Neural Networks and Learning Systems
TBMI 26, 2017

Lecture 3

Supervised learning – Neural networks

Ola Friman
ola.friman@liu.se

Recap - Supervised learning

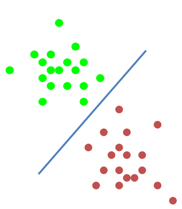
- **Task:** Learn to predict/classify new data from labeled examples.
- **Input:** Training data examples $\{\mathbf{x}_i, y_i\}$ i=1...N, where $\mathbf{x}_i$ is a feature vector and y_i is a class label.
- **Output:** A function $f(\mathbf{x};w_1,\ldots,w_k)$ that can predict the class label of $\mathbf{x}$.

Find a function f and adjust the parameters $w_1,\ldots,w_k$ so that new feature vectors are classified correctly. Generalization!!

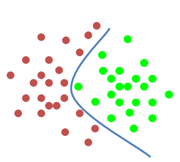
2

Linear separability

Linearly separable



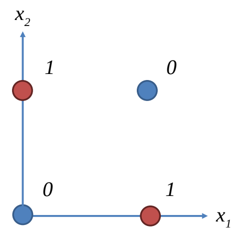
Non-linearly separable



3

The XOR problem

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

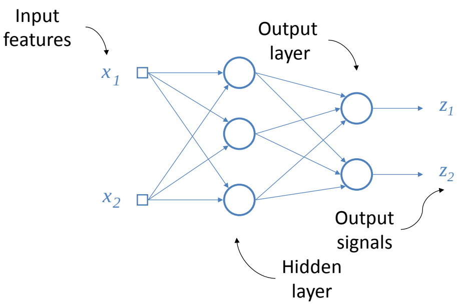


Not linearly separable!

4

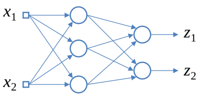
Multi-layer perceptron

a.k.a Neural Network



Many classes - output coding

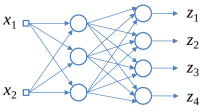
Example with 4 classes



$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \begin{bmatrix}0\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix} & \begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}1\\1\end{bmatrix} \end{array}$$

or

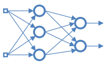
$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \begin{bmatrix}-1\\-1\end{bmatrix} & \begin{bmatrix}-1\\1\end{bmatrix} & \begin{bmatrix}1\\-1\end{bmatrix} & \begin{bmatrix}1\\1\end{bmatrix} \end{array}$$



$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ \begin{bmatrix}1\\0\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \end{array}$$

or
………

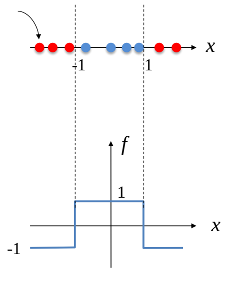
History of neural networks



- 1960's: Large enthusiasm around the perceptron and "connectionism" (Frank Rosenblatt).
- 1969: Limitations of the perceptron made clear in a paper by Minsky & Papert, e.g., the XOR problem.
- "Winter period" – little research
- 1980's: Revival of connectionism and neural networks:
  - Multi-layer networks can solve nonlinear problems (this was known before, but not how to train them!)
  - Backpropagation training algorithm
- 1990's: Reduced interest, other methods seemed more promising
- 2010's: Renewed interest – "Deep learning"

A simple 1D example

Training samples with only one feature value!

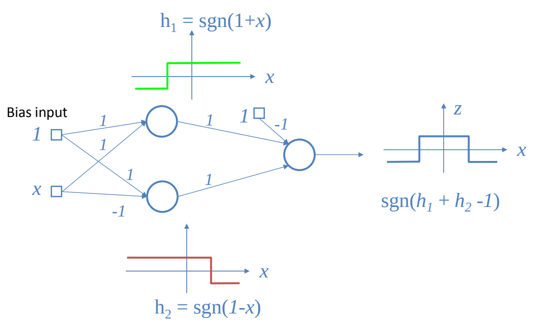


Not separable with a linear function!

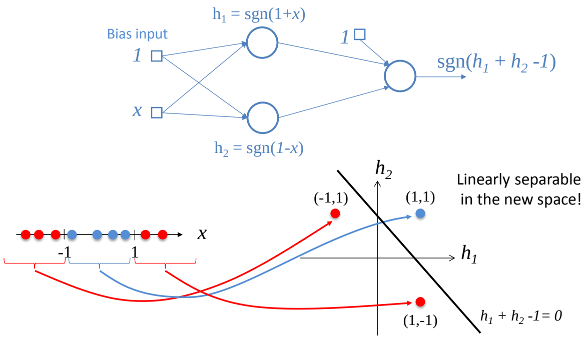
But with a nonlinear function!

$$f(x; w_0, \ldots, w_n) = \begin{cases} -1 & |x| > 1 \\ 1 & |x| < 1 \end{cases}$$

Example solution



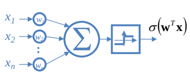
Nonlinear mapping to a new feature space!



Key: The hidden layer(s)

- The output layer requires linear separability. The purpose of the hidden layers is to make the problem linearly separable!
- **Cover's theorem (1965):** The probability that classes are linearly separable increases when the features are nonlinearly mapped to a higher-dimensional feature space.

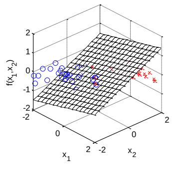
The Perceptron revisited

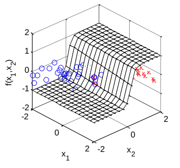


Minimize the following cost function

$$\varepsilon(\mathbf{w})=\sum_{i=1}^{N}\left(\sigma\left(\mathbf{w}^T\mathbf{x}_i\right)-y_i\right)^2$$

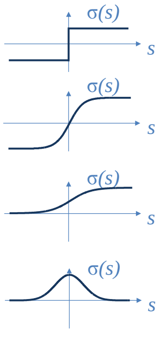
$$\sigma\left(\mathbf{w}^T\mathbf{x}\right)=\mathbf{w}^T\mathbf{x} \qquad \sigma\left(\mathbf{w}^T\mathbf{x}\right)=\tanh\left(\mathbf{w}^T\mathbf{x}\right)$$





Nonlinear activation functions

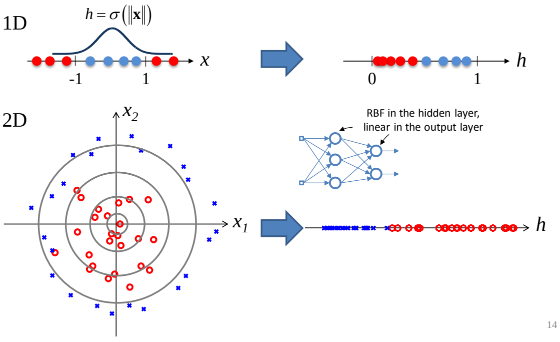
- Step/sign function
  Not differentiable – cannot be optimized!
- Hyperbolic tangent
  $\sigma(s)=\tanh(s) \quad \sigma'=1-\tanh^2(s)=1-\sigma^2$
- The Fermi-function
  $\sigma(s)=\dfrac{1}{1+e^{-s}} \quad \sigma'=\sigma(1-\sigma)$
- Gaussian function
  $\sigma(s;\gamma)=e^{-\frac{s^2}{\gamma^2}} \quad \sigma'(s;\gamma)=-\dfrac{2s}{\gamma}\sigma$



13

Example - Radial Basis Functions

For example a Gaussian



14

Updated minimization algorithm

$$\varepsilon(\mathbf{w})=\sum_{i=1}^{N}\left(\sigma\left(\mathbf{w}^T\mathbf{x}_i\right)-y_i\right)^2$$

$$\frac{\partial\varepsilon}{\partial\mathbf{w}}=2\sum_{i=1}^{N}\left(\sigma\left(\mathbf{w}^T\mathbf{x}_i\right)-y_i\right)\sigma'\left(\mathbf{w}^T\mathbf{x}_i\right)\mathbf{x}_i$$

Gradient descent:

$$\mathbf{w}_{t+1}=\mathbf{w}_t-\eta\frac{\partial\varepsilon}{\partial\mathbf{w}} \quad (\text{Eq.}1)$$

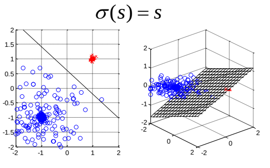
Algorithm:

1. Start with a random **w**
2. Iterate Eq. 1 until convergence

15

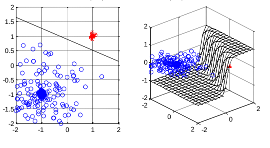
Example

$\sigma(s)=s$



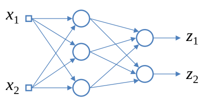
Same as in lecture 2!

$\sigma(s)=\tanh(s)$



16

Training multi-layer neural networks



Cost function

training examples   # output nodes

$$\varepsilon(\mathbf{w}) = \sum_{k=1}^{K} \sum_{m=1}^{M} \left[y_{mk} - z_{mk}(\mathbf{w}) \right]^2$$

all weights   desired output   actual output

18

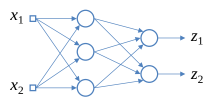
Stochastic gradient descent

Update using one (K=1) training example

$$\varepsilon(\mathbf{w}) = \sum_{m=1}^{M} \left[y_m - z_m(\mathbf{w}) \right]^2$$

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta \frac{\partial \varepsilon}{\partial w_{ij}}$$

From node i to node j
in a layer



19

The chain rule

$$f(g(x)) \qquad\qquad f(g(x), h(x))$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \qquad\qquad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}$$

Examples:

$$f(x) = \sin(x^2) \qquad \frac{\partial f}{\partial x} = \cos(x^2)2x$$

$$f(x) = x^2 \sin(x) \qquad \frac{\partial f}{\partial x} = 2x\sin(x) + x^2\cos(x)$$

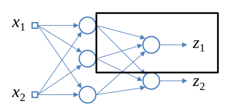
20

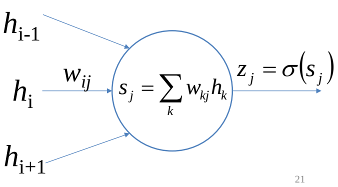
The error backpropagation algorithm

- an exercise of the chain rule!

$$\frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial z_j}\frac{\partial z_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}}$$

$$\varepsilon(\mathbf{w}) = \sum_{m=1}^{M} \left[y_m - z_m(\mathbf{w}) \right]^2$$





21

Backpropagation, cont

$$\varepsilon(\mathbf{w}) = \sum_{m=1}^{M} \left[y_m - z_m(\mathbf{w}) \right]^2 \qquad \frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial z_j} \frac{\partial z_j}{\partial s_j} \frac{\partial s_j}{\partial w_{ij}}$$

$$\frac{\partial \varepsilon}{\partial z_j} = -2\left(y_j - z_j\right)$$

$$\frac{\partial z_j}{\partial s_j} = \sigma'(s_j) = 1 - \sigma(s_j)^2 = 1 - z_j^2$$

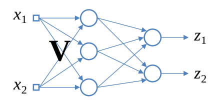
If σ(s) = tanh(s) is used!

$$\frac{\partial s_j}{\partial w_{ij}} = h_i$$

22

Updating the hidden layer

$$\frac{\partial \varepsilon}{\partial v_{ij}} = ?$$



$$\varepsilon(\mathbf{v}) = \sum_{m=1}^{M} \left[y_m - z_m(\mathbf{v}) \right]^2$$

A weight in the hidden layer affects **all** output nodes!

$$\varepsilon\left(z_1(\mathbf{v}), \ldots, z_M(\mathbf{v})\right)$$

$$\frac{\partial \varepsilon}{\partial v_{ij}} = \sum_{k=1}^{M} \frac{\partial \varepsilon}{\partial z_k} \frac{\partial z_k}{\partial v_{ij}} = \ldots.$$

Exercise!

Chain rule! Continue expanding!

23

Backpropagation – Summary

- Two phases:
  - Forward propagation: Propagate a training example through the network
  - Backward propagation: Propagate the error relative the desired output backwards in the net and update parameter weights.
- Batch update: update after all examples have been presented.

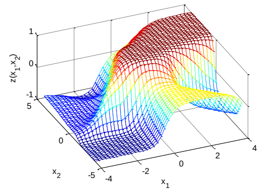
$$\Delta w_{ij} = -\eta \sum_{k=1}^{K} \frac{\partial \varepsilon(k)}{\partial w_{ij}}$$

24

Decision boundaries

Neural networks can produce very complex class boundaries!

$z(x_1, x_2)$

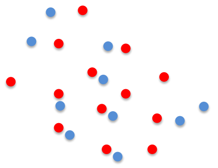


$f(x_1, x_2) = sign(z(x_1, x_2))$

25

Reminder - Magic is not possible!

No neural network, however complex, can separate inseparable classes!



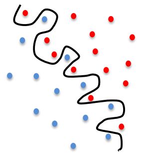
Finding and extracting suitable features are the critical problems in machine learning!

Pros and cons of neural networks

- A multi-layer neural network can learn any class boundaries.
- The large number of parameters is a problem:
  - Local optima → suboptimal performance
  - Overfitting → poor generalization
  - Slow convergence → long training times

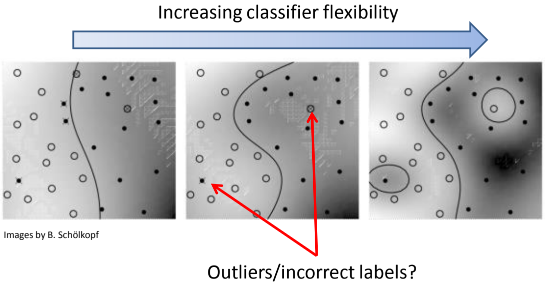
Overfitting

- The large number of parameters makes it possible to produce overly complicated boundaries.



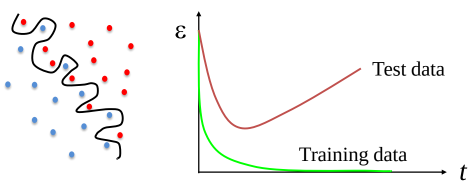
- A too good fit to the training data can perform poorly for new cases, i.e. worse generalization properties!

Example



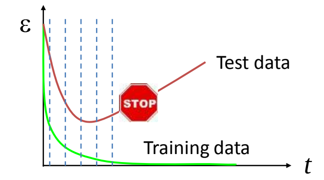
Overfitting

- The error on training data *always* decreases with increased training
- The error on test data (the generalization error) decreases in the beginning, but can then start to increase if overfitting occurs!



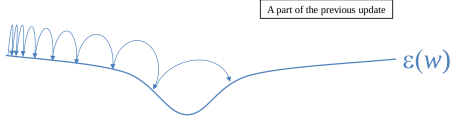
Preventing overfitting in neural networks

- **Early stopping:**
  Pause training regularly and calculate the performance on the test data.
  Caution: Test data becomes training data! Will bias evaluation.



Faster convergence

- Normalize input features, e.g., to the range [-1,1].
- Separate and adaptive step length η for each weight:
  - If the derivative has the same sign in several consecutive steps, η should increase. If the derivative change sign, η should decrease.
- Introduce a *momentum term*: $\Delta w_{ji}(t) = \underbrace{\alpha \Delta w_{ji}(t-1)}_{\text{A part of the previous update}} - \eta \frac{\partial \varepsilon(t)}{\partial w_{ji}}$



How many layers?

- 1 hidden layer is enough to produce any classification boundary.
- Complex boundaries more compactly obtained with many non-linear layers - less nodes in total compared to 1-layer solution.
- With ordinary 'backprop' training, no performance advantage with many hidden layers.
  - Vanishing gradient problem:
    Error gradients become very small for early layers in the network → weights are not updated.