Examination questions for FYTN14 / EXTQ40 2018-01-10 (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!

- How many **bias**-weights do you have in an MLP with the architecture $N - M - O$ (N inputs, M hidden nodes and O output nodes? (1p)
- How is the rectifier linear unit defined (use math or draw a figure)? (1p)
- How many unique weights do you have in a Hopfield model with N nodes? (1p)
- What is the natural output activation function for regression type of problems? (1p)
- Write down a suitable error function for an autoencoder with continuous inputs. (1p)

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

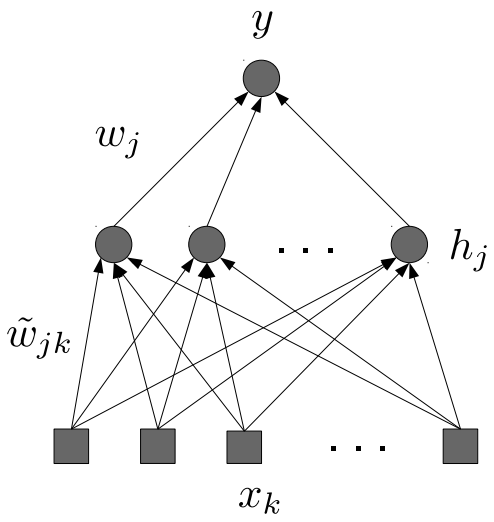


Figure 1:

- Write down the function this network implements, i.e. the output y as a function of inputs and the weights. Note! You do not have to explicitly show the bias nodes/weights. (2p)
- Given a dataset of inputs $(\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(N))$ and corresponding targets $(d(1), d(2), ..., d(N))$, write down the mean summed square error function E. (1p)
- Using gradient descent as the approach to minimize E, derive the weight updates Δw_j for the hidden to output weights. (2p)

3. Assume you have a regression problem with a limited amount of training data (e.g. the regression problem from the first exercise). Overtraining is possible due to noisy data.

Figure 2:

- Illustrate overtraining in the graph above (figure 2) by drawing the training and validation error as a function of the number of hidden nodes. Training error here means the error measured on the training dataset after training with a given number of hidden nodes. Validation error here means the error measured on a separate dataset (not part of the training data) after training with a given number of hidden nodes. (1p)
- Give three different methods that can be used to avoid/reduce overtraining and describe how they work. (3p)
- When using a regularizer that acts on the magnitude of the weights we do not include the bias weights. Why? (1p)

4. Model selection is an important part of machine learning! We can say that model selection finds suitable values for hyper-parameters such as regularization parameters or network design parameters. When doing model selection, the goal is to find a network (with all hyper-parameters) that has optimal *generalization* performance.

- What is meant by generalization performance? (1p)
- Why is the performance measured on the training dataset not a good estimator of the generalization performance? (1p)
- Describe how to estimate the generalization performance using K-fold cross validation. (2p)
- You are now performing a model selection experiment, for a classification problem, where you want to find a good value for the hidden layer dropout probability p_{keep}. Your performance measure is the overall classification accuracy (in %). You perform a series of experiments with different values of p_{keep} and present the results in a diagram (figure 3). In this diagram, draw the expected training accuracy, i.e. the classification accuracy measured on the training data. Also, draw an idealized curve of the validation accuracy, i.e. the classification accuracy measured by e.g. K-fold cross validation and mark the optimal value of p_{keep}. (1p)

Figure 3:

5. Figure 4 shows a simple recurrent network. Here an input sequence $x_1, x_2, ..., x_T$ is producing an output sequence $y_1, y_2, ..., y_T$ using a single hidden node with a feedback weight (W). The hidden activation function is φ_h and the output activation function is φ_o.

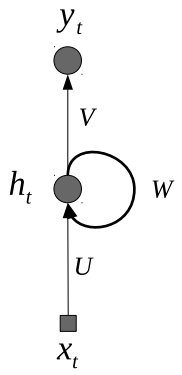


Figure 4:

- Write down the function this network implements, i.e. the output y_t as a function of the input x_t and the weights. Note! You do not have to explicitly show the bias nodes/weights. (1p)

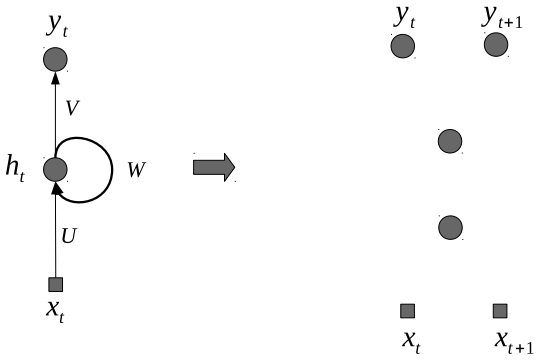
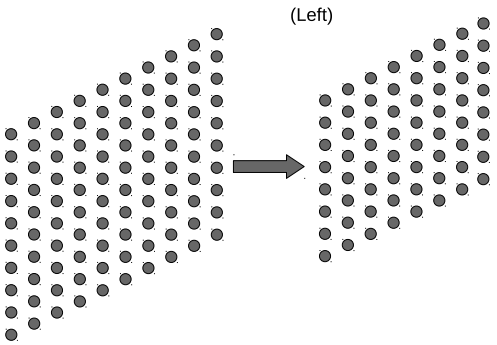


Figure 5:

- This recurrent network can be unfolded in time! Unfolding in time results in a corresponding MLP where the number of hidden layers is the number of sequence steps "unfolded". In figure 5 our simple recurrent network is unfolded in time, two time steps, and represented as a two hidden layer MLP with two inputs and two outputs. Draw all weights in this network and mark them correctly with U, V, W, also mark the hidden nodes with h_t, h_{t+1}. You do not have to draw bias weights! (2p)
- The above network can be trained using the so called backpropagation-through-time (BPTT) method. What does that mean? (1p)
- Describe one problem that may occur using BPTT for long sequences? (1p)

6. Assume we have images of size 10x10 pixels. We are going to use a convolutional neural network (CNN) for the analysis of these images. In the first convolutional layer we will use a single 3x3 kernel (stride = 1 and no zero padding). After convolution we will have a 8x8 layer of "hidden" nodes, as illustrated in figure 6 (left graph). A corresponding fully connected MLP would have 100x64 non-bias weights, while the CNN only have 9 **unique** non-bias weights in this first layer.



(Right)

0.4	6.3	2.3	3.7
1.0	7.4	1.9	1.3
1.4	2.3	5.2	7.6
2.4	6.3	0.6	0.7

Figure 6:

- Explain this reduction in the number of weights, as compared to the fully connected MLP, in terms of *sparse connectivity* and *shared weights*. (3p)
- After convolution one often applies a pooling operation, where maxpooling is very common. In figure 6 (right graph) you see the (numerical) output from a convolution operation. Apply a 2x2 maxpooling filter with stride 2 and write down the new layer that this operation produces. (1p)
- The rectified linear unit (RELU) is very common as an activation function in CNNs. Give one property of the RELU that is useful in CNNs (or other deep networks). (1p)

7. We know that a single perceptron (logistic output) trained on a binary classification problem only can implement a hyperplan as the decision boundary. This can be seen by looking at the condition,

$$y(\mathbf{x}, \mathbf{w}) = A$$

where A is a constant $\in [0, 1]$ and $y(\mathbf{x}, \mathbf{w})$ is the perceptron output given by

$$y(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{x}^T \mathbf{w})}$$

Note: The threshold weight is included in the expression $\mathbf{x}^T \mathbf{w}$.

Let's now consider a simple ensemble of two such perceptrons. The question is whether such an ensemble is linear or not? We have

$$y_c(\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2) = c_1 y(\mathbf{x}, \mathbf{w}_1) + c_2 y(\mathbf{x}, \mathbf{w}_2)$$

where we demand that $c_1 + c_2 = 1, c_1 > 0, c_2 > 0$ and $\mathbf{w}_1 \neq \mathbf{w}_2$. Now look at the corresponding condition for the boundary of the ensemble,

$$y_c(\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2) = A$$

and give further conditions for c_1, c_2 and A in order to have a linear boundary for y_c (5p).

Good Luck !

/Mattias