Handout Perceptrons and Multi-Layered Perceptrons

Bert Kappen,
Department of Biophysics
Radboud University Nijmegen

September 30, 2014

Contents

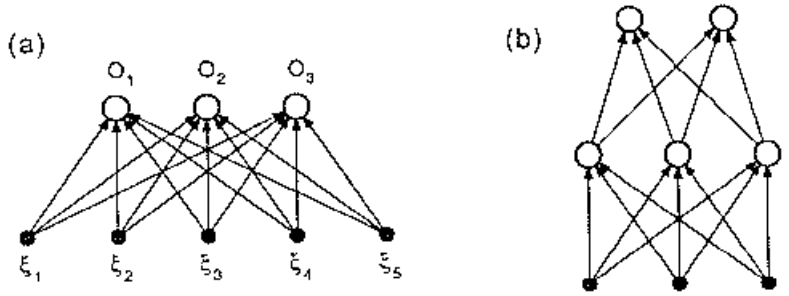


Figure 1: A) Simple Perceptron B) Multi-layered Perceptron

1 Perceptrons

Perceptrons are feed-forward neural networks. Examples are given in Fig. 1. Consider a simple perceptron with one output:

$$o = g(h) = g\left(\sum_{j=1}^{n} w_j \xi_j - \theta\right) = g\left(\sum_{j=0}^{n} w_j \xi_j\right)$$

with weights w_j and inputs ξ_j. $\xi_0 = -1$ and $\theta = w_0$. g is a non-linear function.

Learning: Given a number of input-output pairs $(\xi_j^\mu, \zeta^\mu), \mu = 1, \ldots, P$, find w_j such that the perceptron output o for each input pattern ξ^μ is equal to the desired output ζ^μ:

$$o^\mu = g\left(\sum_{j=0}^{n} w_j \xi_j^\mu\right) = \zeta^\mu, \quad \mu = 1, \ldots, P$$

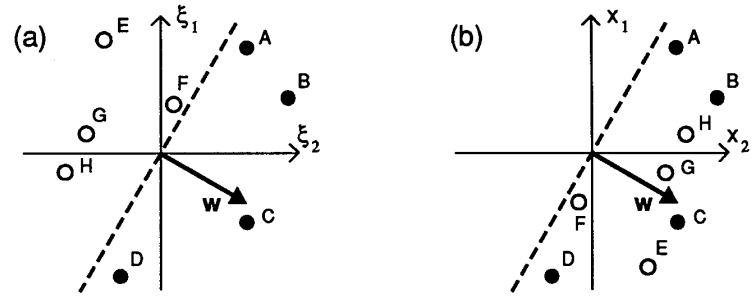
1.1 Threshold units

Consider the simplest case of binary threshold neurons:

$$g(h) = \text{sign}(h)$$

Then, the learning condition becomes

$$\text{sign}(w \cdot \xi^\mu) = \zeta^\mu, \quad \mu = 1, \ldots, P$$



Since $\zeta^\mu = \pm 1$, we have

$$\text{sign}(w \cdot \xi^\mu \zeta^\mu) = 1 \quad \text{or} \quad w \cdot x^\mu > 0$$

with $x_j^\mu = \xi_j^\mu \zeta^\mu$.

1.2 Linear separation

Classification depends on sign of $w \cdot \xi$. Thus, decision boundary is hyper plane:

$$0 = w \cdot \xi = \sum_{j=1}^{n} w_j \xi_j - \theta$$

Perceptron can solve linearly separable problems. An example of a linearly separable problem is the AND problem: The output of the perceptron is 1 if all inputs are 1, and -1 otherwise (see Fig. 2).

By definition, problems that are not linearly separable need more than one separating hyper plane to separate the two classes. An example of a non-linearly separable problem is the XOR problem: The output is equal to the product of the input values (see Fig. 2A). Other problems that are not linearly separable occur when three or more input patterns are linearly dependent (see Fig. 2B).

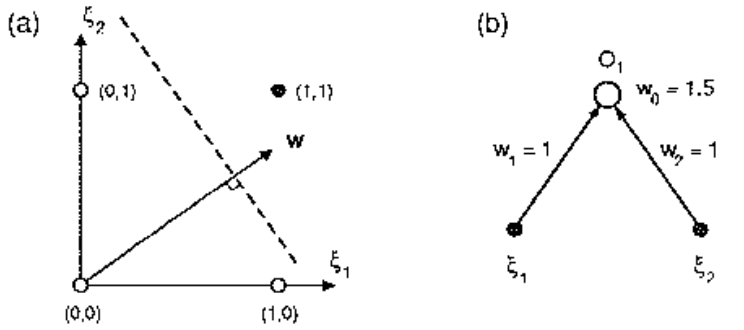
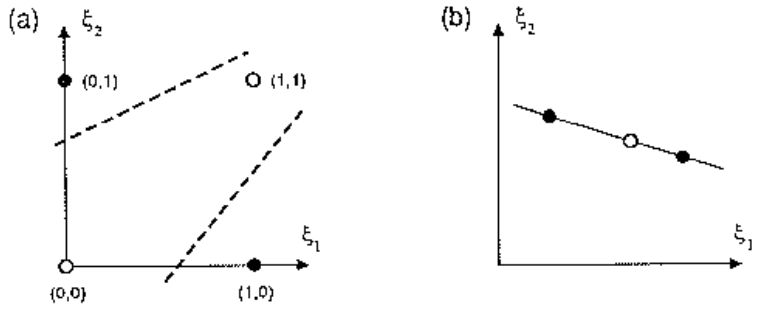


Figure 2: The AND problem for two inputs is linearly separable.



1.3 Perceptron learning rule

We have seen that the desired weight vector satisfies

$$w \cdot x^\mu > 0, \quad \text{all patterns} \tag{1}$$

We define the following perceptron learning rule:

$$\begin{aligned} w_j^{\text{new}} &= w_j^{\text{old}} + \Delta w_j \\ \Delta w_j &= \eta \Theta(-w \cdot x^\mu) \xi_j^\mu \zeta^\mu = \eta \Theta(-w \cdot x^\mu) x^\mu \end{aligned} \tag{2}$$

η is the learning rate. This learning rule is Hebbian in the sense that the change in weight is proportional to the product of input and output activity. The function Θ is 1 for positive arguments and zero otherwise: When presenting pattern μ, learning only occurs, when the condition $w \cdot x^\mu > 0$ is not satisfied for that pattern.

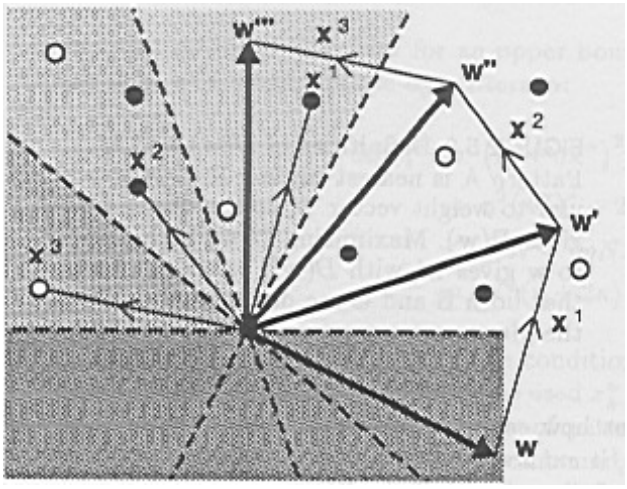


Figure 3: The perceptron learning rule in action. Learning rule Eq. 2 is applied to all patterns in some random or given order. Learning stops, when a weight configuration is found that has positive inner product with all training patterns.

In Fig. 3 we show the behavior of the perceptron learning rule with $\eta = 1$. The dataset consists of three data patterns x^1, x^2 and x^3. The initial weight vector is w. Presenting pattern x^1, we note that $w \cdot x^1 < 0$ and therefore learning occurs. The resulting weight vector is $w' = w + x^1$. Presenting pattern x^2 and x^3 also result in learning steps and we end up in weight configuration w'''. This weight vector has positive inner product with all training patterns and learning terminates.

Depending on the data, there may be many or few solutions to the learning problem, or non at all! In Fig. 4 we give examples of two data sets and their solutions Eq. 1. In Fig. 4A there are more admissible weight vectors and they can have a larger inner product with all training patterns than in Fig. 4B. We define the quality of the solution w by the pattern that has the smallest inner product with w. Since the solution does not depend on the norm of w, we define the quality as

$$D(w) = \frac{1}{||w||} \min_{\mu} w \cdot x^{\mu}$$

The best solution is given by $D_{\max} = \max_w D(w)$.

In Fig. 5, we illustrate this for a given data set and two admissible solutions w and w' and their values of D respectively. Since $D(w') > D(w)$, w' is the preferred solution.

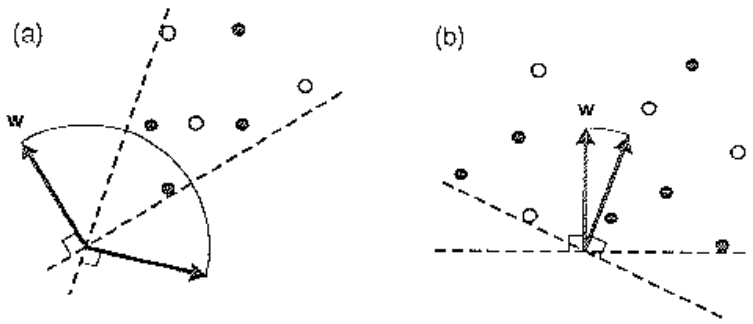


Figure 4: Two examples of data sets and the sets of w that satisfy condition Eq. 1. A) Many solutions B) Few solutions.

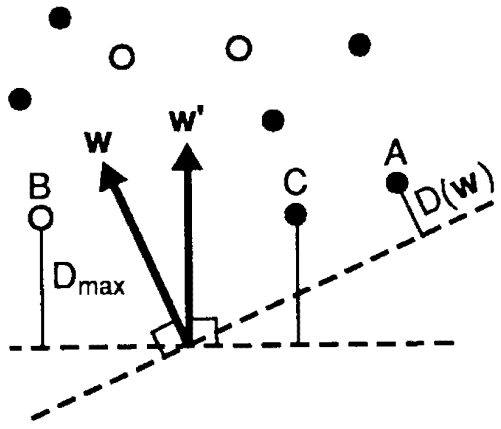


Figure 5: Two admissible solutions w and w' and their values of D respectively. Since $D(w') > D(w)$, w' is the preferred solution.

If we can find a w such that $D(w) > 0$ the problem is linearly separable and learnable by the perceptron learning rule. If the problem is not linearly separable not such solution exists.

1.3.1 Convergence of Perceptron rule

In this section we show that if the problem is linearly separable, the perceptron learning rule converges in a finite number of steps. We start with initial value $w = 0$. At each iteration, w is updated only if $w \cdot x^\mu < 0$. After some number of iterations, let M^μ denote the number of times pattern μ has been used to update w. Thus,

$$w = \eta \sum_\mu M^\mu x^\mu$$

$M = \sum_\mu M^\mu$ is the total number of iterations in which the weight vector is updated. If the learning rule converges, it means that M is finite and does not grow indefinitely.

The proof goes as follows. Assume that the problem is linearly separable, so that there is a solution w^* with $D(w^*) > 0$. We will show that

$$O(\sqrt{M}) \leq \frac{w \cdot w^*}{||w||||w^*||} \leq 1$$

where the second inequality follows simply from the definition of the inner product, and we will show the first inequality below. Thus, M can not grow indefinitely and the perceptron learning rule converges in a finite number of steps.

The proof of the first inequality is elementary:

$$\begin{aligned} w \cdot w^* &= \eta \sum_\mu M^\mu x^\mu \cdot w^* \geq \eta M \min_\mu x^\mu \cdot w^* = \eta M D(w^*)||w^*|| \\ \Delta||w||^2 &= ||w + \eta x^\mu||^2 - ||w||^2 = 2\eta w \cdot x^\mu + \eta^2 ||x^\mu||^2 \leq \eta^2 ||x^\mu||^2 = \eta^2 N \end{aligned}$$

The inequality in the second line makes use of the fact that for each training pattern where learning takes place $w \cdot x^\mu < 0$. The norm of w is thus bounded by

$$||w||^2 \leq \eta^2 N M$$

Combining these two inequality, we obtain Thus,

$$\frac{w \cdot w^*}{|w||w^*|} \geq \sqrt{M} \frac{D(w^*)}{\sqrt{N}} \tag{3}$$

which completes the proof. Note, that the proof makes essential use of the existence of w^* with $D(w^*) > 0$. If $D(w^*) < 0$ the bound Eq. 3 becomes a trivial statement and does not yield a bound on M.

If the problem is linearly separable, we can in conclude that the number of weight updates:

$$M \leq \frac{N}{D^2(w^*)}$$

where N is some trivial constant. We see that convergence takes longer for harder problems (for which $D(w^*)$ is closer to zero).

1.4 Linear units

We now turn to a possibly simpler case of linear units:

$$o^\mu = \sum_j w_j \xi_j^\mu$$

Desired behavior is that the perceptron output equals the desired output for all patterns: $o^\mu = \zeta^\mu, \mu = 1, \ldots, P$. In this case, we can compute an explicit solution for the weights. It is given by

$$w_j = \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^\nu, \quad Q_{\rho\nu} = \frac{1}{N} \sum_j \xi_j^\rho \xi_j^\nu \tag{4}$$

Q is a matrix with dimension $P \times P$ and contains the inner products between the input patterns.

To verify that Eq. 4 solves the linear perceptron problem, we simply check for one of the input patterns (ξ^μ) whether it gives the desired output:

$$\begin{aligned} \sum_j w_j \xi_j^\mu &= \frac{1}{N} \sum_{\rho,u,j} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u \xi_j^\mu \\ &= \sum_{\rho,u} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} Q_{\nu\mu} \\ &= \sum_\rho \zeta^\rho \delta_{\rho\mu} = \zeta^\mu \end{aligned}$$

For this solution to exist, Q must be invertible which means that Q must be of maximal rank (rank P). Therefore $P \leq N$. [1]

When $P < N$ the solution $w_j = \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u$ is not unique. In fact, there exists a linear space of dimension $N - P$ of solutions w. Namely, let

$$\begin{aligned} w_j^0 &= \frac{1}{N} \sum_{\rho\nu} \zeta^\rho \left(Q^{-1}\right)_{\rho\nu} \xi_j^u \\ w_j &= w_j^0 + \xi^\perp \end{aligned}$$

with $\xi^\perp$ an n-dimensional vector that is perpendicular to all training patterns: $\xi^\perp \perp \{\xi^\mu\}$. Then the output of the perceptron is unaffected by $\xi^\perp$:

$$\zeta^\mu = \sum_j w_j \xi_j^\mu = \sum_j (w_j^0 + \xi_j^\perp)\xi_j^\mu = \sum_j w_j^0 \xi_j^\mu$$

1.4.1 Gradient descent learning

Often $P > N$, and thus patterns are linearly dependent. In general, one can define a learning rules through a cost function, that assigns a cost or quality to each possible weight vector. A common cost function is the quadratic cost:

$$E(w) = \frac{1}{2} \sum_\mu \left(\zeta^\mu - \sum_j w_j \xi_j^\mu \right)^2$$

which is minimized when the actual perceptron output $\sum_j w_j \xi_j^\mu$ is as close as possible to the desired output ζ^μ for all patterns μ.

[1] In addition, the input patterns must be linearly independent. If the input patterns are linearly dependent, solution Eq. 4 does not exist unless the corresponding outputs are also linearly dependent. Linear dependence of the inputs implies that there exists α^μ such that

$$\sum_\mu \alpha^\mu \xi_j^\mu = 0$$

This implies that

$$\sum_\mu \alpha^\mu \zeta^\mu = \sum_{\mu j} w_j \alpha^\mu \xi_j^\mu = 0$$

in other words, that the outputs cannot be chosen at freely. For problems with linearly dependent inputs and matched linearly dependent output Eq. 4 can be used by restricting the training set to a linearly independent subset that spans the training set, and computing Q for this subset.

The cost function can be minimized by the so-called gradient descent procedure. We start with an initial random value of the weight vector w and we compute the gradient in this point:

$$\frac{\partial E}{\partial w_i} = -\sum_\mu \left(\zeta^\mu - \sum_j w_j \xi_j^\mu \right) \xi_i^\mu$$

We change w according to the 'learning rule'

$$w_i = w_i + \Delta w_i \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{5}$$

and repeat this until the weights do not change any more.

When η is sufficiently small, it is easy to verify that this gradient descent procedure converges. The proof consists of two observations. One is that for small η, $E(w)$ decreases in each step, and the other is that $E(w)$ is bounded from below, so that it has a smallest value. Therefore E cannot continue decreasing indefinitely and must converge to some stationary value (see Exercises).

1.4.2 The value of η

What is a good value form η? Clearly, when η is very small, convergence is guaranteed, but in practice it may take a very long time. If η is too large, however, convergence is no longer guaranteed. The problem is further complicated by the fact that the optimal choice of η is different for different components of the weight vector w. This is illustrated in Fig. 6, where E as a function of w is drawn. This valley has a unique minimal value for E, but the curvature in two directions is very different. In the long (flat) direction, large steps can be made, but in the orthogonal direction only small steps are allowed. We can analyze the problem, by assuming that the energy has the form

$$E(w) \quad = \quad \frac{1}{2} \sum_i a_i \left(w_i - w_i^* \right)^2 + E_0$$

with w^* the location of the minimum, and a_i the curvatures in the two directions $i = 1, 2$. Eq. 5 becomes

$$\Delta w_i \quad = \quad -\eta \frac{\partial E}{\partial w_i} = -2\eta a_i \left(w_i - w_i^* \right) = -2\eta a_i \delta w_i$$

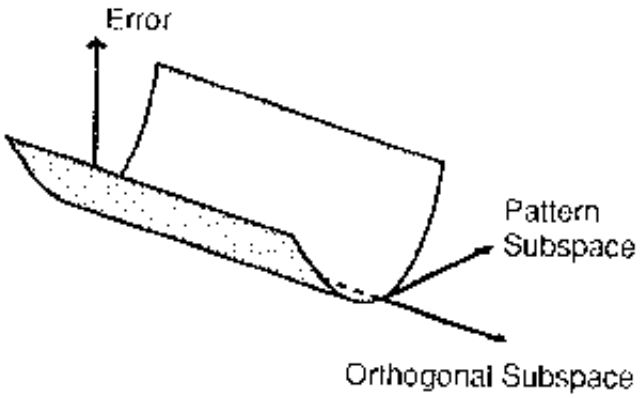


Figure 6: Cost landscape $E(w)$ with different curvatures in different directions.

with $\delta w_i = w_i - w_u^*$. The effect of learning step on δw_i is

$$\delta w_i^{\text{new}} = w_i^{\text{new}} - w_i^* = w_i^{\text{old}} - 2\eta a_i \delta w_i^{\text{old}} - w_i^* = (1 - 2\eta a_i)\delta w_i^{\text{old}}$$

thus, δw_i converges asymptotically to zero iff

$$|1 - 2\eta a_i| < 1. \tag{6}$$

We must find an η that satisfies Eq. 6 for all i. When $1 - 2\eta a_i < 0$, δw_i changes sign in each iteration. The behavior is illustrated in Fig. 7 with $E(w_1, w_2) = w_1^2 + 20w_2^2$ for different values of η.

1.5 Non-linear units

We can extend the gradient descent learning rule to the case that the neuron has a non-linear output:

$$o^\mu = g(h^\mu), \quad h^\mu = \sum_j w_j \xi_j^\mu$$

We use again the quadratic cost criterion:

$$E_1(w) = \frac{1}{2} \sum_\mu (\zeta^\mu - o^\mu)^2$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \sum_\mu (\zeta^\mu - o^\mu)\, g'(h^\mu) \xi_i^\mu$$

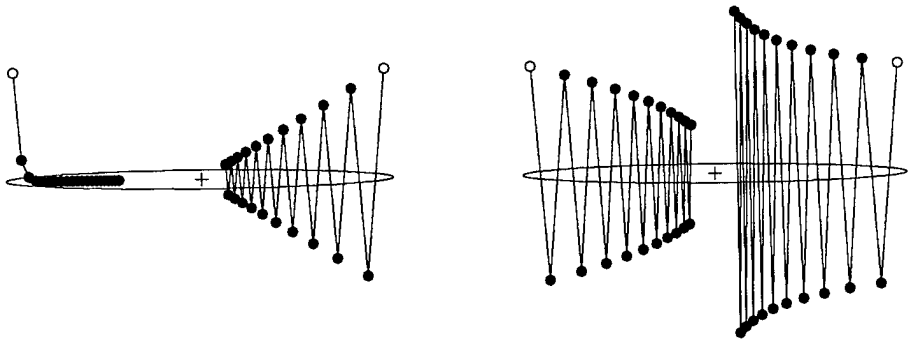


Figure 7: Behavior of the gradient descent learning rule Eq. 5 for the quadratic cost function $E(w_1, w_2) = w_1^2 + 20w_2^2$ for $\eta = 0.02, 0.0476, 0.049, 0.0505$.

When the function g is a monotonous function, it is invertible and one could also formulate a different cost criterion by observing the identity

$$\begin{aligned} \zeta^\mu &= g(h^\mu) \Leftrightarrow g^{-1}(\zeta^\mu) = h^\mu \\ E_2(w) &= \frac{1}{2}\sum_\mu \left(g^{-1}(\zeta^\mu) - h^\mu\right)^2 \end{aligned}$$

Note, that E_2 has a quadratic dependence on w, as in the linear case (but with transformed targets $g^{-1}(\zeta^\mu)$ instead of ζ^μ). In general, optimizing either E_1 or E_2 yield different optimal solutions.

1.6 Multi-layered perceptrons

The gradient descent learning procedure can be trivially extended to the perceptron with multiple layers and multiple outputs as shown in Fig. 1B. In addition to the input variables ξ_k and the output variable o_i, we have a layer of hidden variables v_j for which no training data are observed. The value of the hidden variables is computed in terms of the input variables, and the outputs are computed in terms of the hidden variables:

$$o_i = g\left(\sum_j w_{ij} v_j\right) = g\left(\sum_j w_{ij} g\left(\sum_k w_{jk}\xi_k\right)\right) \tag{7}$$

The output is now a complex function of the input pattern ξ_k and the weights w_{jk} in the first layer of the network and the weights w_{ij} in the second layer of the network.

Given a set of P training patterns $(\xi_k^\mu, \zeta_i^\mu), \mu = 1, \ldots, P$, we again use the gradient descent procedure to find the weights that minimize the total quadratic error:

$$E(w) = \frac{1}{2} \sum_i \sum_\mu \left(o_i^\mu - \zeta_i^\mu \right)^2 \tag{8}$$

with o_i^μ the output on node i for input pattern ξ^μ as given by Eq. 7.

For large neural networks with many hidden units, the simple gradient descent procedure can be quite slow. However, there exist well-known algorithms that significantly accelerate the convergence of the gradient descent procedure. One such method is the conjugate gradient method. Treatment of this method is beyond the scope of this course (see however [**?**] or Matlab for further details).

Note, that the optimal solution that is found depends on the number of hidden units in the network. The more hidden units, the more complex functions between input and output can be learned. So, for a given data set, we can make the error Eq. 8 as small as we like by increasing the number of hidden units. In fact, one can show that the multi-layered perceptron can learn any smooth function, given a sufficiently large number of hidden units.

However, the objective of a learning algorithm is to use the neural network to predict the output on novel data, that were not previously seen. Increasing the number of hidden units does not necessarily improve the prediction on novel data. The situation is illustrated in Fig. 8 for the case of one input variable and one output variable. The crosses denote the data points that were used for training and the smooth curve is the neural network solution. For a small number of hidden units, the solution may look something like Fig. 8A. The solution does not pass through all the data points. For a larger number of hidden units, the solution may look something like Fig. 8B. The solution does pass through all the data points and is more complex. However, the prediction of the more complex network is less accurate than the simple network for the data point indicated by the circle, which was not part of the training set. The extend to which the trained neural network is capable of predicting on novel data is called the generalization performance. The network with the optimal generalization performance must balance two opposing criteria: minimization of the error on the training data requires a large number of hidden units, but the solution should also be sufficiently smooth to give good prediction.

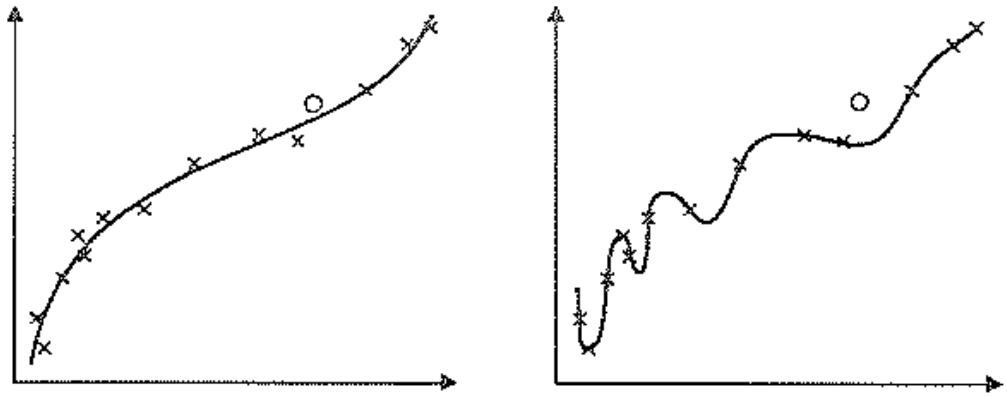


Figure 8: Network output versus network input. A) Network with a small number of hidden units. B) Network with a large number of hidden units. Networks with more hidden units can implement more complex functions and can better fit a given training set. However, more complex networks do not necessarily generalize better on novel data.

1.7 Summary

This chapter is based on [**?**]. Perceptrons are simple models of feed-forward computation in a network of neurons. Binary perceptrons can be used for classification problems. Learning is done using the perceptron learning rule. The learning rule converges in a finite number of iterations if and only if the problem is linearly separable.

Perceptrons can also be constructed with continuous output, either using a linear or non-linear transfer function. These perceptrons can be learned using the gradient descent method. Gradient descent converges asymptotically for any data set.

The quality of the perceptron can be significantly improved by using multiple layers of hidden units. The multi-layered perceptron can learn any function by using a sufficiently large number of hidden units. However, prediction quality on novel data does not generally increase with the number of hidden units. Optimal generalization is obtained for a finite number of hidden units.

1.8 Exercises

1. Check dat $D_{max} = \frac{1}{\sqrt{3}}$ voor het AND probleem en $D_{max} = -\frac{1}{\sqrt{3}}$ voor het XOR probleem. Het AND probleem in de $\xi_i = \pm 1$ codering is gedefinieerd als $\zeta = 1$ als $\xi_1 = \xi_2 = 1$ and $\zeta = -1$ in alle overige gevallen. Het XOR probleem is gedefinieerd als $\zeta = \xi_1 * \xi_2$. Gebruik voor de gewichten vector $w = (w_0, w_1, w_2)$. (Hint: gebruik $w_1 = w_2$ vanwege symmetrie).

2. Beschouw gradient descent in een kostenlandschap gegeven door $E = a_1 x^2 + a_2 y^2$. Bereken de leerparameter η zodanig dat de convergentie in zowel x als y richting even snel is.

3. Beschouw een lineair perceptron (sectie 1.4) om de AND functie te leren.

   - wat zijn de optimale gewichten en drempels? wat is de optimale kosten E?
   - laat zien dat $E > 0$ impliceert dat de inputpatronen lineair afhankelijk zijn.

4. Toon aan dat het gradient descent algoritme Eq. 5 asymptotisch convergeert.