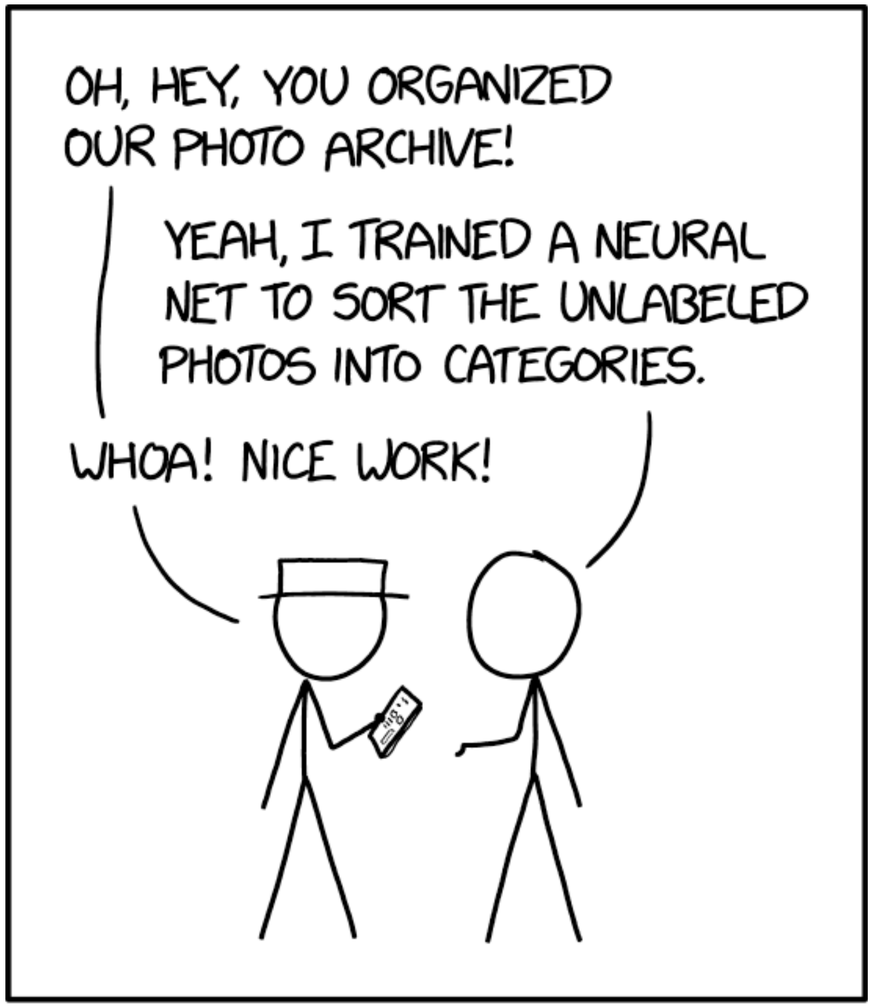
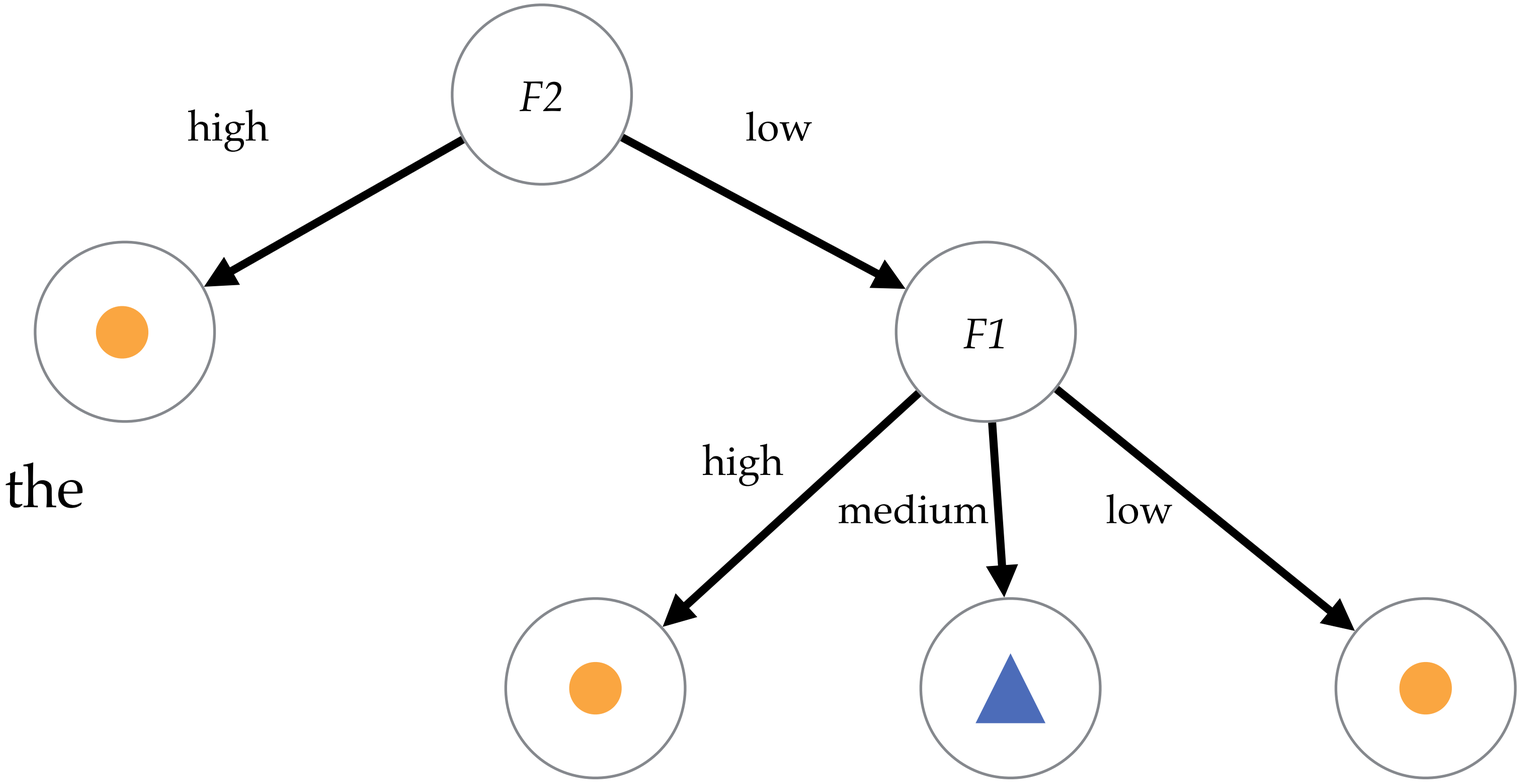
NON-LINEAR (DISCRIMINATIVE) CLASSIFIERS

Multilayer Perceptrons



https://xkcd.com/2173/

Recap: Decision Trees

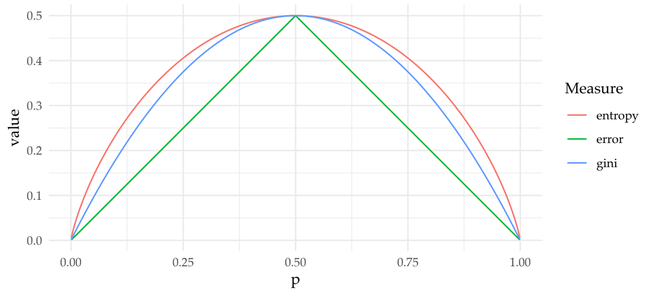


1. Recursively, greedily, partition the space, maximising the gain

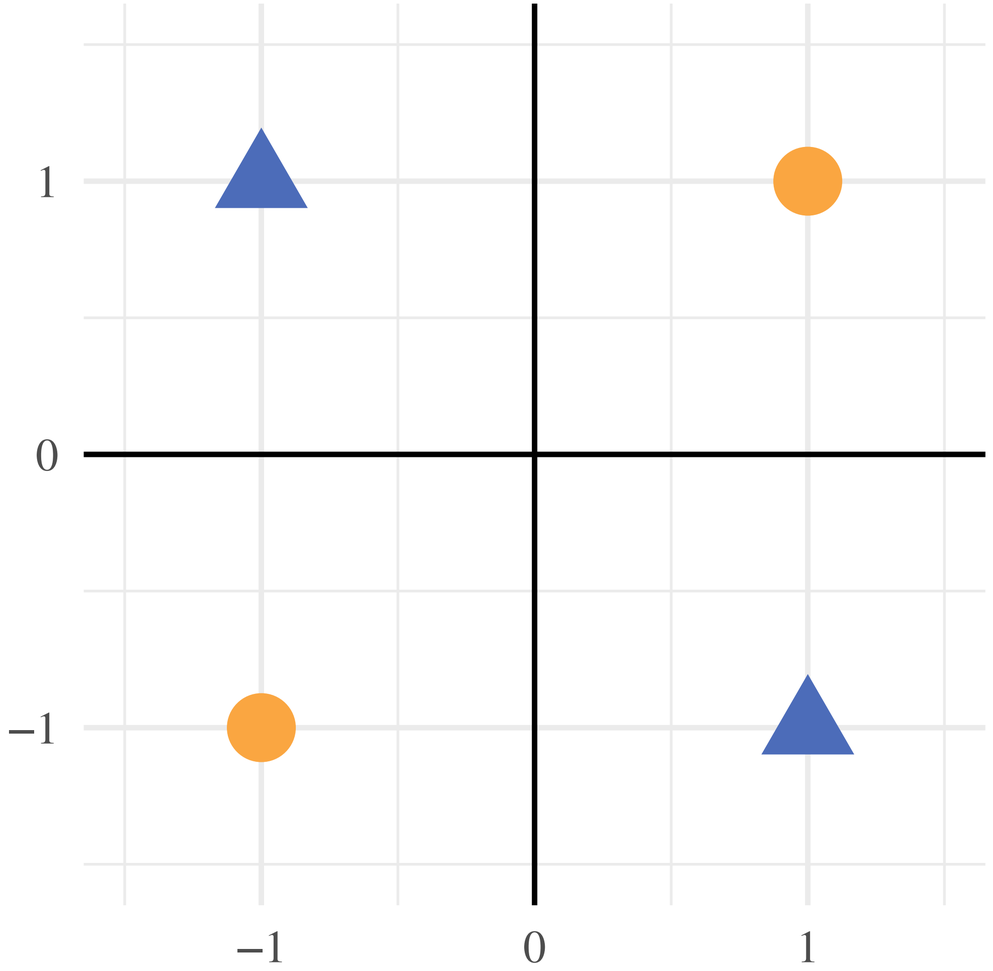
$$I(S) - \sum_{V \in Values(F)} \frac{|S_V|}{|S|} I(S_V)$$

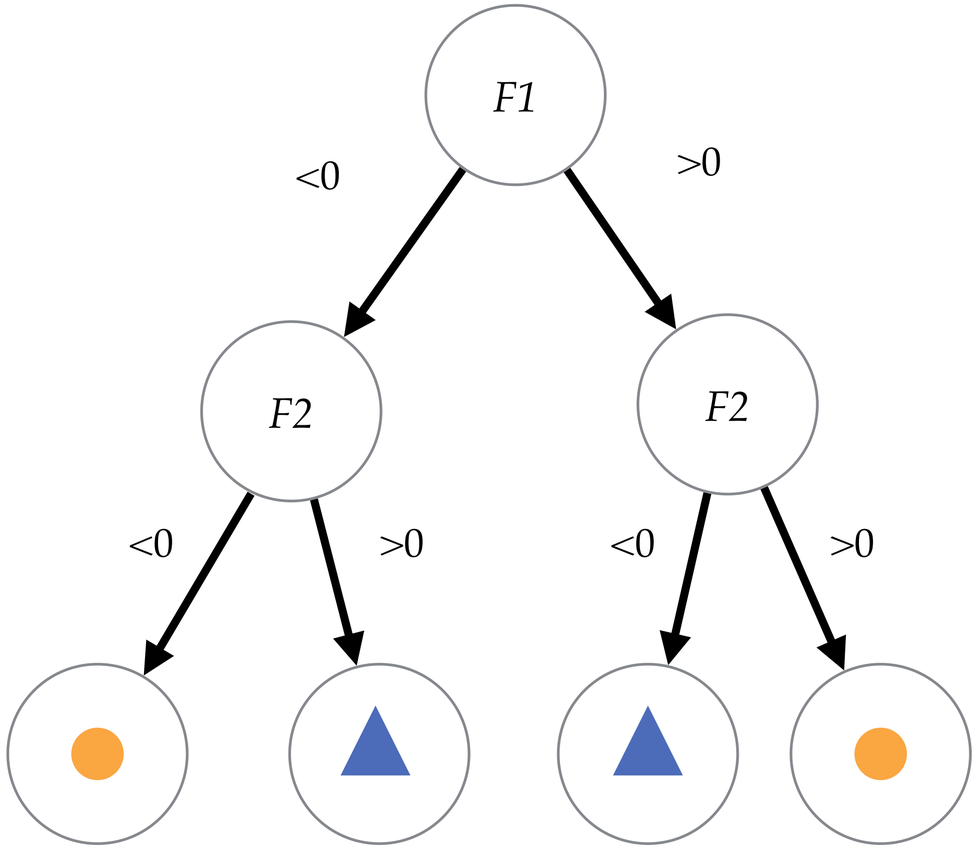
2. Assign each leaf to a class

3. Stop and/or prune to avoid overfitting



Revisiting XOR





Can the decision tree learner solve this?

Classifier Construction using Empirical Risk Minimisation

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

Learning Goals & Reading

LEARNING GOALS

After practicing with the concepts of this week you are able to

- **Explain when and why non-linear classifiers are needed**
- Explain the basic concepts of two non-linear classifiers: multi-layer perceptrons and **decision trees**
- Explain the **underlying algorithm of decision trees** and (multi-layer) perceptrons and how they are trained.
- Implement a decision tree
- Explain why and how one can **combine multiple classifiers**
- **Contrast a decision tree and a random forest**

LITERATURE

Please study the following material from Chris Bishop's "Pattern Recognition and Machine Learning":

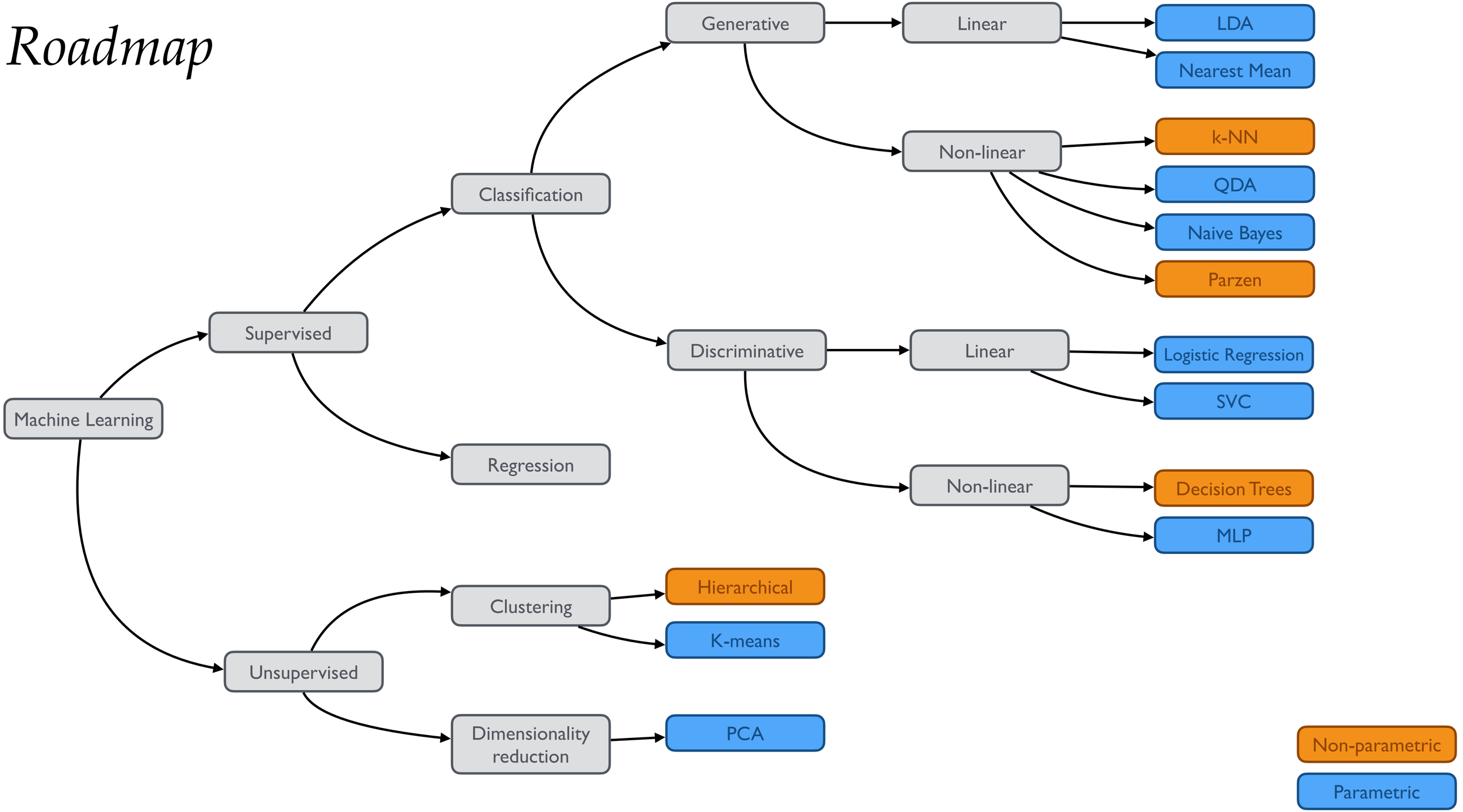
Lecture 6.1

- Section 14.2
- Section 14.4

Lecture 6.2

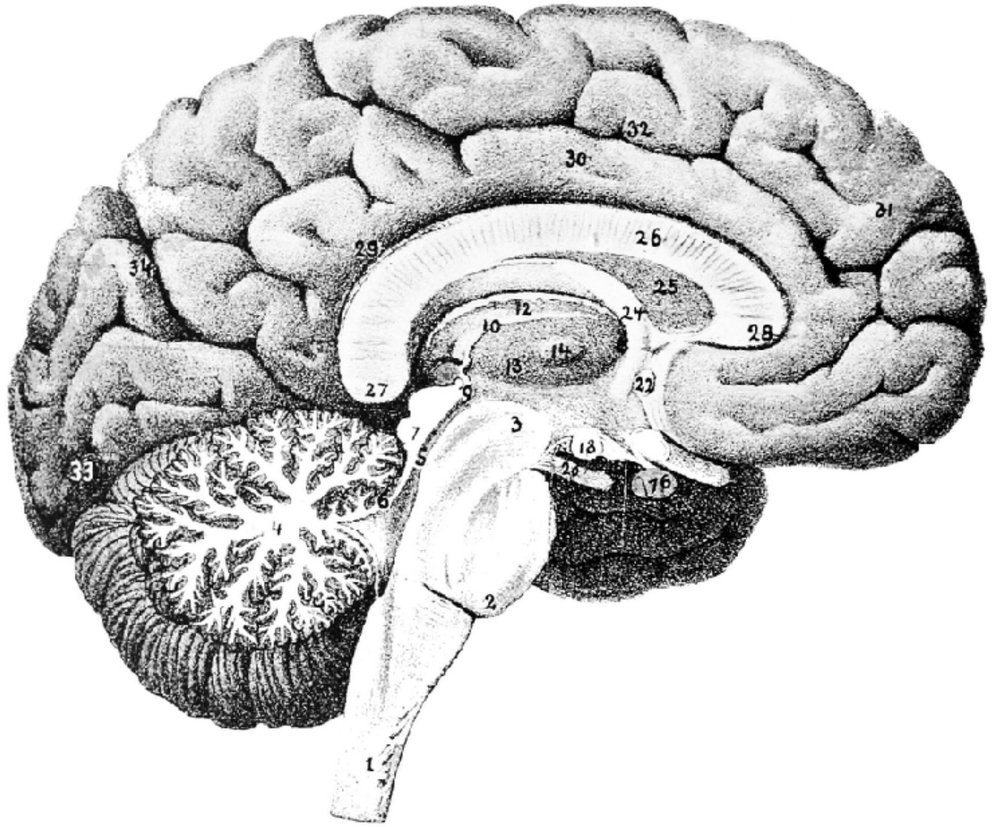
- Section 4.1.7
- Section 5.1
- Section 5.2 up to and including 5.2.1
- Section 5.3 up to and including 5.3.1

Roadmap


Machine Learning
Supervised
Classification
Generative
Linear
LDA
Nearest Mean
Non-linear
k-NN
QDA
Naive Bayes
Parzen
Discriminative
Linear
Logistic Regression
SVC
Non-linear
Decision Trees
MLP
Regression
Unsupervised
Clustering
Hierarchical
K-means
Dimensionality reduction
PCA
Non-parametric
Parametric

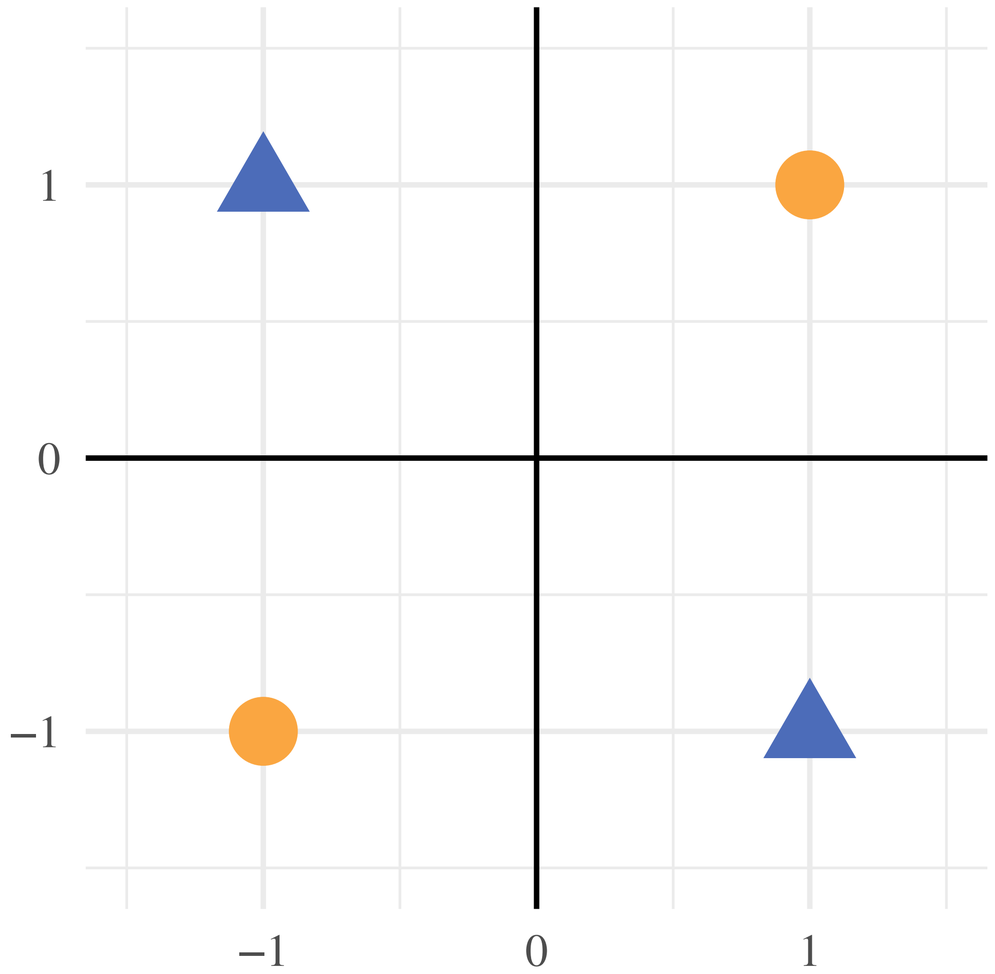
*Today: **the** common learning algorithm used in ML today*

- Multi-layer perceptrons / artificial neural networks / feed-forward “deep” neural networks
- Connected to topics we covered:
  - Logistic regression
  - Gradient descent
  - Classifier combining

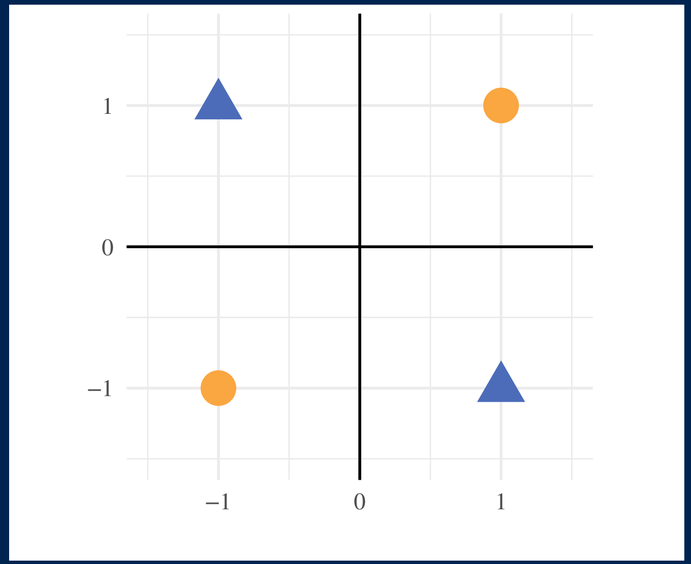


Source: Popular Science Monthly 46

XOR again



Question: feature transformations



1. Consider new feature x_1x_2 and draw the problem in this new feature space. What is the optimal classifier in this new space?

2. Is this classifier a linear classifier in the original feature space?

Practice Question: BMI

Body Mass Index (BMI) is defined as weight (in kg) divided by height^2 (in m^2). Suppose I use a fixed rule: to classify people as healthy when BMI<30.

1. **True or False**: Given the input features weight and height, I can construct a linear classifier that is exactly the same as this rule.

2. **True of False:** Given the input features weight, height and BMI, I can construct a linear classifier that is exactly the same as this rule.

PERCEPTRON

Electronic 'Brain' Teaches Itself

The Navy last week demonstrated the embryo of an electronic computer named the Perceptron which, when completed in about a year, is expected to be the first non-living mechanism able to "perceive, recognize and identify its surroundings without human training or control." Navy officers demonstrating a preliminary form of the device in Washington said they hesitated to call it a machine because it is so much like a "human being without life."

Dr. Frank Rosenblatt, research psychologist at the Cornell Aeronautical Laboratory, Inc., Buffalo, N. Y., designer of the Perceptron, conducted the demonstration. The machine, he said, would be the first electronic device to think as the human brain. Like humans, Perceptron will make mistakes at first, "but it will grow wiser as it gains experience," he said.

recognize the difference between right and left, almost the way a child learns.

When fully developed, the Perceptron will be designed to remember images and information it has perceived itself, whereas ordinary computers remember only what is fed into them on punch cards or magnetic tape.

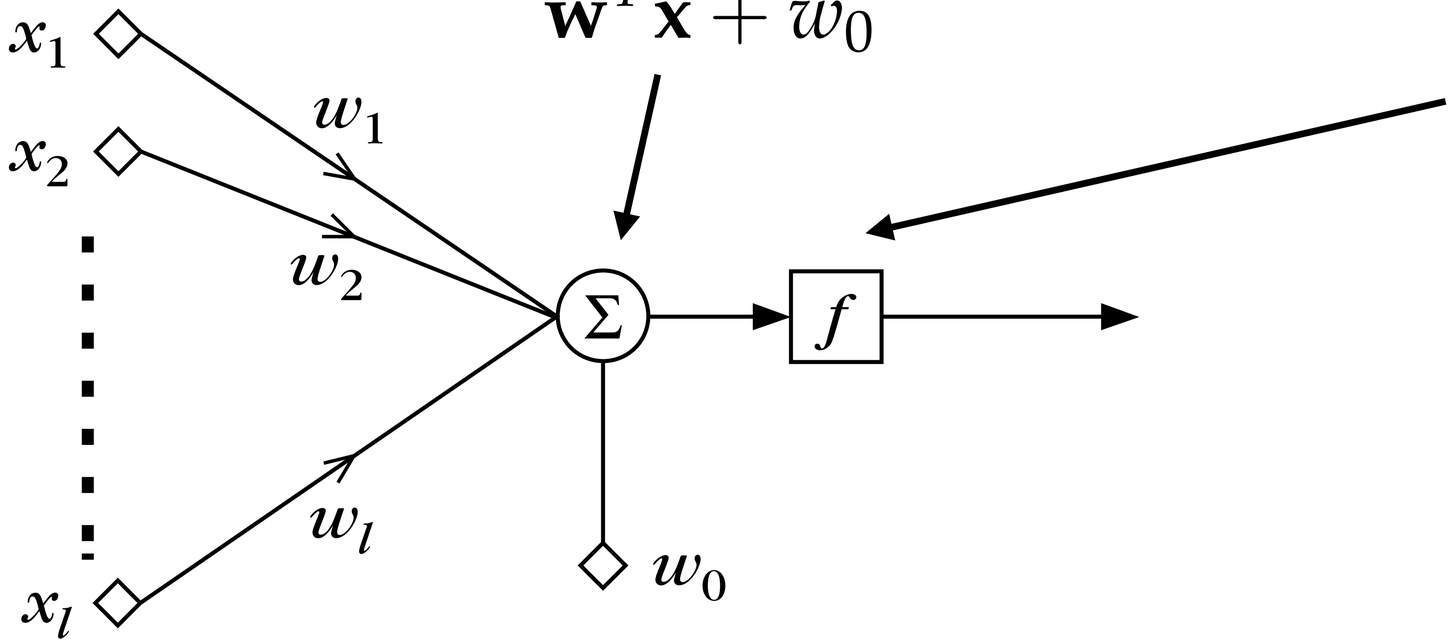
Later Perceptrons, Dr. Rosenblatt said, will be able to recognize people and call out their names. Printed pages, longhand letters and even speech commands are within its reach. Only one more step of development, a difficult step, he said, is needed for the device to hear speech in one language and instantly translate it to speech or writing in another language.

Self-Reproduction

In principle, Dr. Rosenblatt said, it would be possible to build Per-

New York Times July 13, 1958, Section E, Page 9

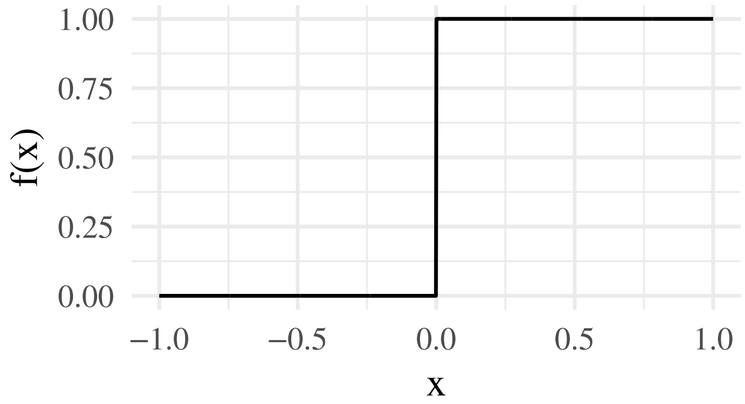
Perceptron



Activation function

Perceptrons use step function

$$f(x) = \mathbb{I}(x > 0)$$



Inspired by a simplified model of neurons in the brain

Perceptron Math

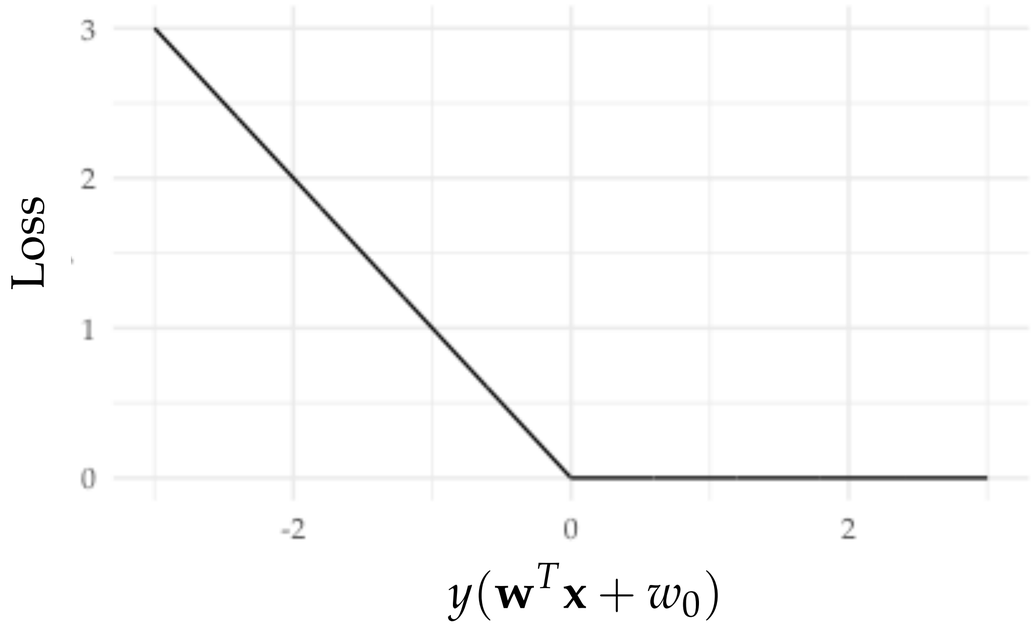
$$y_i \in \{-1, +1\}$$

$$\sum_{i=1}^{N} \max(-y_i(\mathbf{w}^T \mathbf{x}_i + w_0), 0)$$

Loss: perceptron loss

Function class: linear

Optimizer: (stochastic) gradient descent



Perceptron Training

$$\sum_{i=1}^{N} \max(-y_i(\mathbf{w}^T\mathbf{x}_i + w_0), 0)$$

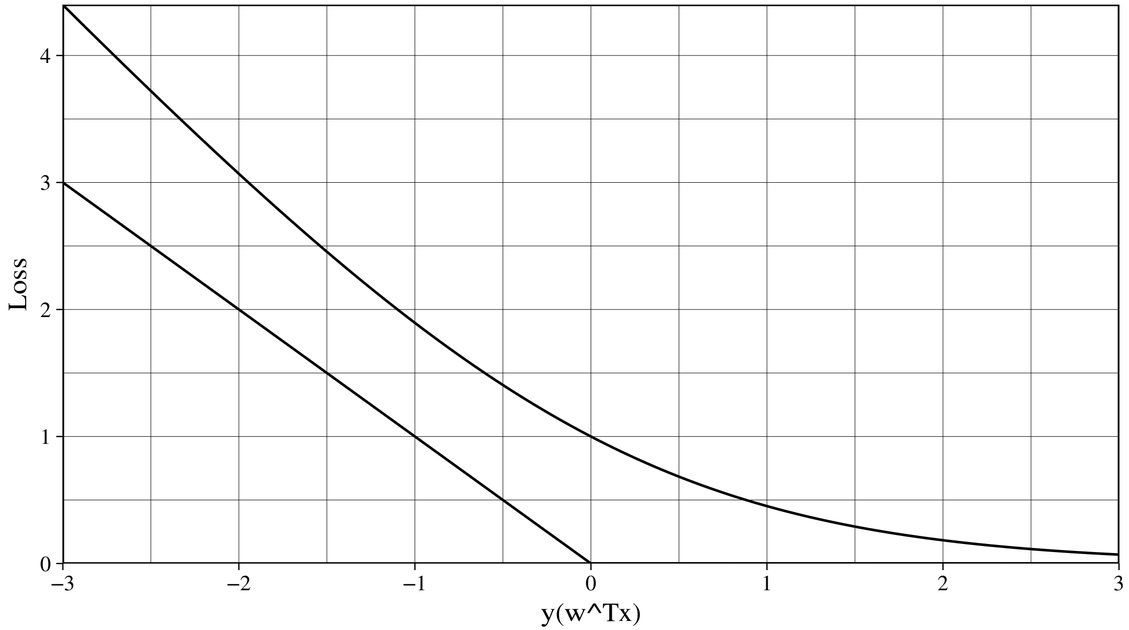
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<0} -y_i\mathbf{x}_i$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \alpha_t \sum_{y_i(\mathbf{w}^{t\,T}\mathbf{x}_i)<0} y_i\mathbf{x}_i$$

Guaranteed to converge if:

1. Problem is linearly separable
2. We choose the right, decreasing, linear rate

Logistic Regression



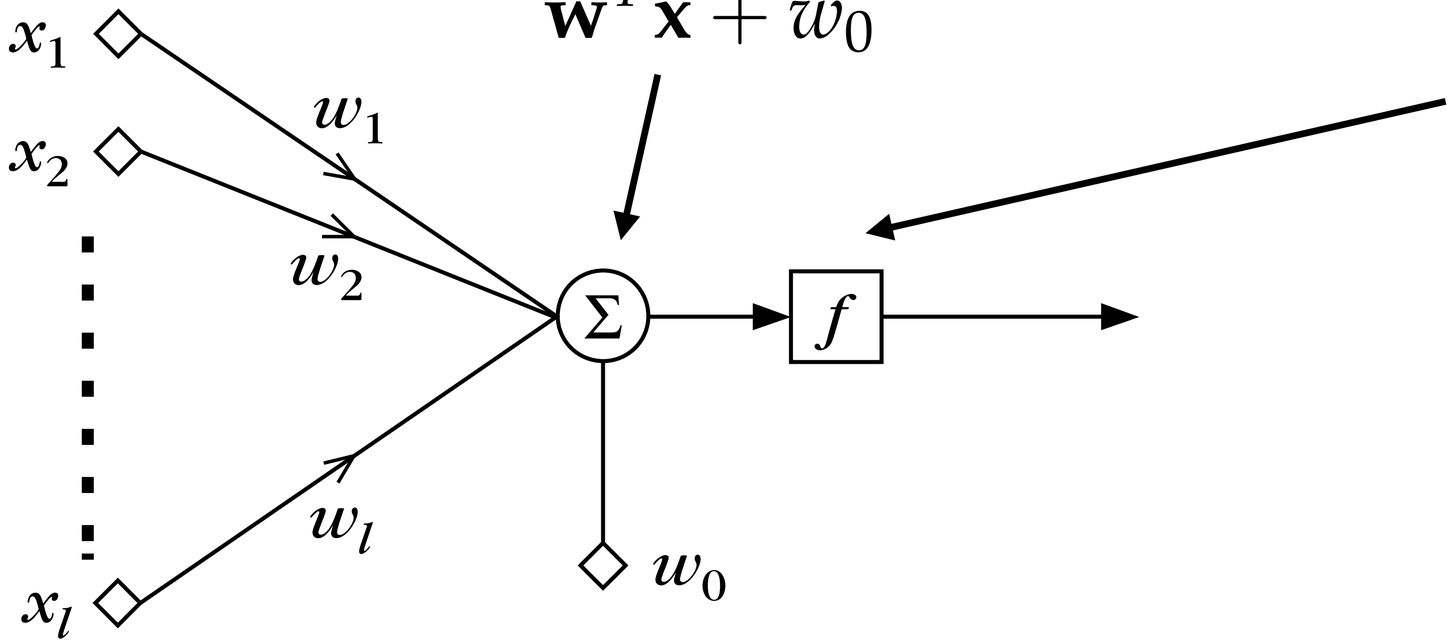
$$\sum_{i=1}^{N} \frac{1}{\log(2)} \log(1 + \exp[-y_i(\mathbf{w}^T\mathbf{x}_i + w_0)])$$

$$\sum_{i=1}^{N} \max(-y_i(\mathbf{w}^T\mathbf{x}_i + w_0), 0)$$

We have seen something similar: logistic regression

But: different “activation function” and different loss

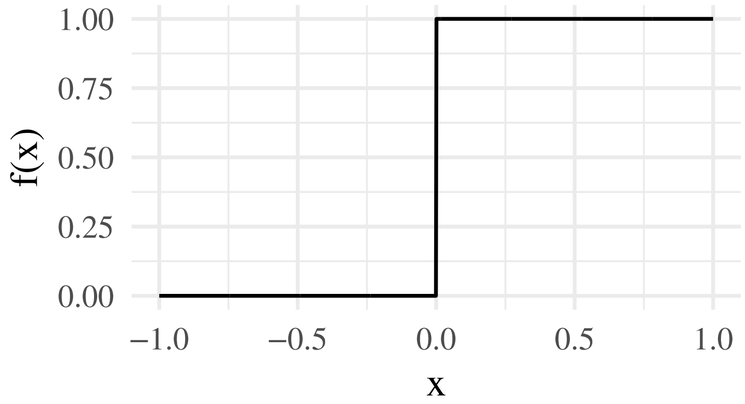
Perceptron



Activation function

Perceptrons use step function

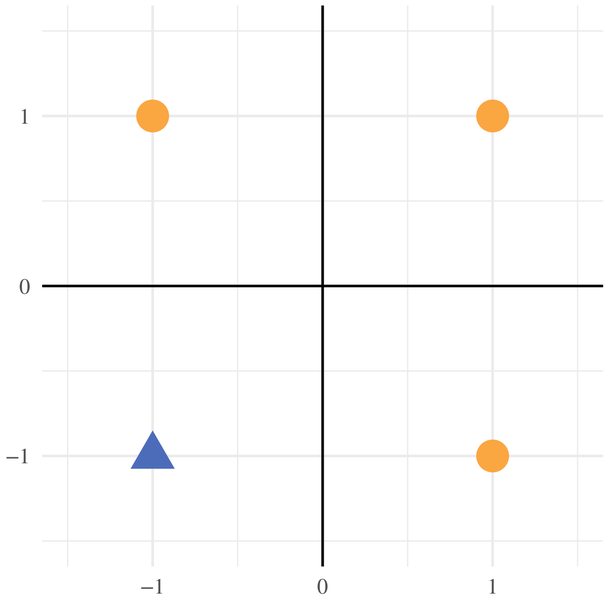
$$f(x) = \mathbb{I}(x > 0)$$

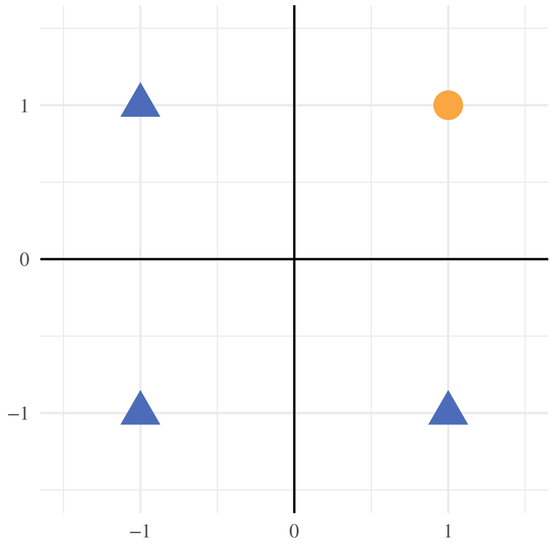


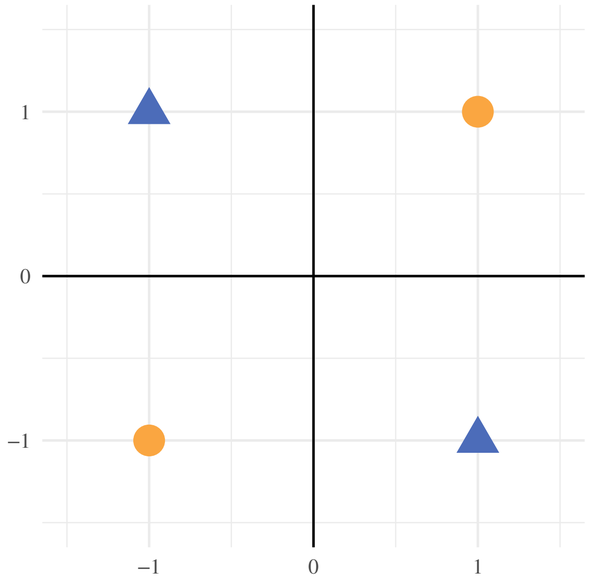
Inspired by a simplified model of neurons in the brain

Question: Learning logic gates

- What logic functions can be learned by the perceptron:
  - OR
  - NOR
  - AND
  - NAND
  - XOR

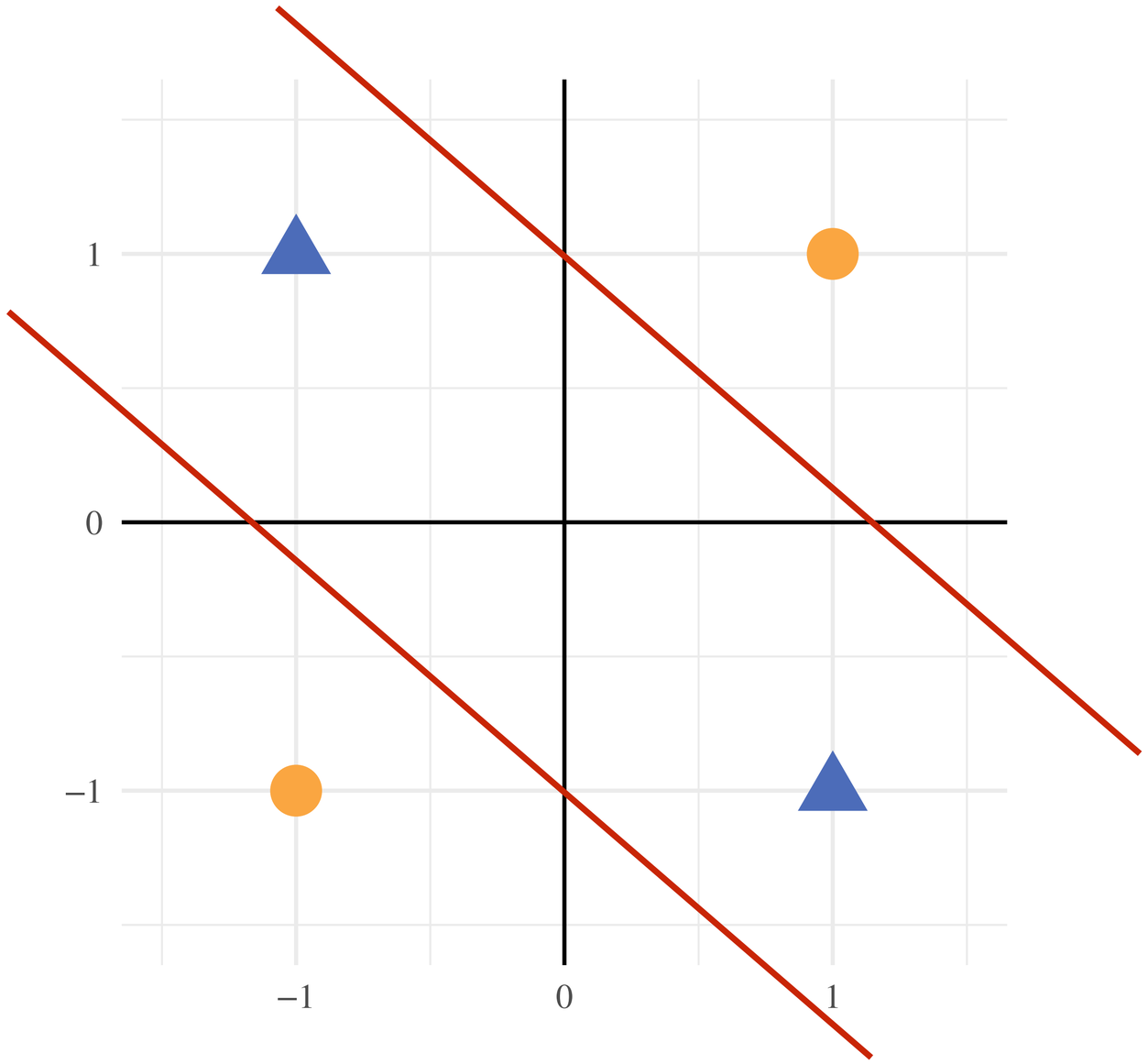




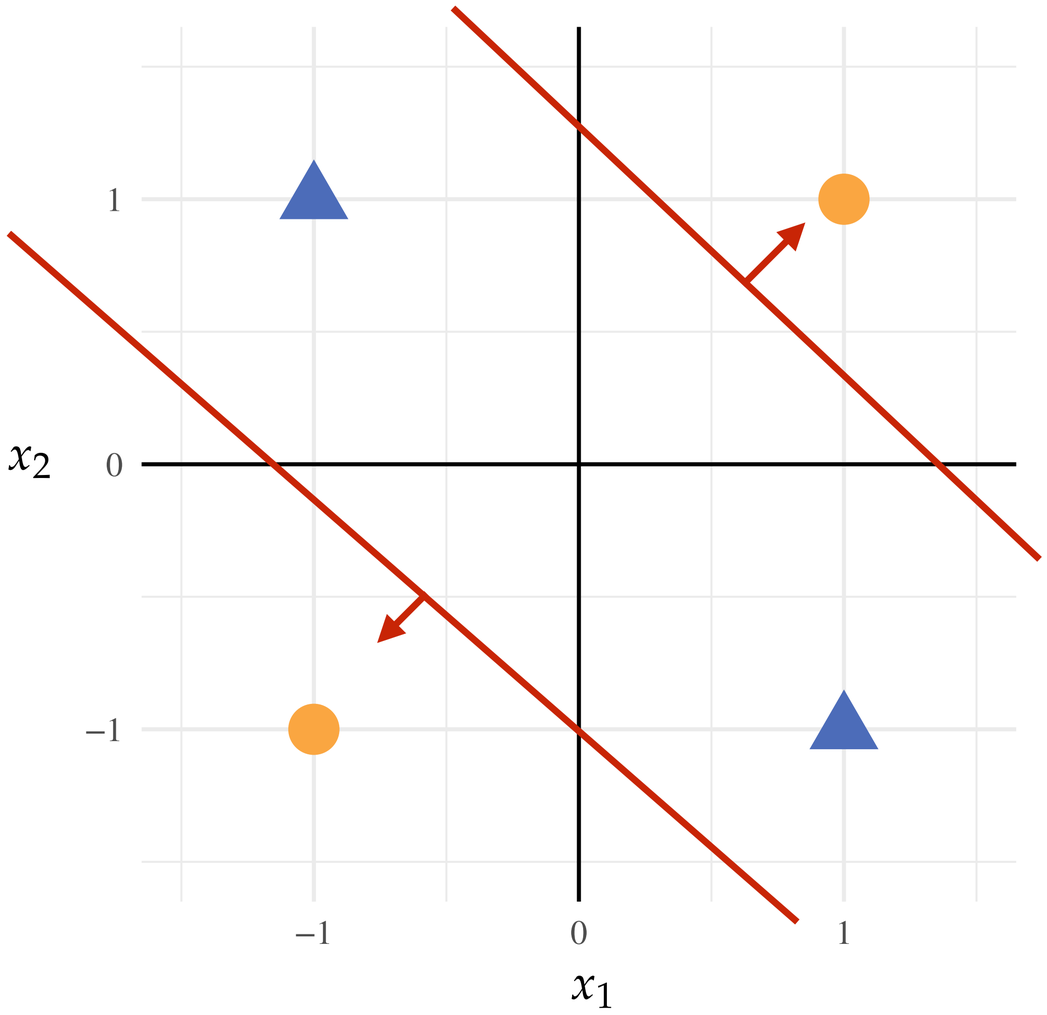


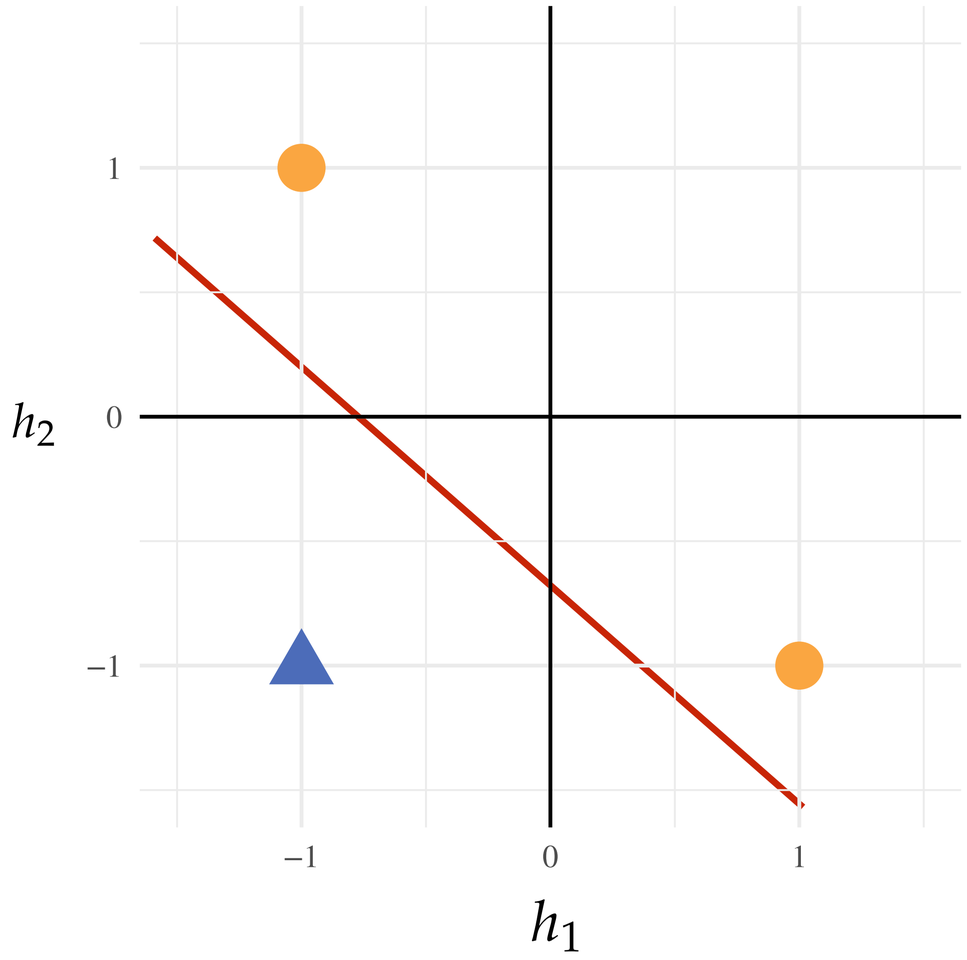
MULTI-LAYER PERCEPTRONS

XOR problem



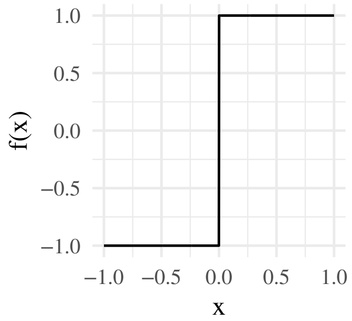
What if we use two classifiers?

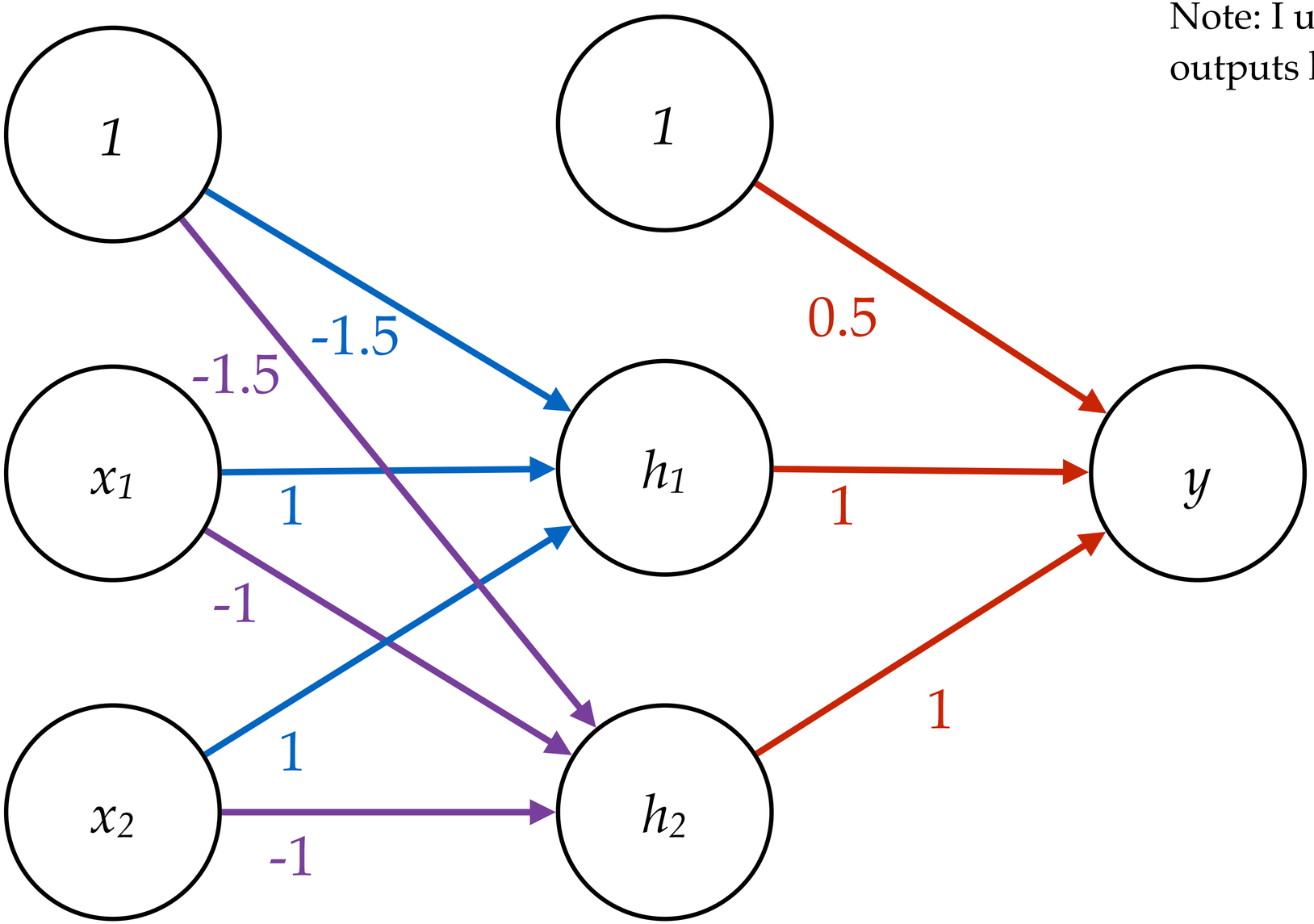




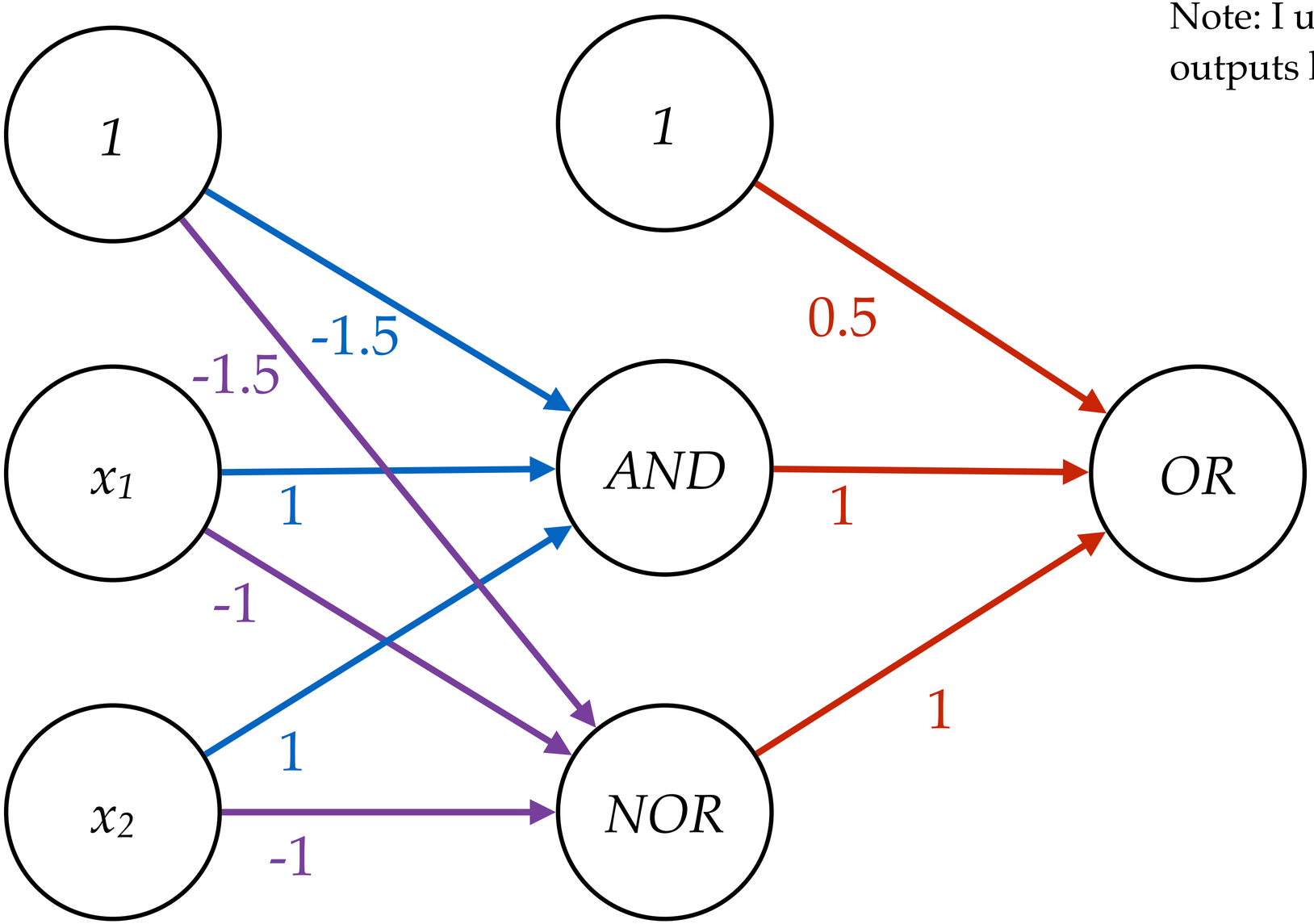
Use two classifiers and consider their outputs as the inputs to a third classifier

Note: I use {-1,+1} values as outputs here instead of {0,1}





Note: I use {-1,+1} values as outputs here instead of {0,1}



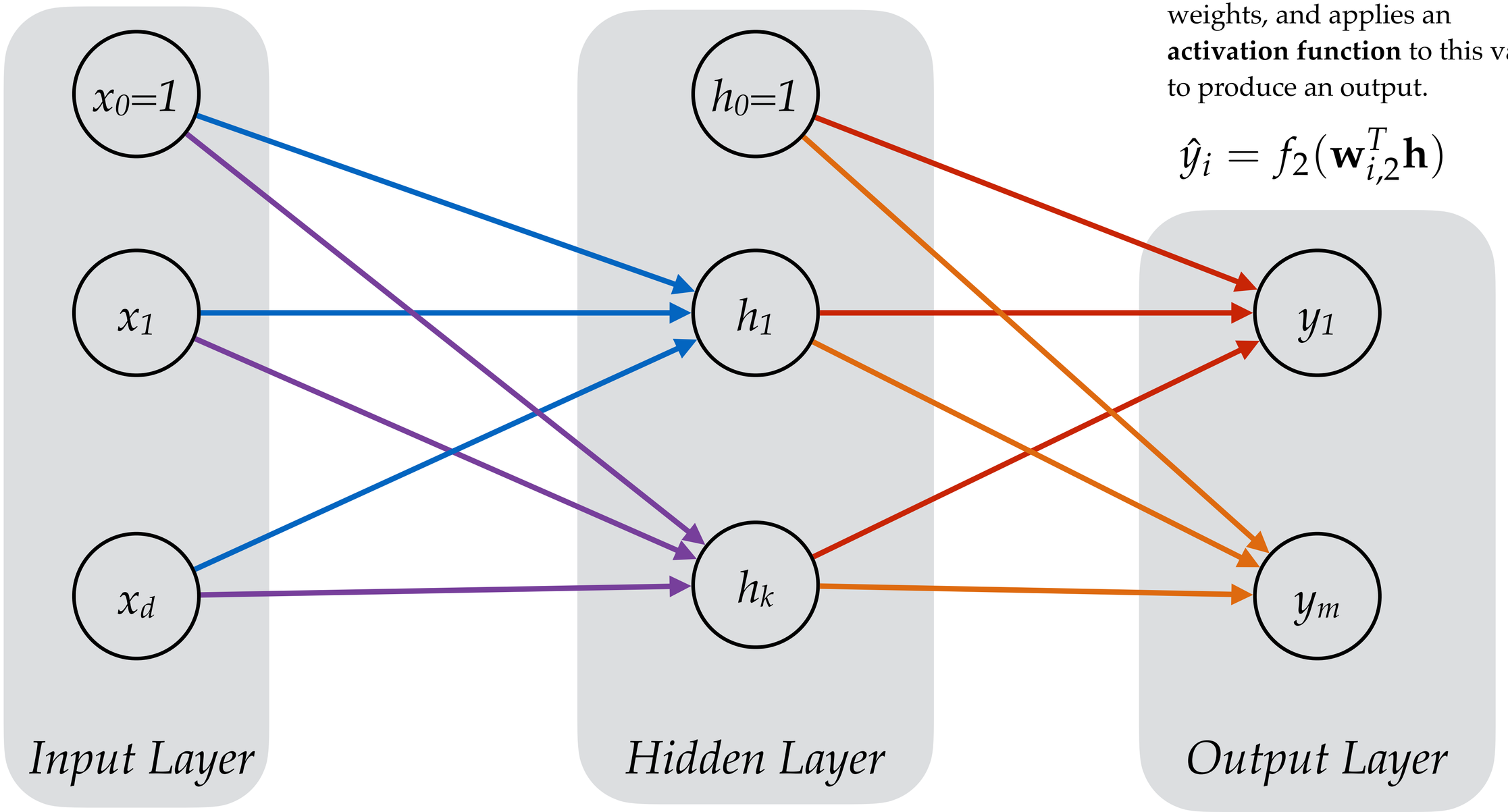
Intuition using logic gates

Terminology

$$h_i = f_1(\mathbf{w}_{i,1}^T \mathbf{x})$$

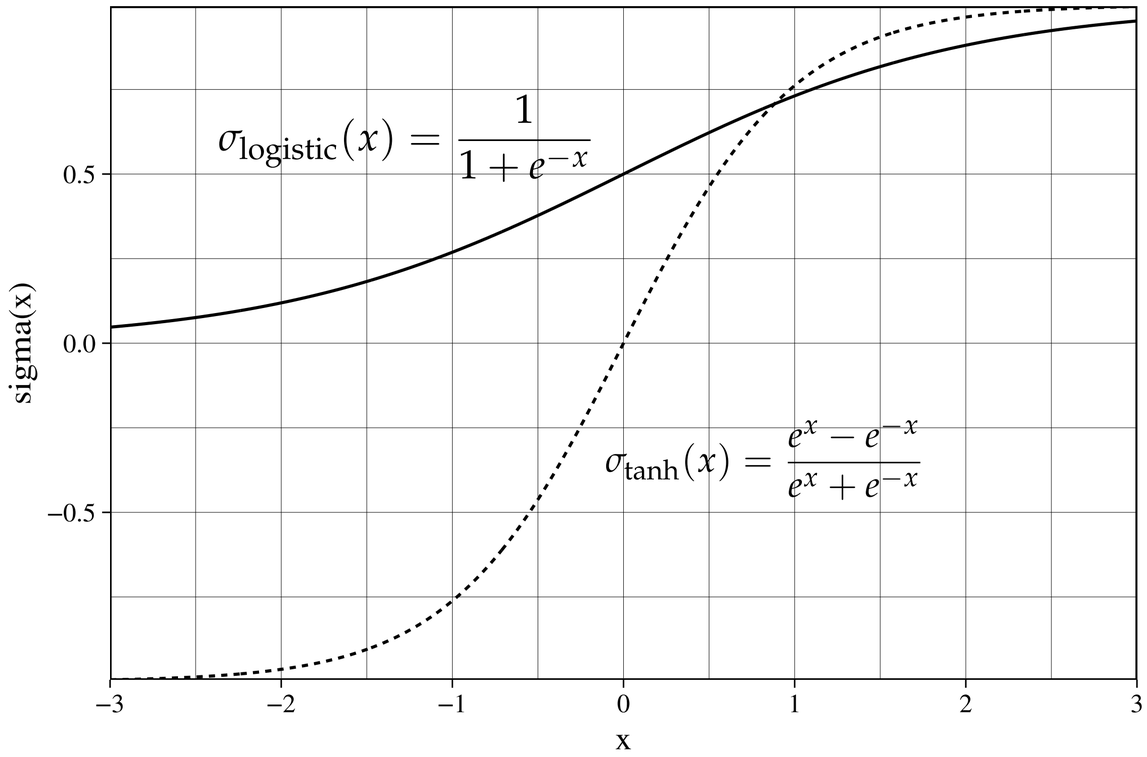
Each **neuron** takes the linear combination of the previous layer's output using its own weights, and applies an **activation function** to this value to produce an output.

$$\hat{y}_i = f_2(\mathbf{w}_{i,2}^T \mathbf{h})$$



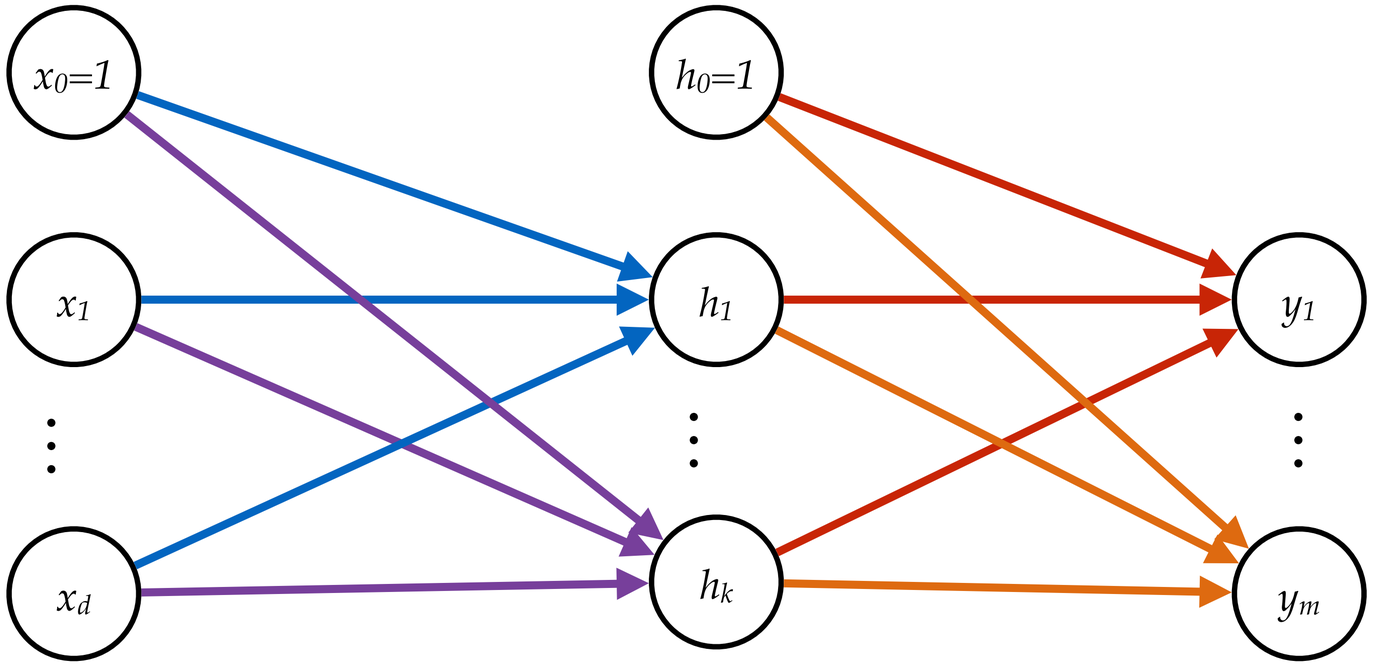
Multi-layer Perceptron

- Confusing naming: MLP as a general term
- Other activation functions:
  - Sigmoid (logistic)
  - Tanh
  - Modern alternatives: ReLU and its variants



Note: tanh is a rescaled logistic

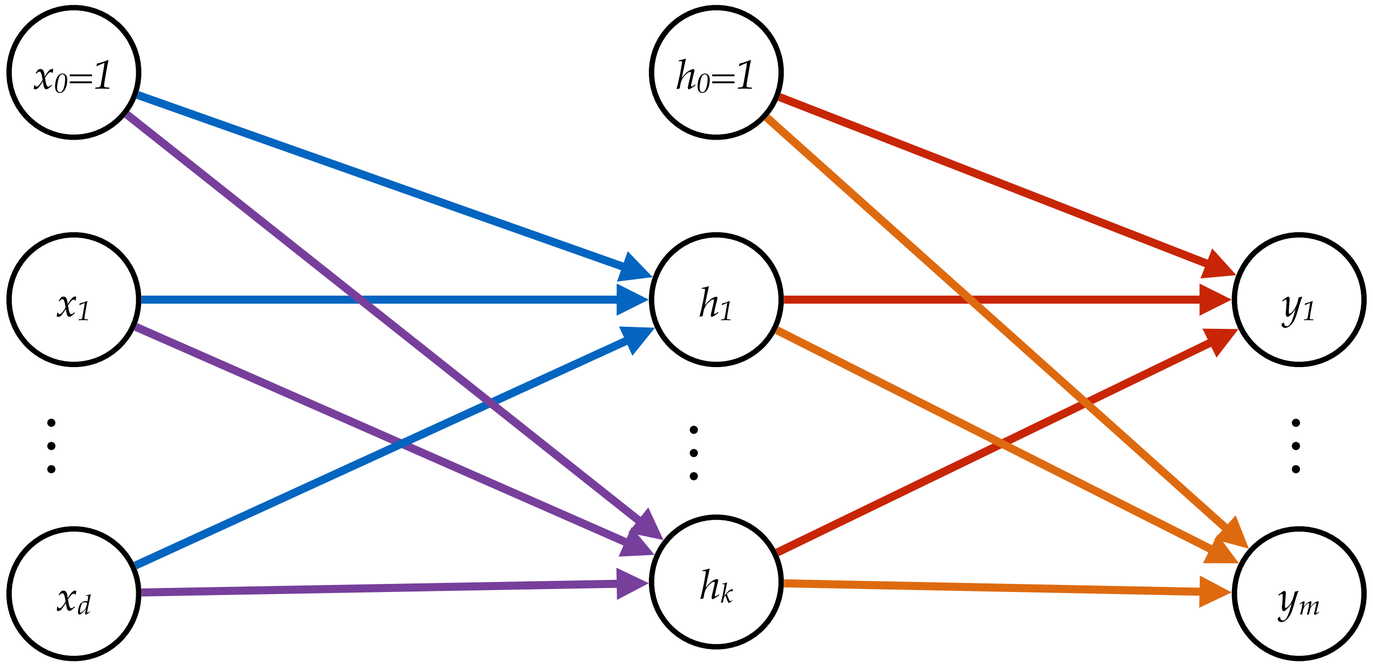
Multi-layer Perceptron



Can describe complex functions, but how do we

1. calculate predictions
2. learn the parameter weights?

Prediction: Forward Pass



$$\mathbf{h} = f_1(\mathbf{W}_1^T \mathbf{x}) \qquad \hat{\mathbf{y}} = f_2(\mathbf{W}_2^T \mathbf{h})$$

Given a model, we can do Prediction/inference in a *forward pass*

Objective function

$$\mathbf{x}$$

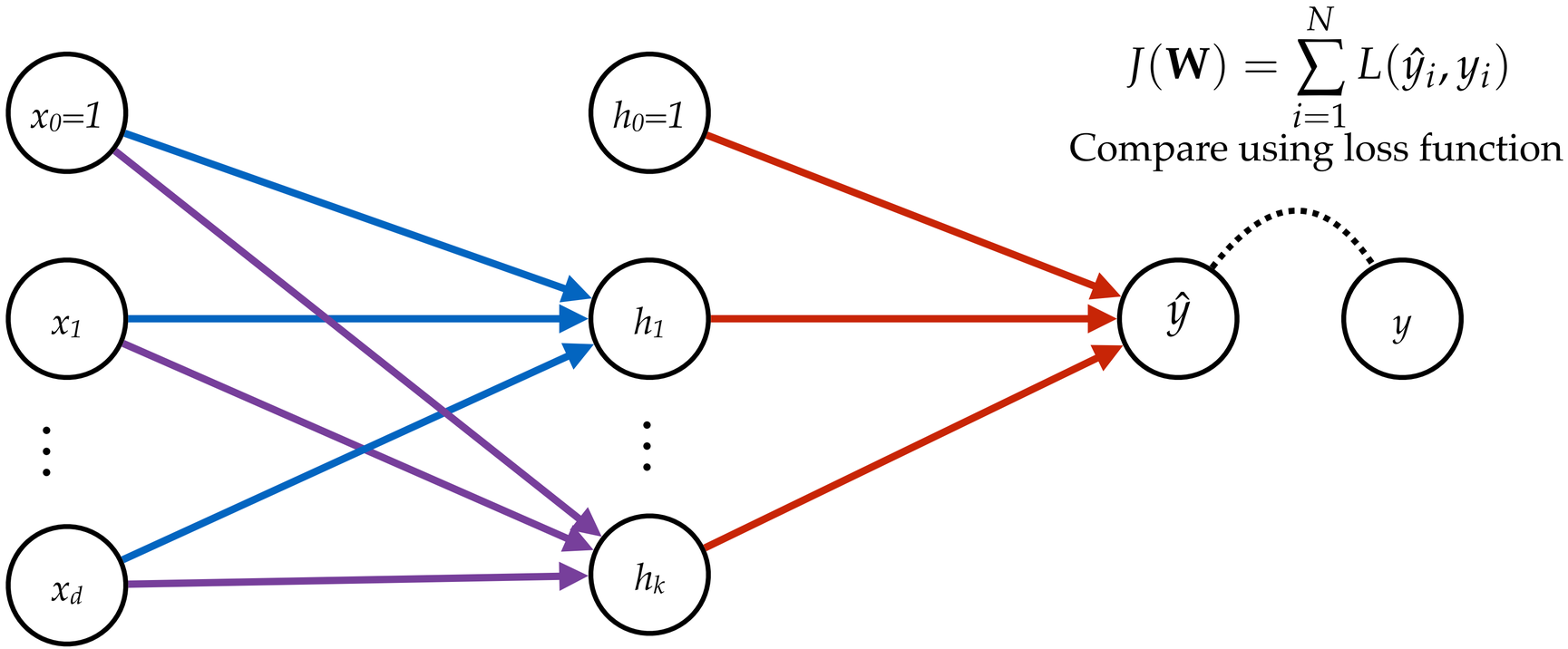
$$\mathbf{h} = f_1(\mathbf{W}_1^T \mathbf{x})$$

$$\hat{y} = f_2(\mathbf{W}_2^T \mathbf{h})$$

$$J(\mathbf{W}) = \sum_{i=1}^{N} L(\hat{y}_i, y_i)$$

$$J(\mathbf{W}) = \sum_{i=1}^{N} L(f_2(\mathbf{W}_2^T f_1(\mathbf{W}_1 \mathbf{x}_i)), y_i)$$

Why learning is hard



If the output is wrong, how should we update the model? Which part of the error should we attribute to the different weights?

Learning the MLP weights

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha_t \left. \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}^t}$$

- Remember the general setup: we are again going to use gradient descent. So we need the gradient
- Intuitively, what the gradient will do is propagate the error at the end of the network, back across the graph to tell us how to change the weights.
- This clever way of efficiently calculating the gradient is called “back propagation”

Backpropagation

$$J(\mathbf{W}_1, \mathbf{w}_2) = \sum_{i=1}^{N} L(f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1 \mathbf{x}_i)), y_i)$$

$$\begin{aligned}(\nabla_{\mathbf{w}_2} J(\mathbf{w}_2, \mathbf{W}_1))^T &= \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot [\nabla_{\mathbf{w}_2} f_2(\mathbf{w}_2^T f_1(\mathbf{W}_1^T \mathbf{x})))]^T \\ &= \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x})))}{\partial \mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x}))} \cdot [f_1(\mathbf{W}_1^T \mathbf{x})]^T\end{aligned}$$

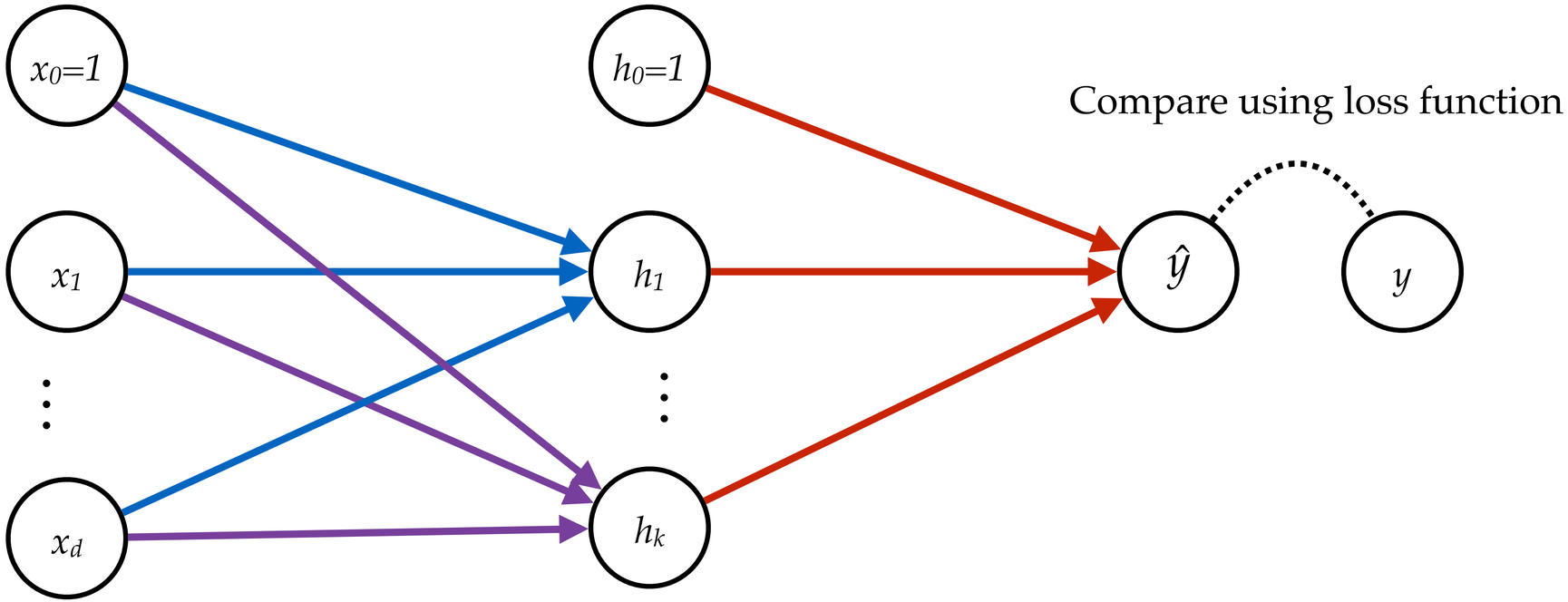
It is important you understand the intuition here: why this is called back propagation and how it relates to the gradient.

Backpropagation

$$(\nabla_{\mathbf{w}_2} J(\mathbf{w}_2, \mathbf{W}_1))^T = \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x}))}{\partial \mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x}))} \cdot [f_1(\mathbf{W}_1^T \mathbf{x})]^T$$

$$(\nabla_{\mathbf{w}_1} J(\mathbf{w}_2, \mathbf{W}_1))^T = \frac{\partial L(y, \hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial f_2(\mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x}))}{\partial \mathbf{w}_2^T f(\mathbf{W}_1^T \mathbf{x}))} \cdot \mathbf{w}_2^T \cdot diag(\frac{\partial f_1(\mathbf{W}_1^T \mathbf{x})}{\partial \mathbf{W}_1^T \mathbf{x}}) \cdot \begin{bmatrix} \mathbf{x}^T \\ \mathbf{0}^T \\ \dots \\ \mathbf{0}^T \end{bmatrix}$$

Backpropagation



Propagate the error at the end back through the network

Learning the MLP weights

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha_t \left. \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}^t}$$

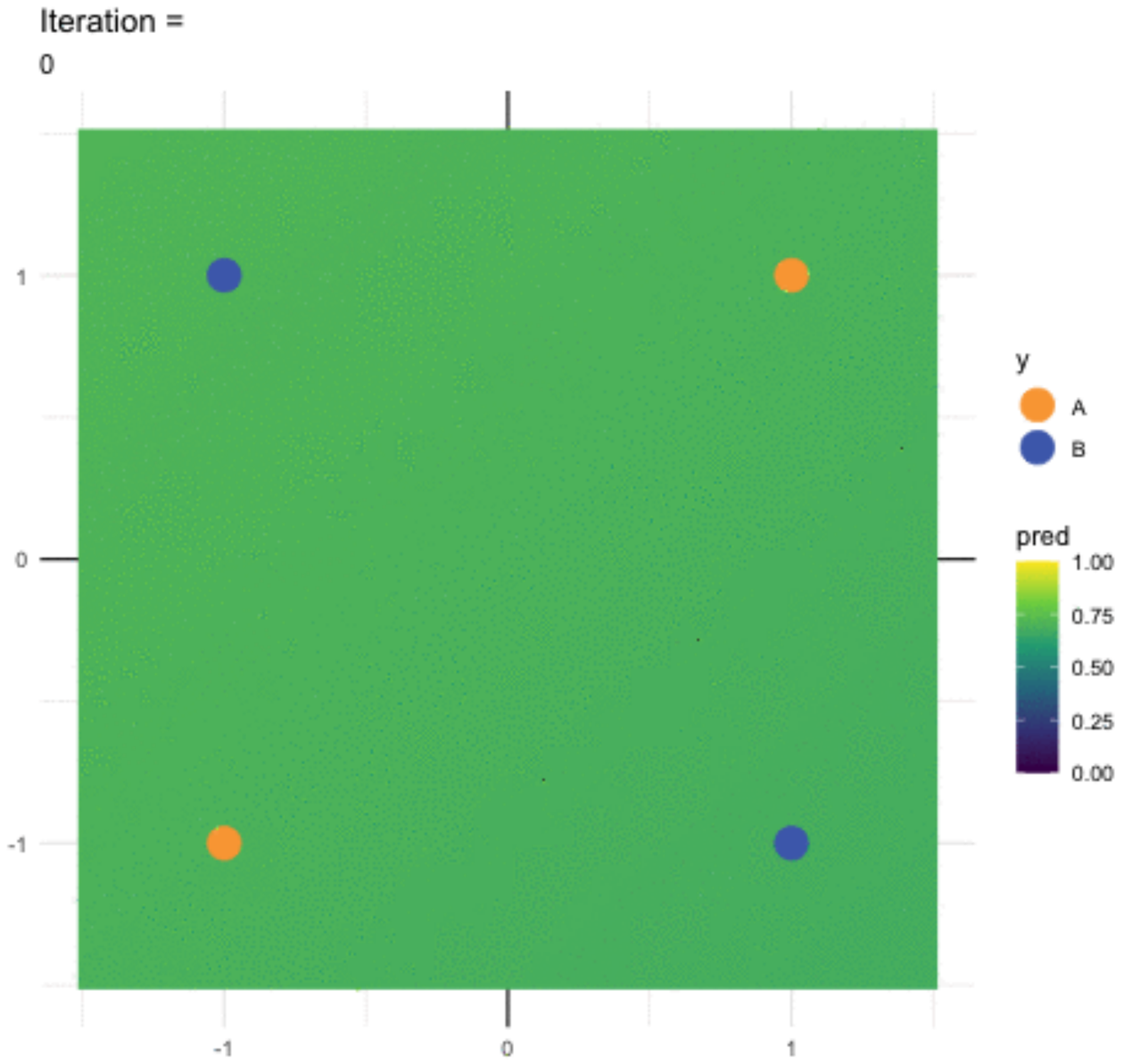
- Remember the general setup: we are again going to use gradient descent. So we need the gradient
- Intuitively, what the gradient will do is propagate the error at the end of the network, back across the graph to tell us how to change the weights.
- This clever way of efficiently calculating the gradient is called “back propagation”

Classifier Construction using Empirical Risk Minimisation

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

XOR example fit



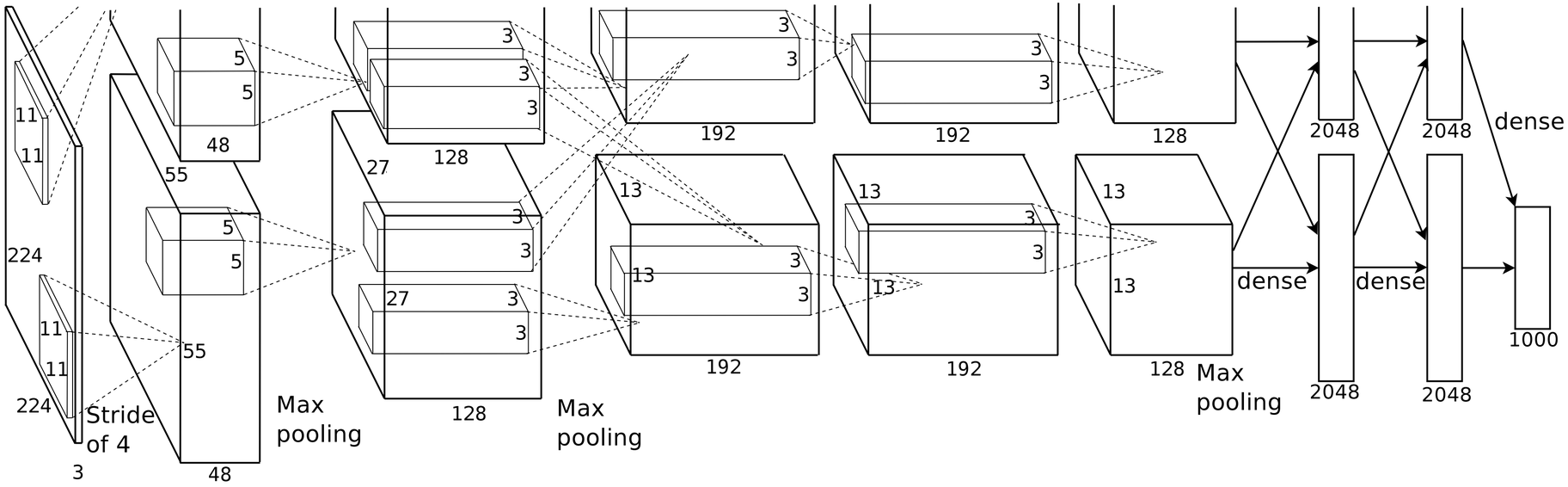
Many variations

- Activation functions and how to combine them
- How many hidden layers?
- How to connect different neurons, share parameters, etc.
- Initialization, optimization procedure, regularization, weight decay, …
- These combinations lead to the models that are used today:
  - Convolutional Neural Networks
  - Residual Networks

Challenges

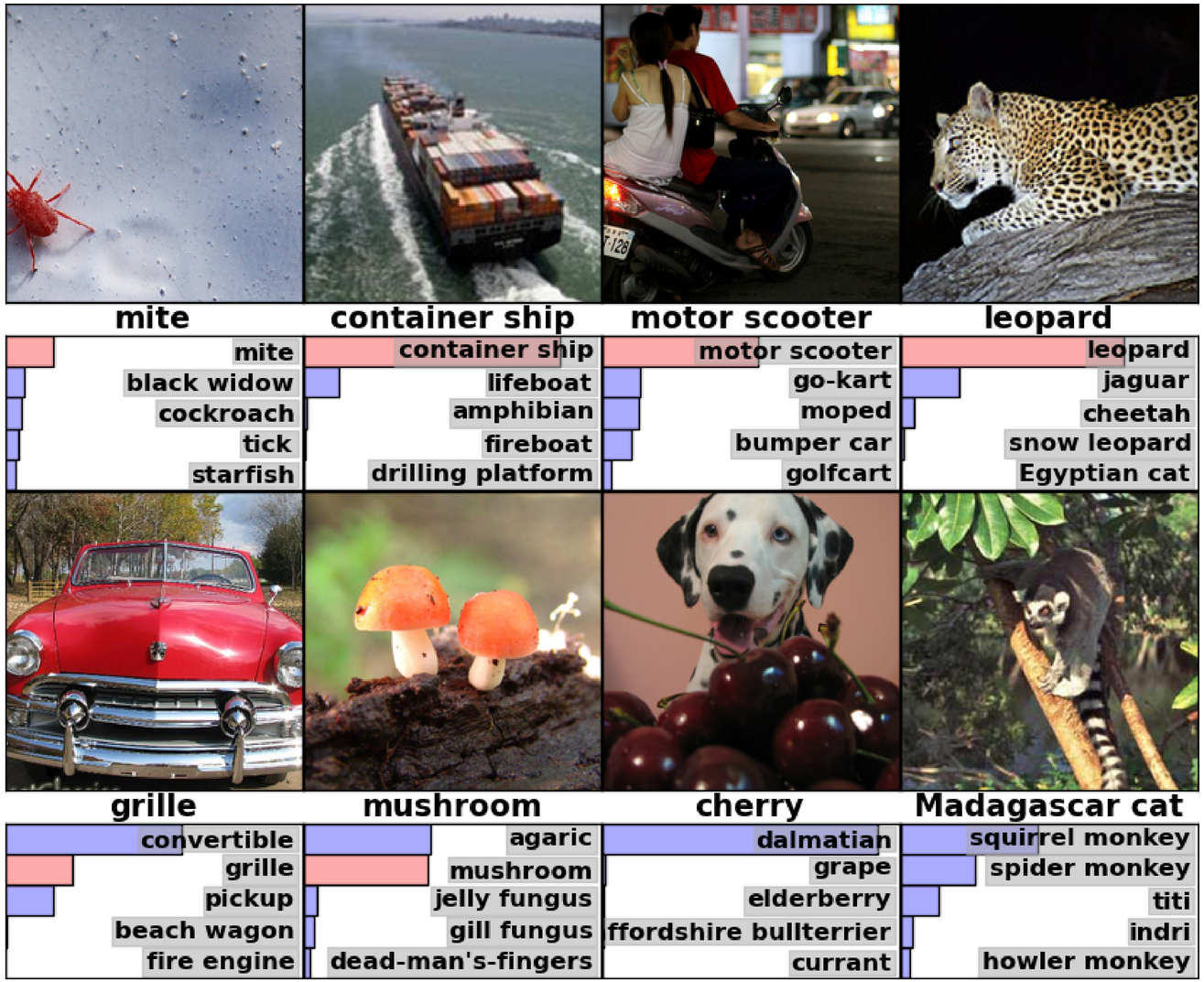
- Non-convex risk function: may get stuck in local optima, converge slowly, etc.
- Many architecture choices: which is optimal?
- Flexible model: risk of overfitting
- There is an *art* to getting networks to work well
- Difficult to interpret the model (black box?)
- With many parameters: computationally demanding (but GPUs, TPUs and other specialised hardware)

Example: ImageNet

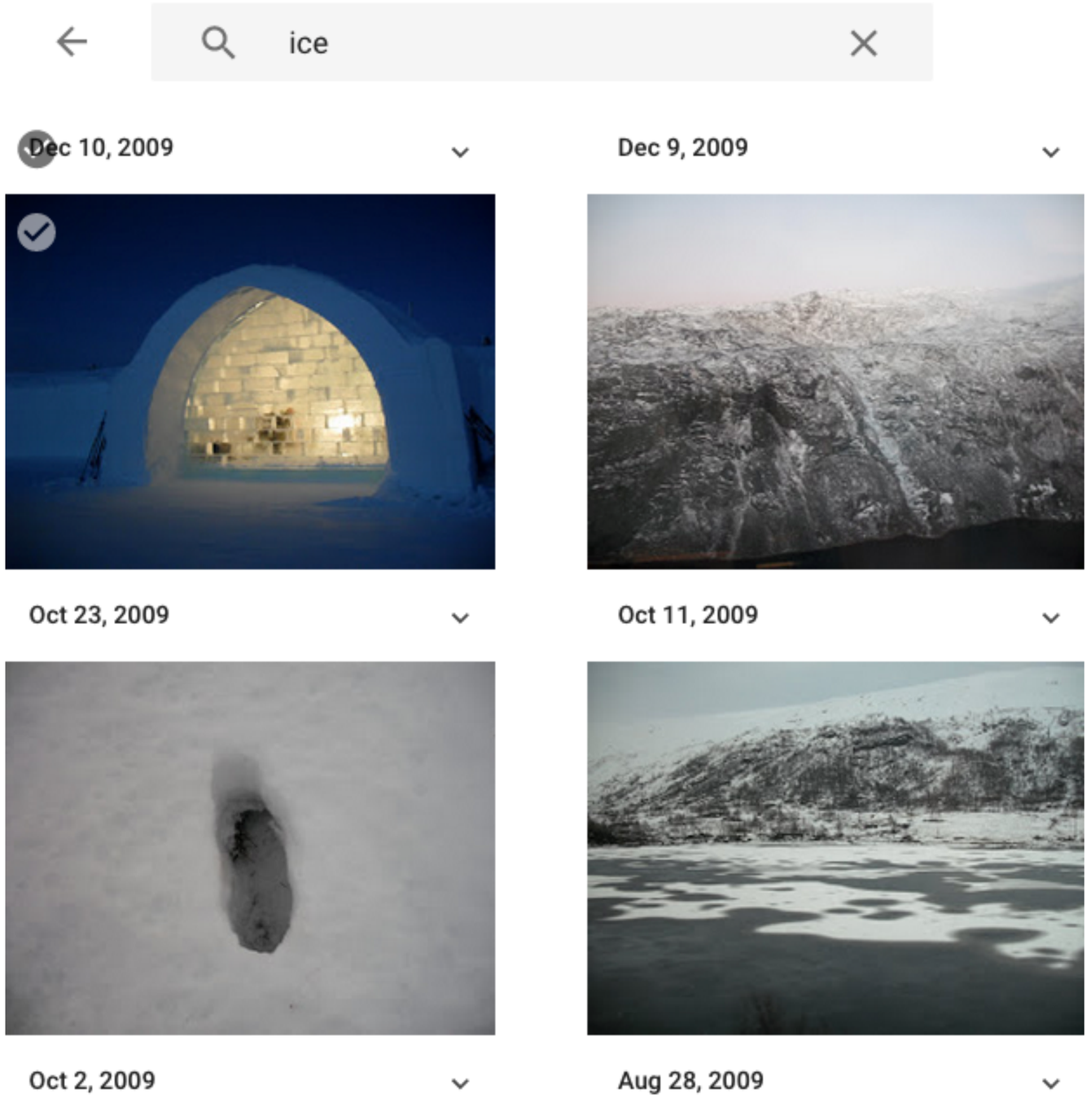


1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

Example: ImageNet

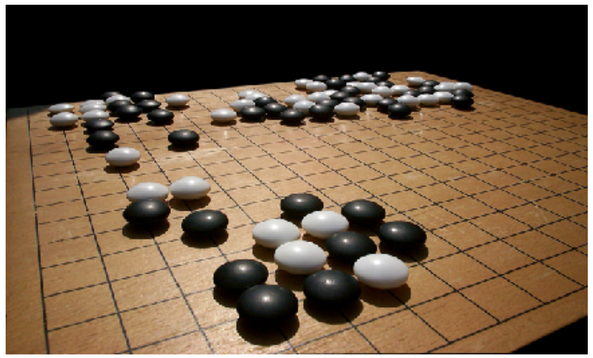


1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.



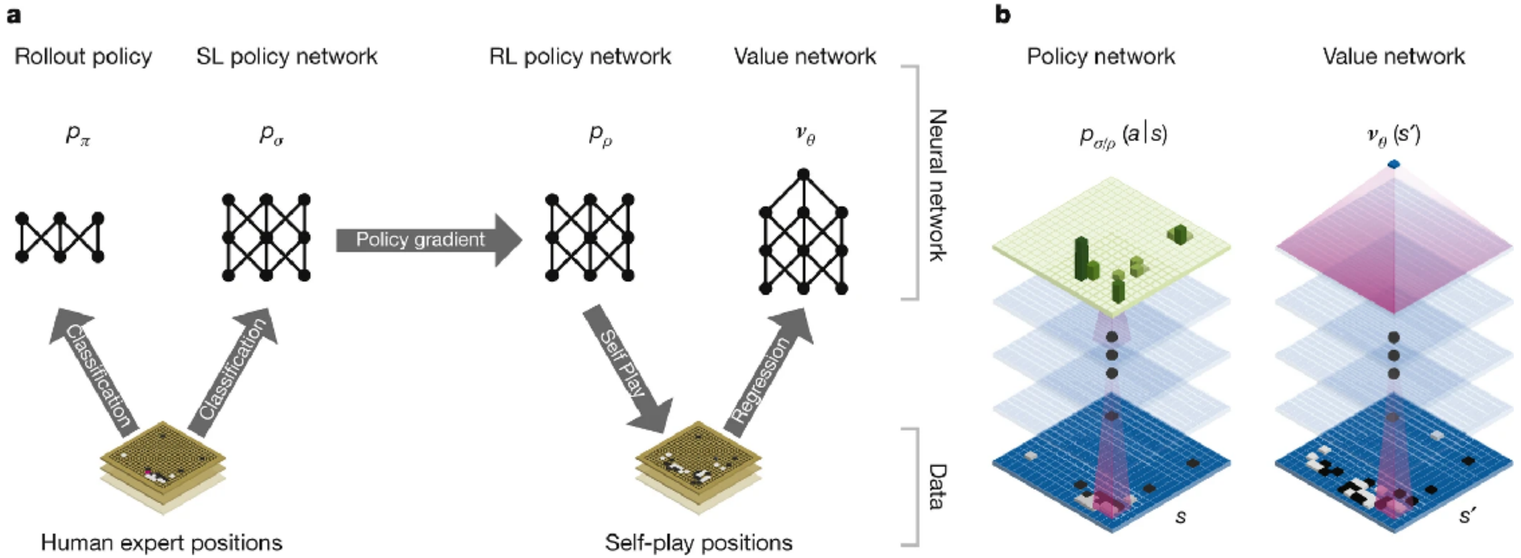
Google Photo search for "ice"

Example: Learning Moves in Go



Artificial Neural Network

Best Next Move



Nature, volume 529, pages 484–489 (2016)

DeepMind's Deep

Q-Network used to play
Atari's Breakout game

Advantages & Disadvantages

- ✅ Flexible model class
- ✅ Good empirical performance on many (structured) problems
- ✅ Can be easily adapted to different learning settings

- ❌ Computationally expensive
- ❌ Lots of hyper parameters
- ❌ Does not convergence to a unique optimum
- ❌ Optimizing them can be an art
- ❌ Hard to interpret

Recap

- Perceptrons are simple linear binary classifiers
- Multi-layer perceptrons are connected architectures of perceptron-like nodes, inspired by a simplified model of the brain
- They are trained using (stochastic) gradient descent, efficiently calculating the gradient using back propagation