Problems for practice in FYTN14/EXTQ40/NTF005F.
Problems marked with asterisk introduce side topics.

Chapter: Introduction

1.1 It is often useful to express the derivative $\varphi'(a)$ of activation functions in terms of $\varphi(a)$ itself. Do so for the following common activation functions.

a) $\varphi(a) = a$

b) $\varphi(a) = \frac{1}{1+\exp(-a)}$

c) $\varphi(a) = \tanh(a)$

d) $\varphi(a) = \max\{0, a\}$ (ignore $a = 0$)

e) $\varphi(a) = \ln[1 + \exp(a)]$

1.2 The logistic function $\varphi(a) = \frac{1}{1+\exp(-a)}$ and the hyperbolic tangent $\varphi(a) = \tanh(a)$ are two common sigmoidal activation functions. Consider a similar function $f(a) = \varphi(\beta a)$, where β is a constant. For the two sigmoidal cases, what is the slope $f'(a)$ for $a = 0$?

1.3 *

Yet another odd sigmoidal function is the algebraic sigmoid:

$$\varphi(a) = \frac{a}{\sqrt{1 + a^2}}$$

Compute the derivative of $\varphi(a)$ with respect to a and express it in terms of $\varphi(a)$. What is the value of $\varphi'(a)$ at the origin?

1.4 Suppose we have an activation function as the f defined in Problem 2.2, where the parameter β governs the slope of the sigmoidal function and a is e.g. a weighted sum of inputs. We would like to absorb the β parameter into a. How can we modify either the weights $(\omega_1, ..., \omega_N)$ or the inputs $(x_1, ..., x_N)$ to accomplish this?

1.5a Show that the threshold fuction $\theta(a)$ (which is 1 for $a > 0$ and 0 otherwise) may be approximated by a logistic function with large weights.

1.5b Show that a linear neuron may be approximated by a logistic function with small weights.

Chapter: Feed-Forward Networks

Note: Problems come in a somewhat random order!

2.1 Figure 1 shows a multi-layer perceptron with direct input to output weights.

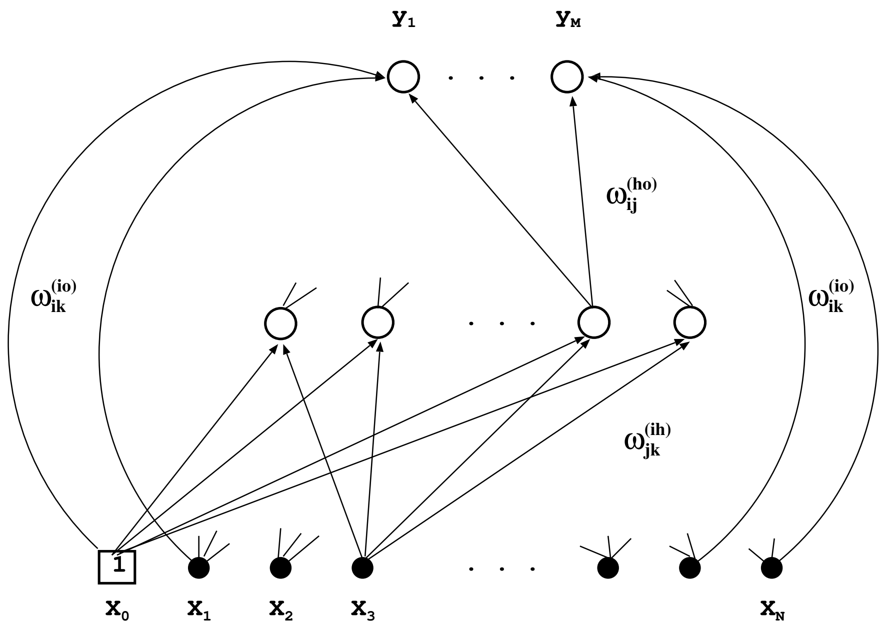

Figure 1: Network with skip-layer weights.

a) Write down the input-output mapping defined by this network.
b) Bias is shown as a special node with constant value 1, but there is no bias node in the hidden layer. Explain why it is not needed!

2.2 Figure 2 shows a simple perceptron. This output y is given by

$$y = \varphi\left(\sum_{k=1}^{P} x_k \omega_k + \omega_o\right)$$

where $\varphi(a)$ can be either sigmoidal or a threshold function. Show that the *decision boundary* implemented by this network is always a hyperplane in P dimensions.

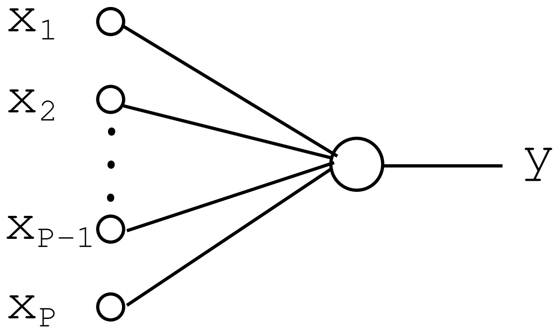

Figure 2: The perceptron

2.3 **a)** Derive the gradient descent updating rule for the simple perceptron (figure 2) that uses a logistic activation function. A summed-square error function $E = \sum_n (y_n - d_n)^2$ is used as loss.

b) Consider a pattern μ that has become misclassified, so that y_μ is very close to $(1 - d_\mu)$. Discuss how this pattern will contribute to the wieght update, and explain why the combination (logistic output) with (summed error loss) can be dangerous.

2.4 * Consider two one-dimensional, Gaussian distributed classes C_1 and C_2 that have a common variance equal to 1. Their mean values are:

$$\begin{aligned} \mu_1 &= -10 \\ \mu_2 &= +10 \end{aligned}$$

a) Write down an expression for the class distributions $p(x|C_1)$ and $p(x|C_2)$.

We would now like to construct a Bayes' classifier for C_1 and C_2. The condition

$$P(C_1|x) = P(C_2|x)$$

defines the Bayes' boundary.

b) Derive this boundary (i.e. rule for classification).

2.5a Figure 3 shows a neural network solving this XOR problem, defined as $(0,0) \to 0$, $(0,1) \to 1$, $(1,0) \to 1$ and $(1,1) \to 0$. The activation function for both nodes is the threshold function. Show that it solves the XOR problem by constructing (a) decision regions, and (b) a truth table for the network.

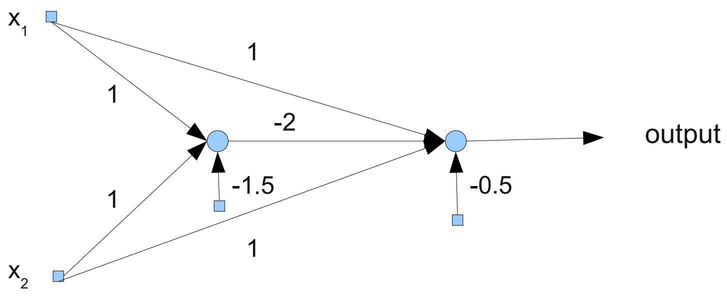

Figure 3: Network solving the XOR problem

2.5b One variant of the XOR problem is the following mapping: $(-1,-1) \rightarrow 0$, $(+1,-1) \rightarrow 1$, $(-1,+1) \rightarrow 1$ and $(+1,+1) \rightarrow 0$. Figure 4 shows a perceptron that can handle this XOR problem. The trick here is to define an input that is the product of x_1 and x_2. So we can see this as transformation of input space to easier handle the XOR problem. But we can also analyze this simple network an see what new boundaries this network is defining in the original 2D input space.

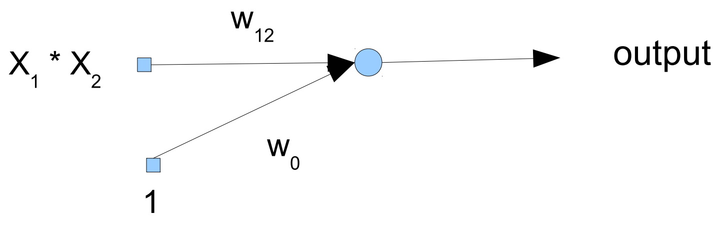

Figure 4: Network solving the XOR problem

This network is given by

$$y = \theta\left(x_1 * x_2 * \omega_{12} + \omega_0\right)$$

(θ is the threshold function). What boundary does this network implement? Should it be able to handle the XOR problem?

2.6 Show that a multi-layer perceptron with linear activation functions is equivalent to a single layer network.

2.7 In an MLP, the global minimum is typically highly degenerate, with many equivalent solutions. As one example, hidden nodes j_1 and j_2 can swap roles, exchanging all weights in, biases, and weights out. For another example, consider and MLP with one hidden layer, and with tanh() as hidden node activation function. Assume that all weights into the hidden node j, and its bias, change sign. Show that the output $\mathbf{y}_n = \mathbf{y}(\mathbf{x}_n, \boldsymbol{\omega})$ can be preserved with a compensating change of all weights out from j. Find the corresponding compensation if the hidden nodes have the logistic activation function. In particular, show that the output bias must also change.

2.8 Consider a 1-hidden layer MLP, with logistic activation function in the hidden layer, i.e. $\varphi(a) = 1/(1 + \exp(-a))$. Show that there exists an equivalent network, which computes exactly the same function, but with hidden activation functions given by $\varphi(a) = \tanh(a)$.

Hint: First find a relation between the logistic and the tanh functions and then find the transformation of the weights needed.

2.9 Show, for a feed-forward network with tanh() as hidden activation function, and a summed-square error function as loss, that the origin in weight space is a stationary point of the loss function, apart from a possibility to train the output node biases.

2.10 Lets return to the multi-layer perceptron defined in Figure 1. Assume a summed-square-error function $E = \sum_n \sum_i (y_i(\mathbf{x}_n) - d_{ni})^2$, where d_{ni} are the targets. The activation functions are $g^h()$ and $g^o()$ for the hidden and output layer, respectively. Derive an expression for

$$\frac{\partial E}{\partial \omega^{ih}_{jk}}, \qquad \frac{\partial E}{\partial \omega^{ho}_{ij}}, \qquad \text{and} \qquad \frac{\partial E}{\partial \omega^{io}_{ik}}$$

2.11 *

Consider a binary classification problem in which the target values are $d \in \{0, 1\}$, with a network output $y(\mathbf{x}, \mathbf{w})$ that represents $p(d = 1|\mathbf{x})$, and suppose that there is a probability ϵ that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the loss function corresponding to the negative log likelihood. Verify that the usual cross-entropy error function is obtained when $\epsilon = 0$. Note that this loss function makes the model robust to incorrectly labelled data, in contrast to the usual loss function.

Hint 1: Introduce $k \in \{0, 1\}$ as the true class label.
Hint 2: Express $p(d = 1|\mathbf{x})$ in terms of $p(k|\mathbf{x})$ and ϵ, where d is the dataset label

(observed).

Chapter: Generalization

Network Ensembles

3.1 Suppose we have L trained models $y_i(\mathbf{x})$ $(i = 1, ..., L)$. All models try to approximate $h(\mathbf{x})$. We can write each mapping $y_i(\mathbf{x})$ as,

$$y_i(\mathbf{x}) = h(\mathbf{x}) + \epsilon_i(\mathbf{x})$$

and the corresponding error D_i for each model as,

$$D_i = \mathrm{E}\left[(y_i(\mathbf{x}) - h(\mathbf{x}))^2\right] = \mathrm{E}\left[\epsilon_i(\mathbf{x})^2\right]$$

where (here) $\mathrm{E}\left[\cdot\right]$ is defined as,

$$\mathrm{E}\left[\epsilon(\mathbf{x})^2\right] = \int \epsilon(\mathbf{x})^2 p(\mathbf{x}) d\mathbf{x}$$

and where $p(\mathbf{x})$ is the distribution of the input data $\mathbf{x}$. Let now D_{av} be the mean error of the L networks

$$D_{\text{av}} = \frac{1}{L}\sum_i D_i$$

Now introduce an ensemble in the form

$$y_{\text{ens}}(\mathbf{x}) = \frac{1}{L}\sum_i y_i(\mathbf{x})$$

The ensemble error is

$$D_{\text{ens}} = \mathrm{E}\left[(y_{\text{ens}}(\mathbf{x}) - h(\mathbf{x}))^2\right]$$

Now assume that the errors $\epsilon_i(\mathbf{x})$ are uncorrelated and have zero mean, i.e. $\mathrm{E}\left[\epsilon_i\right] = 0$ and $\mathrm{E}\left[\epsilon_i \epsilon_j\right] = 0, \quad i \neq j$. Under this assumption show that,

$$D_{\text{ens}} = \frac{1}{L} D_{\text{av}}$$

3.2 Consider a committee machine consisting of L MLP:s, with the following averaging,

$$y_{\text{com}}(\mathbf{x}) = \sum_{i=1}^{L} \alpha_i y_i(\mathbf{x})$$

where α_i is the weighting factor and $y_i(\mathbf{x})$ is the i:th committee member. Given a data set $\{\mathbf{x}_n, d_n\}_{n=1}^{N}$ the task is now to minimize the following error function

$$\begin{aligned} E &= \frac{1}{N}\sum_n \left(y_{com}(\mathbf{x}_n) - d_n\right)^2 \\ &= \frac{1}{N}\sum_n \left(\sum_i \alpha_i y_i(\mathbf{x}_n) - d_n\right)^2 \end{aligned}$$

with respect to α_i. One can solve this minimization problem using Lagrangian multipliers. Show that for the constraint $\sum_i \alpha_i = 1$, the solution is

$$\alpha_i = \frac{\sum_l (C^{-1})_{il}}{\sum_{jl}(C^{-1})_{jl}}$$

where C is the matrix with the matrix elements

$$C_{ij} = \frac{1}{N}\sum_n \epsilon_i(\mathbf{x}_n)\epsilon_j(\mathbf{x}_n)$$

with $\epsilon_i(\mathbf{x}_n) = y_i(\mathbf{x}_n) - d_n$.

Hint: The Lagrangian multiplier technique adds the constraint as an additional term to the function we want to minimize. E.g. $\lambda\left(\sum_i \alpha_i - 1\right)$

3.3 When we create a bootstrap sample from a dataset of size N, we randomly select samples N times, with replacement. Some samples are picked many times, others are never picked.

a) What is the probability that pattern i isn't selected in the first pick for the bootstrap sample?

b) What is the probability that pattern i isn't selected to the bootstrap sample in any of the N picks?

c) Show that the large N limit of the result in (b) is $1/e$.

Chapter: CNN, Autoencoder and GAN

4.1 How many trainable parameters are there in a convolution filter that has a $K \times K$ kernel acting on C channels? Each channel is a 2D picture of size $W \times H$.

4.2 A CNN has three input channels with 6x6 images. A convolutional layer consists of 2 filters with 3x3 kernels and stride 1, without padding. This is followed by 2x2 max pooling with stride 2. The result is flattened and sent as input to an MLP.

a How many input nodes does the MLP get?

b How many trainable parameters does the CNN have, before the MLP?

c The same input images are convoluted with the same 2 filters, but now with zero-padding. This is followed by the same max pooling. What are the new answers to (a) and (b)?

4.3 *

A filter in layer l in a CNN has kernel size K_l and stride S_l. If the input and hidden images are multi-dimensional, consider symmetric kernels and strides, so that the same values apply to all dimensions.

a) Show that size of the receptive field R_l for a node created by the filter in l and the displacement D_l between receptive fields of neighbouring nodes can be determined recursively by

$$\begin{aligned} D_1 &= S_1 \\ R_1 &= K_1 \\ D_{l+1} &= S_{l+1} D_l \\ R_{l+1} &= R_l + (K_{l+1} - 1) D_l \end{aligned}$$

provided $S_l \leq K_l$.

b) A CNN begins with a 3x3 convolution with stride 1, followed by a 2x2 max pooling with stride 2, and then a 3x3 convolution with stride 2. What is the size of the receptive field for a node after the first convolution, the max pooling, and the last convolution, respectively?