데이터 마이닝 특강
Practice session

[Classification]

Junseok Park

2018-08-23

Clustering review

- **Practice of clustering algorithms**
  - K-mean
  - Hierarchical clustering (H-Clustering)r
  - Self organizing map (SOM)

- **Realword data practice**
  - Data preprocessing and transformation
  - K-mean
  - H-Clustering

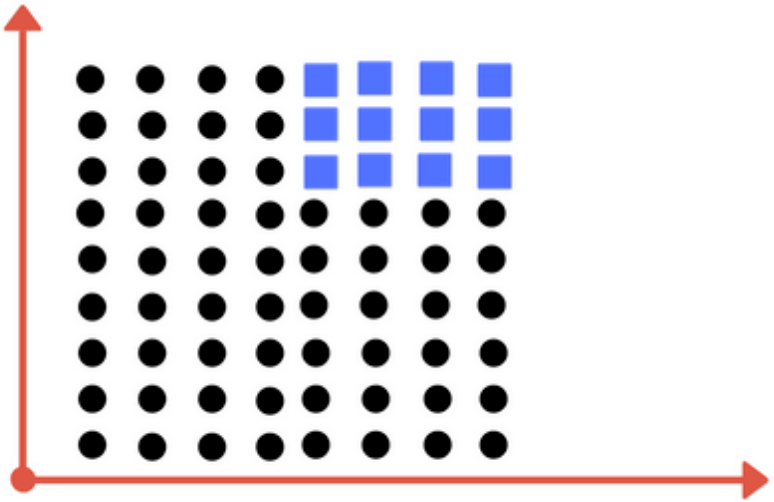
Overview

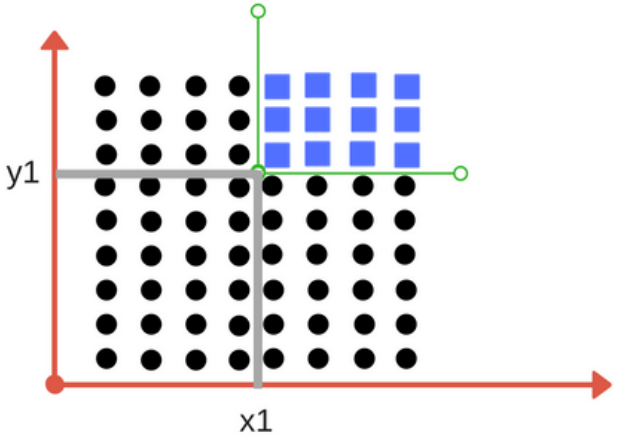
- **Decision Tree**
  - Introduction
  - Practice
- **Multi Layer Perceptron (MLP)**
  - Introduction
  - Practice with toy data
  - Practice with MNIST data
  - Other playground on google

Decision Tree

Decision Tree

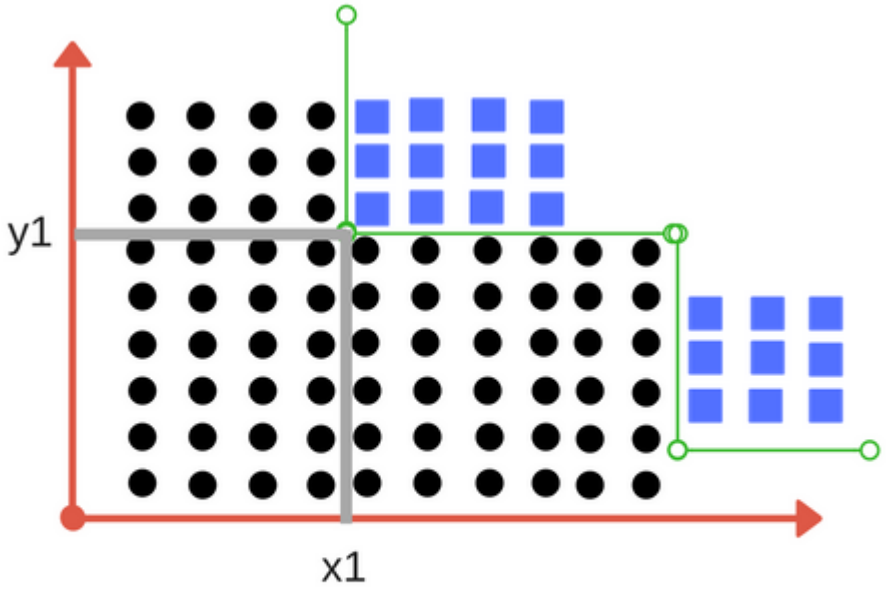
- **Motivation**
  - Is it possible to draw a single separation line?





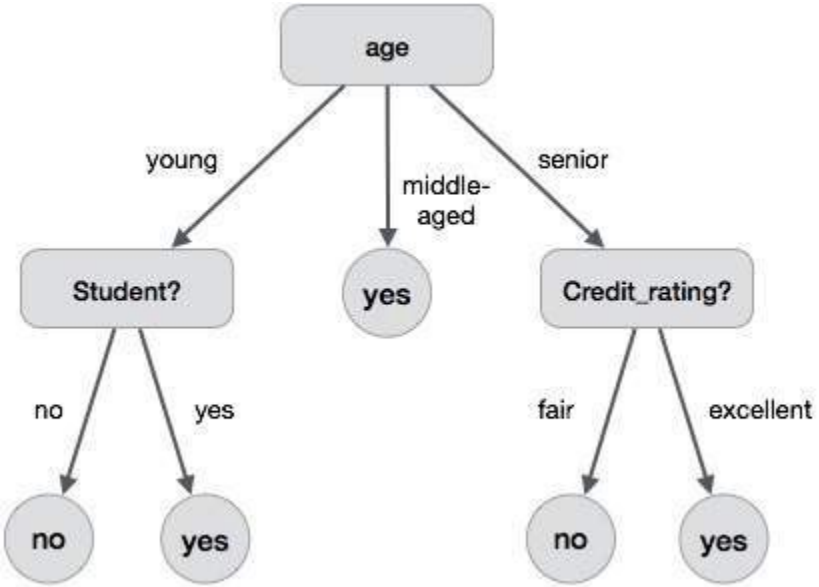
Decision Tree

- **Definition**
  - Decision Tree classifier, repetitively divides the working area(plot) into sub part by identifying lines (repetitively because there may be two distant regions of same class divided by other as show in image below)



Decision Tree

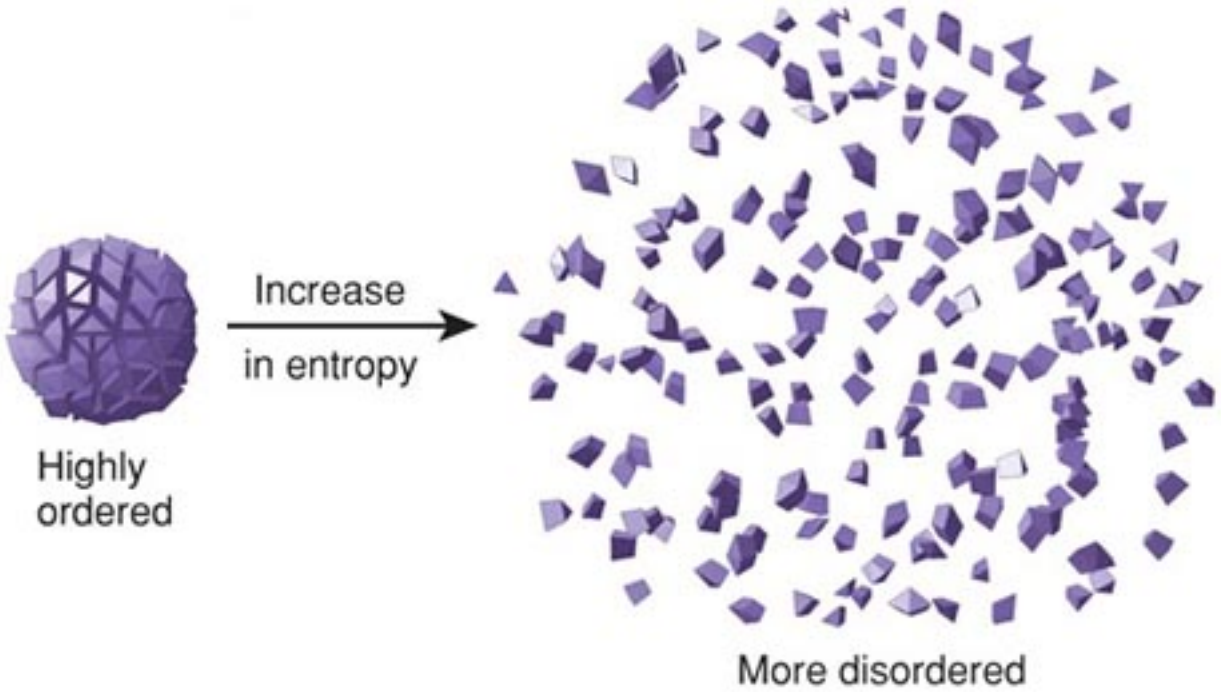
- **Impurity**
  - Impurity is when we have a traces of one class division into other. This can arise due to following reason
    - We run out of available features to divide the class upon.
    - We tolerate some percentage of impurity (we stop further division) for faster performance. (There is always trade off between accuracy and performance).



"Gini Impurity"

Decision Tree

- **Entropy**
  - Entropy is degree of randomness of elements or in other words it is ***measure of impurity.*** *Mathematically, it can be calculated with the help of probability of the items as:*



https://socratic.org/questions/what-does-entropy-measure

Decision Tree

- **Entropy**
  - It is negative summation of probability times the log of probability of item

$$H = -\sum p(x) \log p(x)$$

For example

if we have items as number of dice face occurrence in a throw event as 1123,

the entropy is

p(1) = 0.5

p(2) = 0.25

p(3) = 0.25

entropy = -(0.5*log(0.5)) – (0.25*log(0.25))-(0.25*log(0.25)

= 0.45

Decision Tree

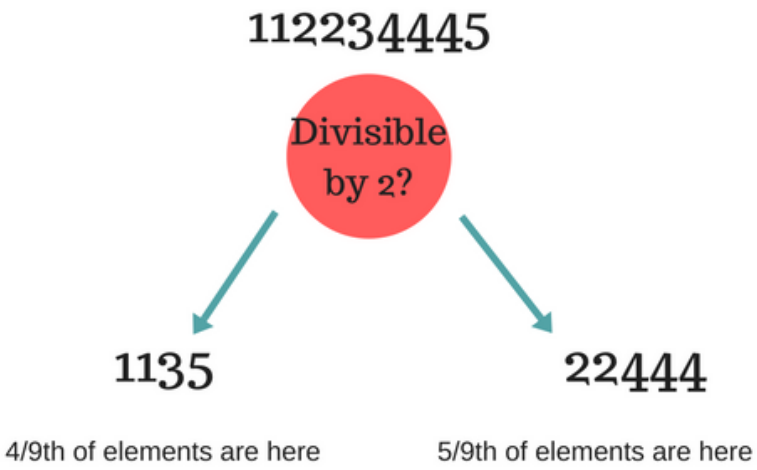
- **Information Gain**
  - a measure of changes in entropy

$$G(x, a) = H(x) - H(x|a)$$

```
Information Gain(n) =
 Entropy(x) – ([weighted average] * entropy(children for features)
```

Decision Tree

- **Information Gain**

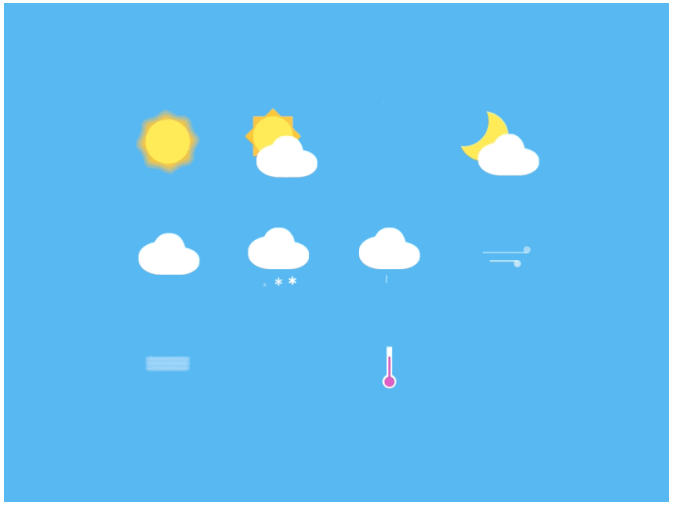



```
Entropy at root level : 0.66
Entropy of left child : 0.45 , weighted value = (4/9) * 0.45 = 0.2
Entropy of right child: 0.29 , weighted value = (5/9) * 0.29 =
0.16
Information Gain = 0.66 - [0.2 + 0.16] = 0.3
```

Dataset (Cont'd)

- **Story of the data**
  - Sara likes weekend sailing. Though, not under any condition. Past twenty Wednesdays I have asked her if she will have any company, what kind of boat she can rent, and I have checked the weather forecast. Then, on Saturday, I wrote down if she actually went to the Sea.

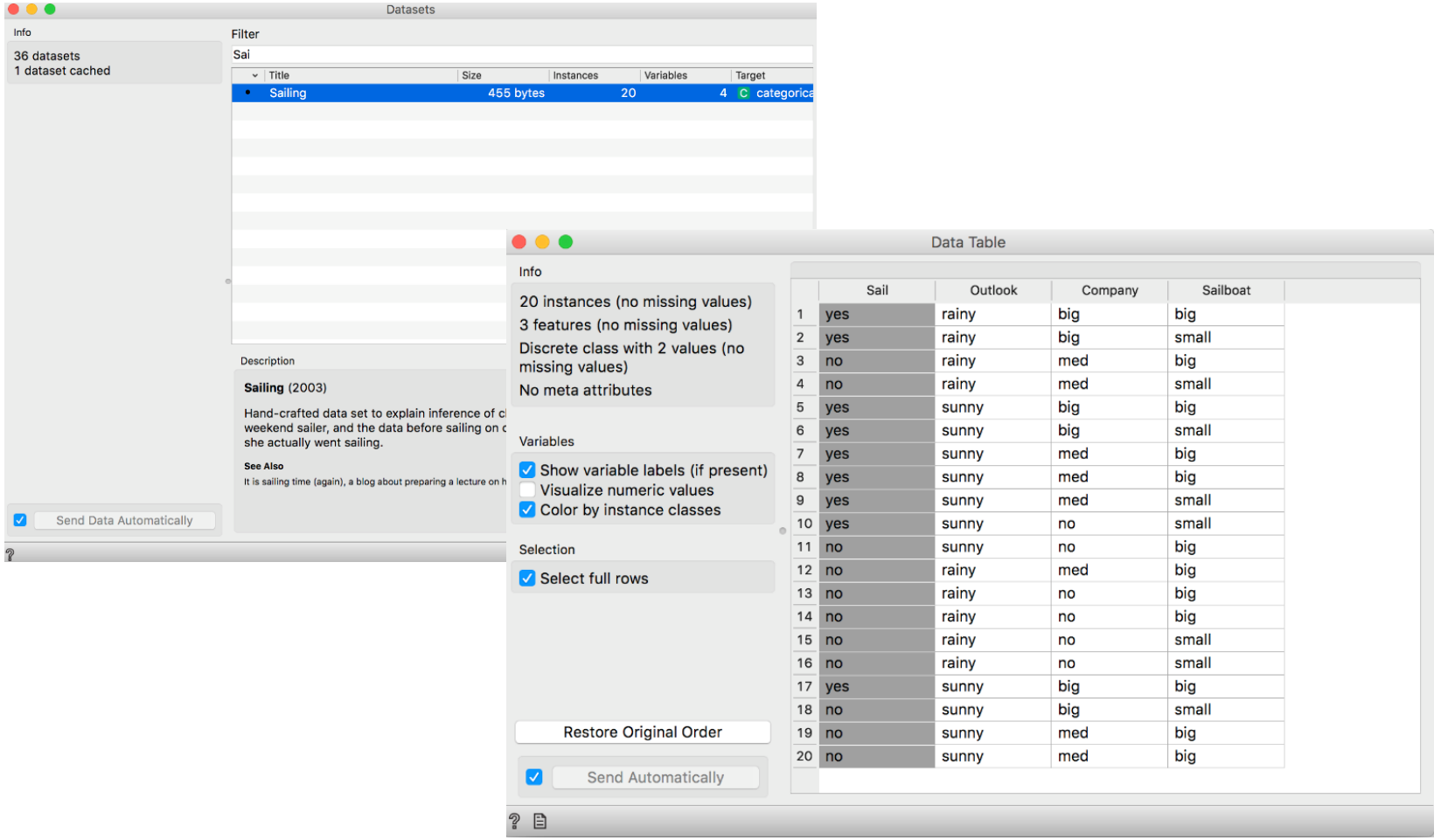
Sailing boat
Images source : https://www.bassproboatingcenters.com/brands.html



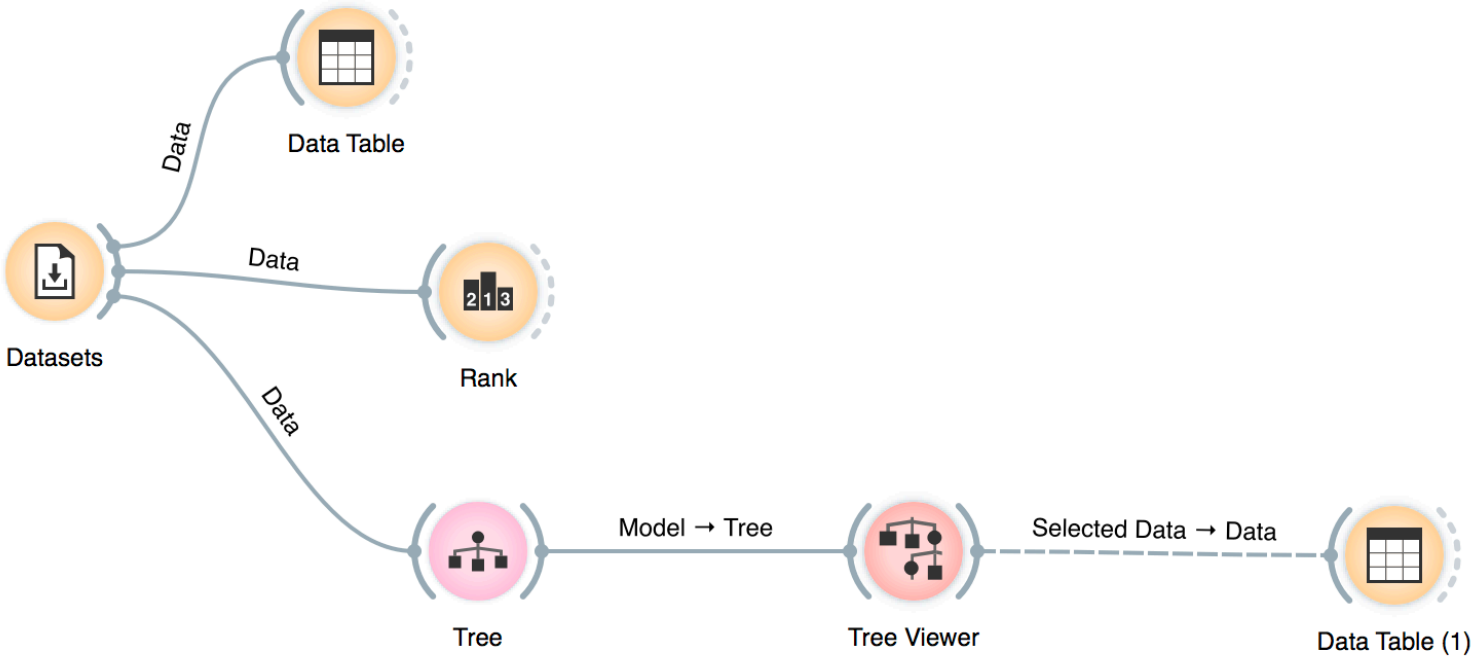
Weathers
Images source : https://www.pinterest.co.kr/pin/459367230724170802/

Dataset

- **Sailing data**
  - data that contains three attributes

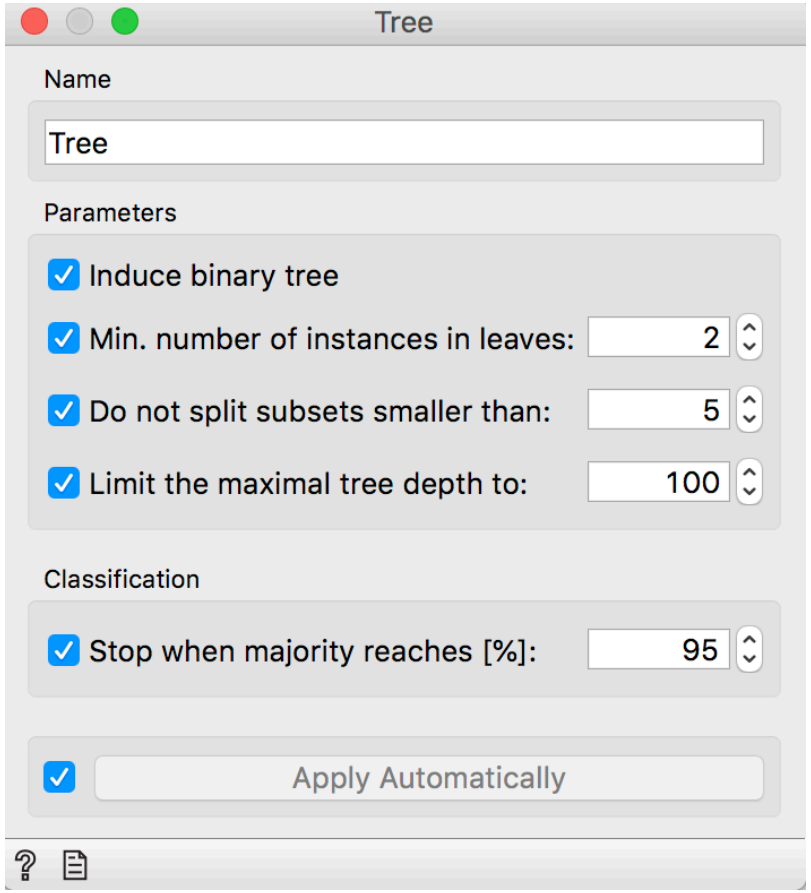


Build Schema



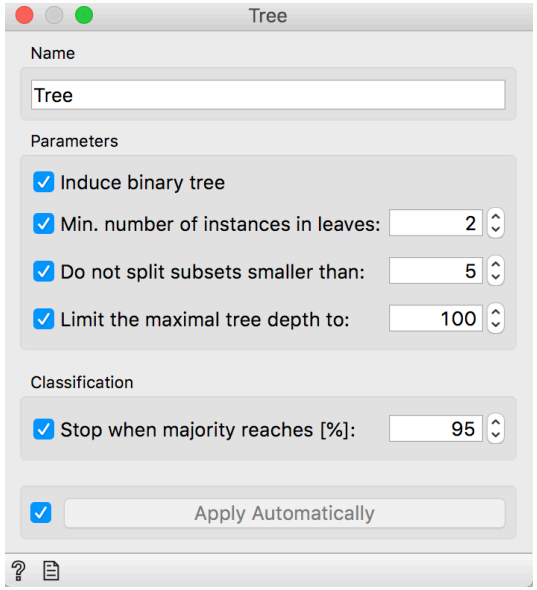
Hyperparameters and configuration

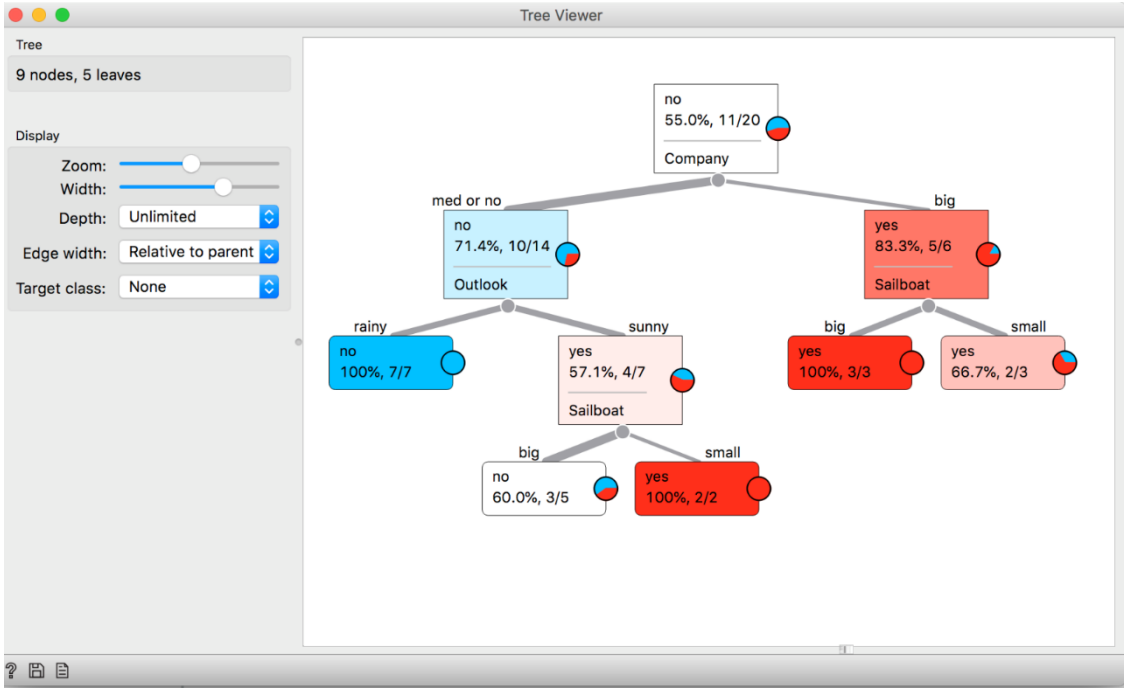
- **Parameters**
  - Induce binary tree
    - build a binary tree (split into two child nodes)
  - Min. number of instances in leaves
    - if checked, the algorithm will never construct a split which would put less than the specified number of training examples into any of the branches
  - Do not split substs smaller than
    - forbids the algorithm to split the nodes with less than the given number of instances
  - Stop when majority reaches[%]
    - stop splitting the nodes after a specified majority threshold is reached



Observation of decision tree

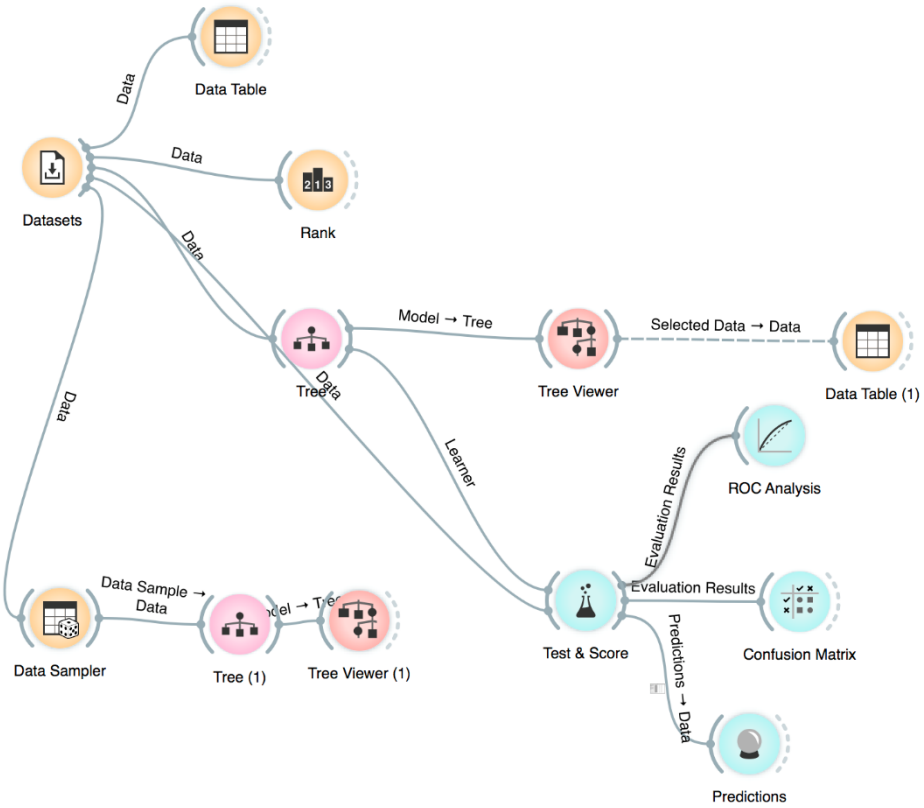
- **Self practices (5 min)**





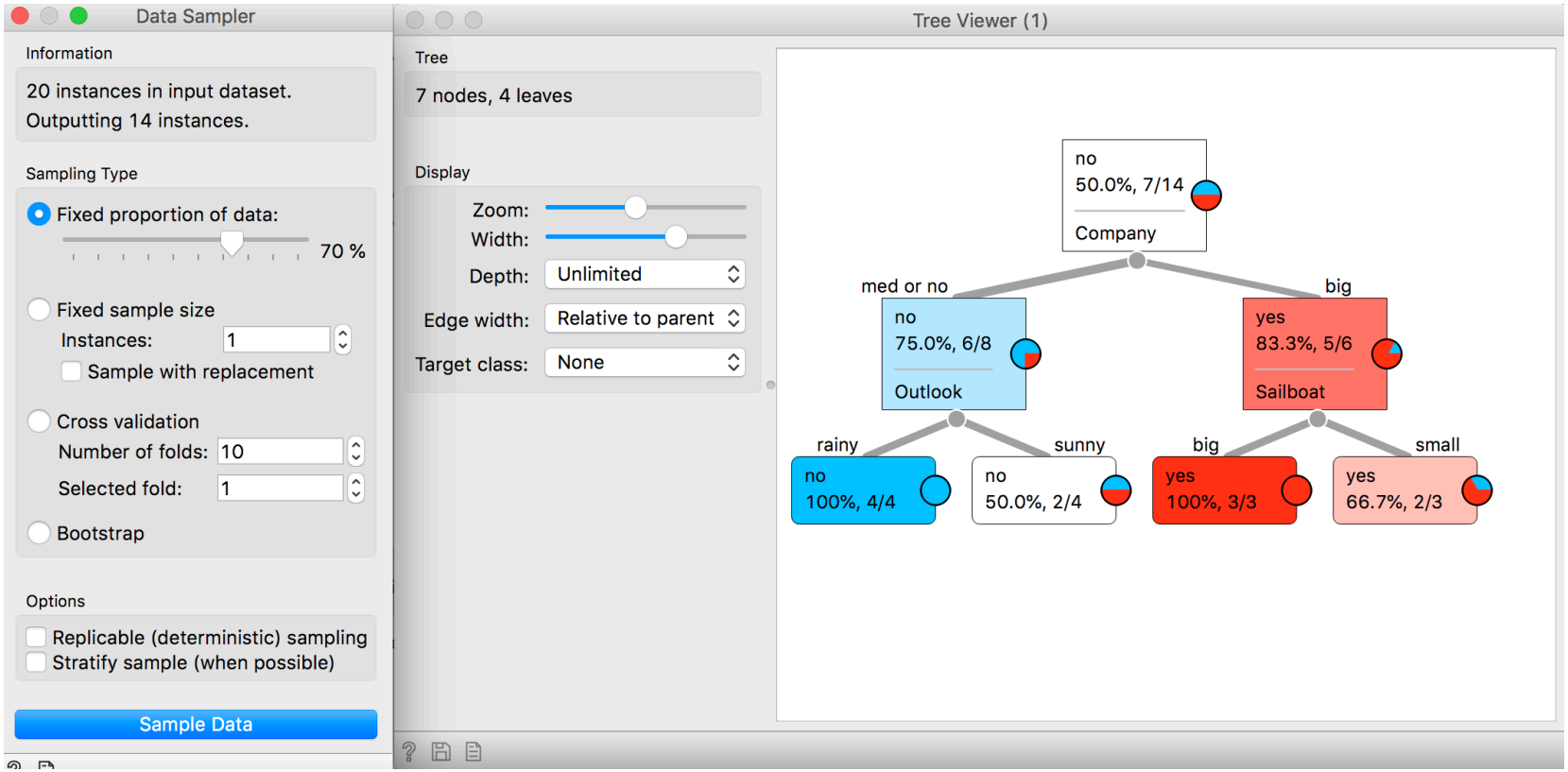
Build schema

- **Self Practices (10 min)**
  - Comparison tests
  - Demonstration for instability of decision tree



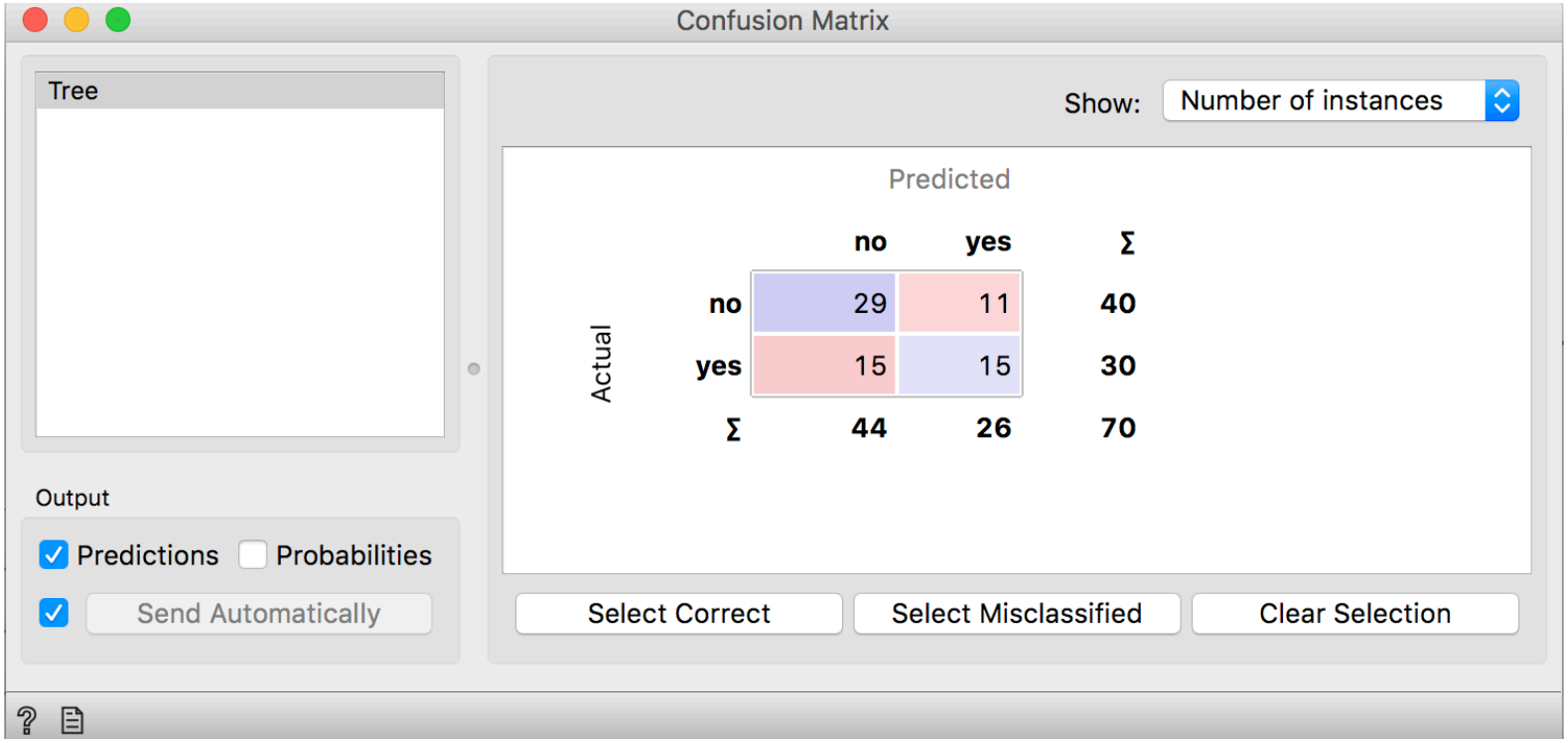
Characteristic of decision tree

- **Instability**
  - Decision tree is instable



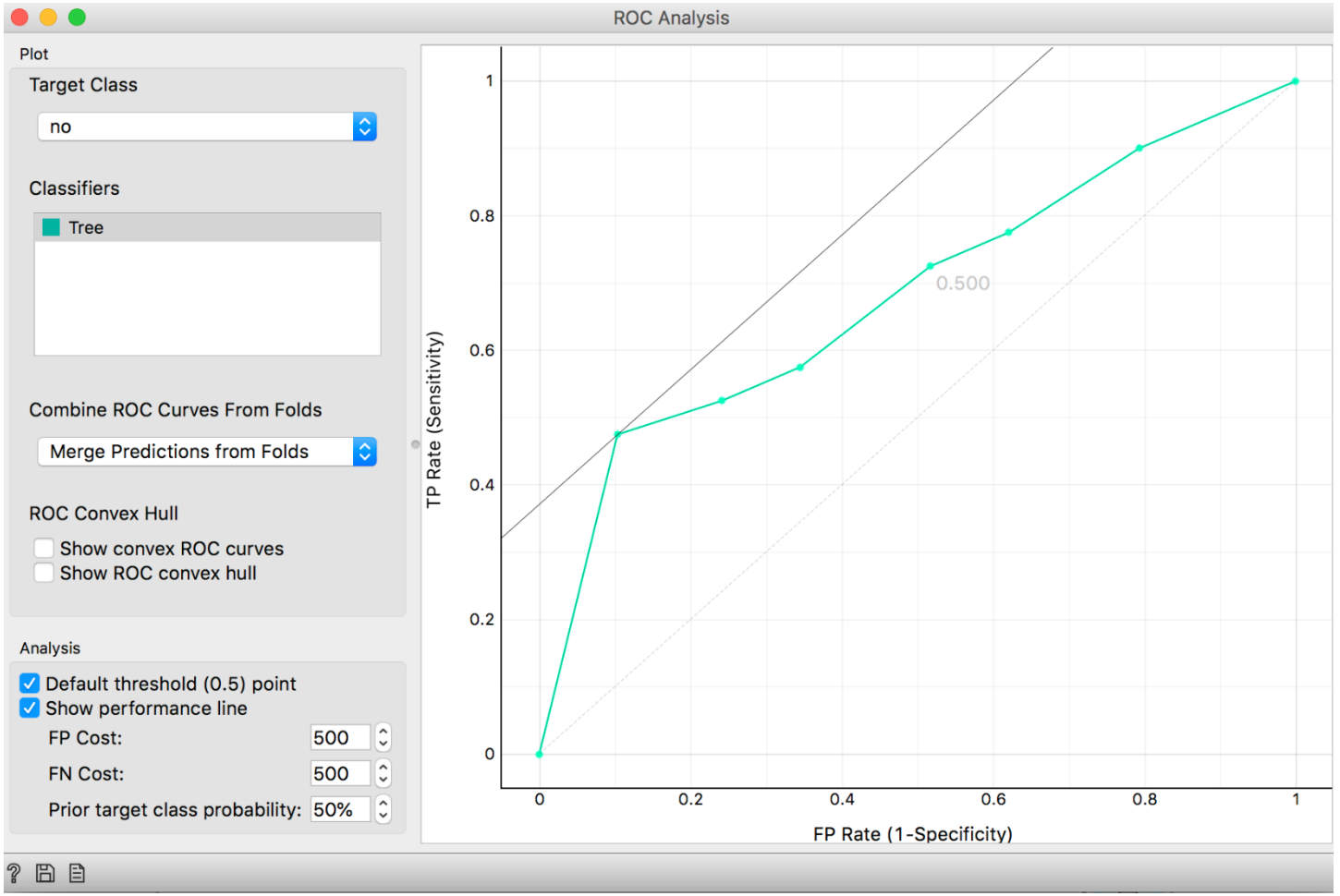
Performance test (Cont'd)

- **Confusion matrix**



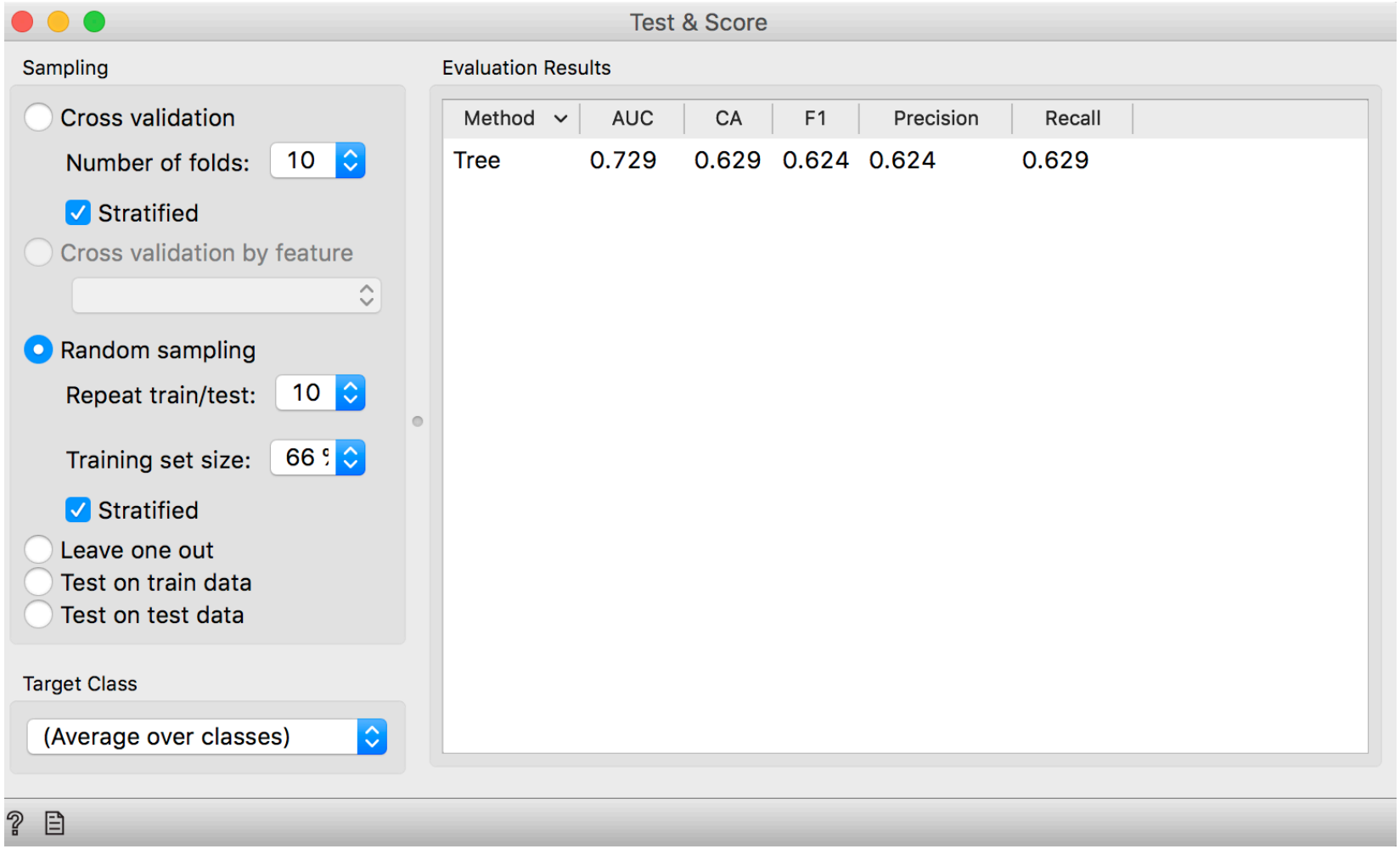
Performance test (Cont'd)

- **Area under curve(AUC) and**



Performance test (Cont'd)

- **Comparison tests**



Performance Test

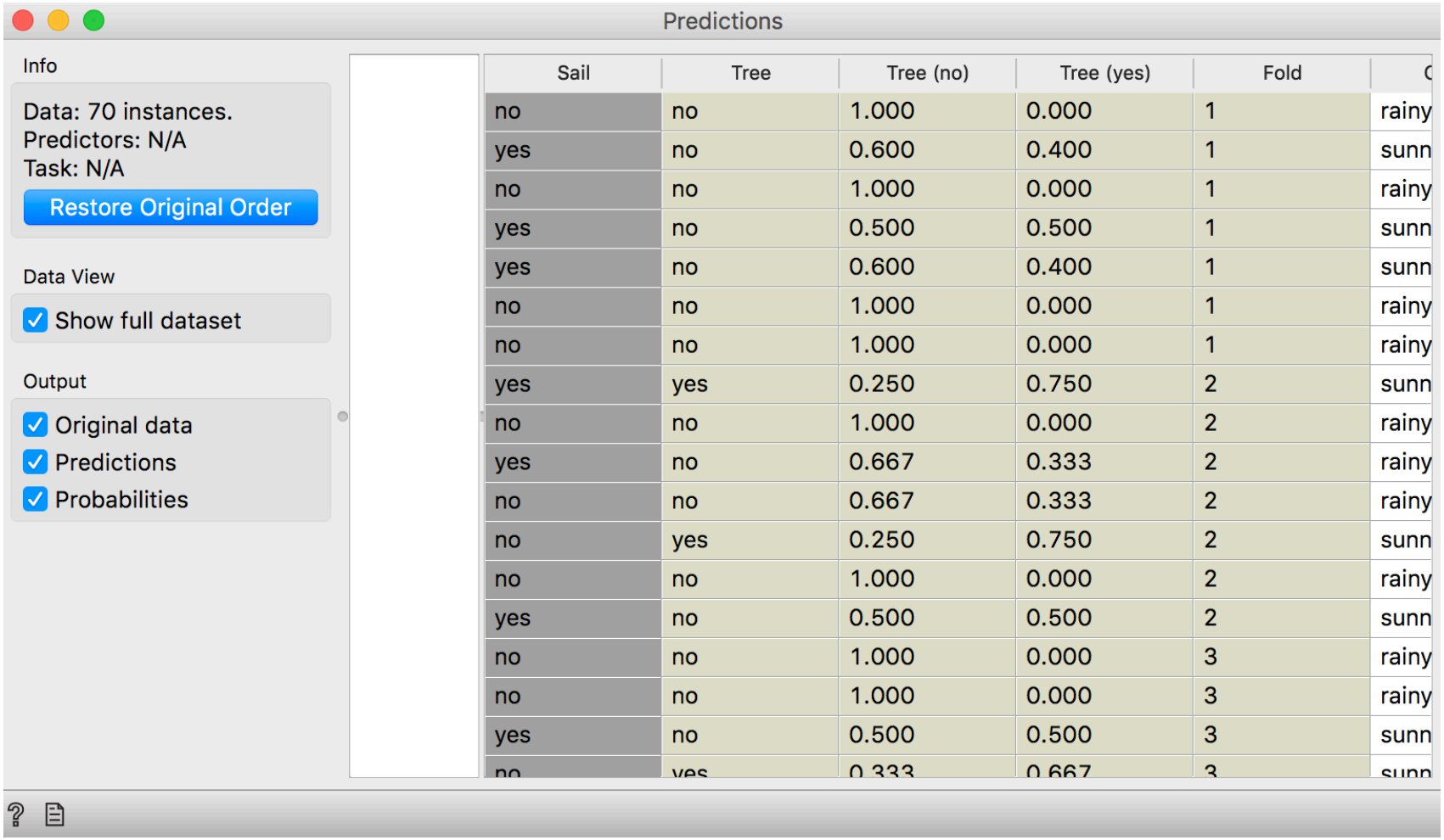
- **Classification Accuracy (CA) Score**
  - In multilabel classification, this function computes subset accuracy:

- **Precision**
  - the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

- **Recall**
  - The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

- **F1**
  - a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:
  - F1 = 2 * (precision * recall) / (precision + recall)

Results

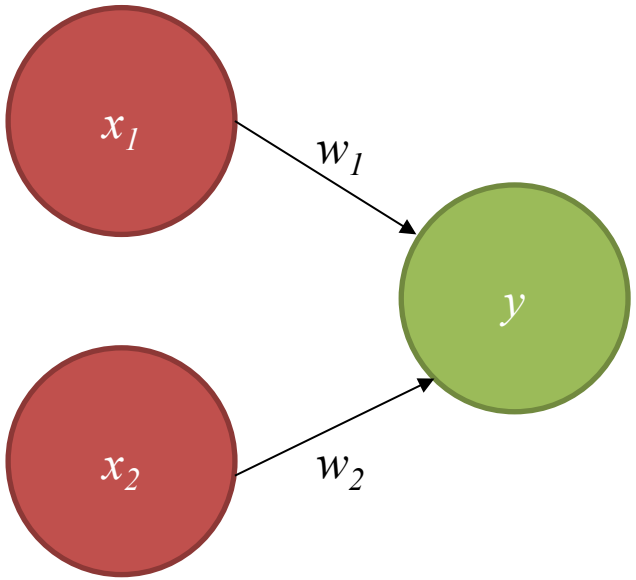


Sail	Tree	Tree (no)	Tree (yes)	Fold	C
no	no	1.000	0.000	1	rainy
yes	no	0.600	0.400	1	sunn
no	no	1.000	0.000	1	rainy
yes	no	0.500	0.500	1	sunn
yes	no	0.600	0.400	1	sunn
no	no	1.000	0.000	1	rainy
no	no	1.000	0.000	1	rainy
yes	yes	0.250	0.750	2	sunn
no	no	1.000	0.000	2	rainy
yes	no	0.667	0.333	2	rainy
no	no	0.667	0.333	2	rainy
no	yes	0.250	0.750	2	sunn
no	no	1.000	0.000	2	rainy
yes	no	0.500	0.500	2	sunn
no	no	1.000	0.000	3	rainy
no	no	1.000	0.000	3	rainy
yes	no	0.500	0.500	3	sunn
no	yes	0.333	0.667	3	sunn

Multi Layer Perceptron

Perceptron

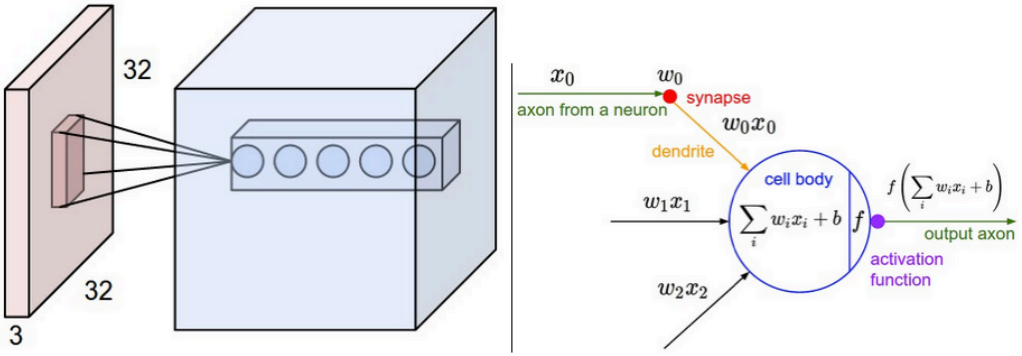
- **Perceptron**
  - Assign inherent weight per input signal
  - Circle : Neuron or Node
  - Line : weight
  - Theta : Threshold



$$y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

Neural Network Overview

- **Neural Network**
  - (before) Linear score function : $f = Wx$
  - (now) 2-layer Neural Network : $f = W_2 \max(0, W_1 x)$
    or 3-layer Neural Network : $f = W_3 \max(0, W_2 \max(0, W_1 x))$



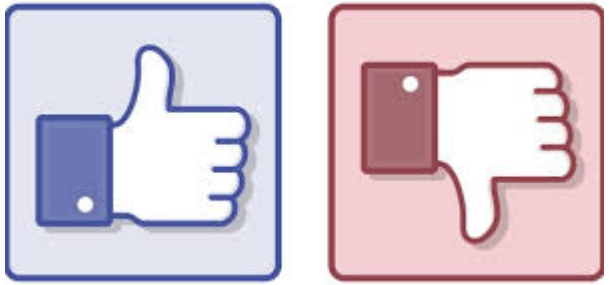
Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

- **Learn about how we can interpret these computations from the neuron/network perspective**

Modeling one neuron

- **Single neuron as a linear classifier**

neuron's capacity



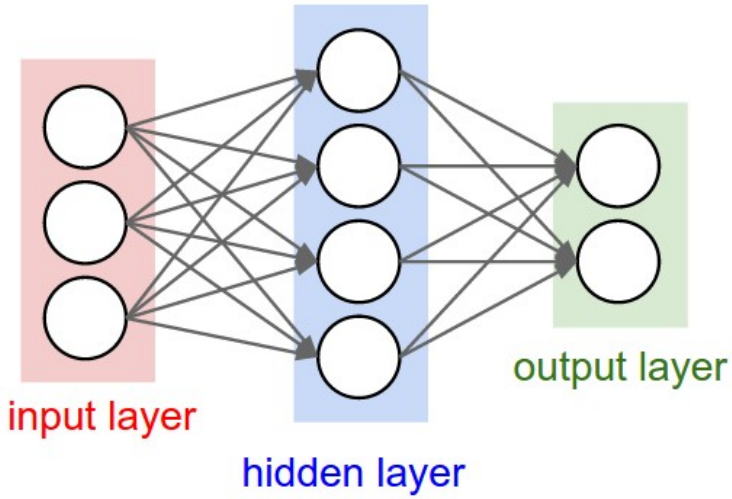
Like **Dislike**
(activation near one) (activation near zero)

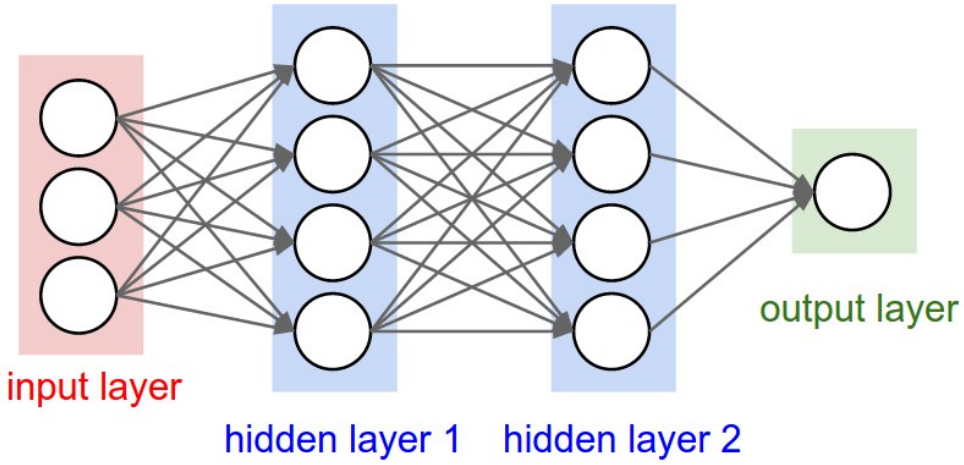
  - Single neuron can be turned into a linear classifier with an appropriate loss function

Binary Softmax Classifier *Interpret* $\sigma(\sum_i w_i x_i + b)$ *as* $P(y_i = 1 \mid x_i; w)$	**Binary SVM Classifier** *Max-margin hinge loss*
Regularization interpretation *Gradual forgetting*	

Neural Network Architecture

- **Layer-wise organization**
  - **Neural Networks as neurons in graphs**
  - The most common layer type is **fully-connected layer**

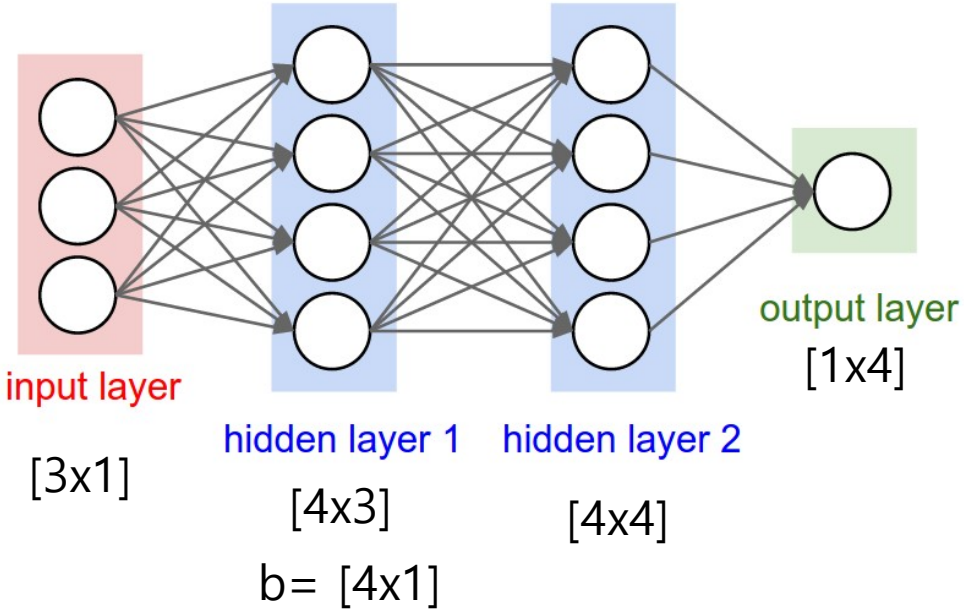




- **Naming conventions** : *Artificial Neural Networks(ANN), Multi-Layer Perceptrons(MLP)*, neurons can be referred as *unit*.
- **Output layer** : commonly no activation function. Class scores, real-valued numbers, real-valued target
- **Sizing neural networks** : 4+2 neurons, 3*4+4*2 weights, 4+ 2 biases for total 26 learnable parameters (left : A 2-layer Neural network)

Neural Network Architecture

- **Example feed-forward computation**
  - *Repeated matrix multiplications interwoven with activation function*



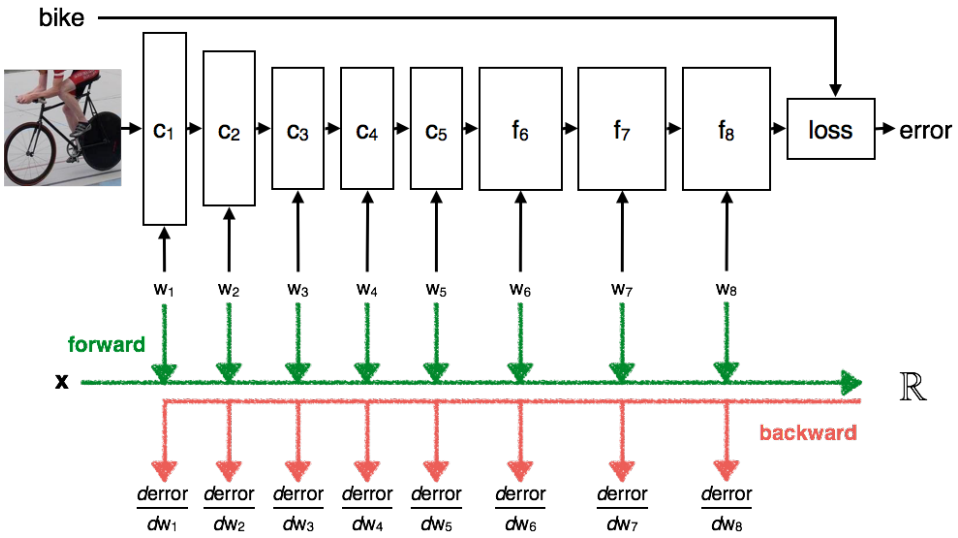
  - *The forward pass of a fully connected layer corresponds to one matrix multiplication followed by a bias offset and an activation function*

Neural Network Architecture

- **Representational Power**
  - At least one hidden layer are *universal approximators* = the neural network can approximate any continuous function

    (G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function, 1989*)

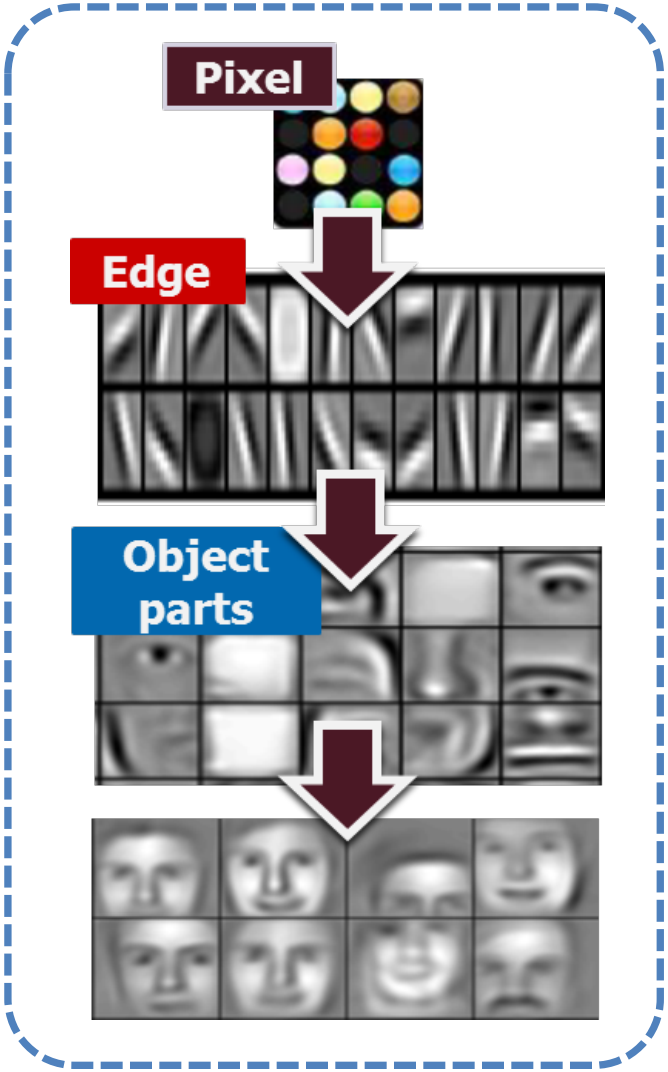
  - Why use more layers and go deeper?
    - Deep neural network works better in an empirical observation, despite the fact that their representation power is equal.
    - For example, CNN depth is most important component for a good recognition system



Introduction

- **A deep network**
  - a neural network with multiple hidden layers ($hidden\ layer \geqq 2$)

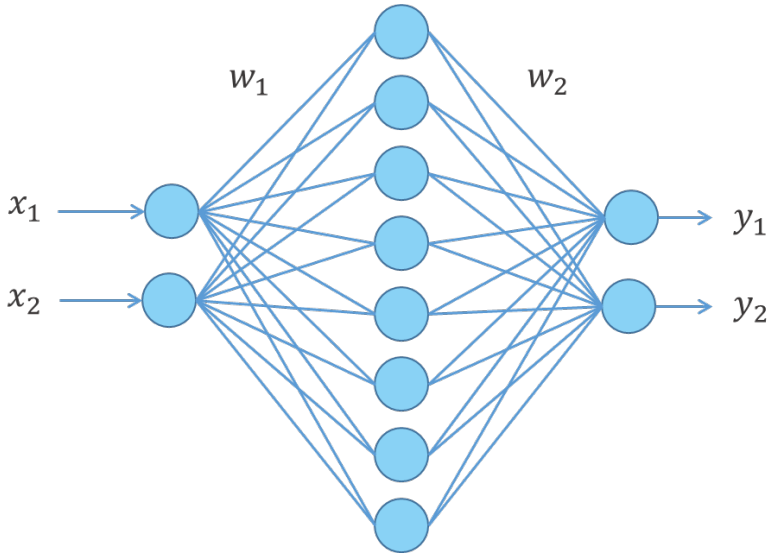
- **A deep learning**
  - a set of algorithms that model high-level abstractions in data by using deep networks.



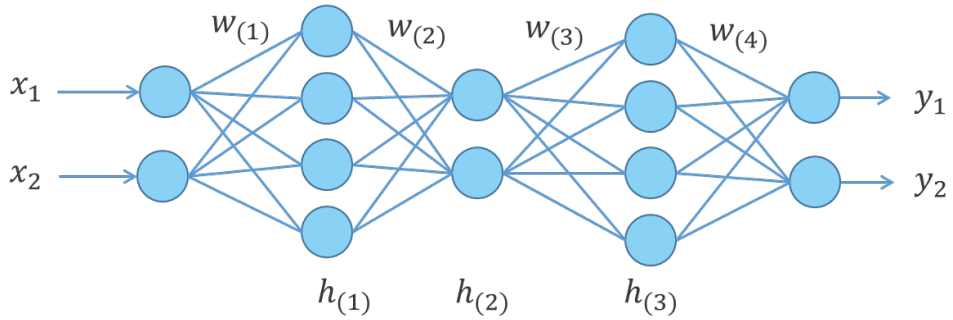
[High-level abstraction]

Characteristic of deep

- **Deep means more hidden layers**
  - The representation gets more hierarchical and abstract
  - It increases the model complexity, which leads to higher accuracy



[Shallow networks]



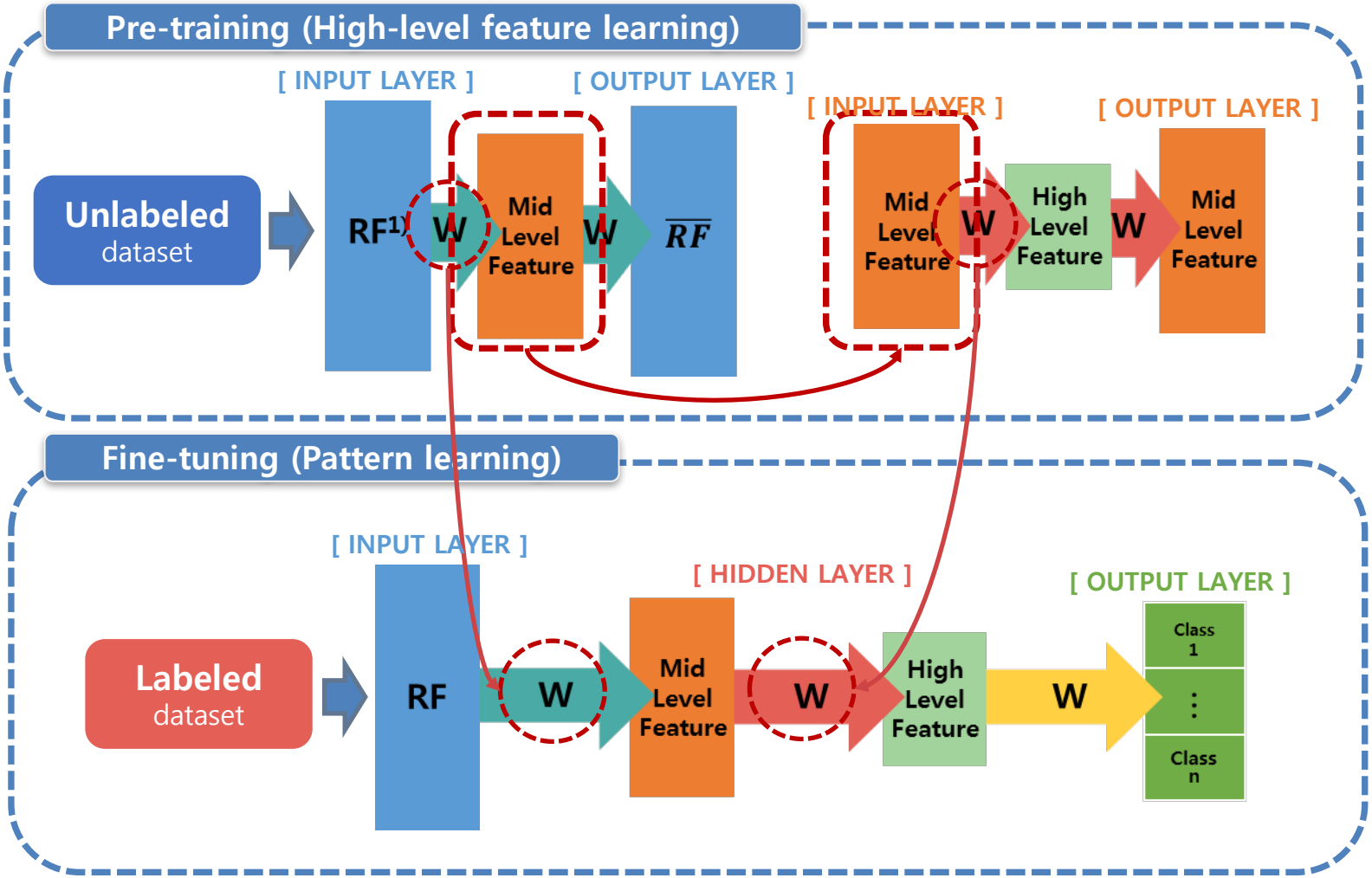
[Deep networks]

$$\#\ of\ paths\ from\ x_1\ to\ \ y_1 = 8$$

DL can learn discriminative features using hierarchical property

Deep Learning Process

- **Deep architecture-based modeling**



1) RF: Raw Features

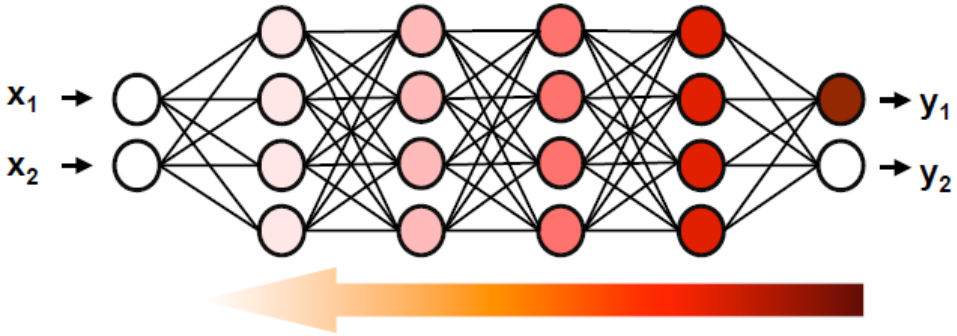
Good performance of Deep learning

- **Solve vanishing gradient problem**
  - Pre-training based on unsupervised learning
  - Rectified Linear Unit (ReLU)

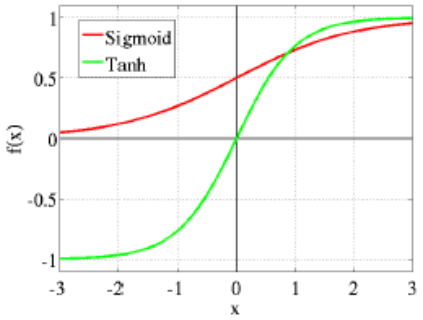
- **Solve overfitting problem**
  - Dropout

Vanishing gradient problem

- **Problems with non-linear activation**
  - Gradient is progressively getting more dilute
  - Below top few layers, correction signal is minimal



Backward error information vanishing

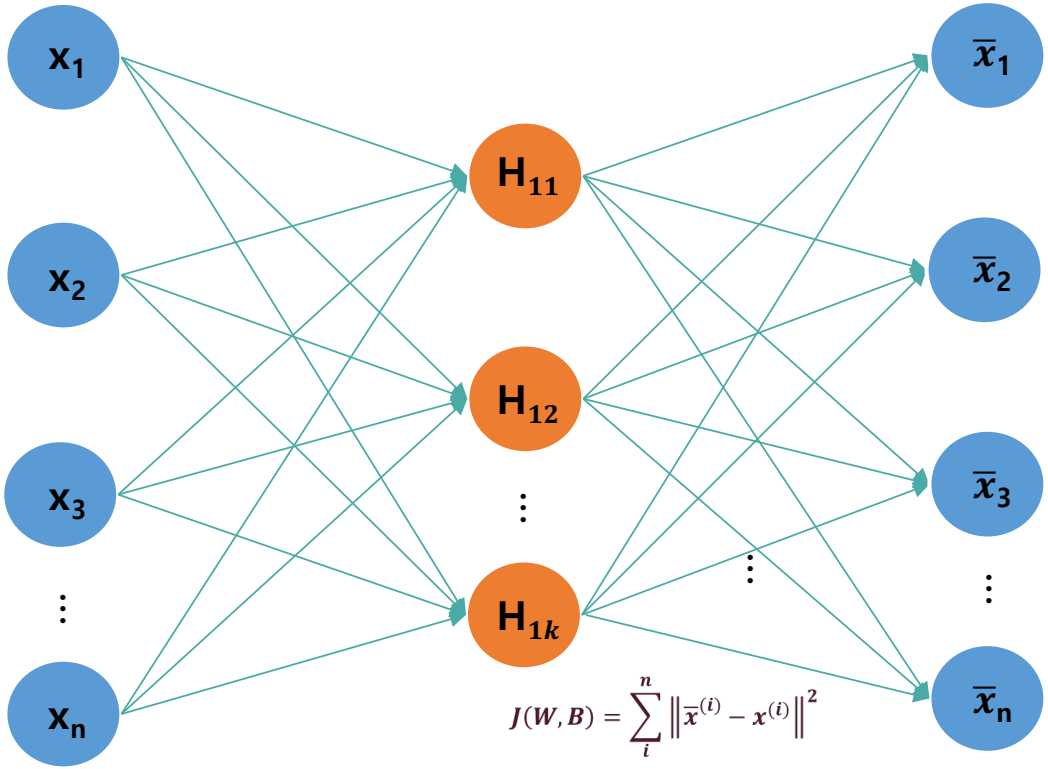


Activation function

$\frac{1}{1+e^{-x}}$ ➔ sigmoid

Pre-trainining

- **Bottom-up layerwise unsupervised learning**
  - Auto Encoder (AE)
    - update weights to minimize values between real and model

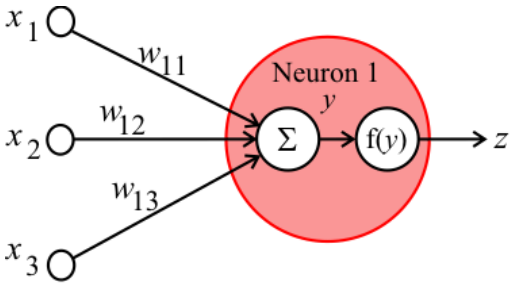


$$J(W, B) = \sum_{i}^{n} \left\| \bar{x}^{(i)} - x^{(i)} \right\|^2$$

➔ Finding weight and bias vectors that minimize reconstruction errors with back-propagation via gradient decent.

Activation function

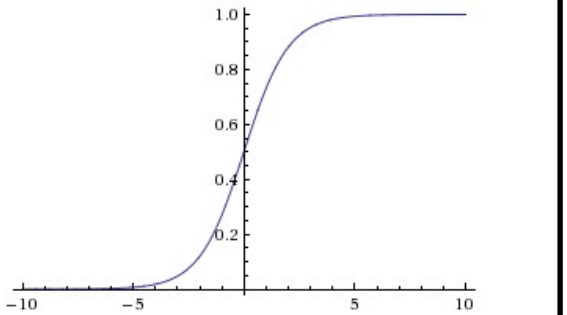
- **Activation functions**
  - Takes a single number and performs a certain fixed mathematical operation on it



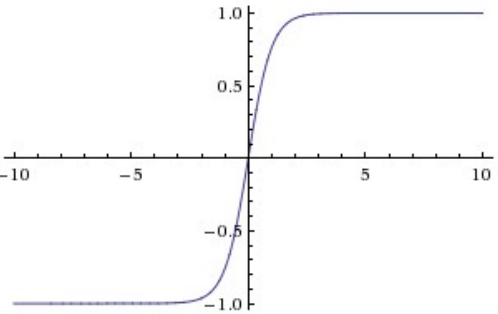
1 $z = f(y) = \text{logsig}(y) = \dfrac{1}{1 + e^{-y}}$

2 $z = f(y) = \tanh(ay)$

1

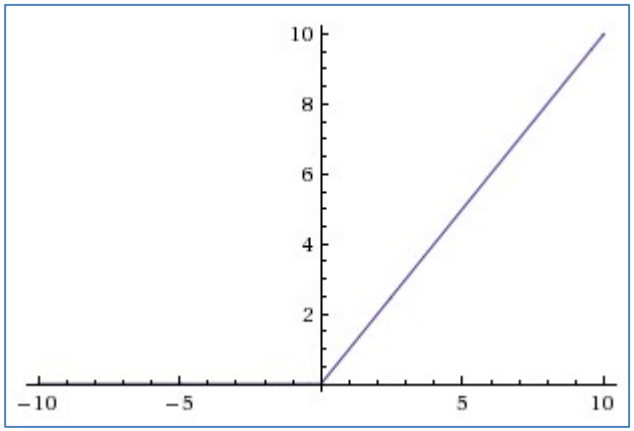


2



Rectified Linear Unit (ReLU)

- **Defined as**

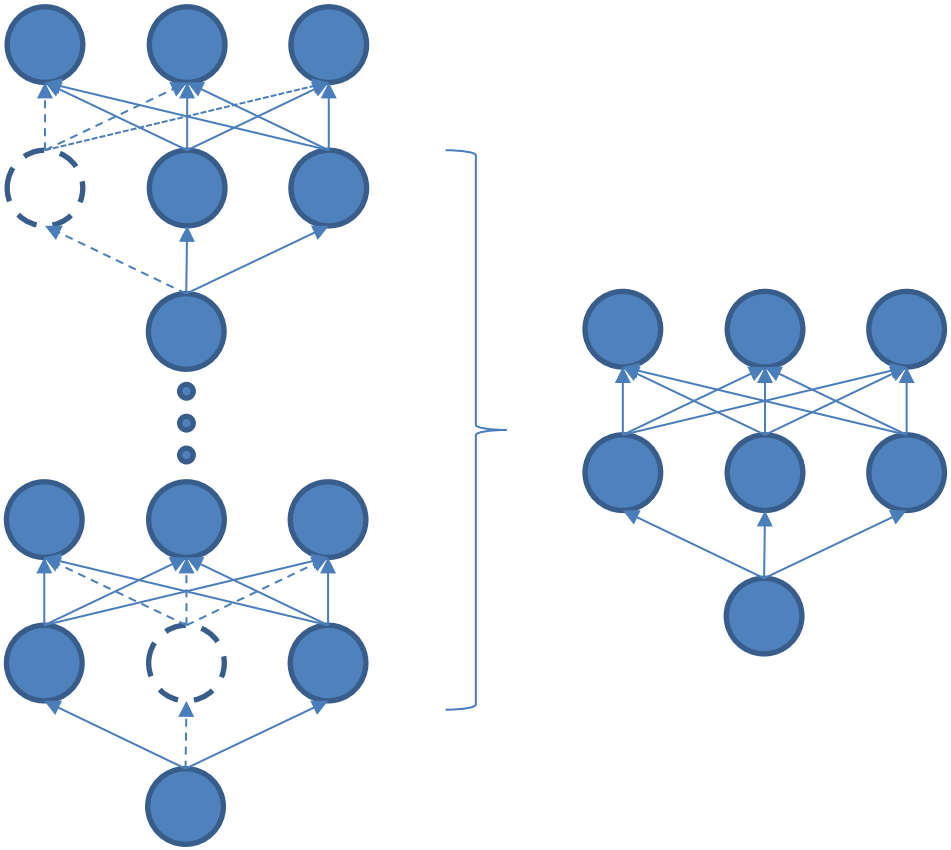


$$f(x) = \max(0, x)$$

- **Advantage**
  - ReLU has a linear slope at positivie value
    - ReLU solves gradient vanishing problem

Dropout

- **Dropout**



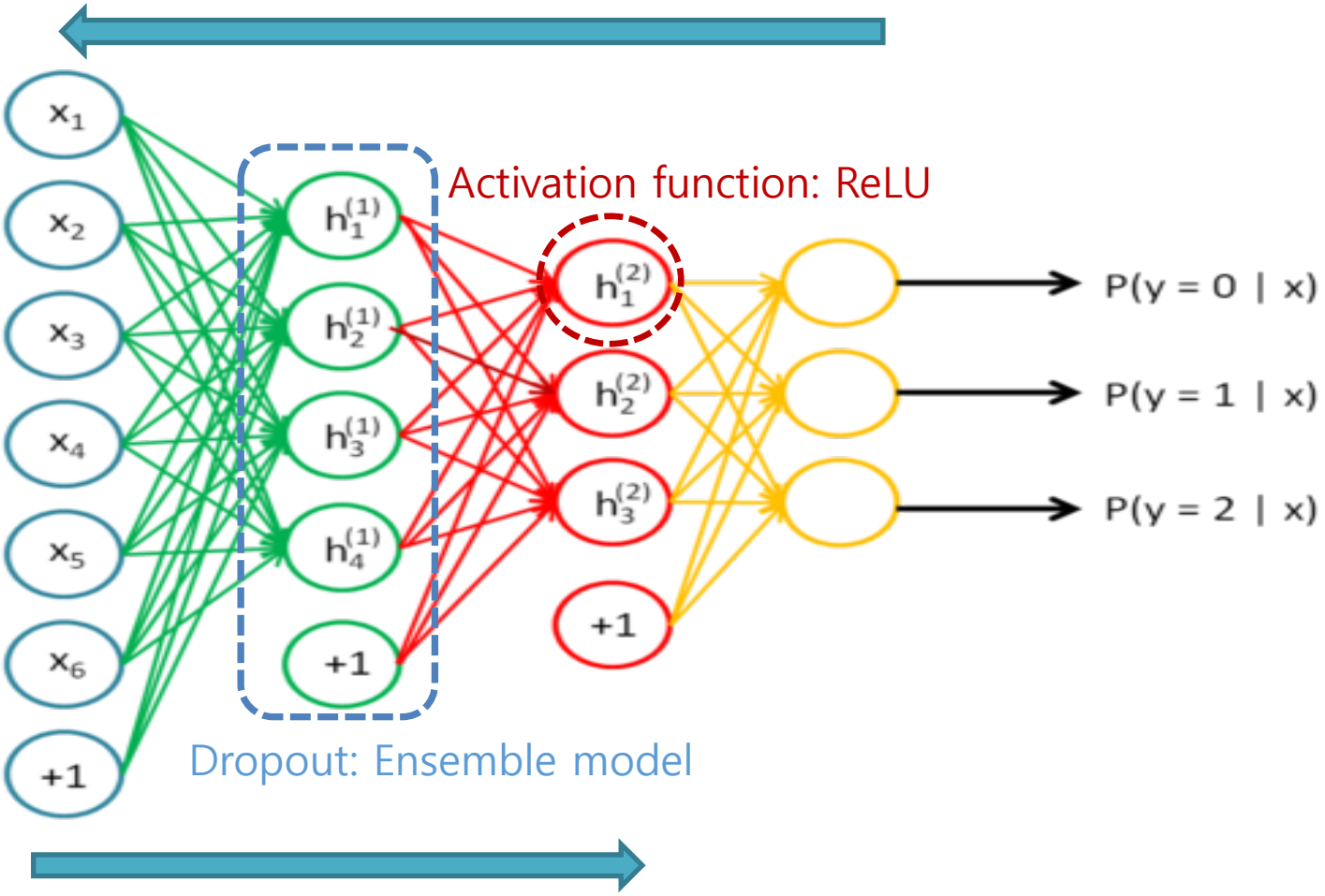
Multi-neural networks

✓ **Randomly and temporarily omitting hidden units from the network with a probability along with all their incoming and outgoing connections**

✓ **Effecting to average the predictions produced by a very large number of different networks**

Summary

Fine-tuning with supervised learning (back-propagation)



Pre-training with unsupervised learning (AE, RBM)

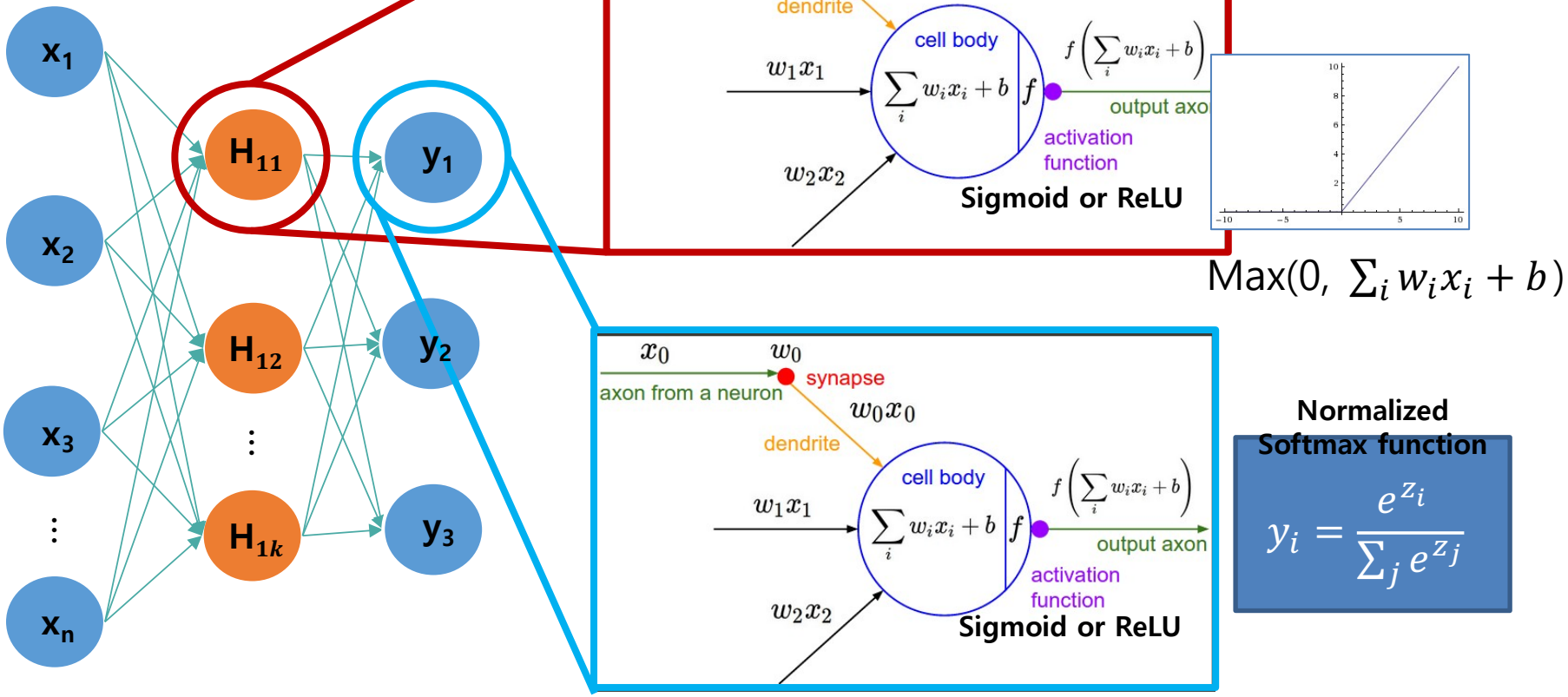
Learning NN

- **Forward propagation**
  - Calculation of output values (prob. or scalar)
  - Classification

- **Backward propagation**
  - Weight update

Learning NN-Forward propagation

- Structure of NN



Learning NN-loss (cost) function

- **linear regression**

$$L = \frac{1}{2}(t - y)^2$$

MSE

- **Nominal class**

$$L = -(t_i * \ln y_i + (1 - t_i) \ln(1 - y_i))$$

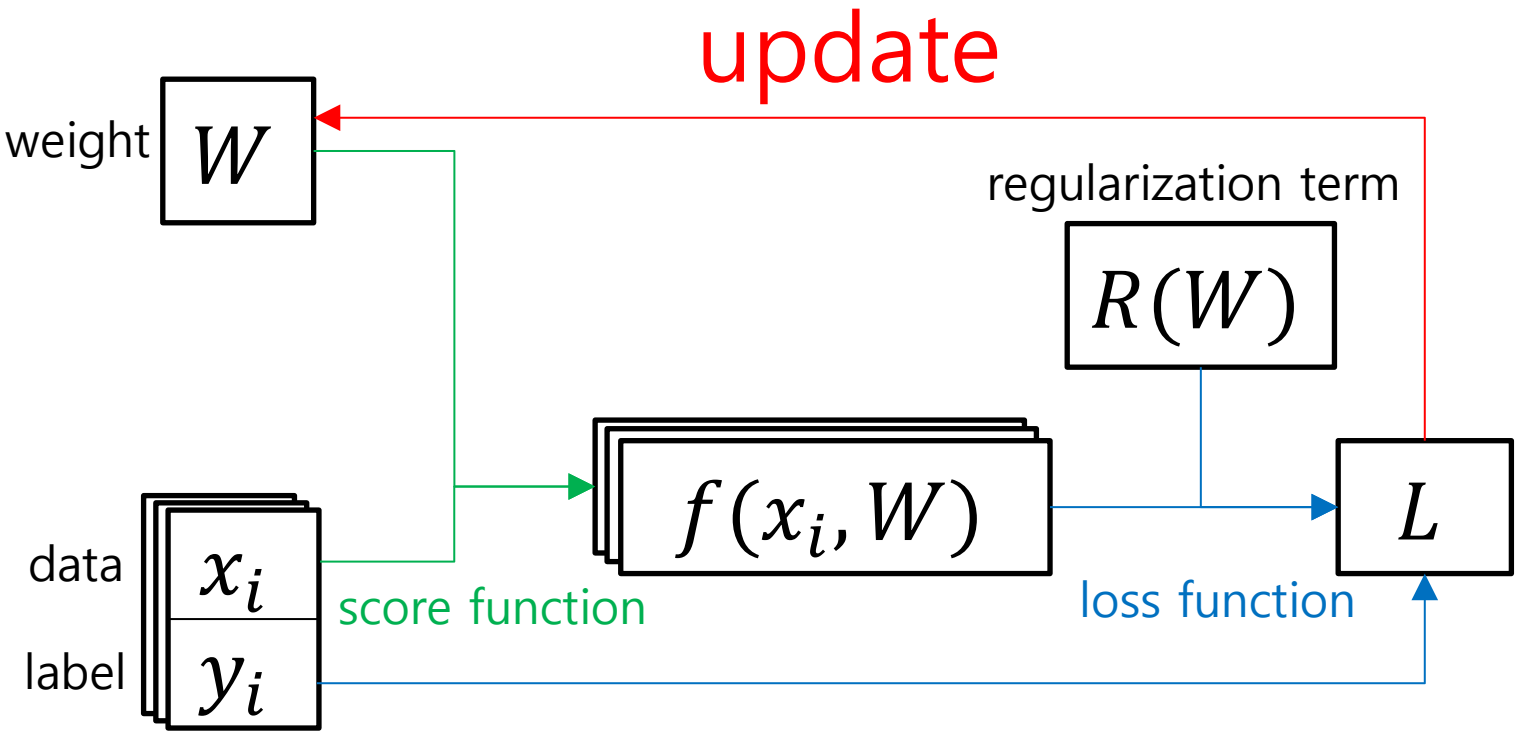
Cross entropy (=negative log likelihood)

Learning NN-backward propagation

- **Weight update**
  - ✓ $W_{new} = W_{pre} + \rho \frac{\partial L}{\partial w}$

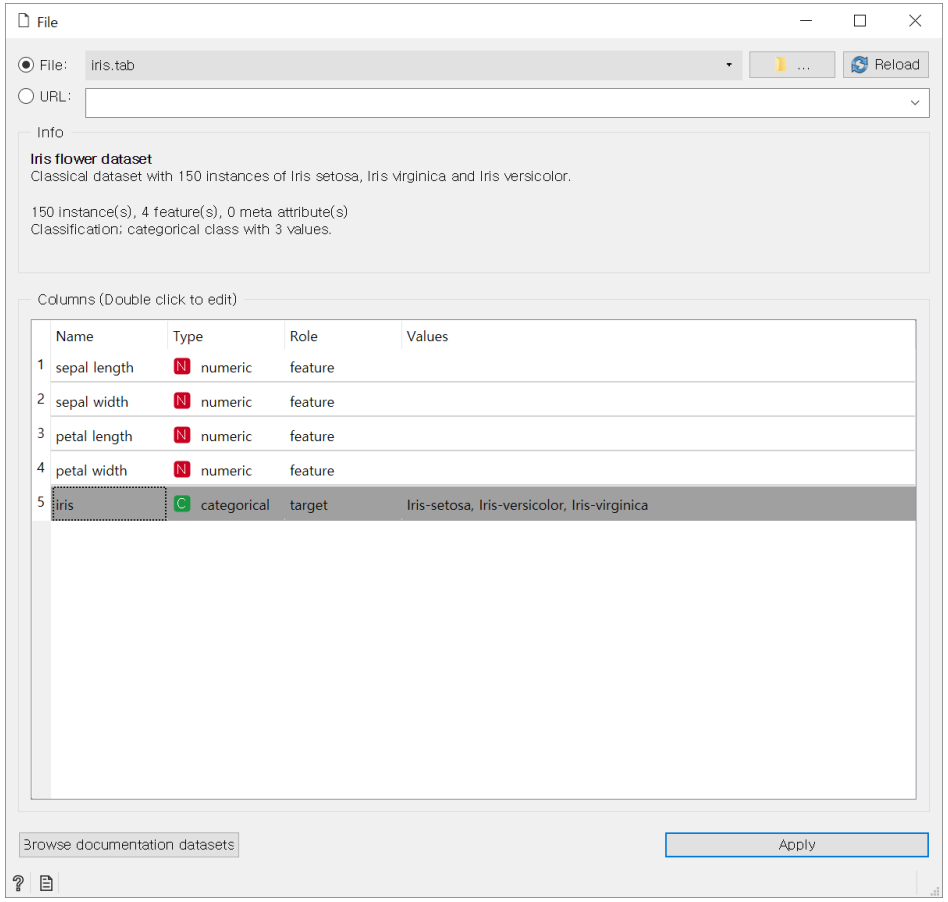
Learning NN Summary

- **Update W to Normalize L**

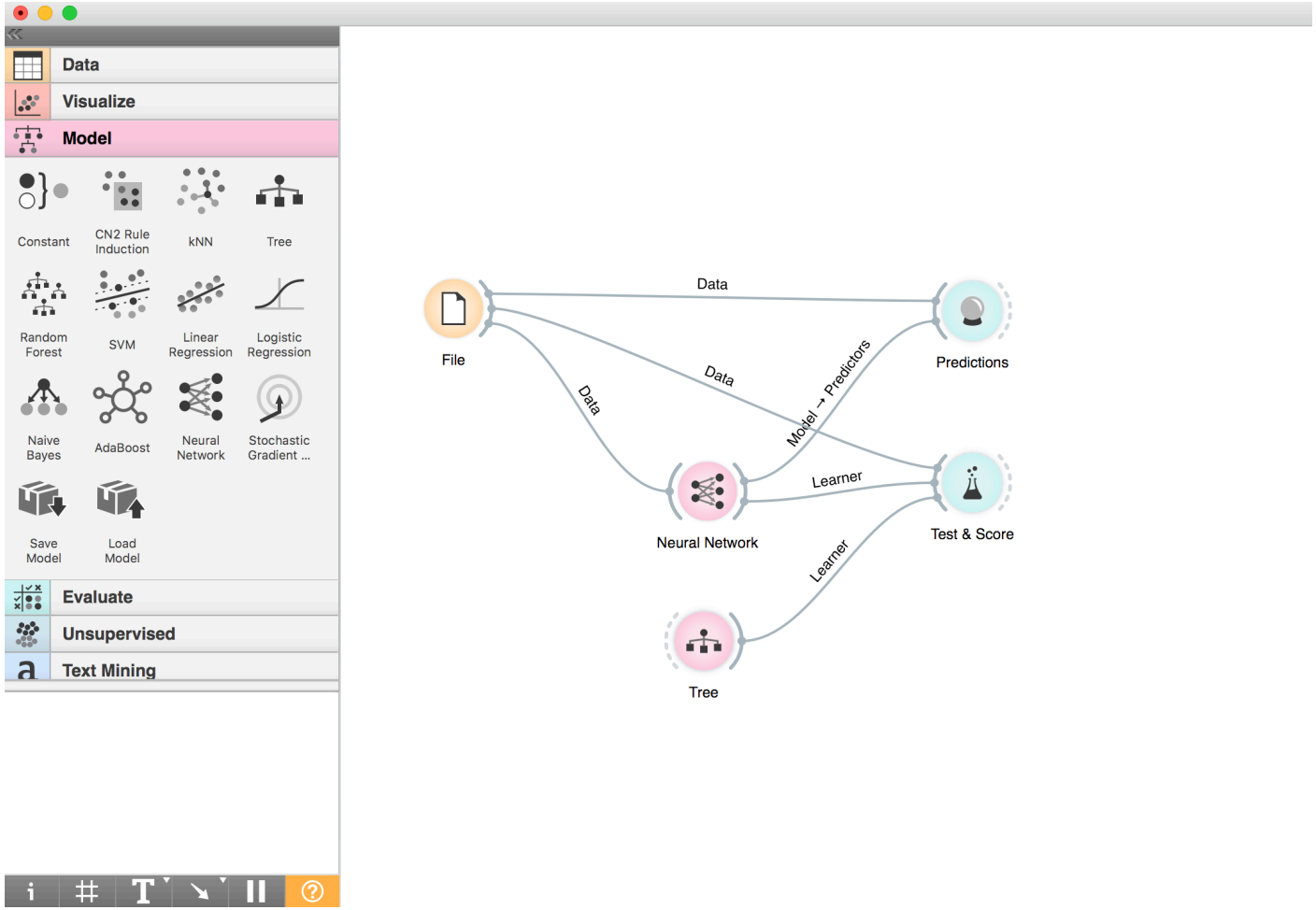


Dataset

- **iris data**

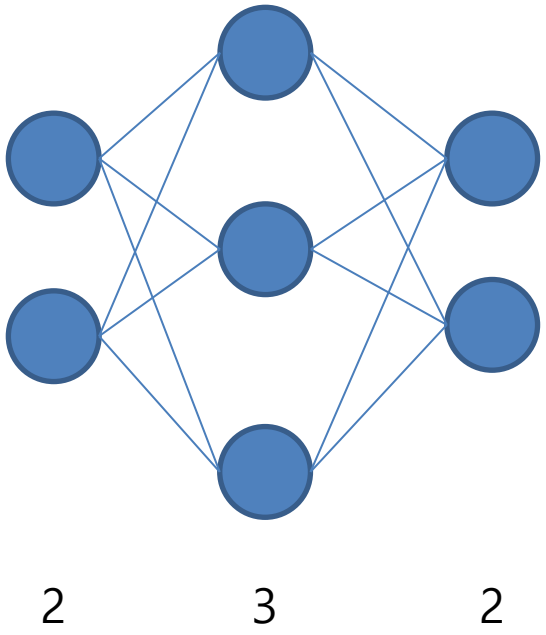


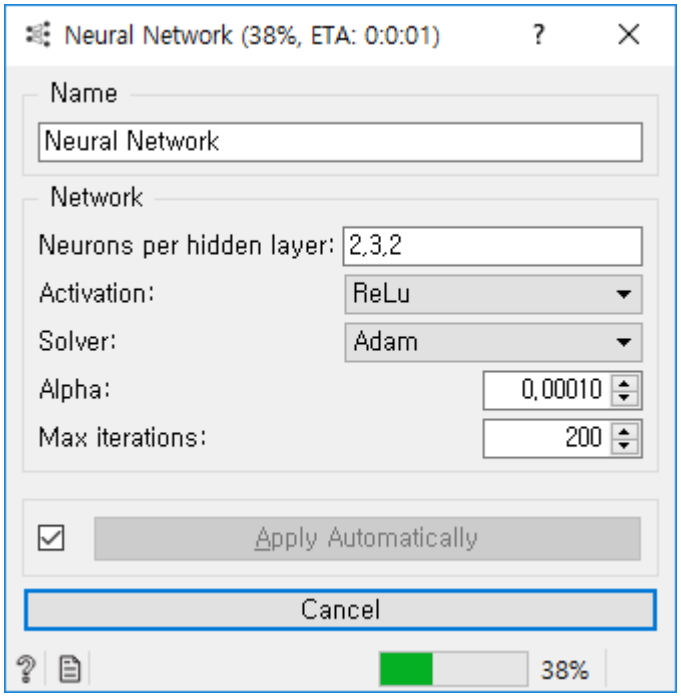
Build Schema



Build NN architecture

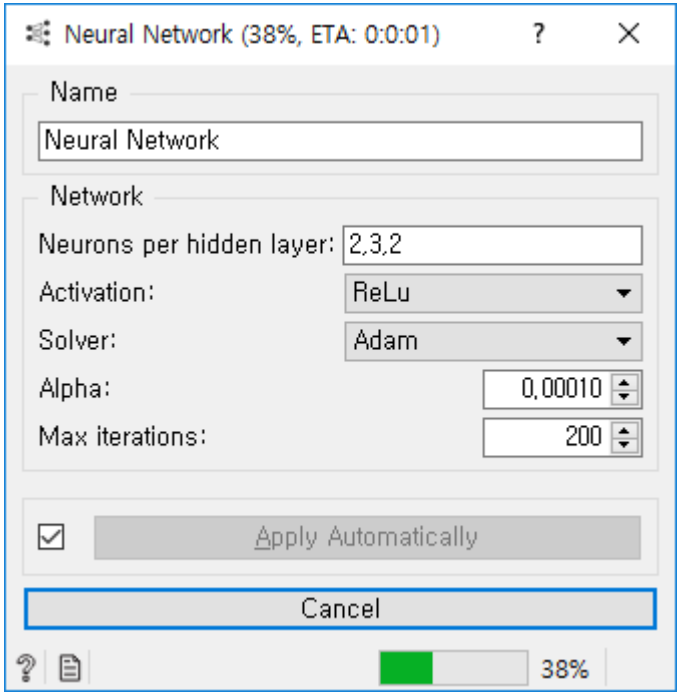
- **Build NN architecture on Orange3**



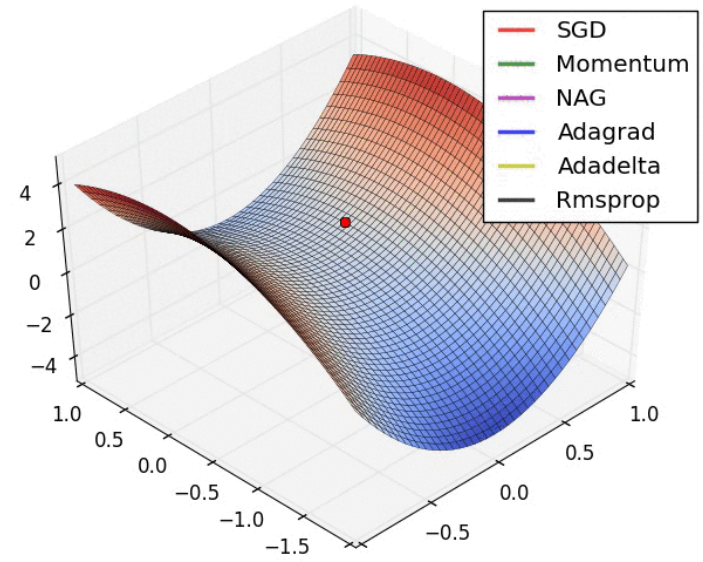


Hyperparameters and configuration

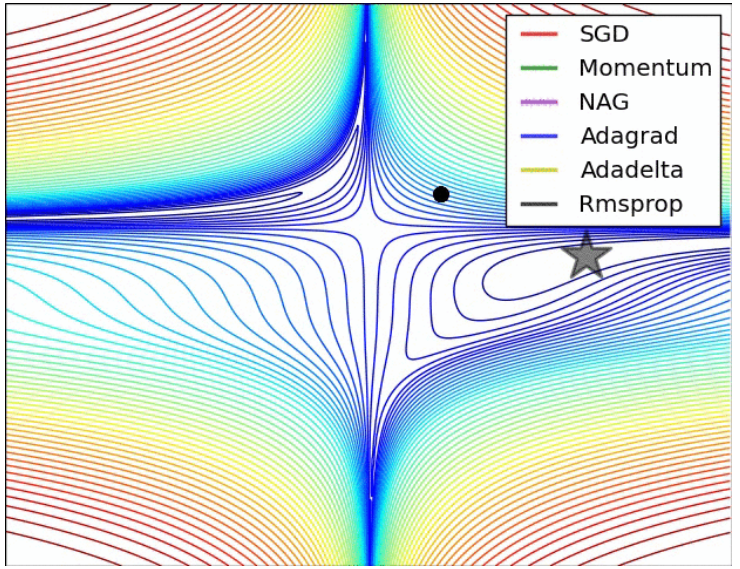
- **Neurons per hidden layer: defined as the ith element represents the number of neurons in the ith hidden layer. E.g. a neural network with 3 layers can be defined as 2, 3, 2.**
- **Activation function for the hidden layer:**
  - Identity: no-op activation, useful to implement linear bottleneck
  - Logistic: the logistic sigmoid function
  - tanh: the hyperbolic tan function
  - ReLu: the rectified linear unit function
- **Solver for weight optimization:**
  - L-BFGS-B: an optimizer in the family of quasi-Newton methods
  - SGD: stochastic gradient descent
  - Adam: stochastic gradient-based optimizer
- **Alpha: L2 penalty (regularization term) parameter**
- **Max iterations: maximum number of iterations**



Weight Optimization



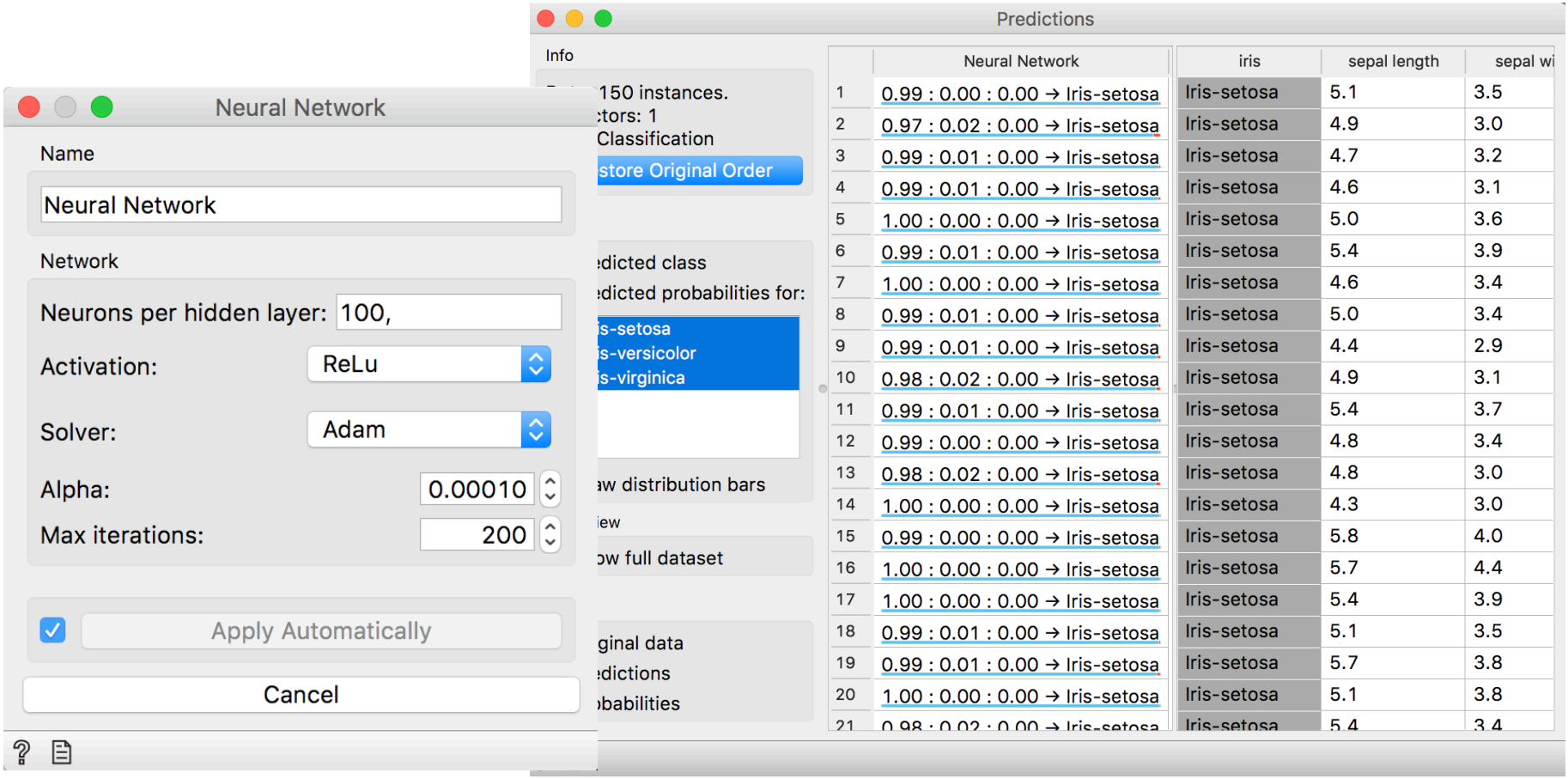
Gradient decent optimization algorithms at log valley



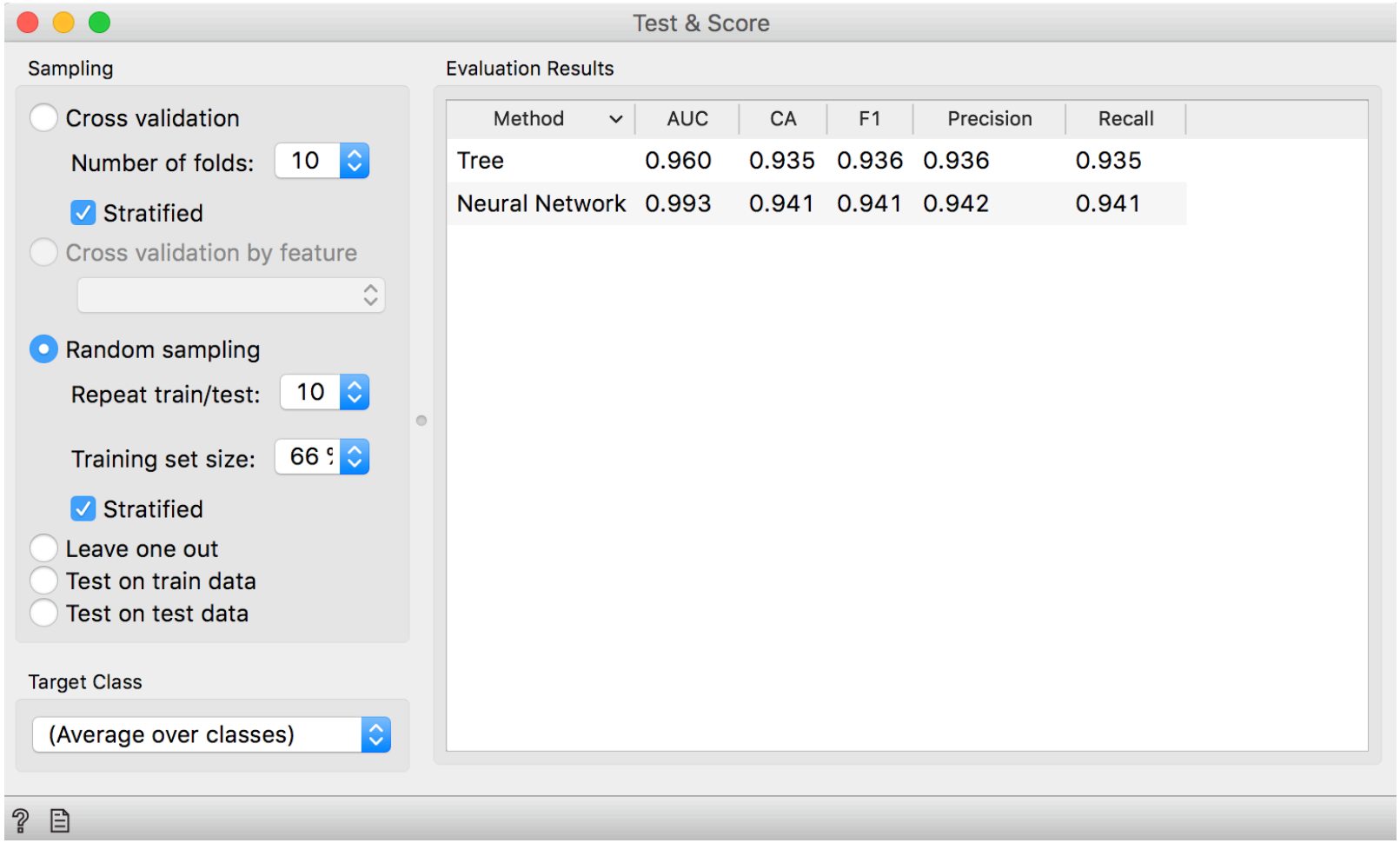
Gradient Descent Optimization Algorithms at Beale's Function

Ref : https://www.reddit.com/r/MachineLearning/comments/2gopfa/visualizing_gradient_optimization_techniques/cklhott/

Observation of NN



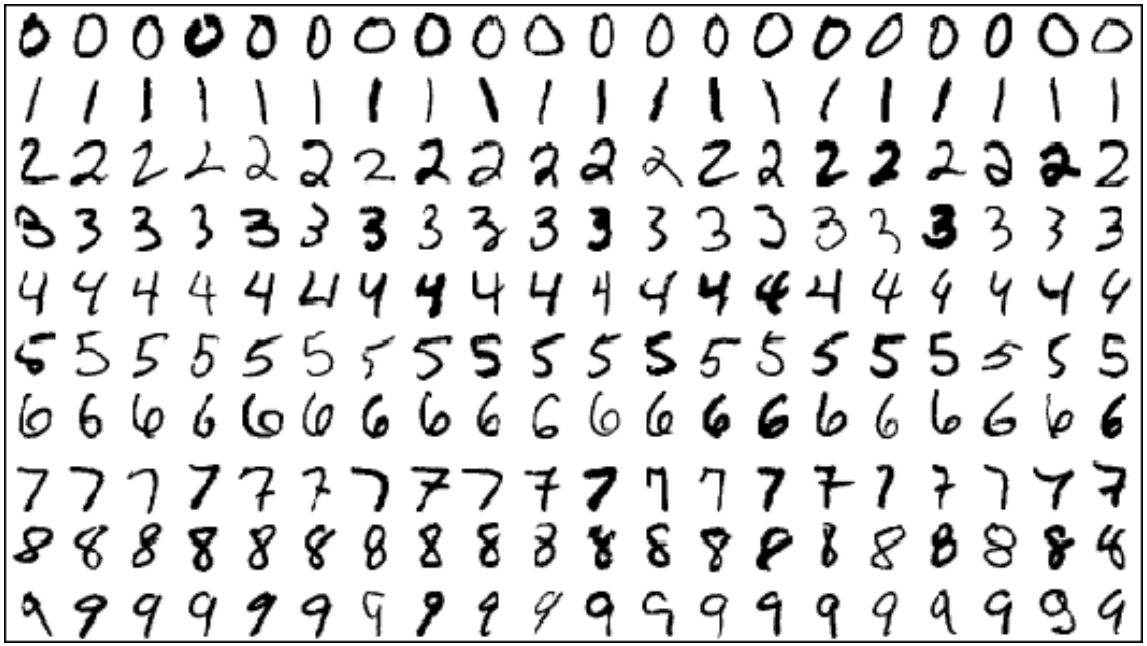
Performance comparison



Real Dataset

- **Digit data**
  - a famous digit data set
  - https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
  - a data set in the Orange format with the image files (https://blog.biolab.si/download/868/)

- **Self task**
  - to classify handwritten digits based on their bitmap representation



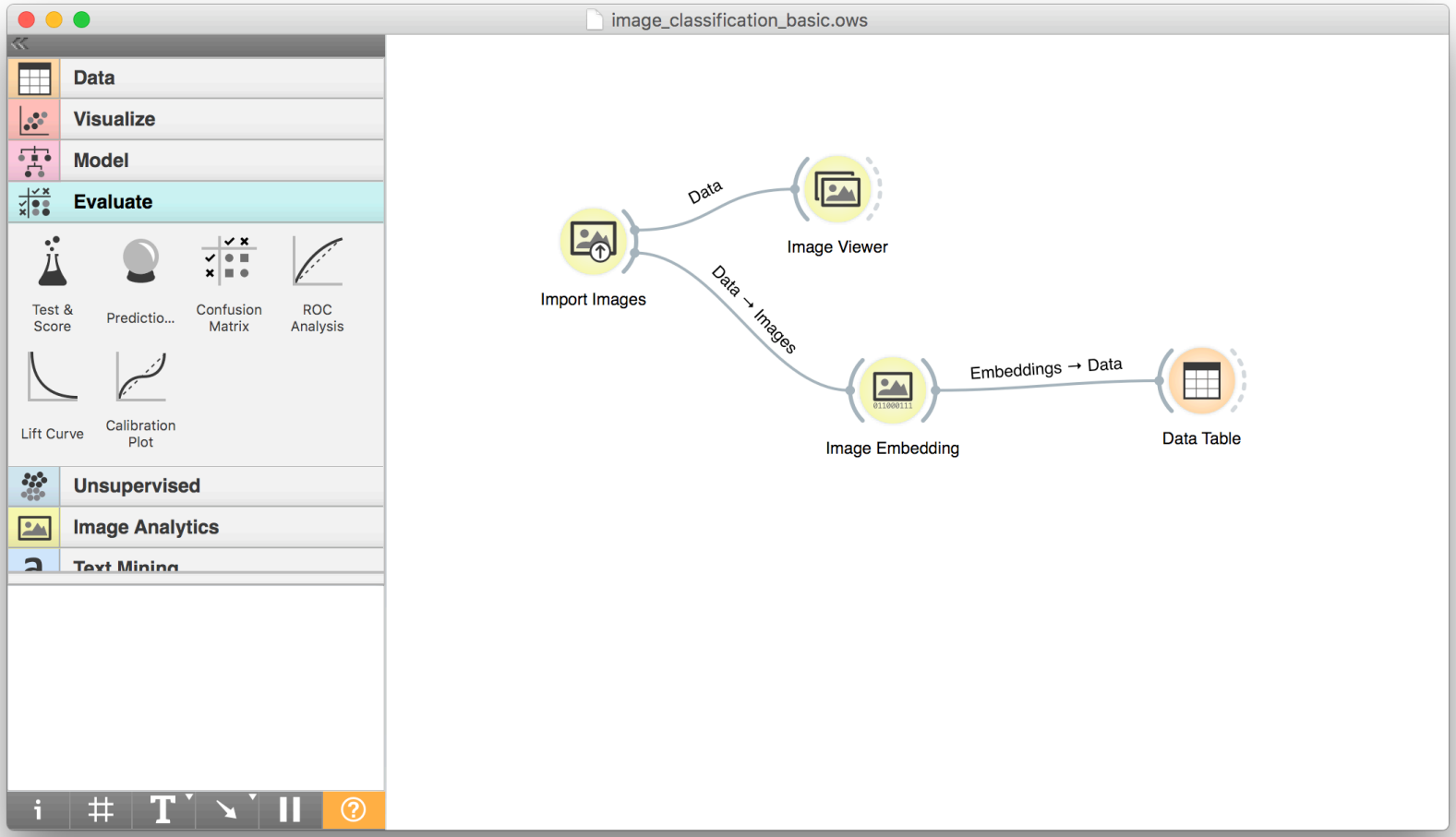
Build Schema

Self Study

- **classify handwritten digits based on their bitmap representation**
  - how do the images of the misclassified digits look like?
  - Improve performance by neural network

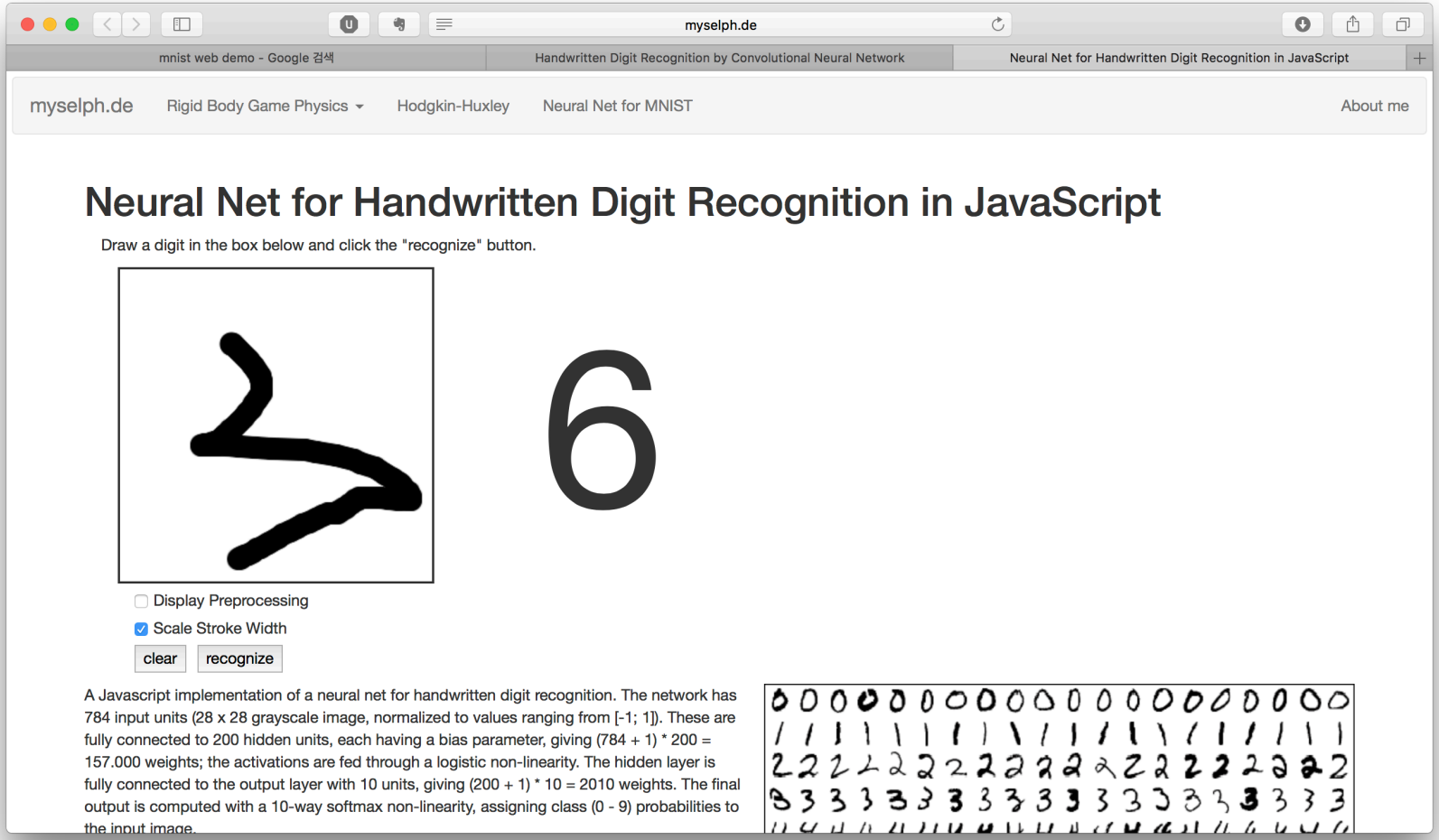
Build Schema

- **Image classification example**



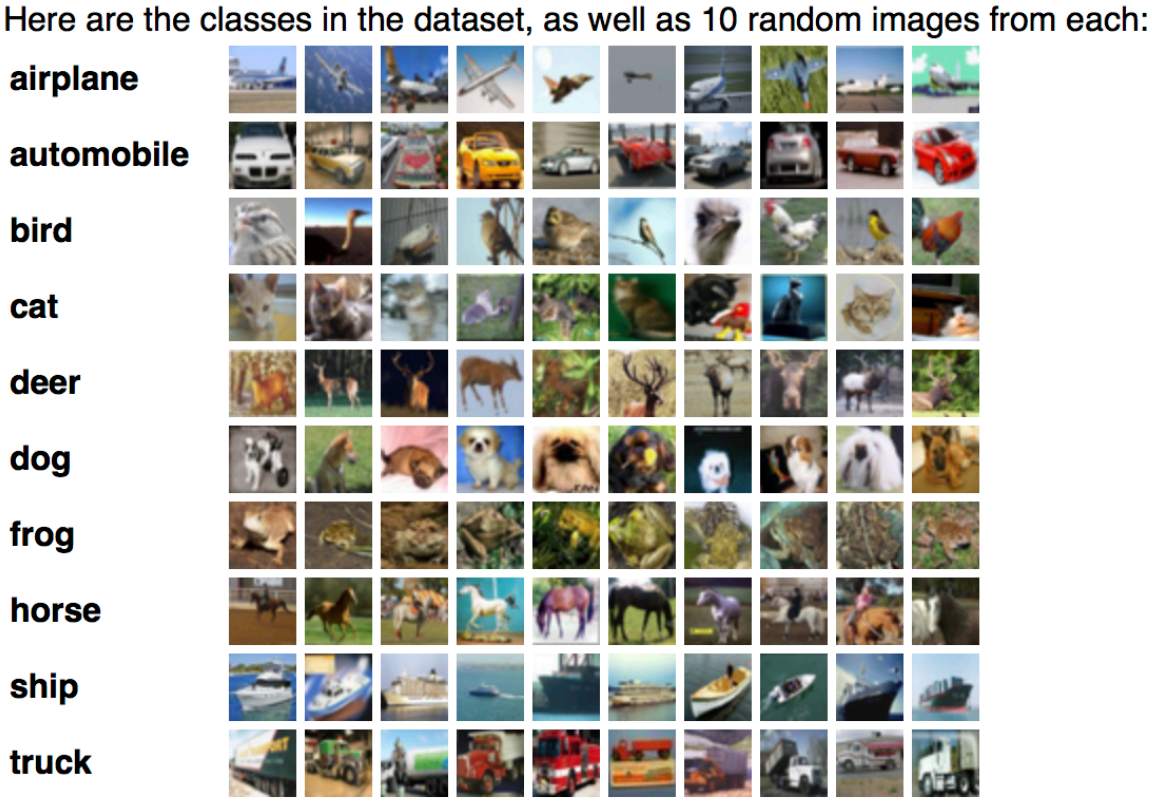
MNIST web demo

- **http://myselph.de/neuralNet.html**



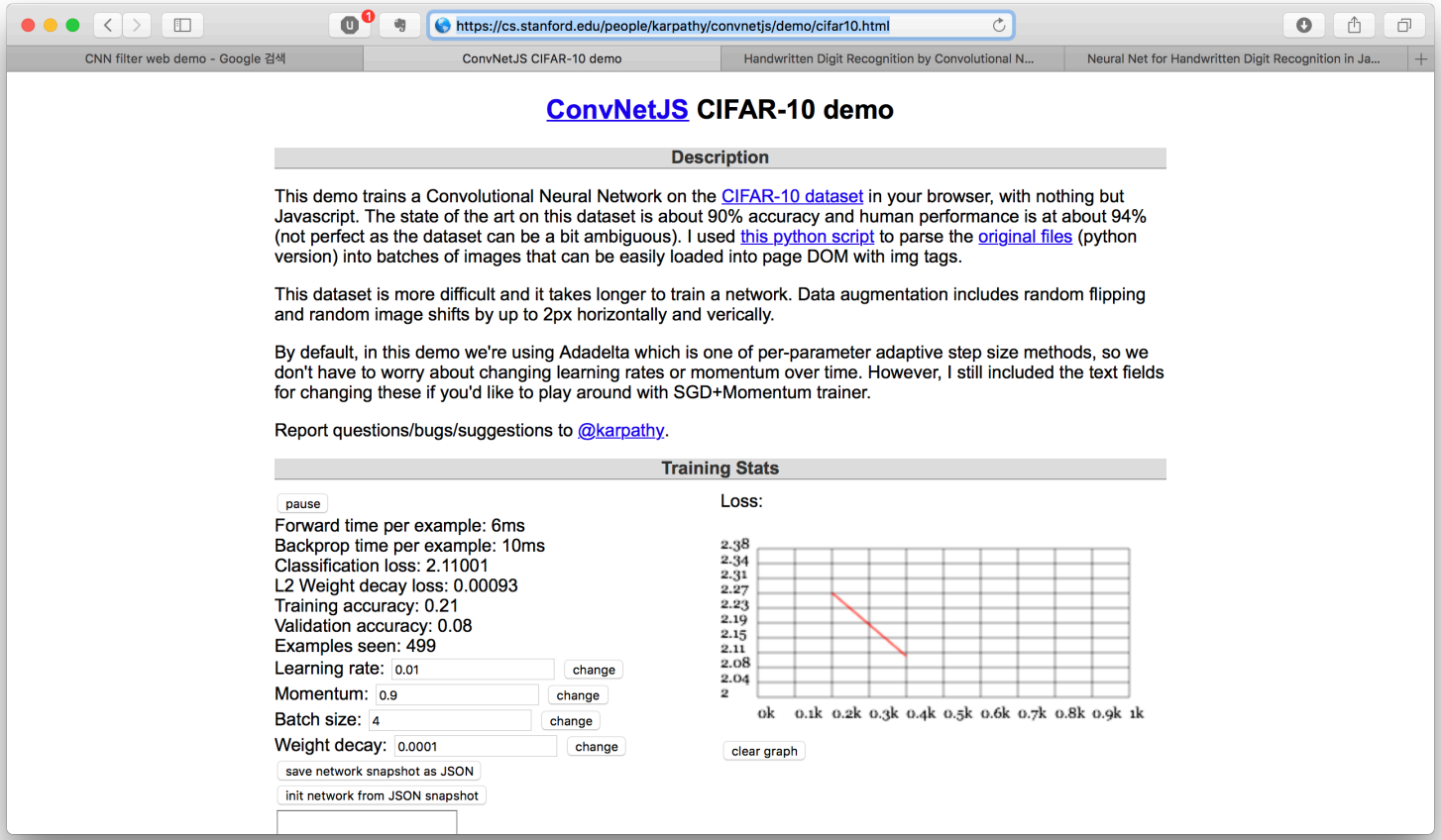
CIFAR-10

- **The CIFAR-10 dataset**
  - consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.



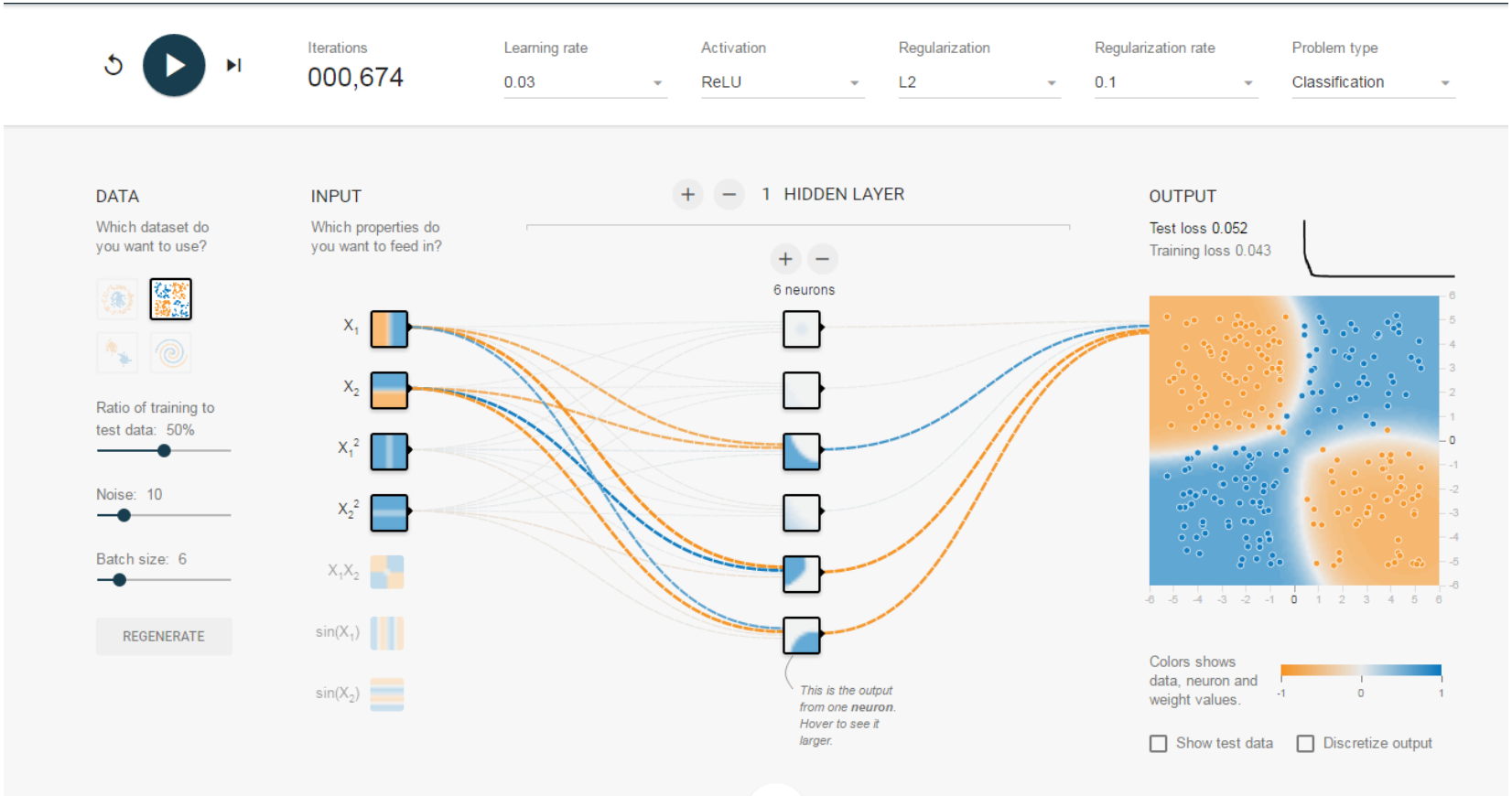
Convolution Neural Network Example

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html



NN Playground

- http://playground.tensorflow.org



References

- https://medium.com/machine-learning-101/chapter-3-decision-trees-theory-e7398adac567
- https://blog.biolab.si/tag/classification-tree/
- https://docs.orange.biolab.si/3/visual-programming/widgets/model/neuralnetwork.html
- https://blog.biolab.si/tag/images/
- https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/
- Stanford CS224d-Lecture6, 2017
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html
- https://www.cs.toronto.edu/~kriz/cifar.html

Thank you