

Examination questions for FYTN14 / EXTQ40 2019-01-16 (14.00-19.00)

Approximately 17.5 points are required for passing the exam.

1. No derivations or explanations are needed for the following 5 questions!

- a) You have a categorical input variable that can take the following “values”: 'winter', 'spring', 'summer' and 'fall'. Give a one-hot encoding of 'winter'. (1p)
- b) A smooth version of the ReLU activation is given by $\log(1+e^x)$. True or false? The derivative of this activation function is the logistic activation function. (1p) (Note: $\log()$ is the natural logarithm.)
- c) True or false? The dropout regularization technique works by randomly removing nodes (input and/or hidden) in the network for every pattern used during training. (1p)
- d) How many distinct patterns can a Hopfield model with N nodes express? (1p)
- e) You have an MLP with two hidden layers and three **linear** nodes per hidden layer. This MLP is used for binary classification problems. Can the decision boundary be non-linear? (1p)

2. Figure 1 shows an MLP with one hidden layer and a single output. The activation function for the

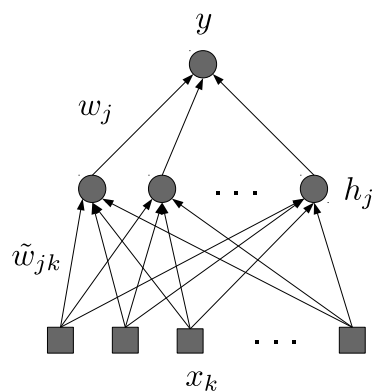


Figure 1:

hidden layer is φ_h and the activation function for the output layer is φ_o .

- Write down the function this network implements, i.e. the output y as a function of inputs \mathbf{x} and the weights. Let the bias to the output activation function be called θ , and the bias to hidden node j be called θ_j . (1p)
- Given a dataset of inputs $\{\mathbf{x}\} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \dots)$ and targets $\{d\} = (d_1, d_2, d_3 \dots)$, we want to train our network so that outputs $y(\mathbf{x}_n)$ are “as similar as possible” to targets d_n . One approach is to minimize an error function $E(\{y\}, \{d\})$. Give examples of two commonly used error functions and describe what kind of targets they are suitable for. (2p)
- Using gradient descent as the approach to minimize E , derive the update $\Delta\theta_j$ for the biases to the hidden nodes. Assume all derivatives $\frac{\partial E}{\partial y_n}$ to be known. (2p)

3. Suppose you have split a dataset into a training part that you use to train an MLP, and a validation part that you use to examine the MLP's performance on previously unseen data.

You train different MLP:s, all with one hidden layer but with different number of hidden nodes, and record the error (value of your error function). The results are shown in the table below:

Number of hidden nodes	2	4	8
Training E	3.4	3.1	2.3
Validation E	3.5	3.3	3.9

- a) In the table we can compare training and validation error for each number of hidden nodes. We can also see how each error evolves as the number of hidden nodes increases. Discuss the observed relations. Are they reasonable? Why? In particular, introduce the concept of overtraining in your discussion. (2p)
- b) Assume we are provided more data, so that we can train our networks with a larger training set. We keep the validation set unchanged. We train new MLP:s on the larger training set, trying the same number of hidden nodes as above. How do you expect the training and validation errors to change? The error function is of the form $E = \frac{1}{N} \sum_n E_n$, where N is the number of samples and E_n a function depending only on sample n . (1p)
- c) Restricting the number of hidden nodes is a way to *regularize* the network, in order to avoid overtraining. Give two other examples of regularization and describe how they work. (2p)

4. A single validation dataset may be too small to give a reliable estimate of how well a MLP performs on unseen data (the “generalization” performance).

- a) Describe the bootstrap method to get a more reliable estimate of the generalization performance. (2p)
- b) As in problem 3, the validation dataset can be used to perform *model selection*, i.e., determine *hyper-parameters* such as hidden nodes, learning rate in gradient descent, etc.. When optimal hyperparameters have been found, we can use them as we train a new network on the entire dataset (training+validation). Explain why the validation result from the model selection step is not a good estimator of the performance of this final MLP on new data! (1p)
- c) Describe a way to get an unbiased estimate of the generalization performance of an MLP trained with optimized hyperparameters. (2p)

5. Recurrent networks!

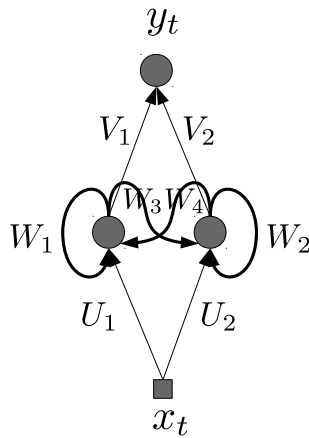


Figure 2:

- We can deal with sequence data using an ordinary MLP without any feedback connections. We call this network a *time delay network*. Explain how the time delay network is trained for the task of predicting the next value of a simple one dimensional time series $x(t), t = 1, \dots, T$. (2p)
- Figure 2 shows a recurrent network with one input and one output node. It has two hidden nodes and four feed-back connections, marked W_1, W_2, W_3, W_4 . Unfold this network one time step in time. Mark all weights in the unfolded network with the correct labels, i.e. $(U_1, U_2, W_1, W_2, W_3, W_4, V_1, V_2)$ (2p)
- Recurrent networks, as the one shown in figure 2, can be used to model sequence data. When dealing with long sequences it can however be tricky to train the unfolded network because of the vanishing gradient problem. Why can gradients “vanish”? (1p)

6. Convolutional neural networks (CNN)!

a) We can define hyper-parameters to be all “settings” (except the weights) that defines the network and how to train it. CNNs have many hyper-parameters. Assume you should construct a CNN for an image classification task. List **different** kinds of hyper-parameters when defining and training your CNN. (0.5 points per hyper-parameter, max 3p)

b) Assume you have an input image (4x4) as below.

$$\begin{bmatrix} 1 & 2 & -2 & 3 \\ -1 & 3 & 1 & -1 \\ 5 & 0 & -2 & 1 \\ -1 & 1 & -3 & 2 \end{bmatrix}$$

Your convolutional filter looks like this:

$$\begin{bmatrix} 1 & 3 \\ 3 & 0 \end{bmatrix}$$

Use stride = 2, no zero-padding and assume a ReLU activation function. Compute the resulting filtered image (after activation, but before a possible max-pooling). You can assume a zero bias weight! (1p)

c) You have the following setup for a binary image classification task using CNNs. Image size: 10x10. One convolutional layer: 1 kernel, size 3x3 (stride = 1, no zero-padding) + max-pooling 2x2 (stride = 2). Dense layer: 2 hidden nodes, Output layer: 1 single output node. How many weights (including bias weights) does this CNN have? (1p)

7. Decision boundaries!

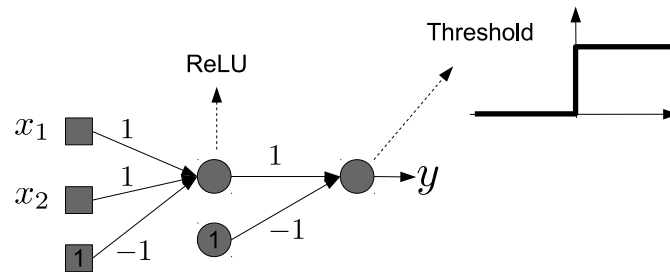


Figure 3:

Figure 3 shows a network with two inputs, a single ReLU hidden node and a single output node with a threshold activation function. This network is used for a binary classification. The values of the weights and bias-weights are shown in the figure.

Figure 4 shows two simple perceptrons with logistic output activation functions, also used for binary classification.

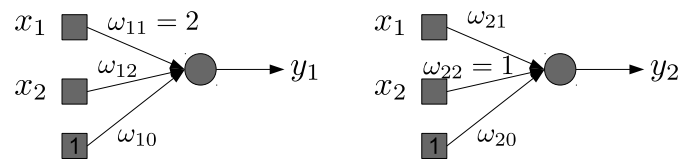


Figure 4:

They form an ensemble according to,

$$y_{\text{ens}} = \frac{1}{2}(y_1 + y_2),$$

and the cut value 0.5 is used to define the two classes for y_{ens} . Find one set of weights $(\omega_{12}, \omega_{10}, \omega_{21}, \omega_{20})$, such that the ensemble y_{ens} and the first network implements the same decision boundary. (5p)

Good Luck!

Mattias & Patrik