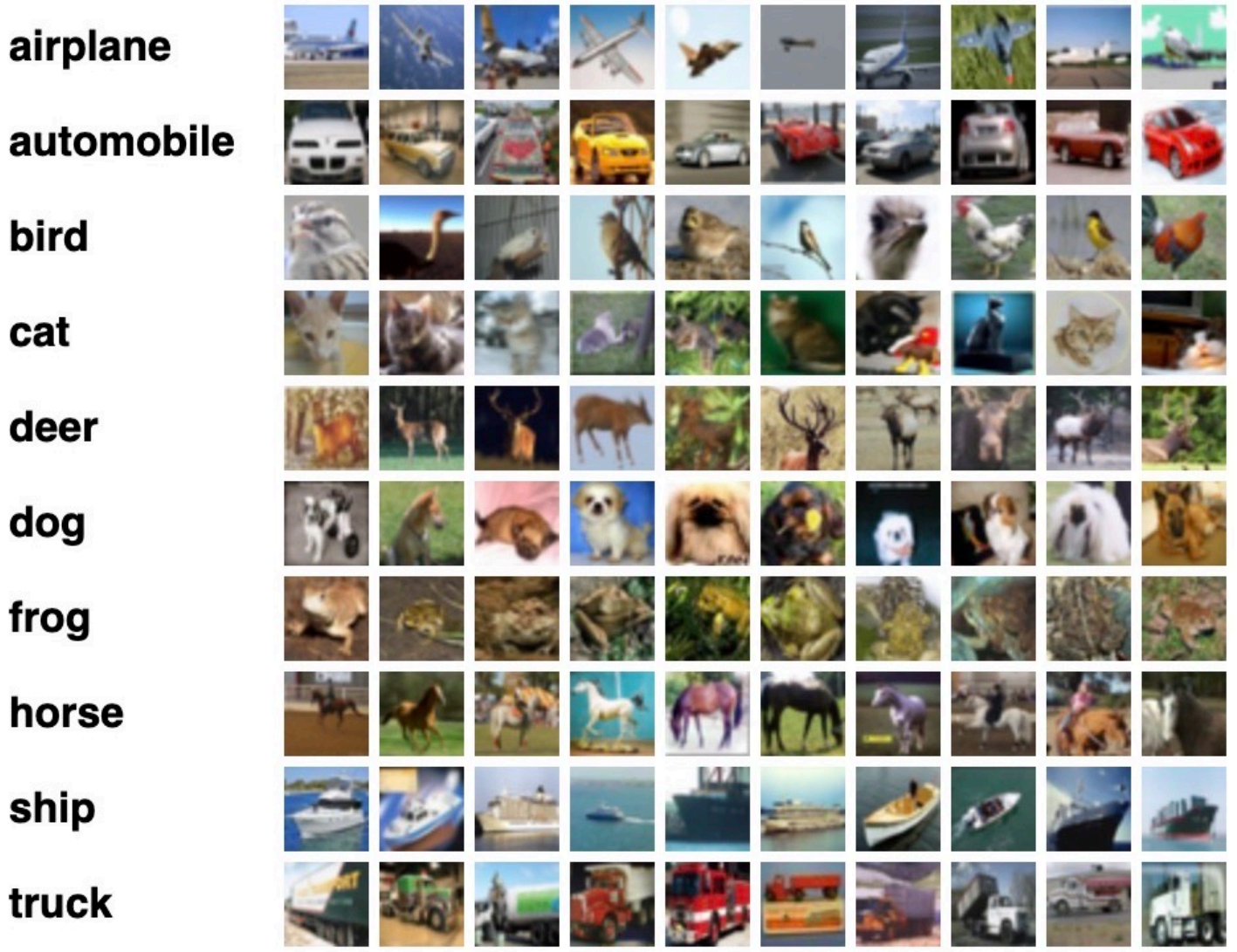
- We have a subset of the real world - a sample from the general population
- Examples in the sample are divided into test and training samples
- Each example has a label for classification





Cifar100 Dataset

- A dataset consisting of color images labeled into 100 classes
- The size of each image is 32 by 32 pixels
- 3 colors - three numbers for each pixel
- A total of 3072 numbers for each image - features for our training
- 50 000 images in train set and 10 000 in test set



Confusion matrix

Prediction: + − − + − +

Image:

True Positive (TP)

True Negative (TN)

False Negative (FN)

False Positive (FP)

- **True Positive**, TP: Correctly identified as appropriate
- **True Negative**, TN: Correctly identified as not conforming
- **False Positive**, FP: Incorrectly identified as compliant
- **False Negative**, FN: Incorrectly identified as not conforming

Dataset STL-10

Results from our practice

train

	precision	recall	f1-score	support
0	0.9940	1.0000	0.9970	500
55	1.0000	0.9900	0.9950	500
58	0.9960	1.0000	0.9980	500
accuracy			0.9967	1500
macro avg	0.9967	0.9967	0.9967	1500
weighted avg	0.9967	0.9967	0.9967	1500

---

test

	precision	recall	f1-score	support
0	0.8120	0.9500	0.8756	100
55	0.7396	0.7100	0.7245	100
58	0.7471	0.6500	0.6952	100
accuracy			0.7700	300
macro avg	0.7662	0.7700	0.7651	300
weighted avg	0.7662	0.7700	0.7651	300

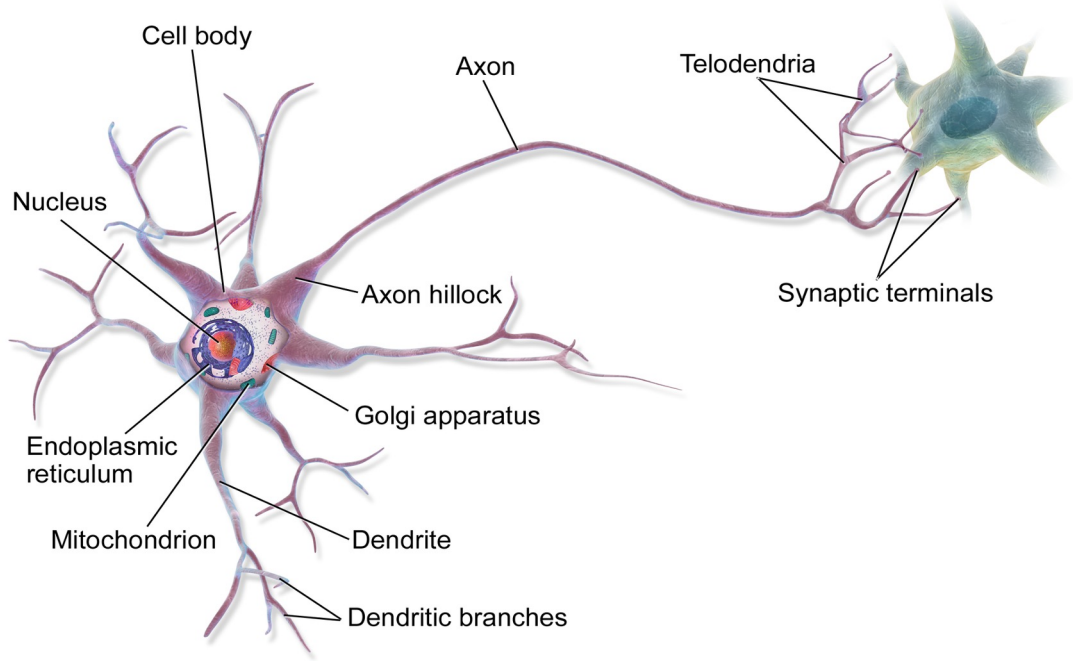
Precision
- Percentage of positive tags that are correctly identified
- Precision = (# true positives) / (# true positives + # false positives)

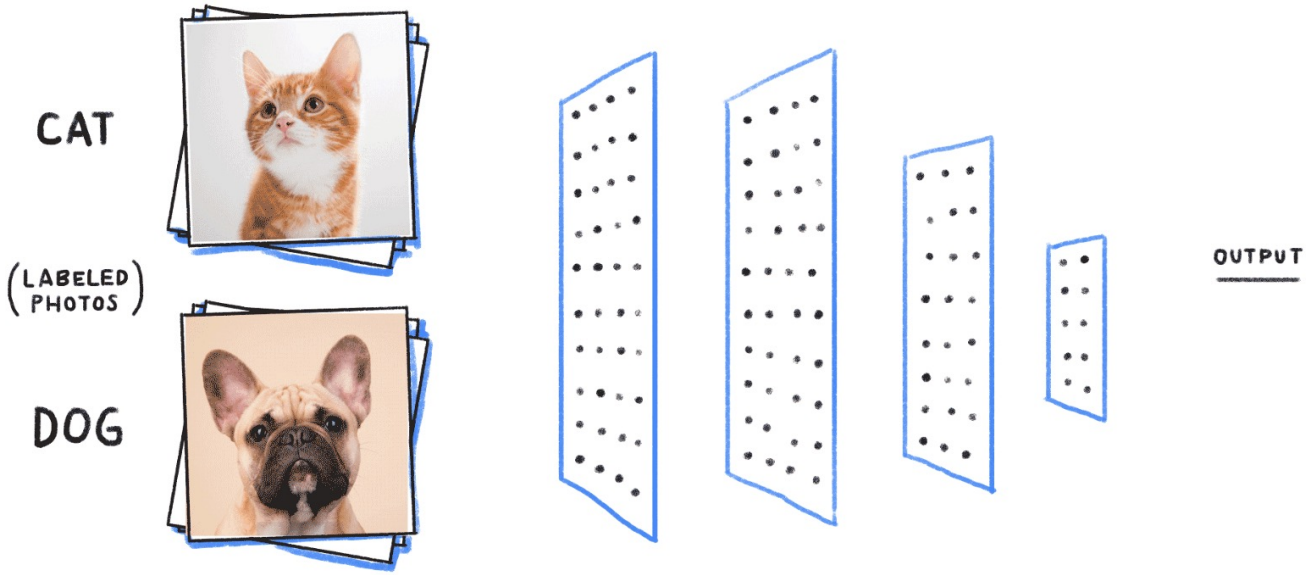
Recall
- Percentage of positive examples that were correctly identified
- Recall = (# true positives) / (# true positives + # false negatives)

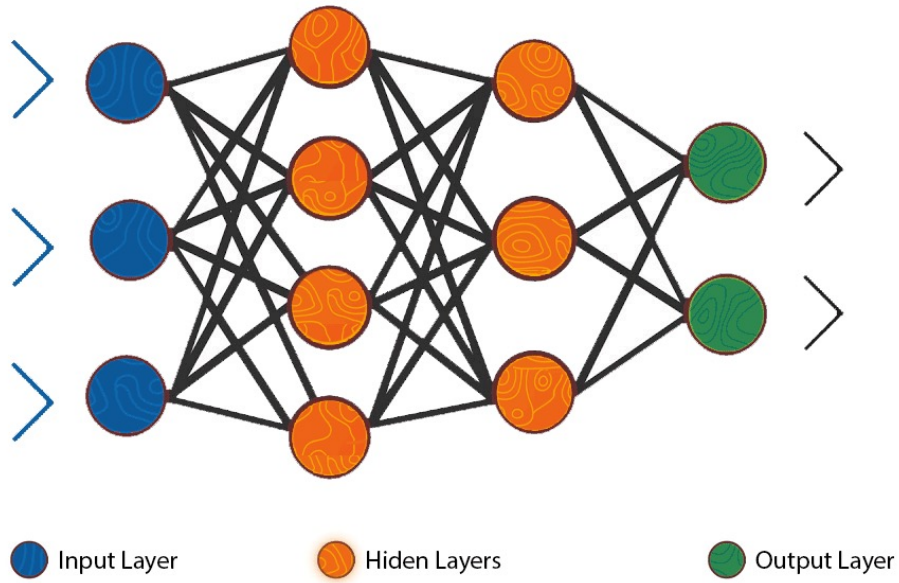
Accuracy
- Percentage of positive marks
- Accuracy = (# true positives + # true negatives) / (# of samples)

Neural Network

- Consists of layers, layers is a set of neurons
- Three types of layers: input data, hidden layers, output of model
- Fully Connected network is good at classification
- But it has many connections and weights





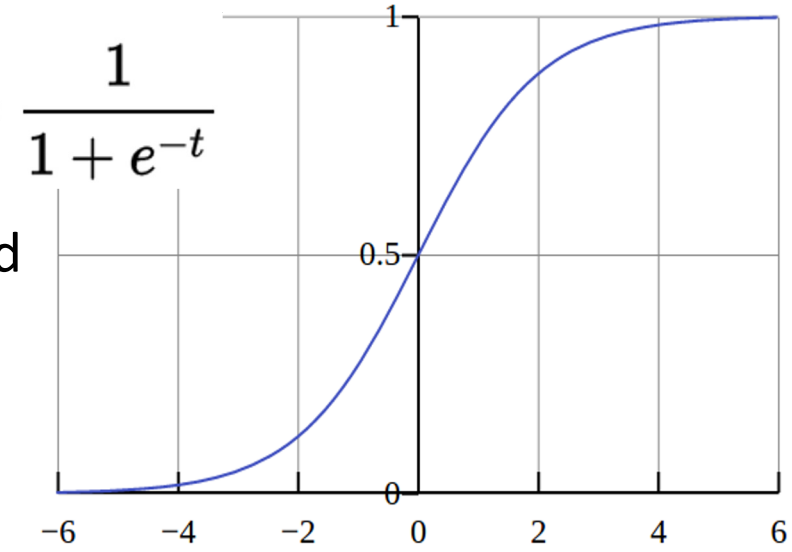


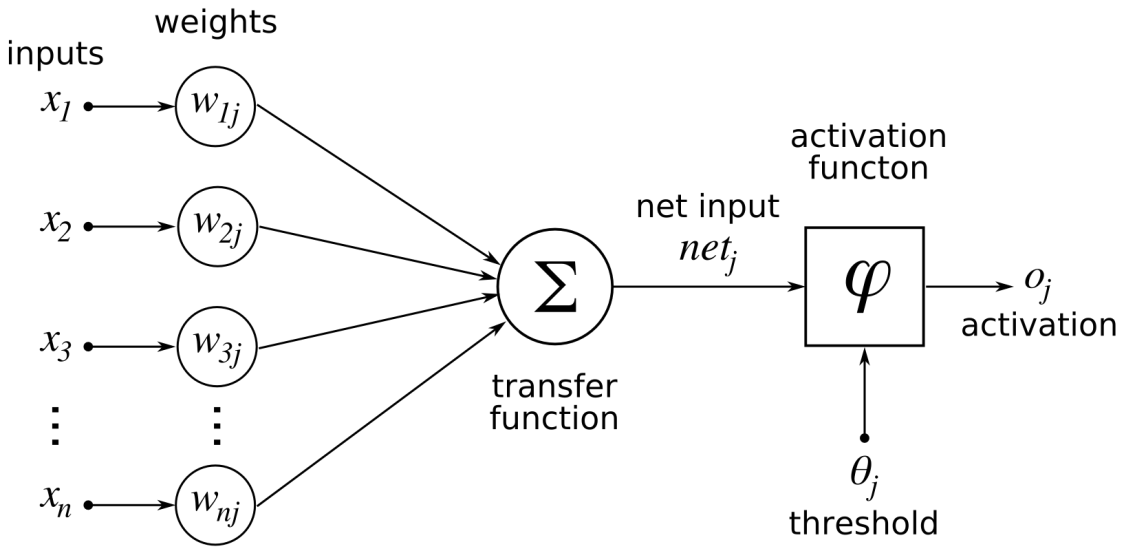
Neuron

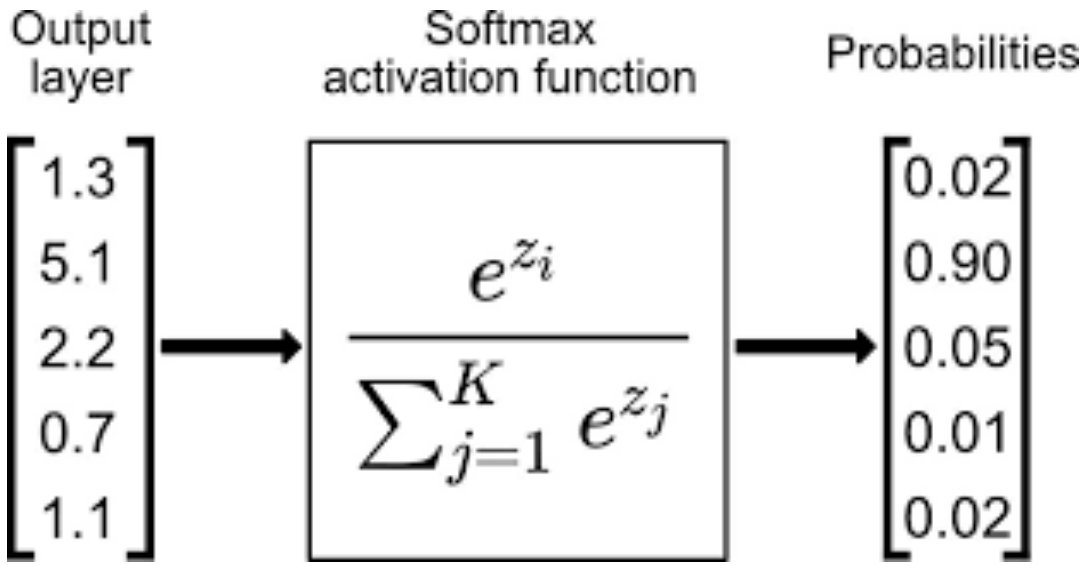
- Each neuron is a sum of weighted inputs
- We train weights and biases
- We need activation function after each neuron to have nonlinear function of several layers

$$S(t) = \frac{1}{1 + e^{-t}}$$

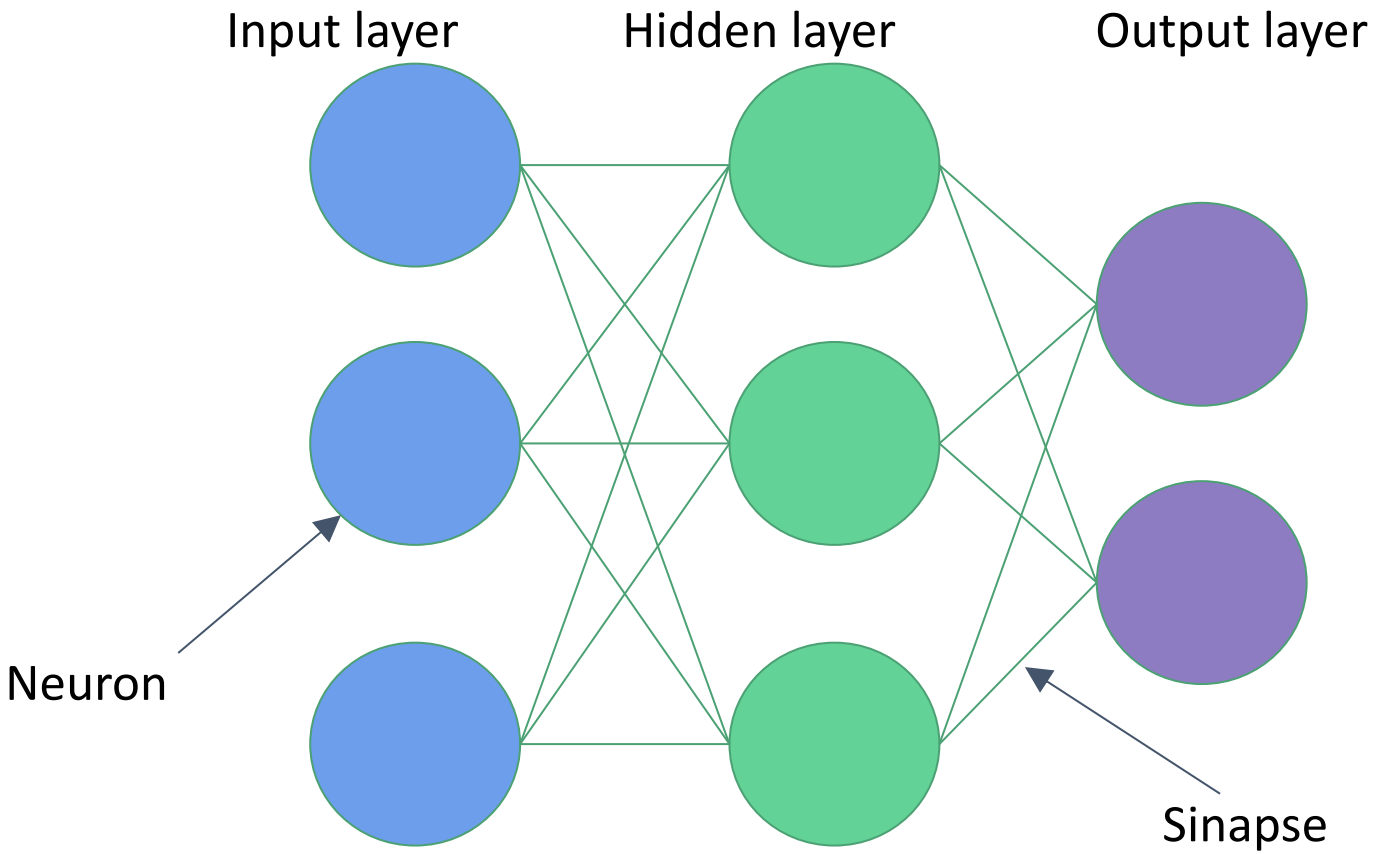
Sigmoid



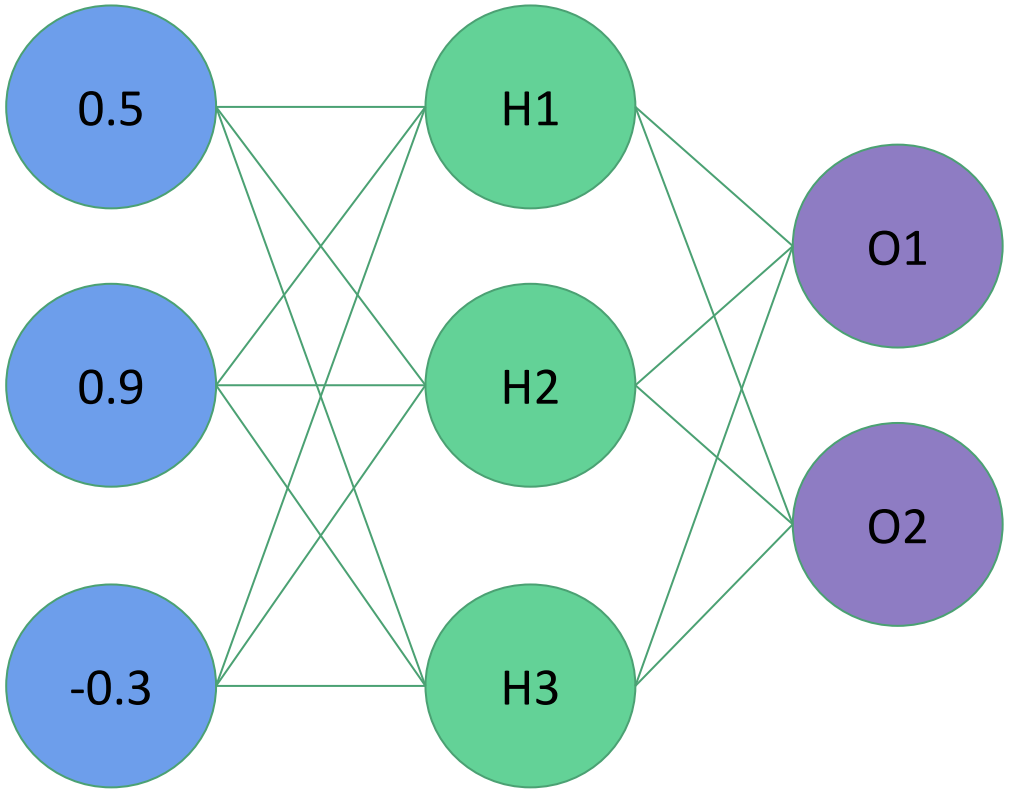




Fully Connected Neural Network



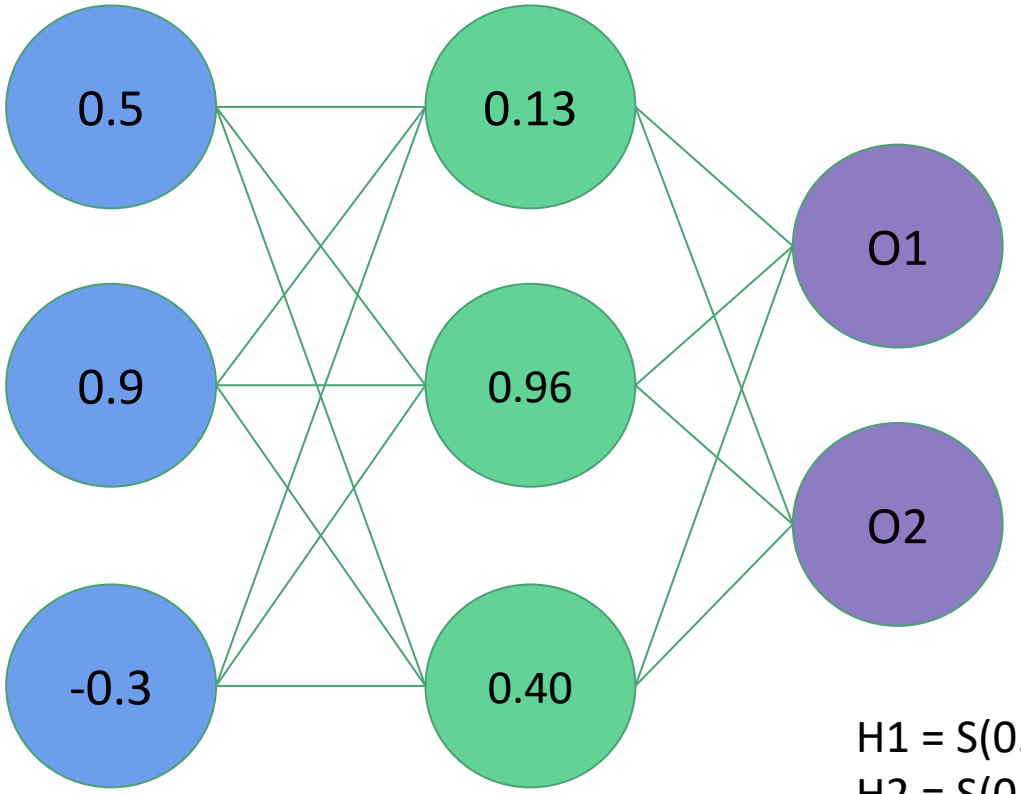
Inference



Weights H1 = (1.0, -2.0, 2.0)
Weights H2 = (2.0, 1.0, -4.0)
Weights H3 = (1.0, -1.0, 0.0)

Weights O1 = (-3.0, 1.0, -3.0)
Weights O2 = (0.0, 1.0, 2.0)

Inference

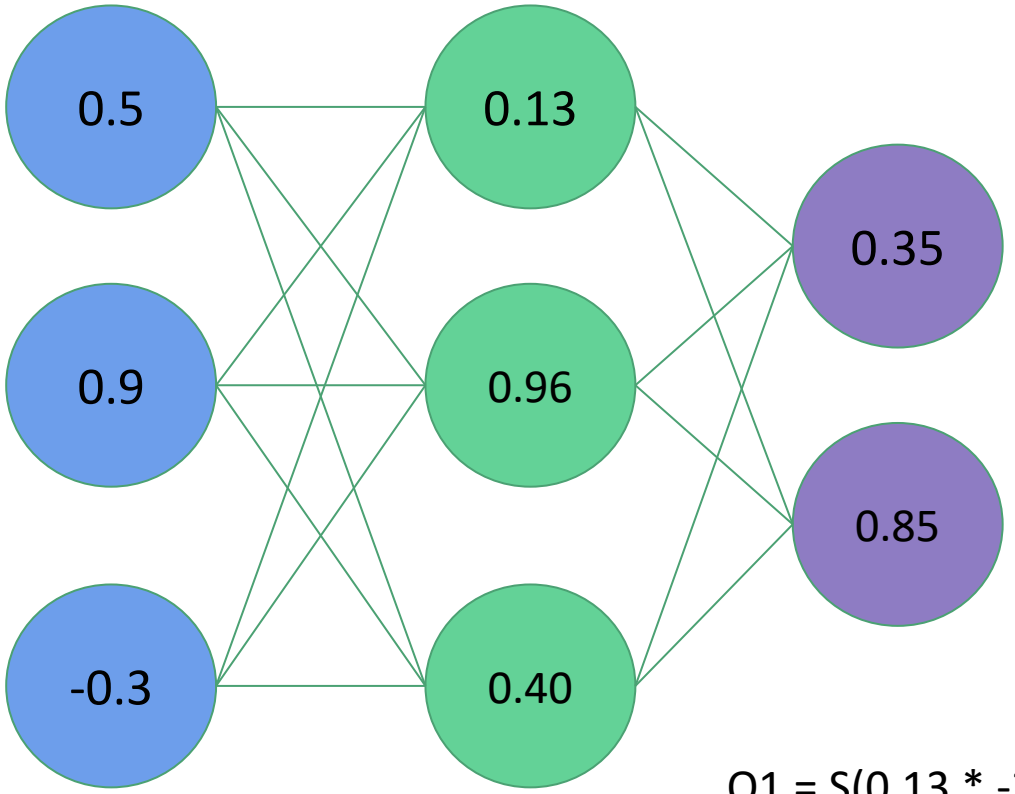


Weights H1 = (1.0, -2.0, 2.0)
Weights H2 = (2.0, 1.0, -4.0)
Weights H3 = (1.0, -1.0, 0.0)

Weights O1 = (-3.0, 1.0, -3.0)
Weights O2 = (0.0, 1.0, 2.0)

H1 = S(0.5 * 1.0 + 0.9 * -2.0 + -0.3 * 2.0) = S(-1.9) = 0.13
H2 = S(0.5 * 2.0 + 0.9 * 1.0 + -0.3 * -4.0) = S(3.1) = 0.96
H3 = S(0.5 * 1.0 + 0.9 * -1.0 + -0.3 * 0.0) = S(-0.4) = 0.40

Inference

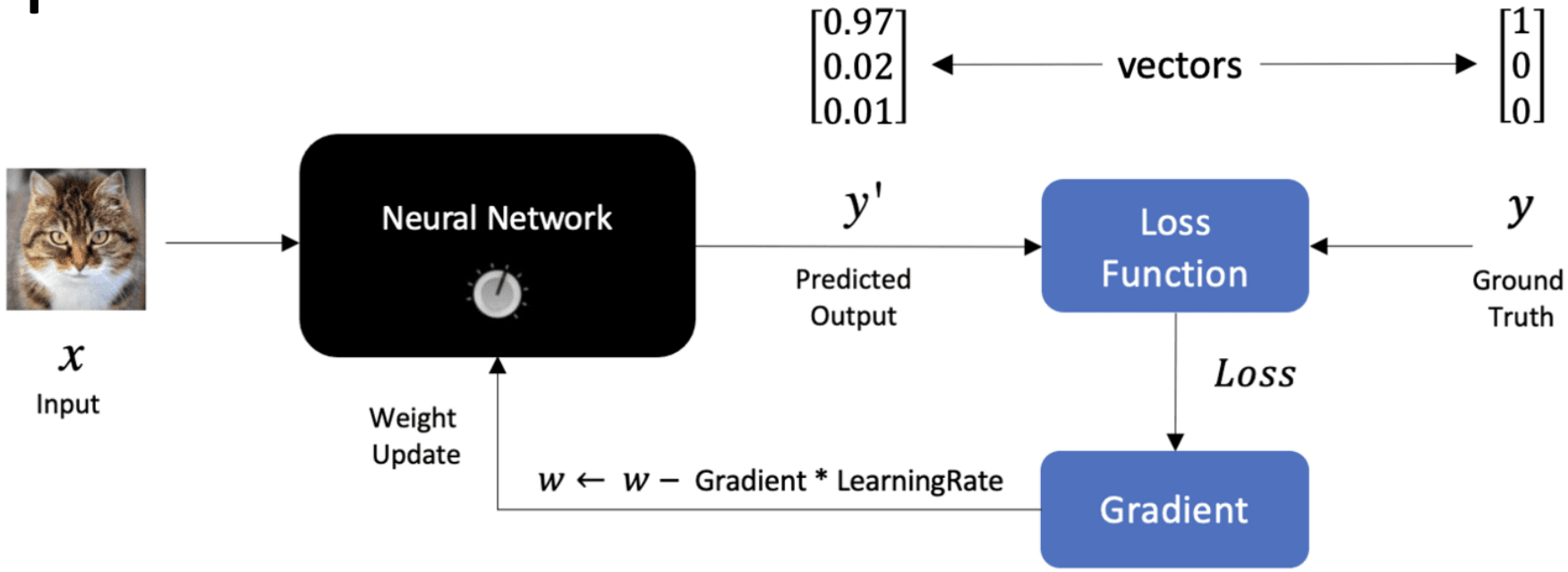


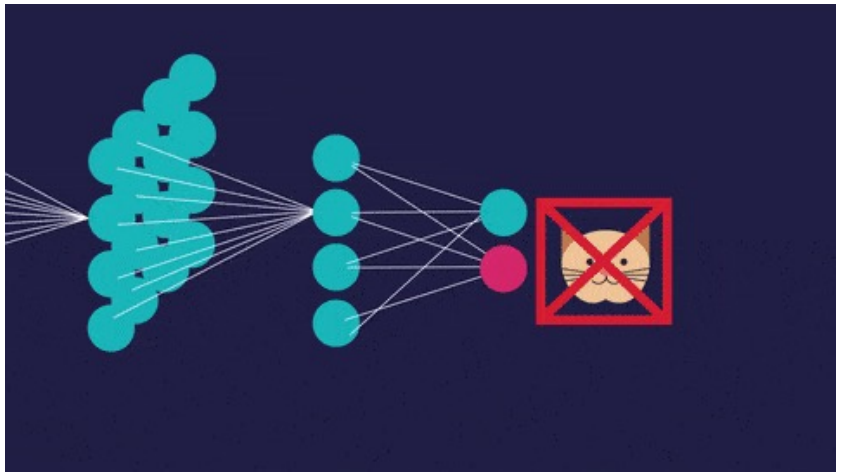
Weights H1 = (1.0, -2.0, 2.0)
Weights H2 = (2.0, 1.0, -4.0)
Weights H3 = (1.0, -1.0, 0.0)

Weights O1 = (-3.0, 1.0, -3.0)
Weights O2 = (0.0, 1.0, 2.0)

O1 = S(0.13 * -3.0 + 0.96 * 1.0 + 0.40 * -3.0) = S(-0.63) = 0.35
O1 = S(0.13 * 0.0 + 0.96 * 1.0 + 0.40 * 2.0) = S(1.76) = 0.85

Training loop



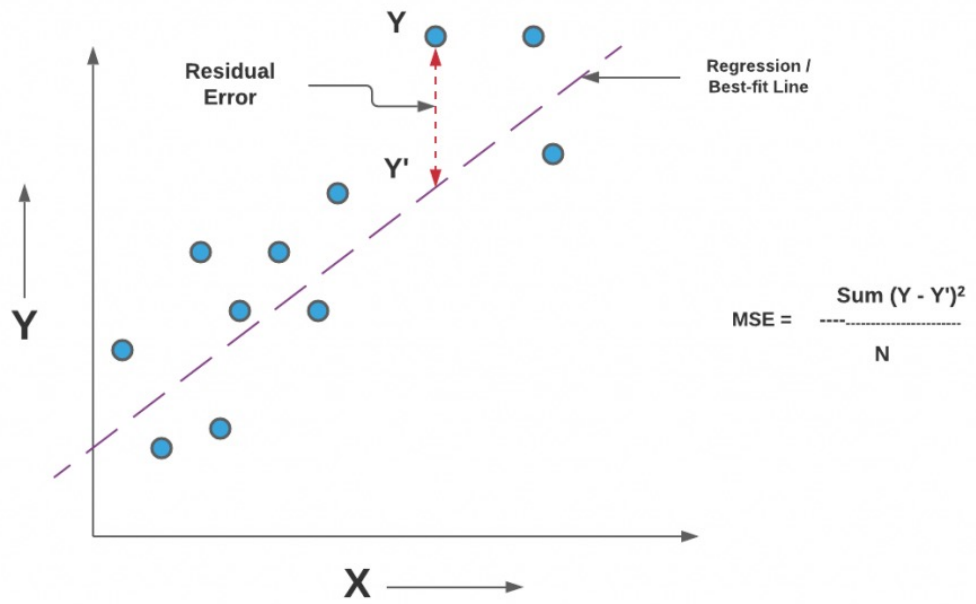


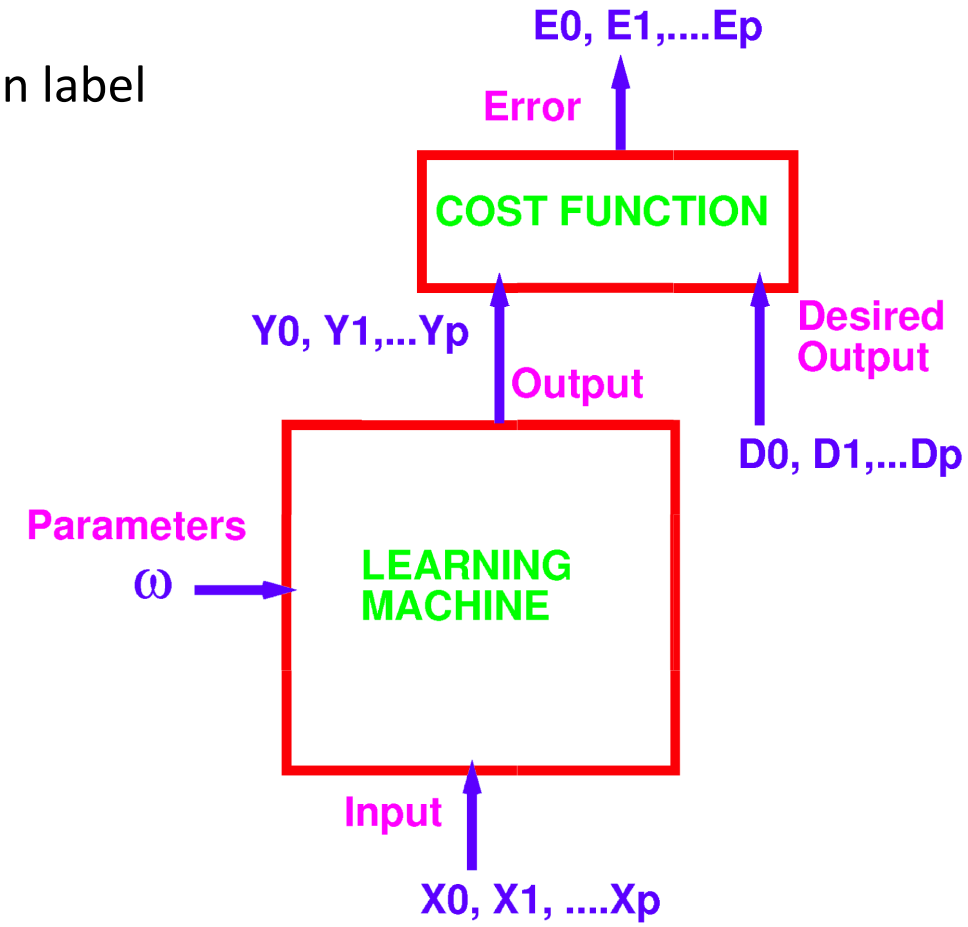
- Training a neural network is supervised learning
- We need labels from an expert to compare them with the neural network's prediction
- If a neural network makes mistakes, it needs to be train
- When training, we change the weights of the neural network

Loss function

- To compare prediction of our model and right answer in label we need to calculate loss function

- For classification we use Cross Entropy loss function
- For regression we use Mean Square Error

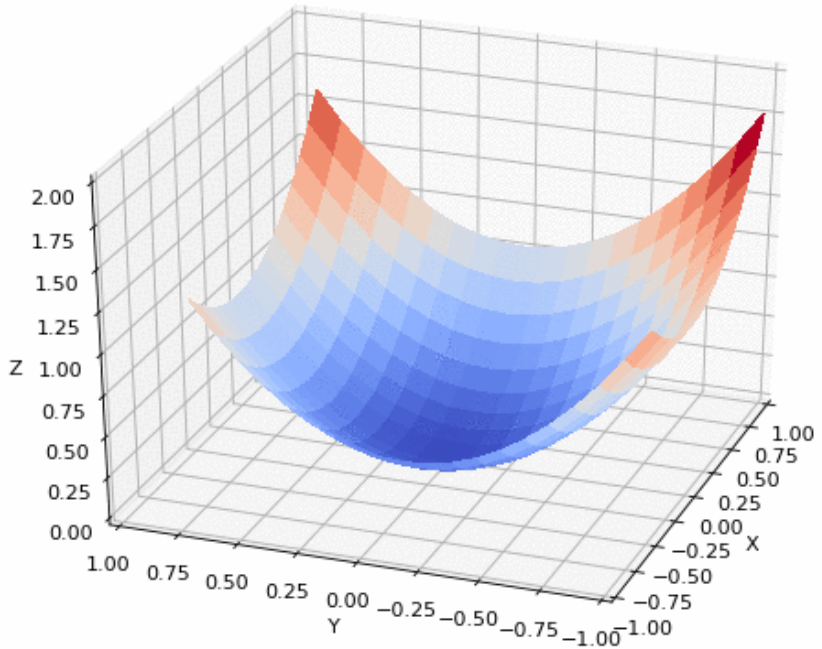




Average Error: $E = \frac{1}{p} \Sigma E\kappa$

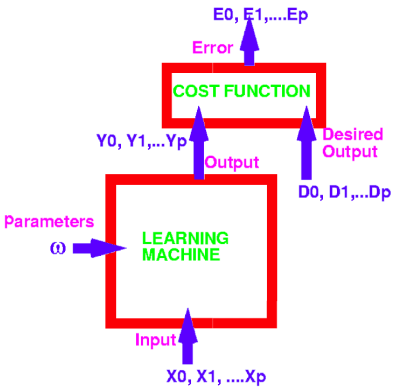
Gradient Descend

- We need minimum of loss function – minimum of error
- But we don’t know this point – function is too complex
- Using Gradient Descend we change weights of neural network to get model with minimal error
- It's like going down a mountain in the fog. You only see the area around you



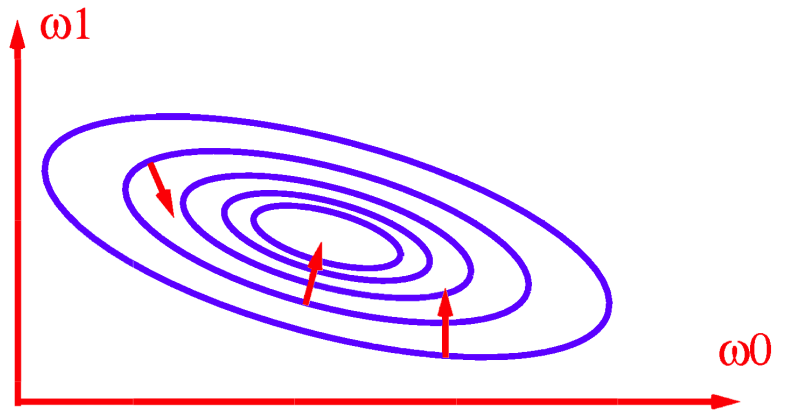
Average Error:

$$E(\omega) = \frac{1}{p} \sum E\kappa(\omega)$$



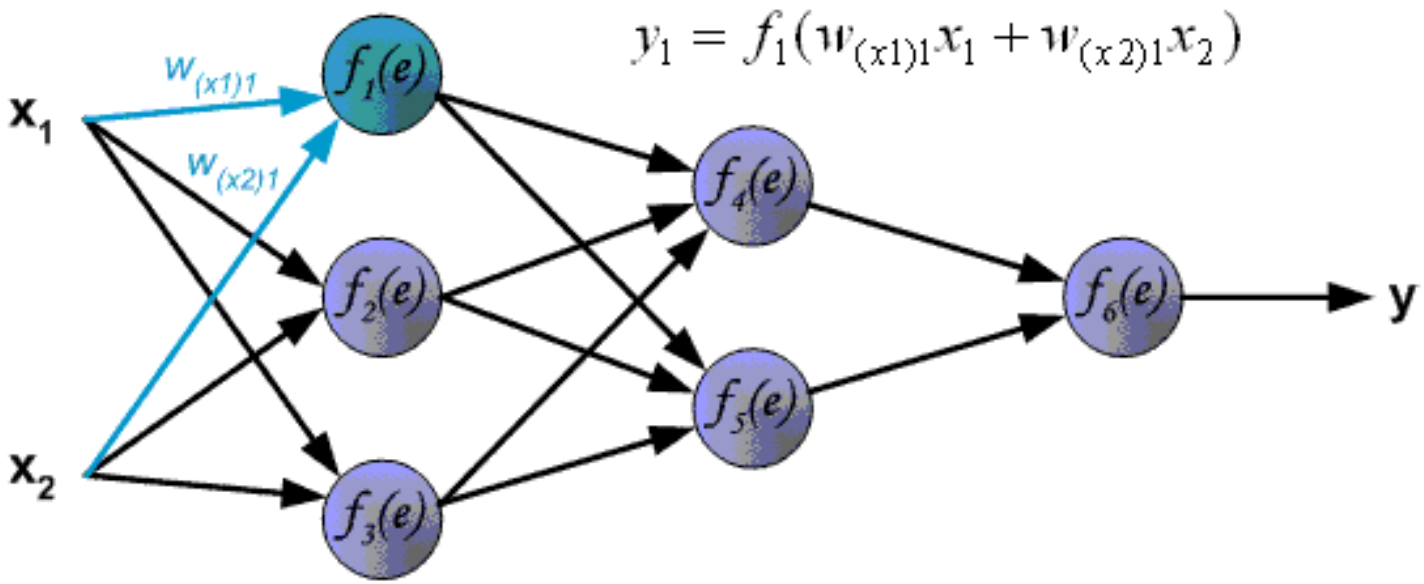
Gradient Descent:

$$\omega(\tau+1) = \omega(\tau) - \eta \frac{\partial E}{\partial \omega}$$



Error Backpropagation

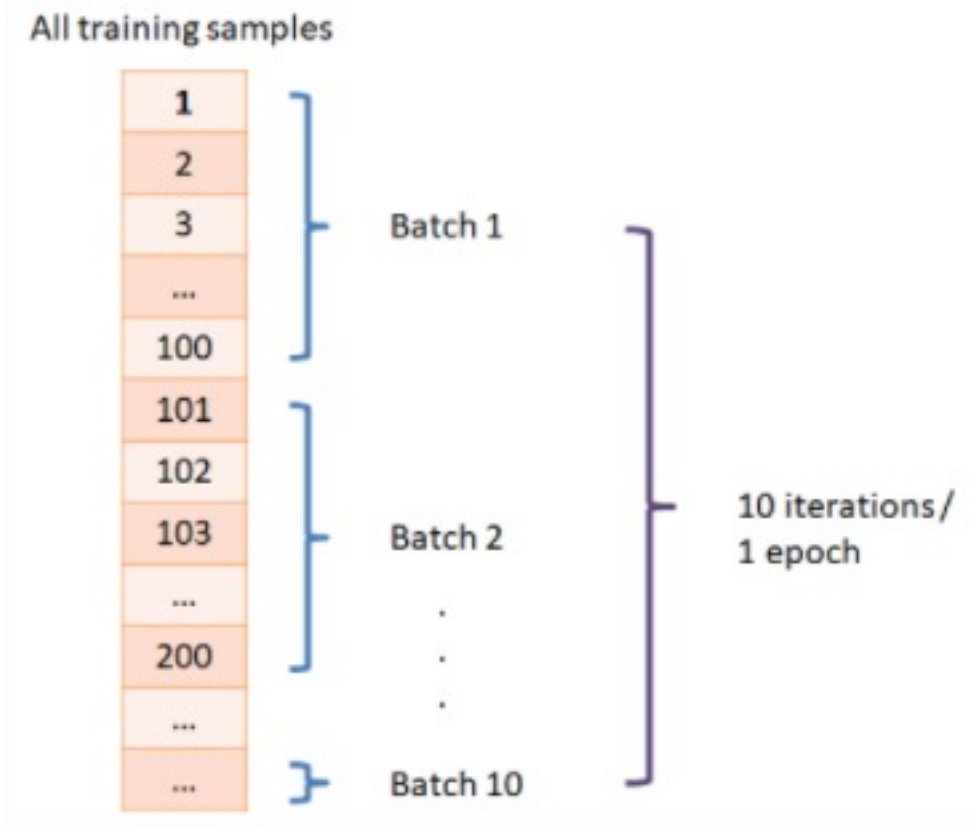
FP



- Question: Which weights should be updated and by how much?
- We need to calculate the partial derivative for each parameter - weight and bias
- To calculate the derivative of such a complex function, we find it step by step, layer by layer

Iterations and epochs

- One iteration – one training step, we use samples from one mini-batch
- One epoch – iterations for whole training set



Batch

$$\omega(\rho+1) = \omega(\rho) - \eta \frac{\partial E}{\partial \omega}$$

```
Repeat {
  for all examples in training set {
     forward prop          // compute output
     backward prop         // compute gradients
     accumulate gradient }
  update parameters }
```

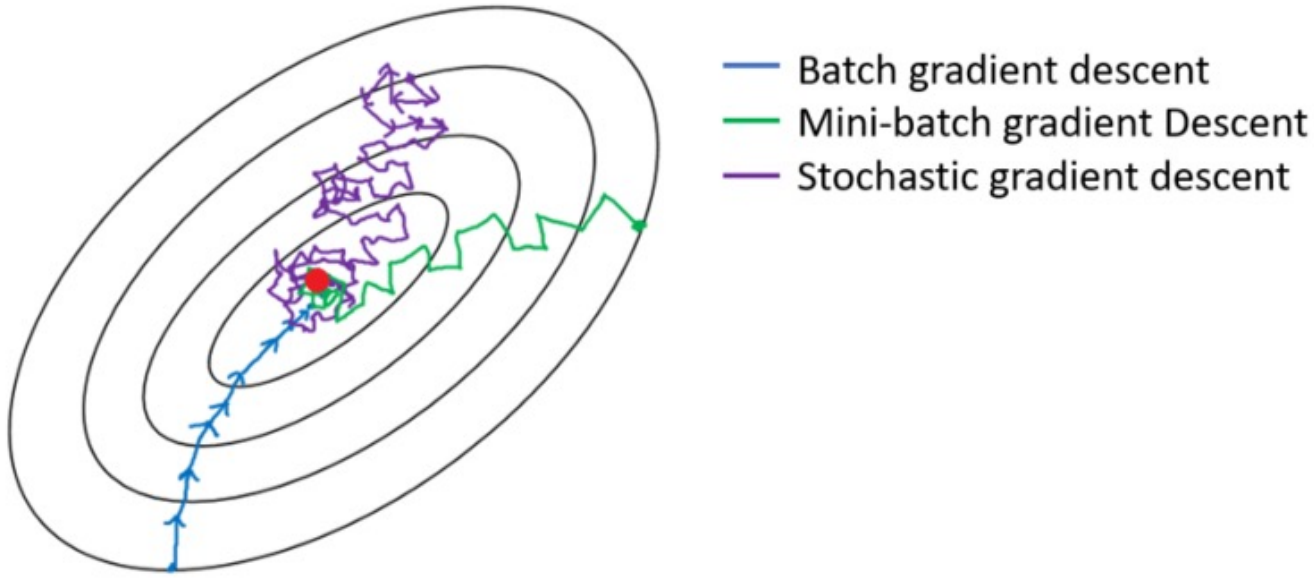
Stochastic

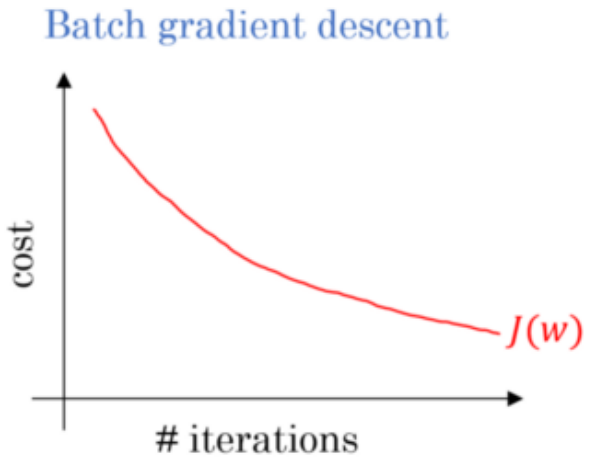
$$\omega(\tau+1) = \omega(\tau) - \eta \frac{\partial E\tau}{\partial \omega}$$

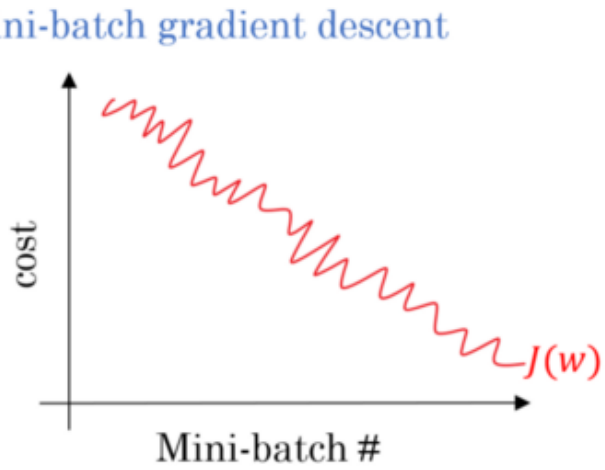
```
Repeat {
  for all examples in training set {
     forward prop          // compute output
     backward prop         // compute gradients
     update parameters }}
```

Batch vs Stochastic

- We calculate the loss function for each sample, but then we add them into one loss function to calculate the gradient and new weights
- Training on the entire dataset takes a long time. Therefore, each time we take a small portion of data - a minibatch
- Each step is calculated faster, but the learning becomes more chaotic

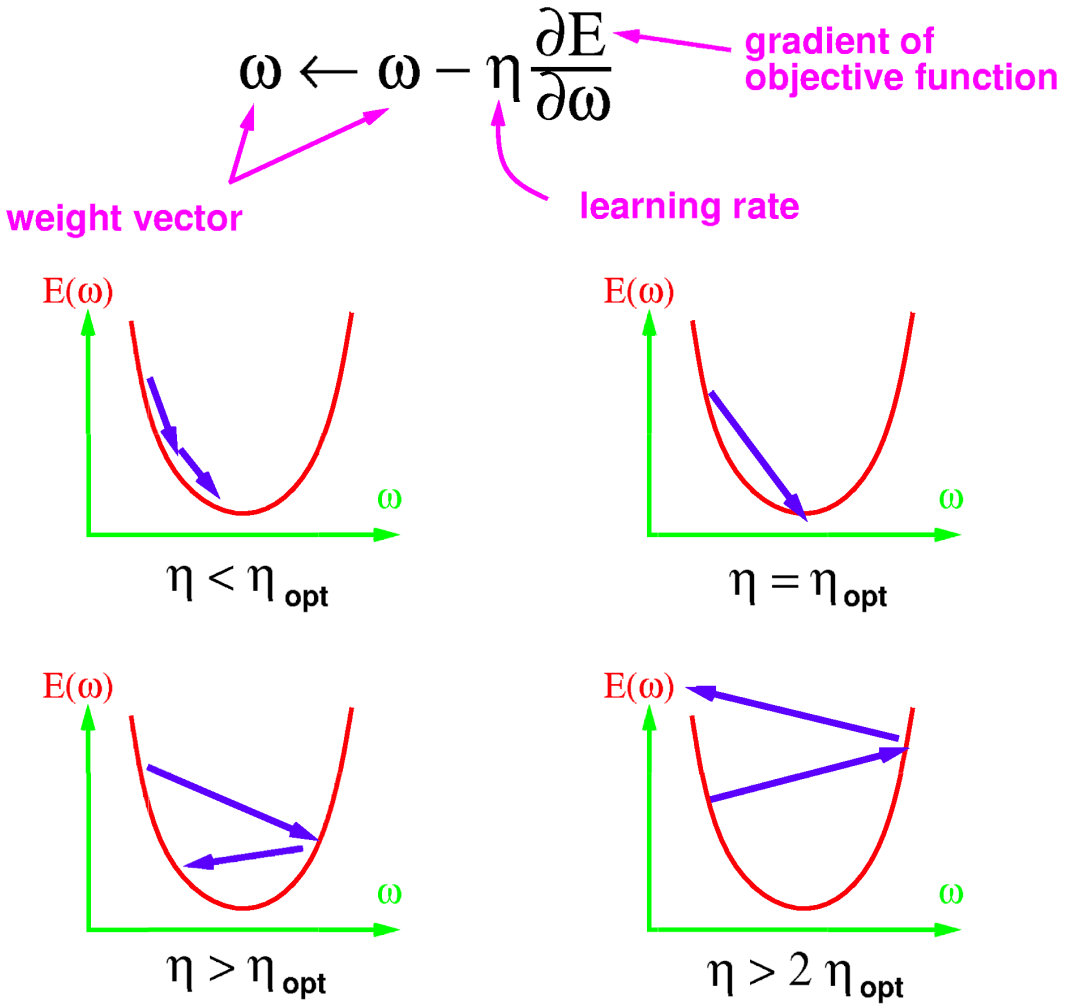




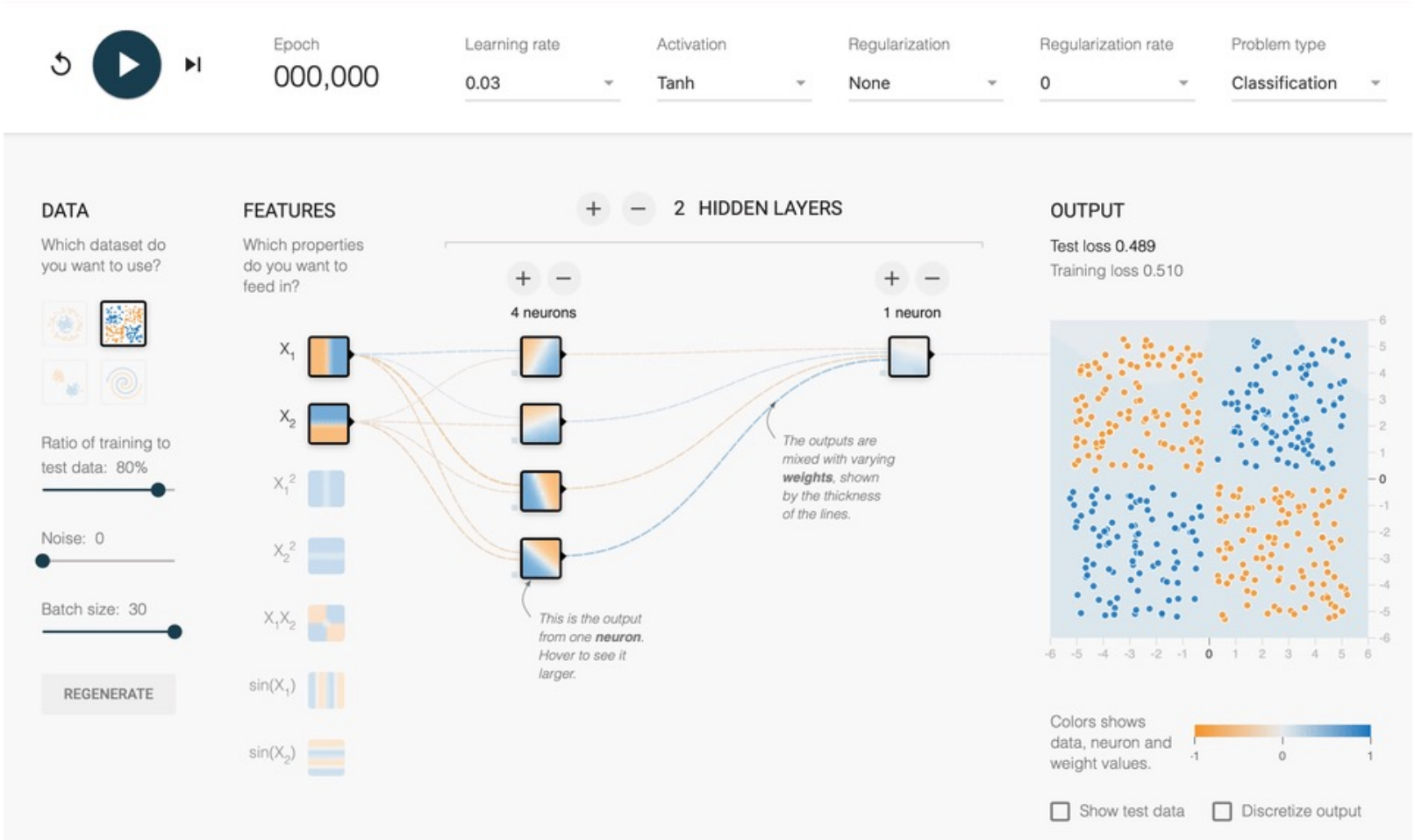


Learning rate

- Network parameters (weights) change during training
- Hyperparameters are constant, they control learning
- One of the hyperparameters is the learning rate. The learning speed is low - we teach accurately, but for a long time. If the speed is high, it's a big step right away, but we can jump over the right place
- Other hyperparameters - number of epochs, batch size, number of hidden neurons

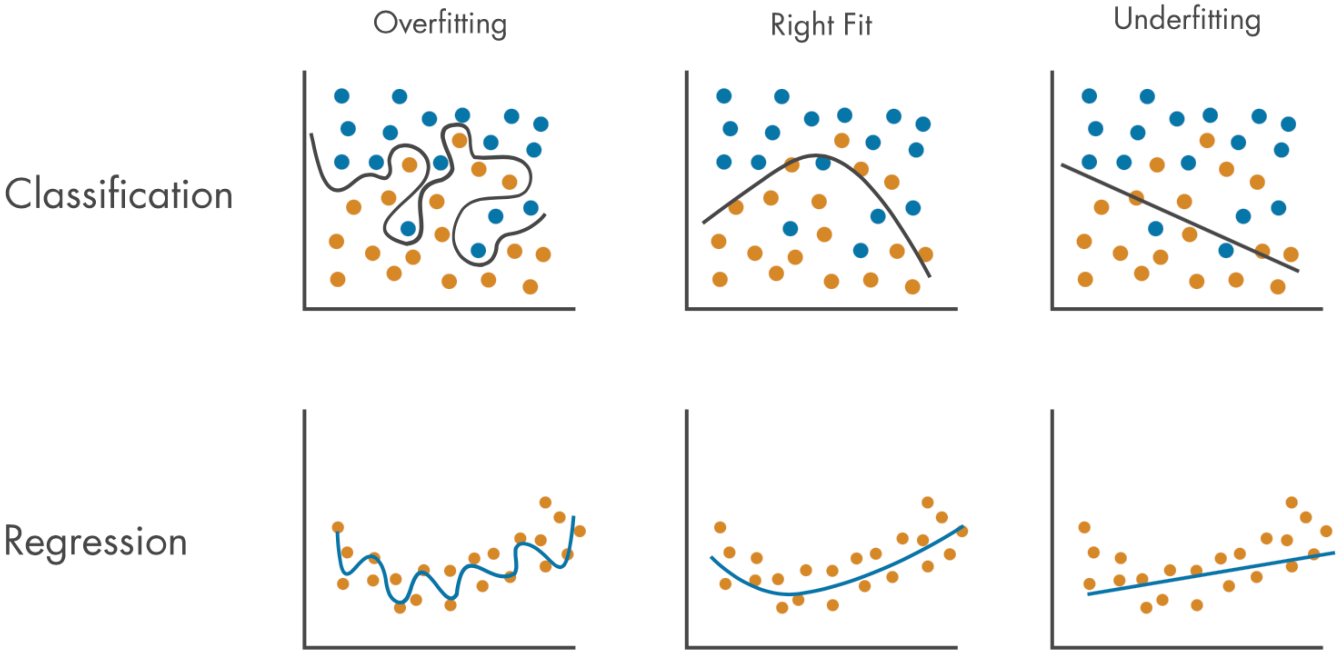


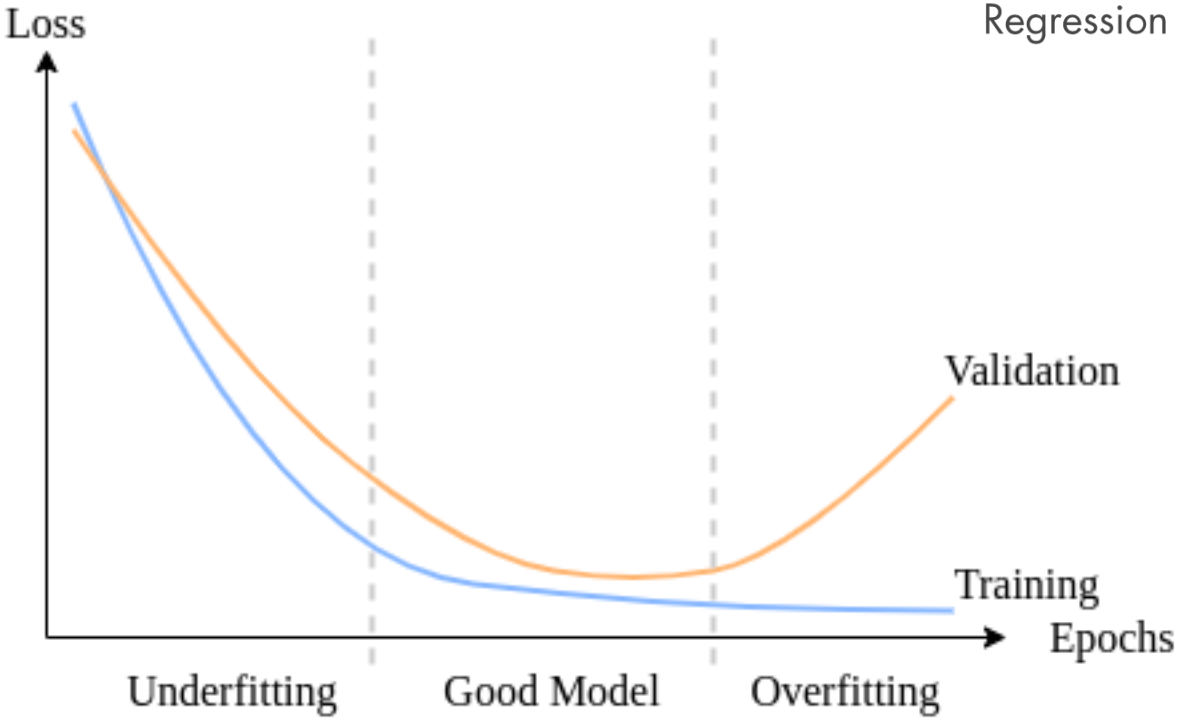
Playground



https://playground.tensorflow.org

Overfitting

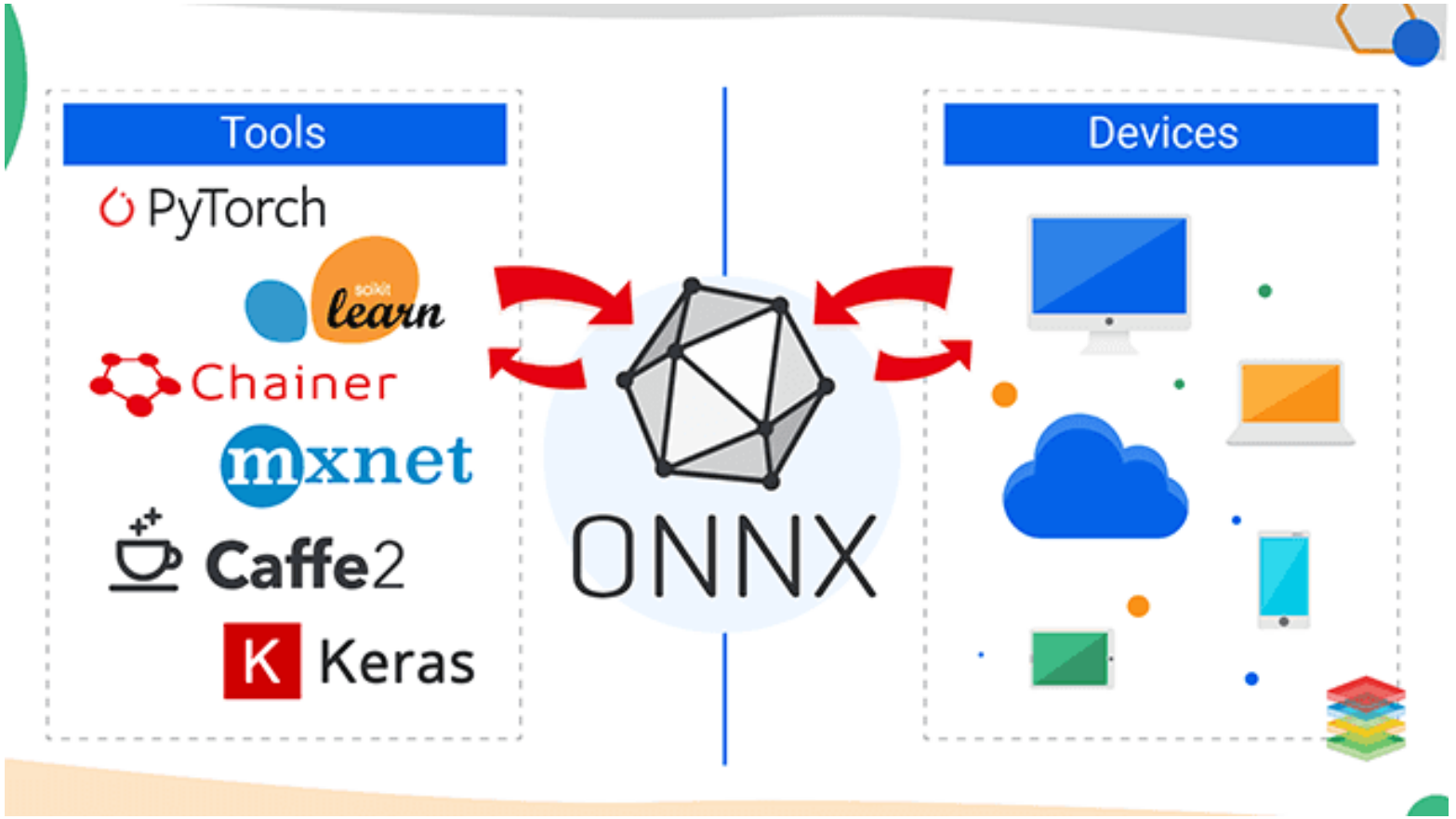




- When training a model that is too complex for a long time, overfitting occurs.
- It is very easy to find - the error on the training set continues to decrease, but on the test set it is constant or growing

ONNX

- ONNX - library for converting models between different technologies
- ONNX allows researchers and developers to choose the right combination of tools to solve a problem
- ONNX.js allows you to run models in the browser, that is, on the user side



Bauman Moscow State Technical University

Thank you!

Tel./WhatsApp: +7(499)263-64-62 Email: 1830@bmstu.ru