UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

MASTER IN COMPUTER SCIENCE

Pattern Recognition

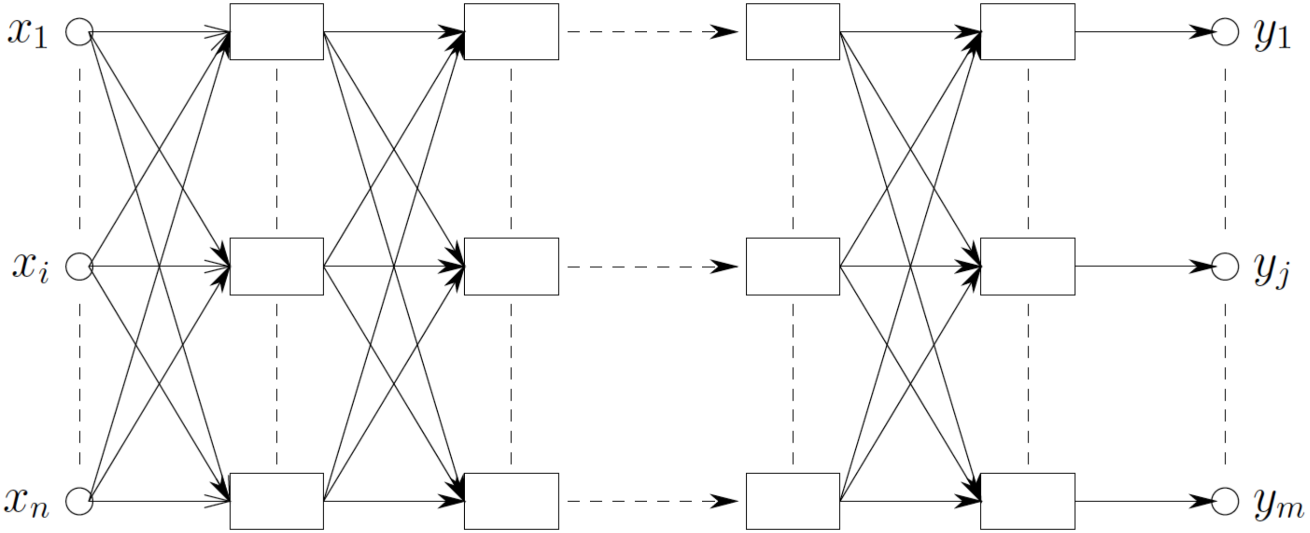
Lecture 5 : Artificial Neural Networks

Dr. Andreas Fischer

andreas.fischer@unifr.ch

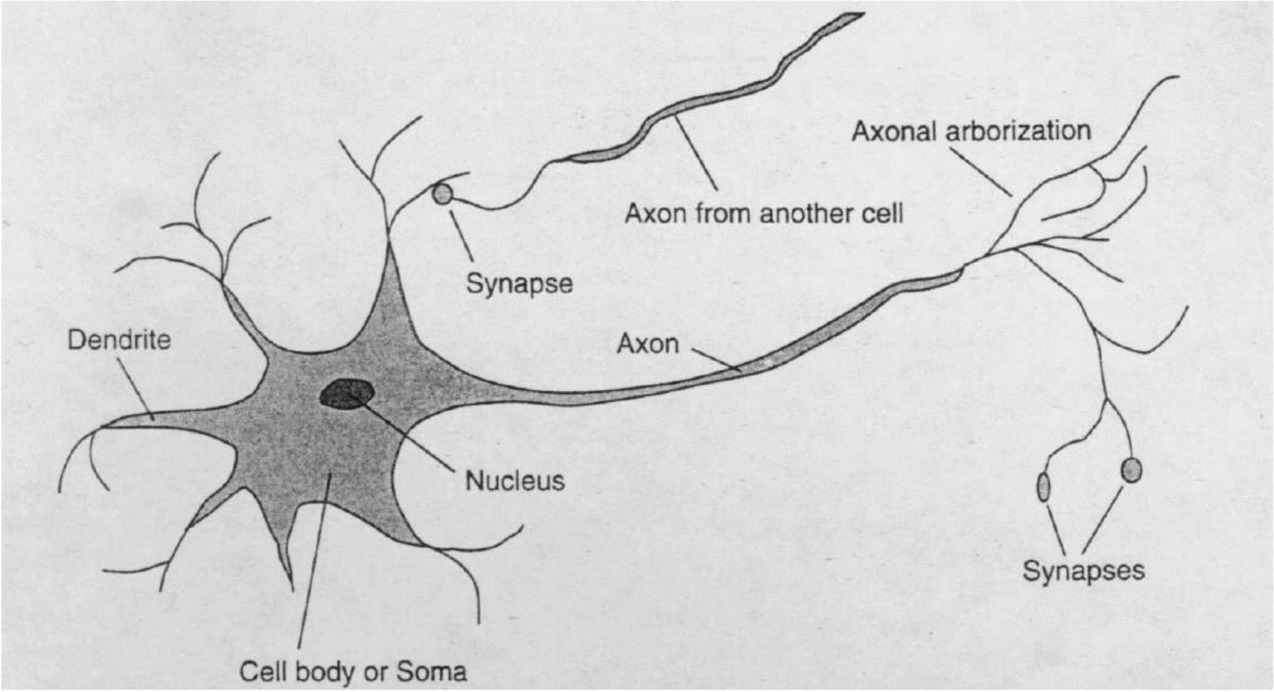
Artificial Neural Networks (ANN)

- Discriminative or generative classifier
- Statistical representation: X = R^n
- *Universal approximator:* posterior p(C|x) or likelihood p(x|C)



Biological Brain

- Functionality inspired by human brain but not an attempt to replicate it.
- The human brain has ~10^{11} neurons linked with ~10^{15} synapses.
- Electrochemical signals are transmitted between linked neurons, the signal is transmitted if the action potential exceeds a threshold.
- Learning: synapses and their transmission properties change over time.



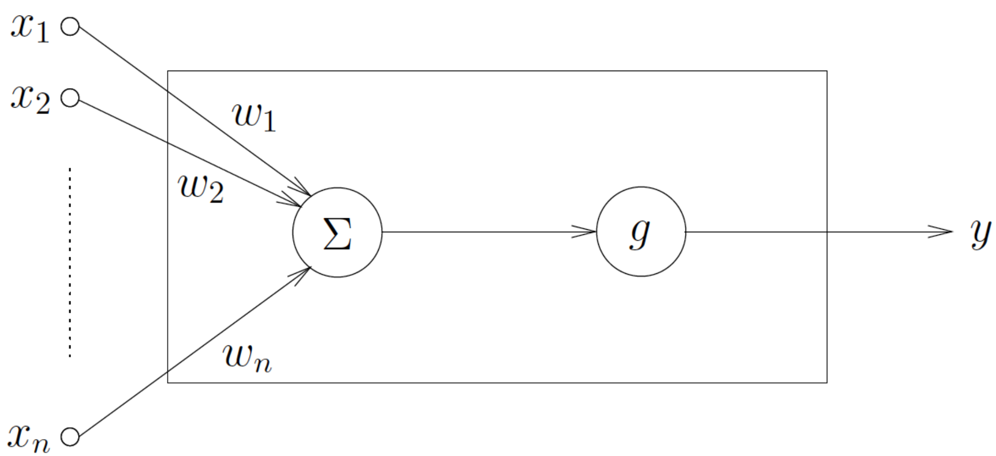
Perceptron

Perceptron

- The perceptron is a simple mathematical model of the neuron, also called Adaline (adaptive linear element) or LTU (linear threshold unit).
- It calculates the function:

$$y = g(w'x) = g\left(\sum_{i=1}^{n} w_i x_i\right)$$

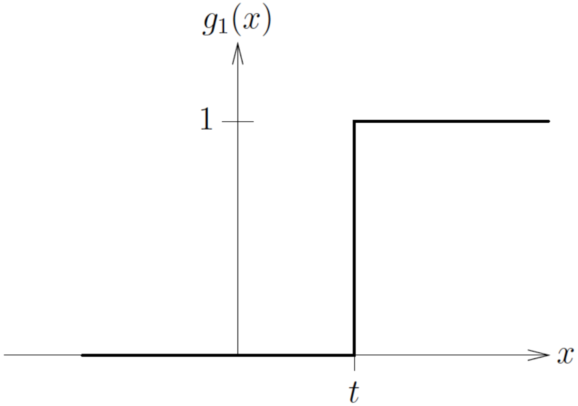
where x_i are real-valued features, w_i are real-valued weights, and g(.) is a threshold function.



Threshold Function

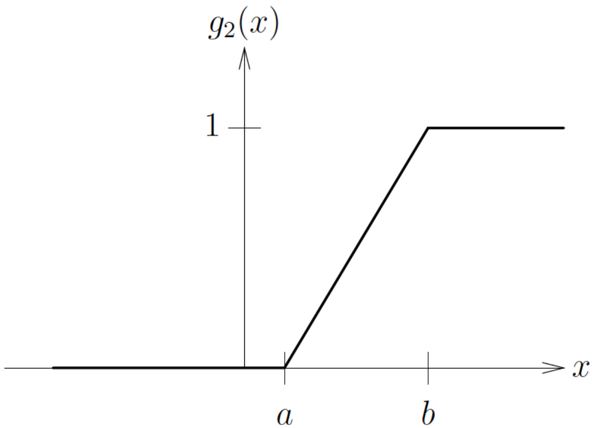
- Hard threshold:

$$g_1(x) = \begin{cases} 0, & x < t \\ 1, & x \geq t \end{cases}$$



- Linear threshold:

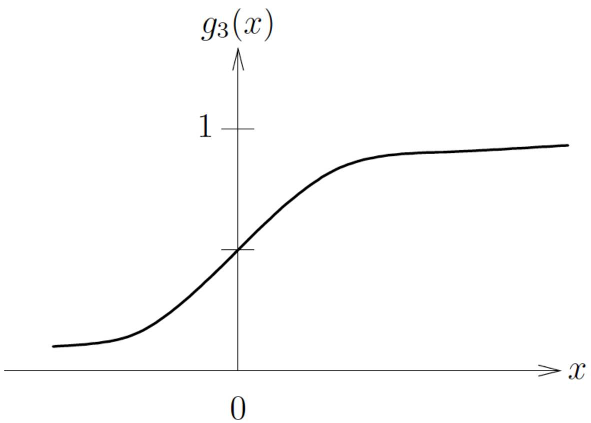
$$g_2(x) = \begin{cases} 1 & , x > b \\ \dfrac{1}{b-a}(x-a) & , a \leq x \leq b \\ 0 & , x < a \end{cases}$$



- Sigmoid function (logistic function), c > 0:

$$g_3(x) = \frac{1}{1+\exp(-c \cdot x)}$$

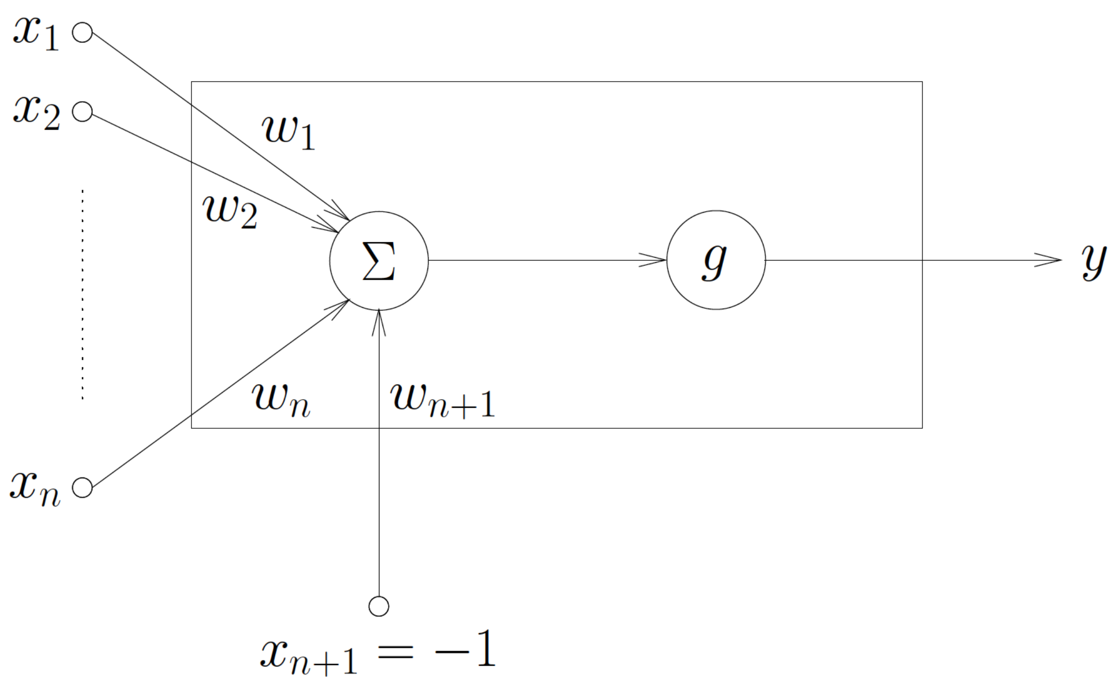
$$\frac{d}{dx} g_3(x) = g_3(1 - g_3(x))$$



Bias

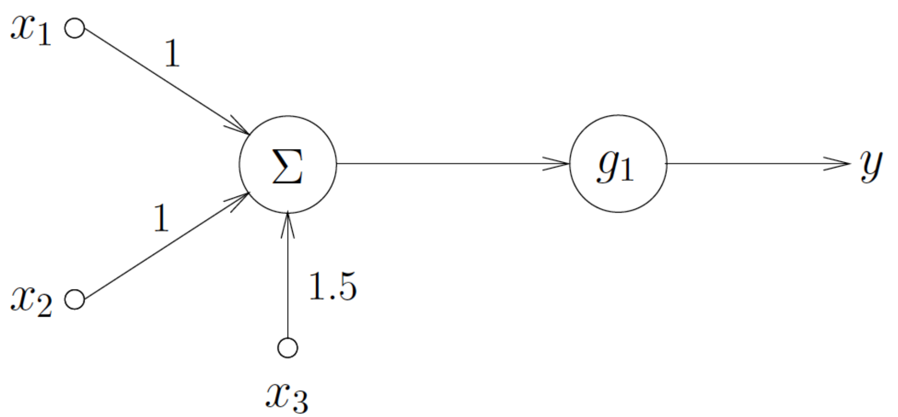
- Threshold value is set to 0 by introducing a *bias* x_{n+1} with constant value $x_{n+1} = -1$ and weight $w_{n+1} = t$:

$$y = \begin{cases} 0 , & w_1 x_1 + \cdots + w_n x_n - w_{n+1} < 0 \\ 1 , & \text{otherwise} \end{cases}$$



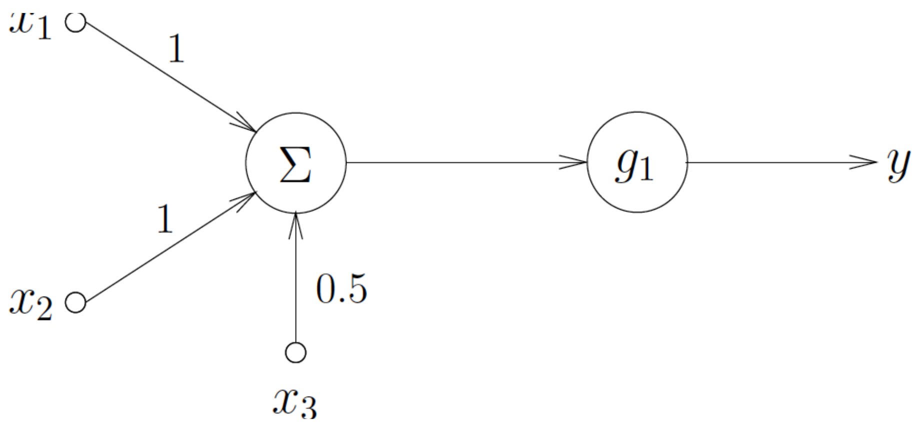
Example

AND:



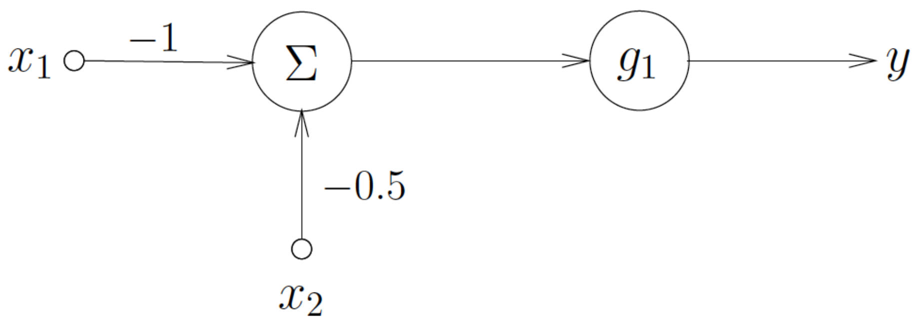
$y \equiv x_1$ AND x_2

OR:



$y \equiv x_1$ OR x_2

NOT:



$y \equiv$ NOT x_1

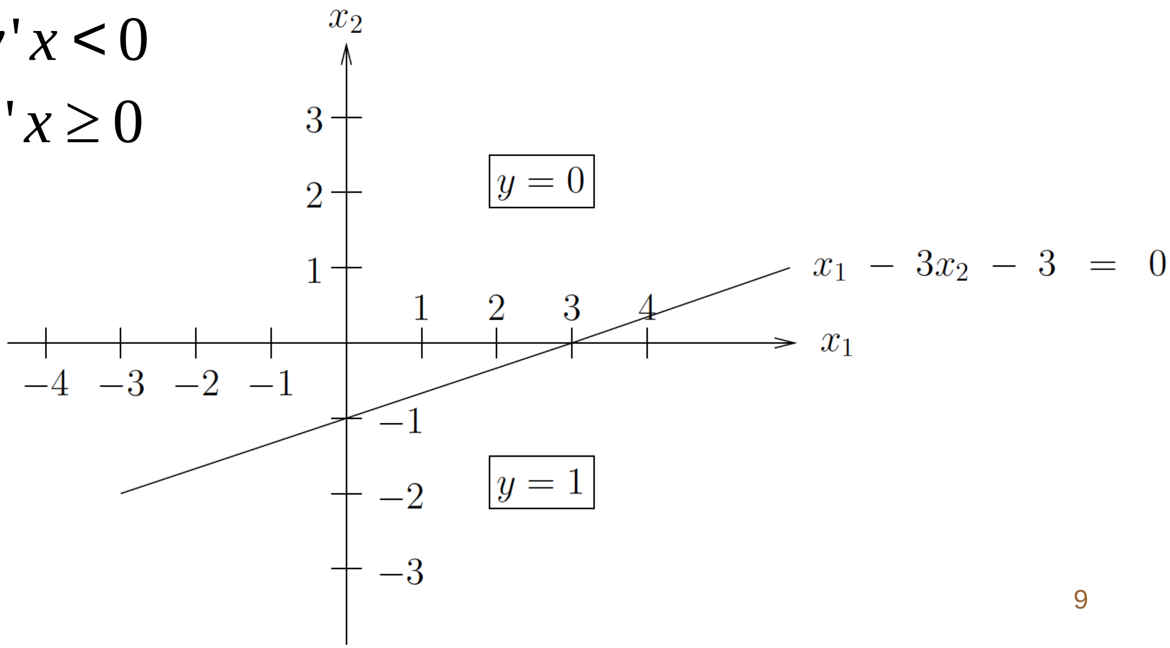
Perceptron Classifier

- Perceptron can be used to separate two classes with a hyperplane, the weights can be learned automatically.
- Training set S = $\{(x_1,y_1),\ldots,(x_N,y_N)\}$:

$$y_i = \begin{cases} 0 \Leftrightarrow x_i \in C_1 \\ 1 \Leftrightarrow x_i \in C_2 \end{cases}$$

- Classification with weights $w_1,\ldots,w_{n+1}$ (including bias):

$$y = \begin{cases} 0 \Leftrightarrow w'x < 0 \\ 1 \Leftrightarrow w'x \geq 0 \end{cases}$$



Perceptron Algorithm

- Solution is not unique but it can be shown that a solution is guaranteed if the classes are linearly separable.
- Learning rate typically c ∈ [0.1,1]
- The algorithm shows the *online* mode, where the weight vector is updated after each sample. In the *offline* or *batch* mode, the mean weight vector is computed after several samples have been processed.

Require: learning samples $(x_1, y_1), \ldots, (x_N, y_N)$, learning rate c

Ensure: weights w with $g(w'x_i) = y_i$

randomly initialize w

while $g(w'x_i) \neq y_i$ for some i **do**

    for all x_i **do**

        $y \leftarrow g(w'x_i)$

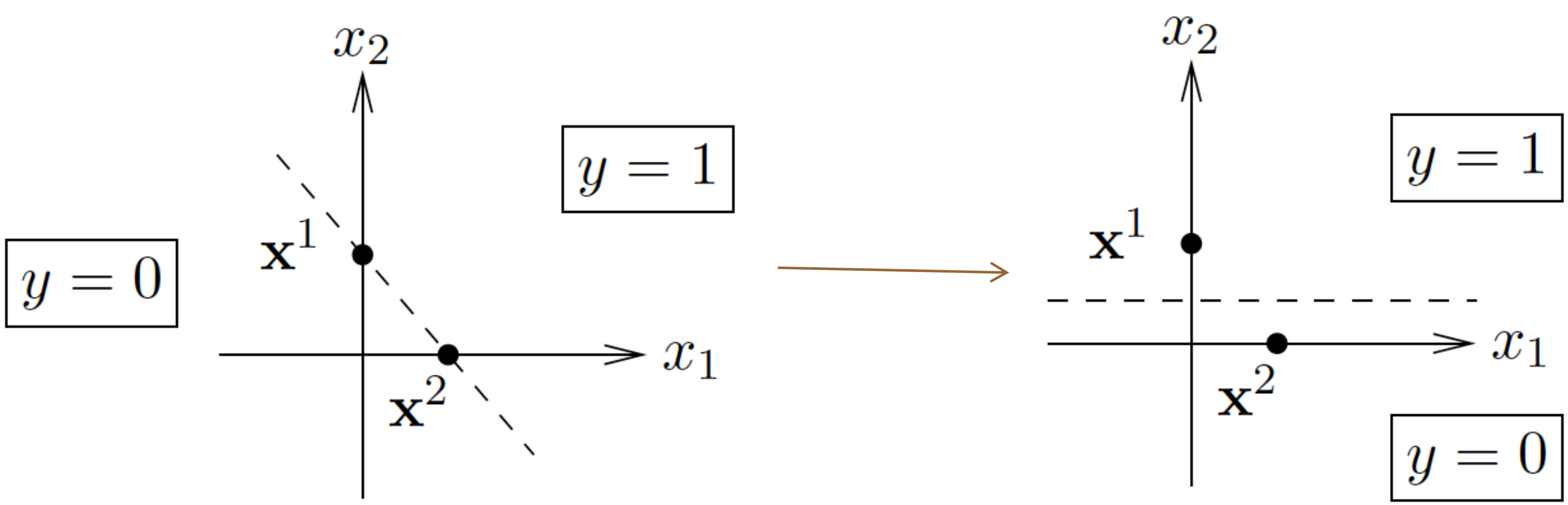
        $w \leftarrow w + c(y_i - y)x_i$

    end for

end while

Example 1

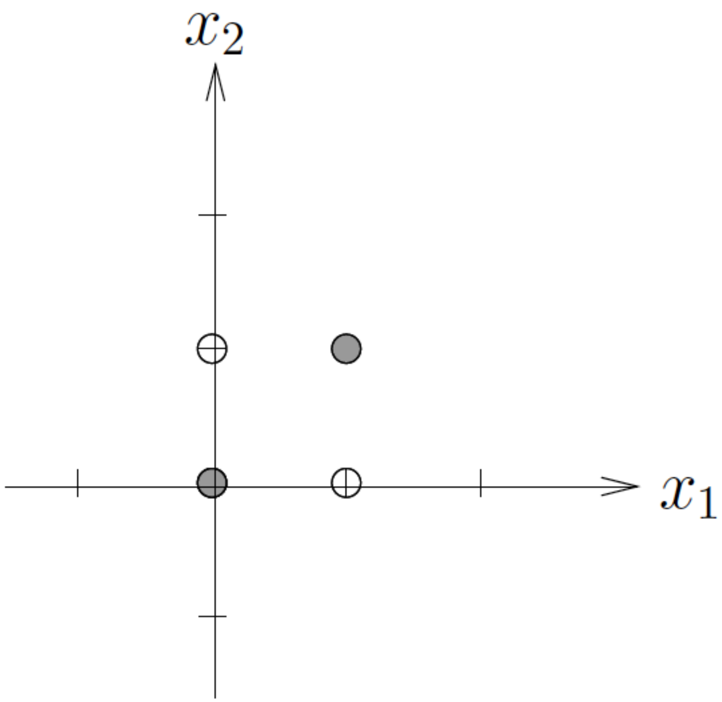
- Input: (x_1 = (0,1,-1), y_1 = 1), (x_2 = (1,0,-1), y_2 = 0), c = 1
- Algorithm:
  1. w_0 = (1,1,1)
  2. w_0'x_1 = 0, y = g(0) = 1, OK
  3. w_0'x_2 = 0, y = g(0) = 1, NOK
     $w_1 = w_0 + (0\text{-}1)x_2 = w_0 - x_2$ = (0,1,2)
  4. w_1'x_1 = -1, y = g(-1) = 0, NOK
     $w_2 = w_1 + (1\text{-}0)x_1 = w_1 + x_1$ = (0,2,1)
  5. g(w_2'x_1) = 1 and g(w_2'x_2) = 0, OK, terminate
- Result: hyperplane $2x_2 - 1 = 0$



Example 2

- Linear classifier, cannot solve XOR.

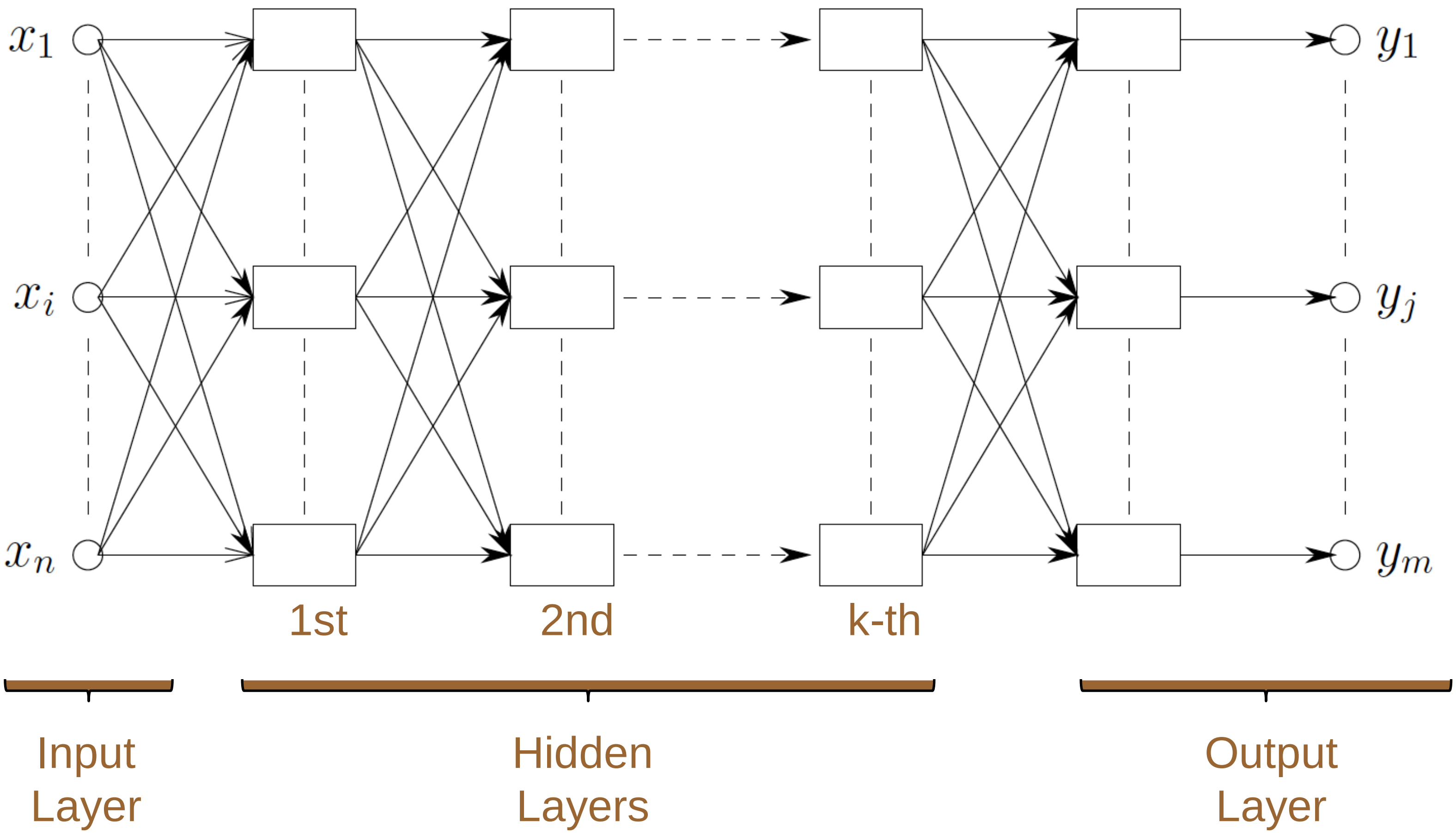
XOR



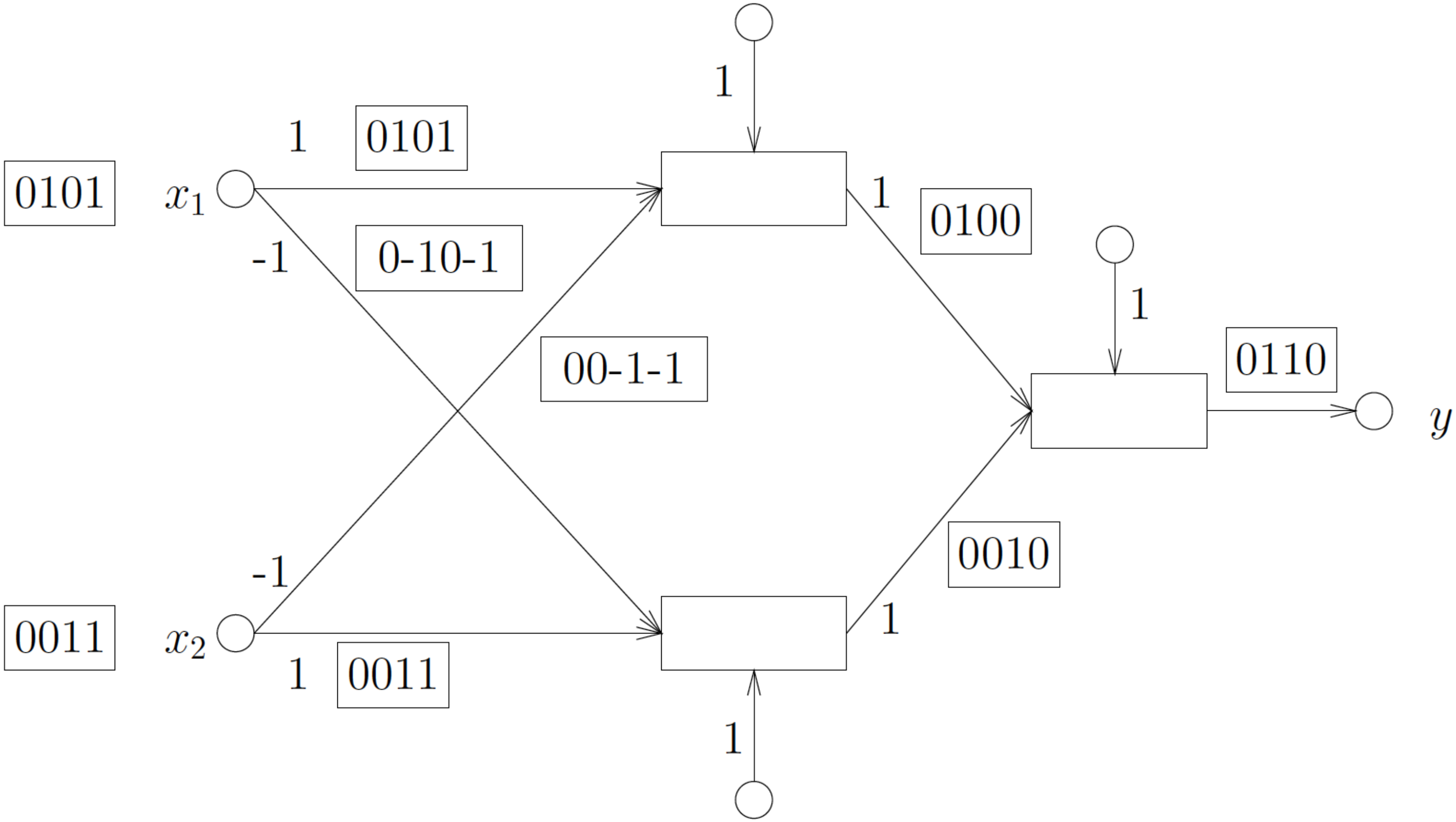
Multilayer Perceptron

Multilayer Perceptron (MLP)

- Multilayer feedforward neural network: multiple hidden layers, information is processed in forward direction.
- Note that the bias of each individual neuron is not shown explicitly.



Example



$$y = \mathrm{XOR}(x_1, x_2)$$

Backpropagation Algorithm

- Output y = $(y_1,\ldots,y_m)$ in R^m.
- Differentiable threshold function needed, often sigmoid function $g_3(x)$.
- The algorithm shows the online mode (update after each sample).

Require: learning samples $(x_1, y_1), \ldots, (x_N, y_N)$, learning rate c
Ensure: trained weights w

randomly initialize w
repeat
  for all x_i **do**
    compute output y
    error $e = y_i - y = (e_1, \ldots, e_m)$
    for all neurons i in the output layer **do**
      compute new weights $\hat{w} = update_{out}(w, i, e, c)$
    end for
    for all hidden layers h from the last to the first **do**
      for all neurons j in the hidden layer h **do**
        compute new weights $\hat{w} = update_{hidden}(w, j, c)$
      end for
    end for
    $w = \hat{w}$
  end for
until termination criterion is met

Weight Update

- The error is backpropagated via derivative g’ of the threshold function.

Require: weights w, output neuron i, error e, learning rate c
Ensure: updated weights $\hat{w}$

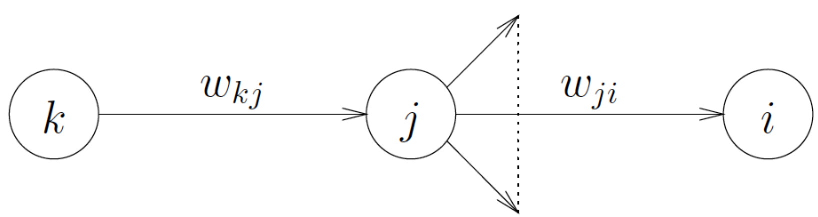
$\delta_i = g'(in(i)) \cdot e_i$
for all predecessors j of neuron i **do**
    $\hat{w}_{ji} = w_{ji} + c \cdot out(j) \cdot \delta_i$
end for

Require: weights w, hidden neuron j, learning rate c
Ensure: updated weights $\hat{w}$

$\delta_j = g'(in(j)) \cdot \sum_i w_{ji}\delta_i$
for all predecessors k of neuron j **do**
    $\hat{w}_{kj} = w_{kj} + c \cdot out(k) \cdot \delta_j$
end for

$$in(i) = \sum_j w_{ji} out(j)$$

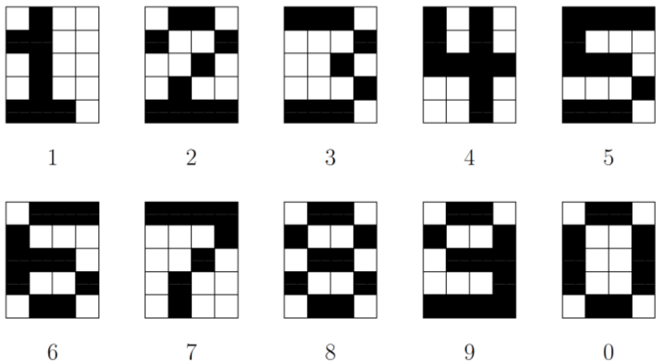
$$out(j) = g(in(j))$$

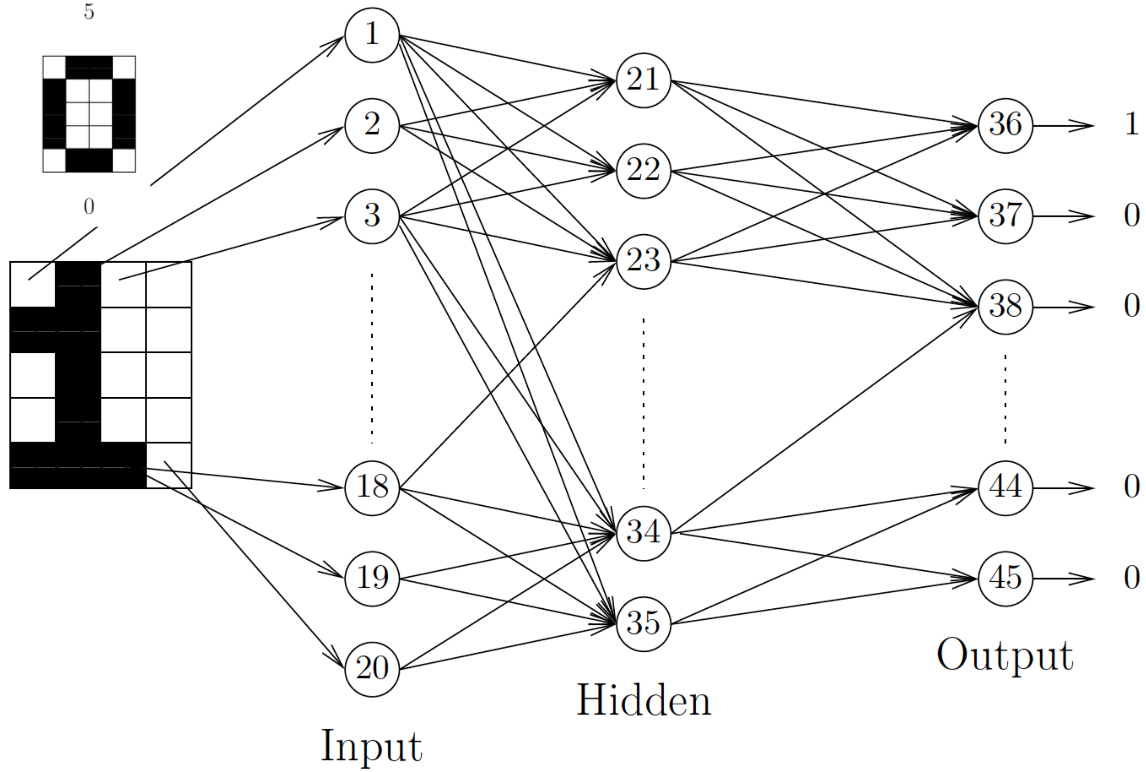


Input Layer/ Hidden Layer | Hidden Layer | Output Layer/ Hidden Layer

Example

- Digit recognition, each input corresponds to a binary pixel value.
- Output with *1-of-n* encoding, (1,0,…,0) for “1”, (0,1,0,…,0) for “2”, etc.
- Digit with the maximum output value is chosen. Often, a *softmax* normalization is used for the outputs such that they sum up to 1: $\hat{y}_i = \frac{\exp(y_i)}{\sum_{j=1}^{m} \exp(y_m)}$



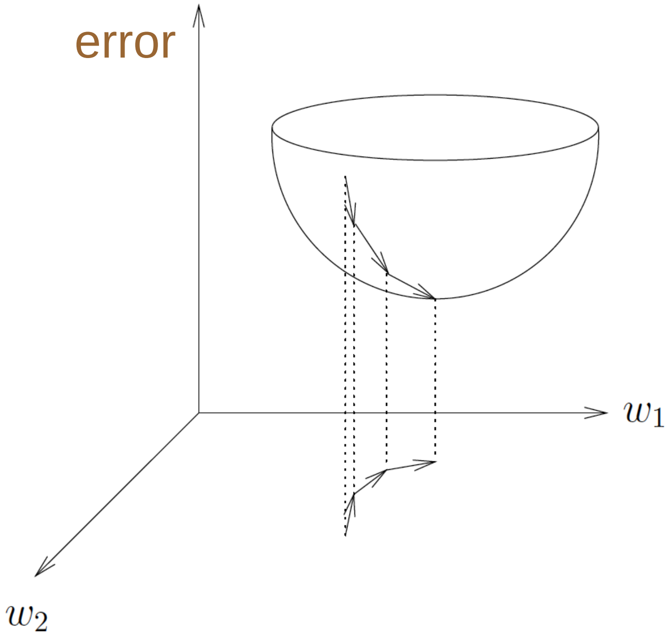


Universal Approximator

- It can be shown that MLP are *universal approximators*, that is they are able to approximate arbitrary functions from R^n to R^m.
- For classification, non-linear posterior functions $p(C|x)$ or likelihood functions $p(x|C)$ are particularly interesting.
- In practice, several parameters have to be selected experimentally during cross-validation:
  - Number of hidden layers: often one layer is sufficient.
  - Number of neurons per hidden layer (e.g. 50, 100, …): as a rule of thumb, N / 10 weights w_{ij} are reasonable for N learning samples.
  - Learning rate c (e.g. 0.1, 0.2, …, 1.0).
- Result depends on random initialization of the weights and the order of the training samples, which should be randomized as well.
  - Several random initializations are usually tested during cross-validation.

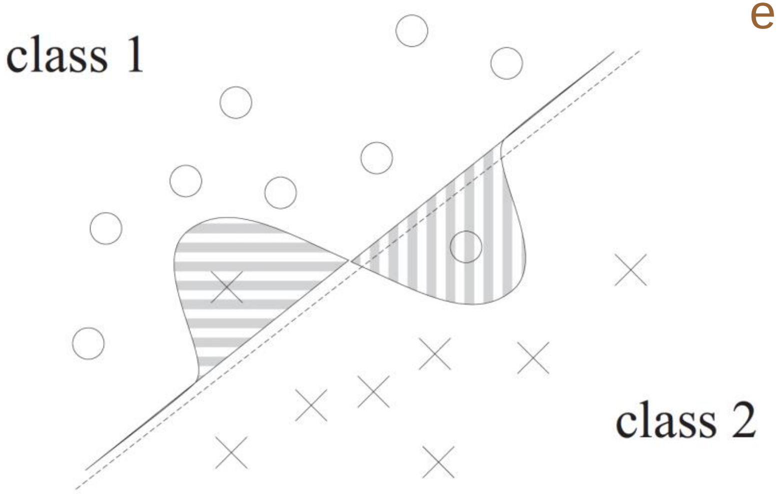
Local Optima

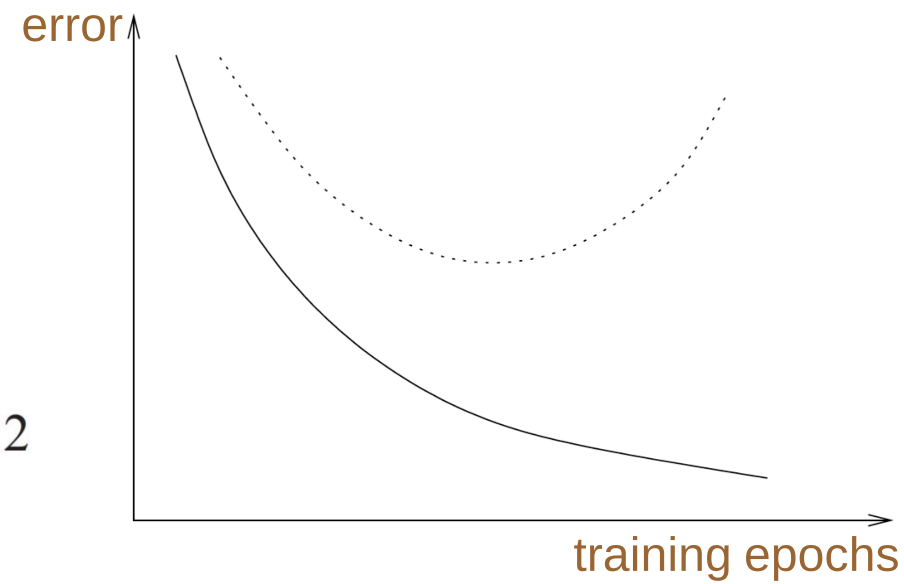
- The backpropagation algorithm is a gradient descent search procedure that is prone to find only *local optima* instead of the global optimum.
- Possible termination criteria for backpropagation include:
  - Error reduction falls below a threshold.
  - Number of training *epochs* above a threshold. One training epoch refers to the processing of the complete training set.



Overfitting

- If trained over many epochs an MLP can adapt very closely to the training samples. However, this may lead to *overfitting* and harm the *generalization* capability of the MLP for unseen samples.
- A good strategy to avoid overfitting during cross-validation is to stop training with respect to the error on the independent validation set.





true class boundary

overfit class boundary

training set

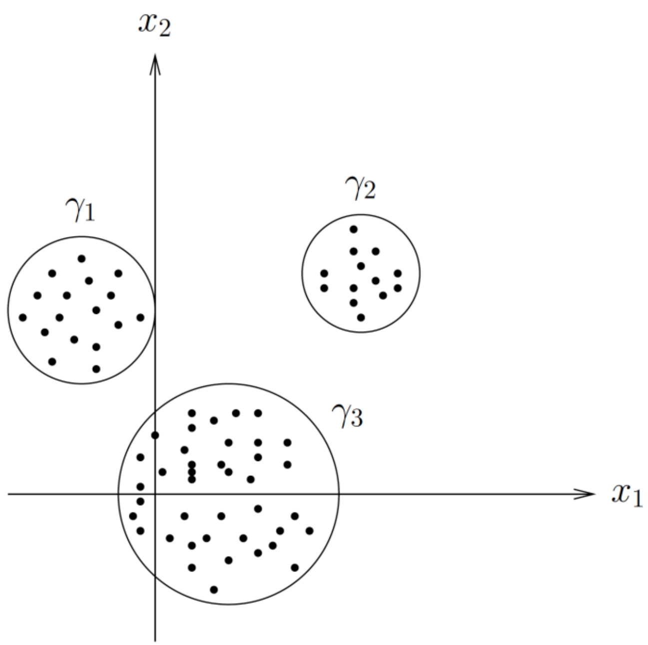
validation set

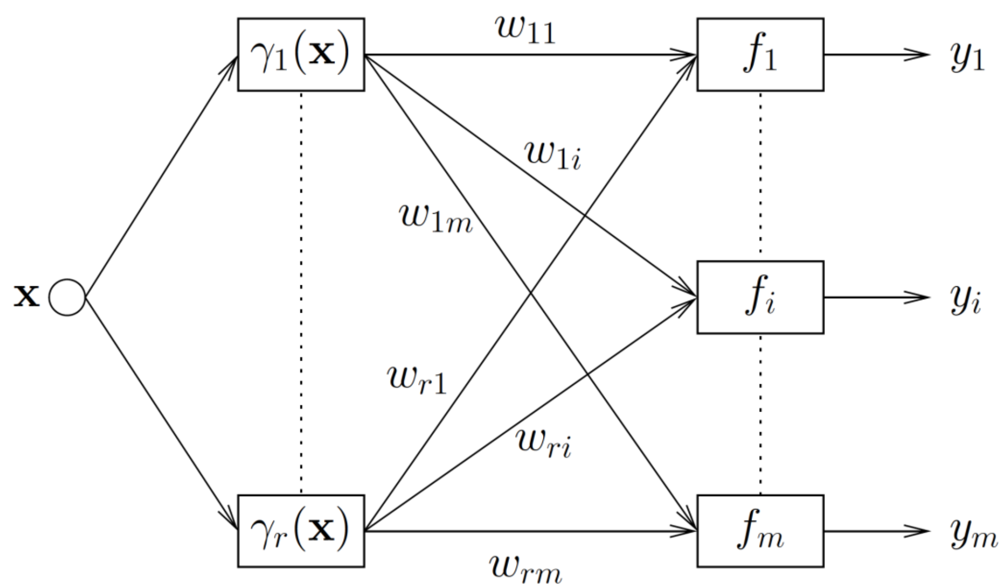
Other Network Architectures

Radial Basis Function (RBF) Networks

- Hidden neurons are radial basis functions, for example Gaussians.
- Mean and covariance estimated from the training samples after clustering (often k-means).

$$g_i(x) = \frac{1}{\sqrt{|\Sigma_i|(2\pi)^n}} \exp\left(-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right)$$

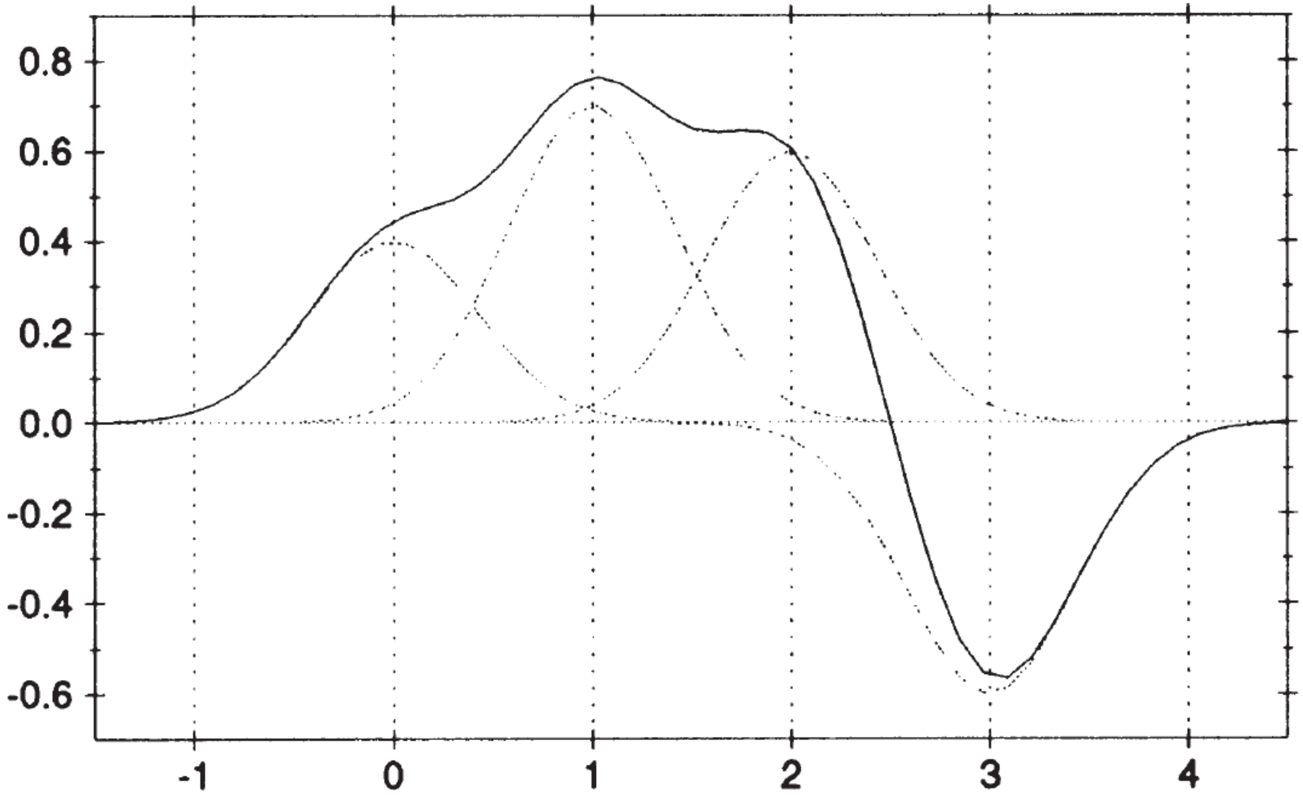




Example

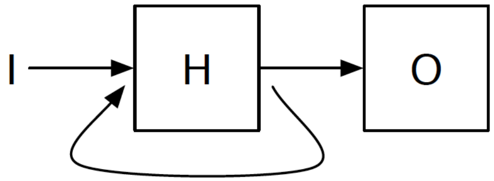
- Approximate likelihood function p(x|C).

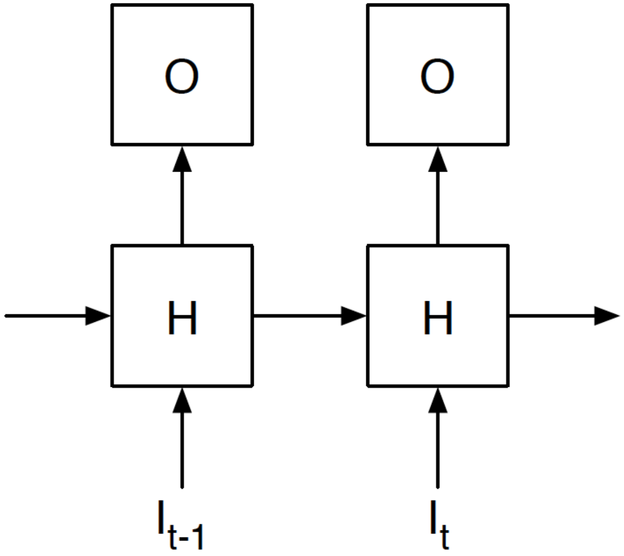
$$y(x) = \sum_{j=1}^{4} w_{j1} g_j(x)$$



Recurrent Neural Networks (RNN)

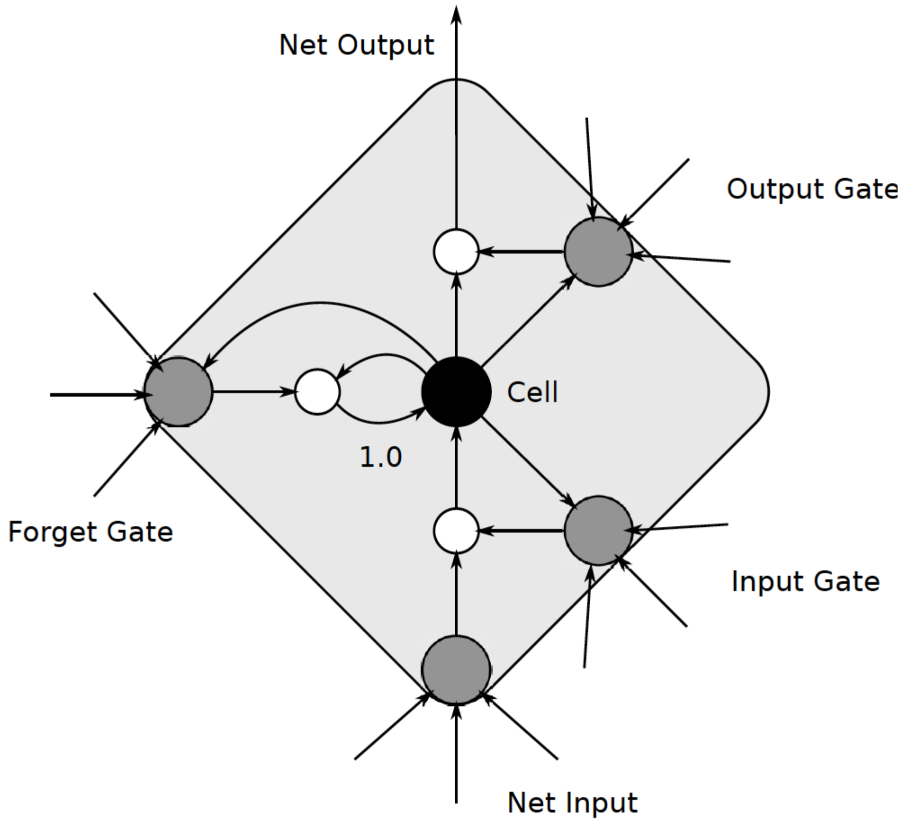
- Hidden layer connected to itself, therefore keeps the information over several time steps.
- Ideally suited for sequence recognition such as speech, movement, handwriting, etc.
- Training strategies include *backpropagation through time*.
- Further reading: http://karpathy.github.io/2015/05/21/rnn-effectiveness/





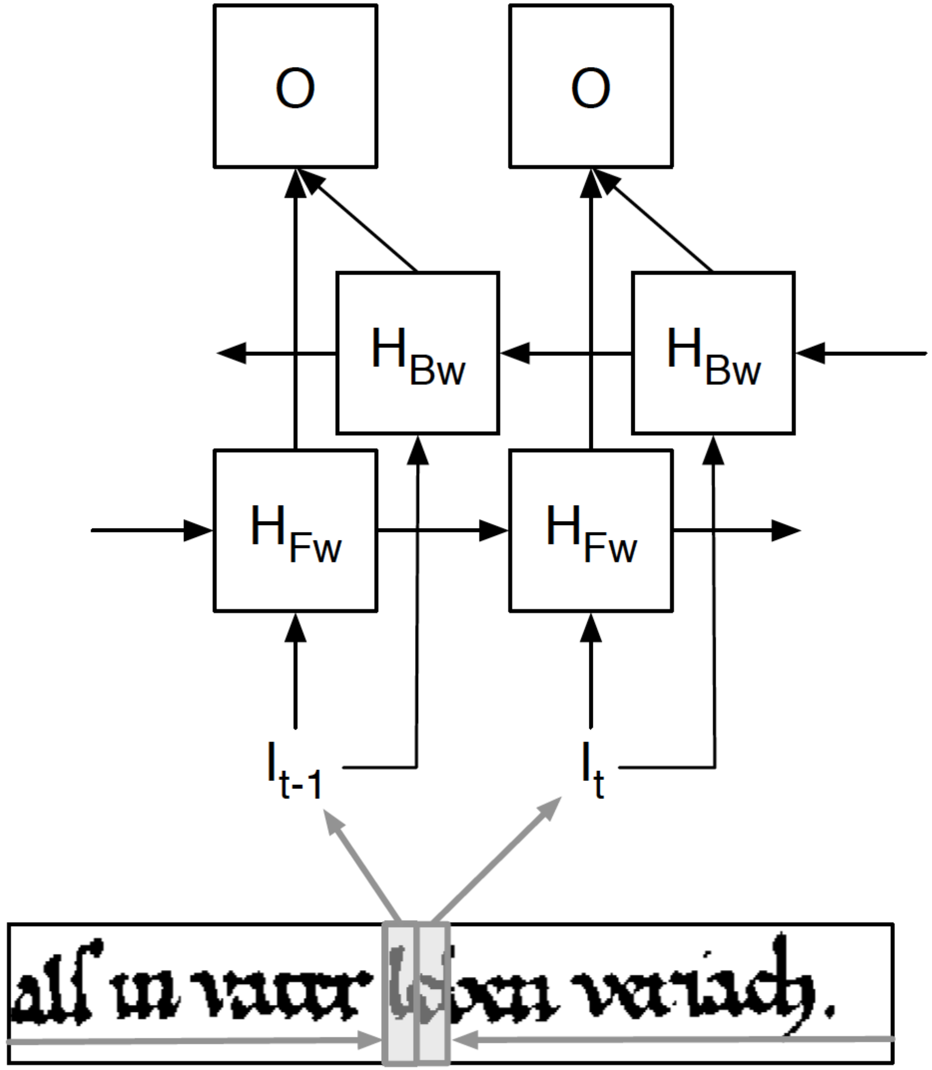
Long Short-Term Memory (LSTM) Networks

- Address the vanishing gradient problem, that is the exponential decay of information in RNN over several time steps.
- Instead of standard neurons, LSTM cells are used. They learn the relevant context using input, output, and forget gates.



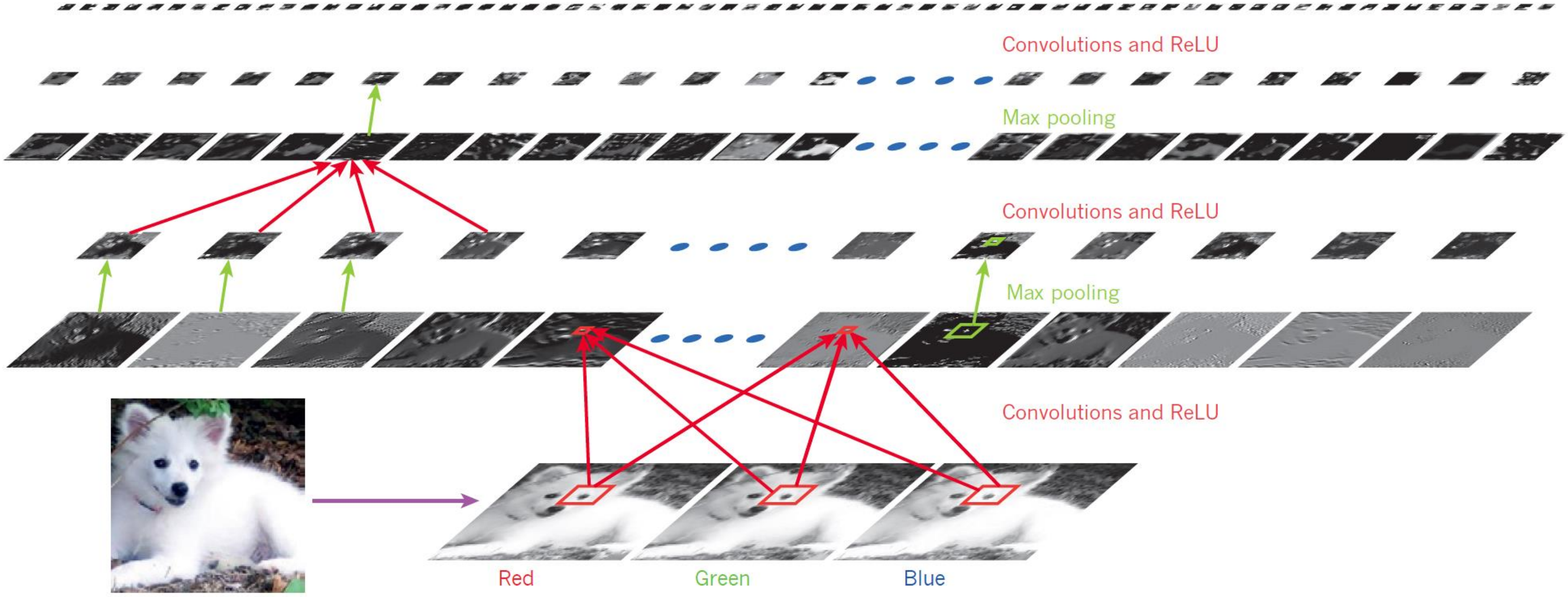
Example

- Handwriting recognition using bidirectional LSTM networks.



Deep Neural Networks

- Recent success in training deep networks with a large number of layers and neurons; successful pretraining strategy with autoencoders.
  - G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. Science 313:504-507, 2006.
  - Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature 521:436-444, 2015.
- Software frameworks readily available, for example PyTorch: https://pytorch.org/



Example

- https://openai.com/blog/dall-e/
- A 12-billion parameter network able to draw images based on a textual description.