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)

Answers:

1. Each layer, except the output layer have a bias node connecting to all nodes in the next layer. Hence, $M + O$ bias weights.
2. $\max(0, x)$
3. $\frac{N*(N-1)}{2}$
4. linear, $\varphi(x) = x$
5. $$E(\mathbf{w}) = \frac{1}{NK} \sum_{n=1}^{N} \sum_{i=1}^{K} \left(y_i(\mathbf{x}(n), \mathbf{w}) - x_i(n) \right)^2$$

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.

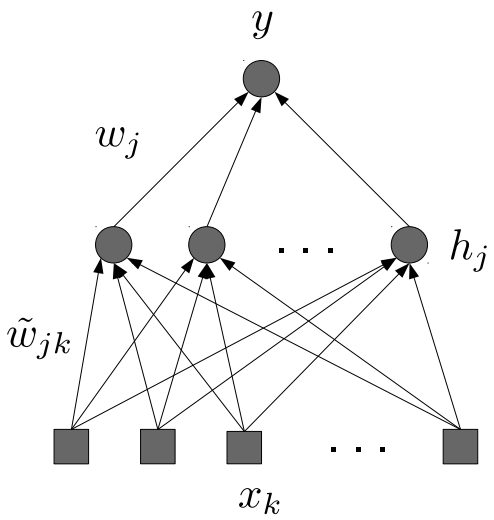


Figure 1:

- 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)

Answers:

1.
$$y(\mathbf{x}, \mathbf{w}) = \varphi_o\Big(\sum_j w_j \varphi_h\big(\sum_k \tilde{w}_{jk} x_k\big)\Big) = \varphi_o\Big(\sum_j w_j h_j\Big)$$

2.
$$E(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\big(y(\mathbf{x}(n), \mathbf{w}) - d(n)\big)^2$$

3. Just write the error as:
$$E(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\big(y(\mathbf{x}(n), \mathbf{w}) - d(n)\big)^2 = \frac{1}{2N}\sum_{n=1}^{N}\big(y(n) - d(n)\big)^2 = \frac{1}{2N}\sum_{n=1}^{N} E(n)$$

Gradient descent general update formula

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} \qquad \text{where} \qquad \frac{\partial E}{\partial w_j} = \frac{1}{2N} \sum_n \frac{\partial E(n)}{\partial y(n)} \frac{\partial y(n)}{\partial w_j}$$

$$\frac{\partial E(n)}{\partial y(n)} = 2\big(y(n) - d(n)\big)$$

$$\frac{\partial y(n)}{\partial w_j} = \varphi_o'\Big(\sum_{j'} w_{j'} h_{j'}\Big) h_j \qquad so \qquad \Delta w_j = -\eta \frac{1}{N} \sum_n E(n) \varphi_o'\Big(\sum_{j'} w_{j'} h_{j'}\Big) h_j$$

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.

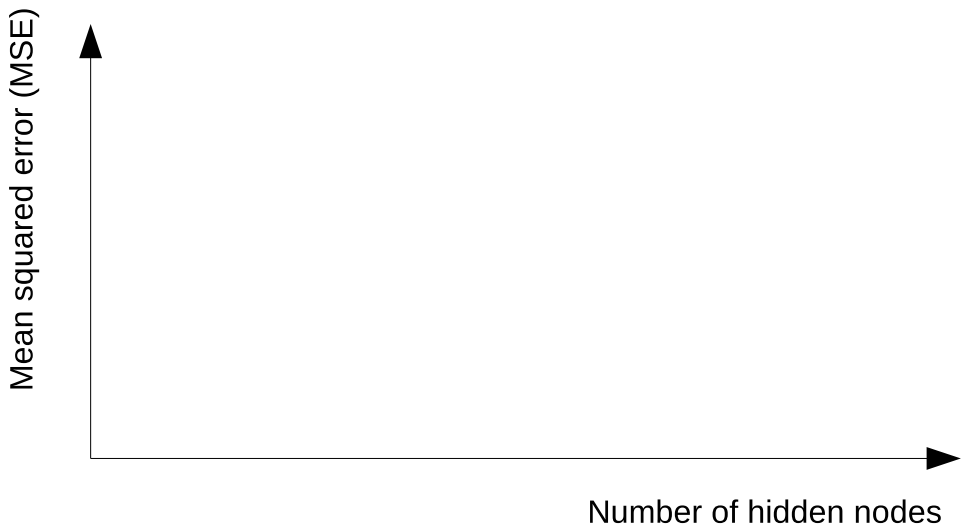


Figure 2:

- 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)

Answers:

1. Se figure 3
2. – Early stop: Limit the overtraining by simply stop the minimization of the error function when the validation error starts to increase.
   – Methods where you add terms to the error function, $\hat{E} = E + \alpha\Omega$. (i) weight decay $\Omega = \sum_i w_i^2$ (ii) lasso $\Omega = \sum_i |w_i|$ (iii) modified weight decay $\Omega = \sum_i (w_i/w_o)^2/(1+(w_i/w_o)^2)$. They all work by constraining the size of the weights in the network. Small weights means limited functionality of the network and less overtraining.
   – A bit similar to the above, Maxnorm. Here all weights weight vectors are constrained by $|\mathbf{w}_i| < c$. Same idea as above.

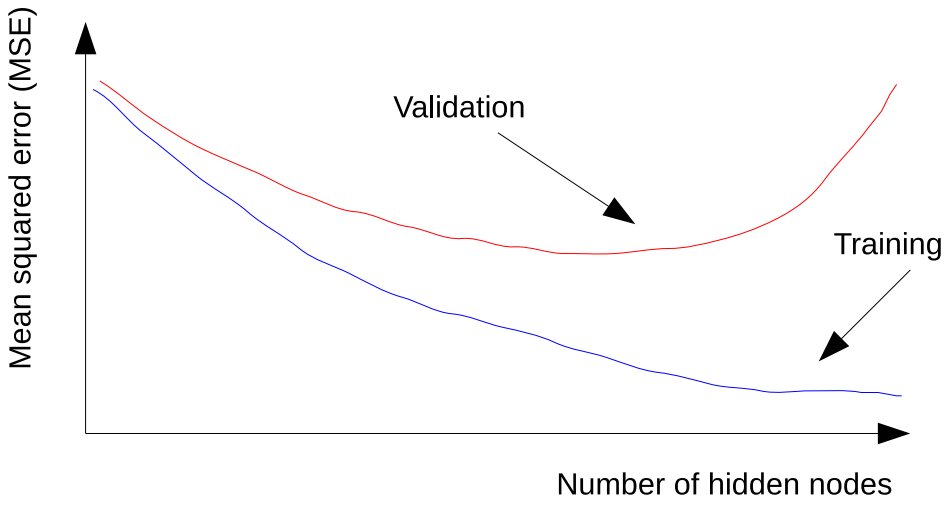


Figure 3:

- Ensemble techniques. Here we take advantage of the fact that an ensemble want overall good networks that disagrees as much as possible. So an ensemble of many slightly overtrained members reduces overtraining of the ensemble.
- Dropout. Temporarily removing random nodes during training. Can be seen as a huge ensemble of many different architectures.

3. Bias weights do not increase e.g. the curvature of decision boundaries or the ability to make a perfect fit to a noisy regression problem. It rather moves the decision boundary and adds constants to regression fits. Bias nodes must therefore be allowed grow.

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 4). 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 4:

Answers:

1. Like this:
   (i) The performance of the model on unseen data coming from the same underlying distribution as the training data.

   (ii) The performance measured on a dataset that was not part of any training or decision process when developing the model. The new dataset should be sampled from the same distribution as the training dataset.

2. We know that a neural network is a universal approximator so we can in principle found a high training performance for any given problem. Hence the training performance does not reflect the generalization performance.

3. A figure of some kind is probably useful here. But in words, divide the dataset into K parts with equal sizes. Systematically use each of the K parts as a validation dataset, while training on the remaining part. The average of the K validation performances is an estimation of the generalization performance.

4. See figure 5

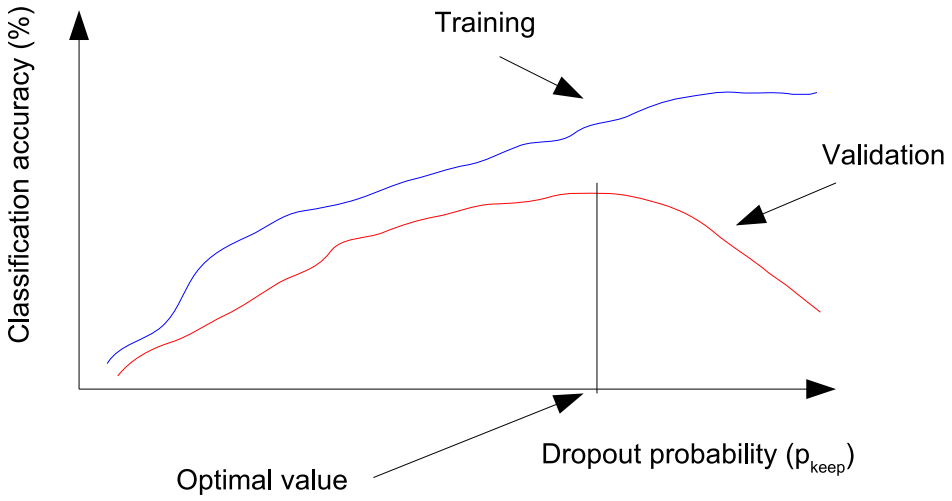


Figure 5:

5. Figure 6 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.

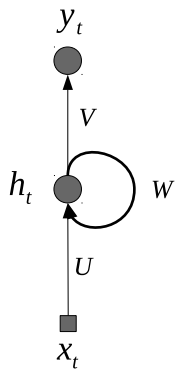


Figure 6:

- 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)

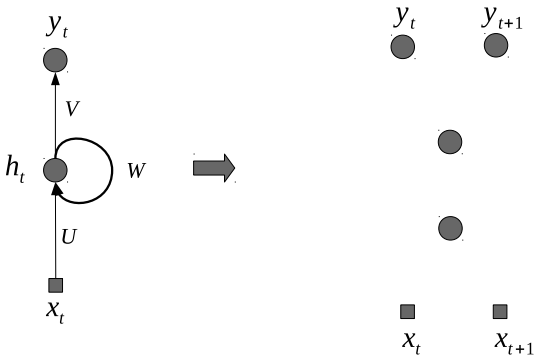


Figure 7:

- 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 7 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)

Answers:

1. The hidden node is given by

$$h_t = \varphi_h(Ux_t + Wh_{t-1})$$

and the output is then given by

$$y_t = \varphi_o(Vh_t)$$

2. See figure 8

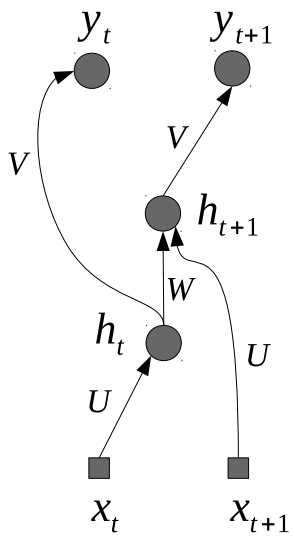
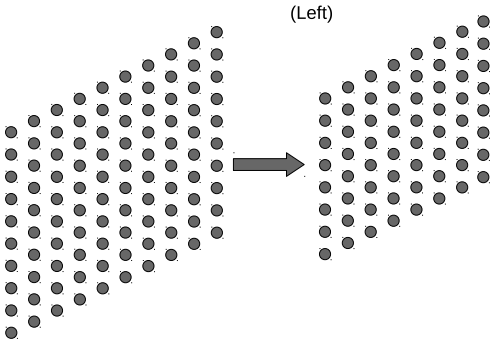


Figure 8:

3. Since we can unfold the recurrent network for a given sequence and that results in an MLP. An MLP can be training using backpropagation given input and target data. For the recurrent network that means having an input sequence and corresponding target values (sequence or other "target"). Training an unfolded recurrent network using backpropagation is called backpropagation through time.

4. For a long sequence we will have a very deep MLP (many hidden layers). Training such a deep MLP using backpropagation the problem of vanishing gradients may occur. When doing backpropagation in a deep MLP we will end up with expressions for the weight update where many derivatives of activation functions are multiplied together. If activation functions are sigmoidal, then we can have a problem of gradient becoming small.

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 9 (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 9:

- 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 9 (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)

Answers:

1. (i) MLP explanantion. Each of the hidden nodes is only connected to a limited number of input nodes. In our case 3x3 input nodes (the kernel size). This we call sparse connectivity and reduces the number of weights from 100*64 to 9*64. Furthermore all of the hidden nodes share the same set of weights. This we call weights sharing and then reduces the unique number of weights from 9*64 to 9, excluding bias weights.

   (ii) Kernel explanantion. Each pixel in the hidden layer is the result of a convolution with a kernel of a size smaller that the original image, which gives rise to the sparse connectivity. The fact that the same kernel parameters are used for all pixels in the hidden layer is giving us the weight sharing.

2. 2x2 matrix. First row: 7.4, 3.7, second row: 6.3, 7.6

3. Examples: (i) no vanishing gradients. (ii) fast computation (iii) sparse connectivity leadning to reduced overtraining.

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).

Answer:

We have that

$$y_c(\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2) = \frac{c_1}{1 + \exp(-\mathbf{x}^T\mathbf{w}_1)} + \frac{c_2}{1 + \exp(-\mathbf{x}^T\mathbf{w}_2)}$$

I will now use a simplified notation where $xw_1 = \mathbf{x}^T\mathbf{w}_1$ and $xw_2 = \mathbf{x}^T\mathbf{w}_2$. The condition for the boundary of the ensemble now reads,

$$\frac{c_1}{1 + \exp(-xw_1)} + \frac{c_2}{1 + \exp(-xw_2)} = A$$

which can be written as

$$c_1\big(1 + e^{-xw_2}\big) + c_2\big(1 + e^{-xw_1}\big) = A\big(1 + e^{-xw_1}\big)\big(1 + e^{-xw_2}\big)$$

Collect terms

$$1 - A + (c_1 - A)e^{-xw_2} + (c_2 - A)e^{-xw_1} - Ae^{-x(w_1+w_2)} = 0$$

where I have used that $c_1 + c_2 = 1$. Generally an expression like $e^{-\mathbf{x}^T\mathbf{w}_1} + e^{-\mathbf{x}^T\mathbf{w}_2} + const = 0$ will not correspond to $\mathbf{x}$ lying on a hyper-plane for all possible values of the weights. We need to get rid of some terms. We see from the above equation that if $c_1 = A$ and $c_2 = A$ we are down to the following expression.

$$1 - A - Ae^{-x(w_1+w_2)} = 0$$

which corresponds to a hyper-plane. Now since $c_1 + c_2 = 1$ this leaves us with only one possible solution $c_1 = c_2 = A = 1/2$.

Good Luck !

/Mattias