Examination questions for FYTN14 / EXTQ40 2020-09-11 (11.00-16.00)

Approximately 17.5 points are required for passing the exam.

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

a) How many **bias-nodes** do you have in an MLP with the N hidden layers? (1p)

b) Compute the derivative of the $1/(1+\exp(-x))$ activation function and express the derivative in terms of the function value. (1p)

c) True of false? Bagging is a method to generate datasets used to train networks in an ensemble? (1p)

d) What is the natural output activation function for binary classification problems? (1p)

e) Give an example of how to normalize continuous input values before training a neural network. (1p)

Answers:

1a $N+1$

1b $f(x) = 1/(1+\exp(-x))$, $f'(x) = f(x)(1-(f(x))$

1c True

1d The sigmoid. $f(x) = 1/(1+\exp(-x))$

1e (i) Subtract the mean and divide by the standard deviation. (ii) Push the values into the range [-1,1] or [0,1]

2. Figure 1 shows an MLP with one hidden node and a single output. The hidden activation function is φ_h and the output activation function is φ_o. This network have direct input to output weights.

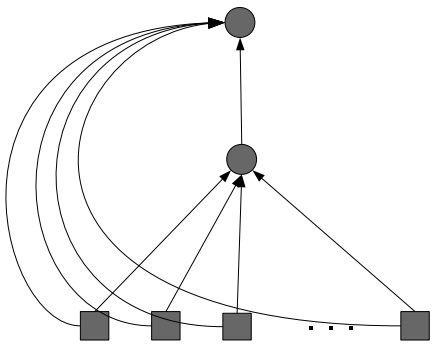

Figure 1:

a) Introduce symbols for the different weights and write down the function this network implements, i.e. the output as a function of inputs and the weights. Note! You should explicitly add the bias weights even though are not visible in the figure. (2p)

b) Write down the mean squared error function and the cross-entropy error function, and describe what kind of problems they are suitable for. (1p)

c) Many weight update approaches are based on the gradient of the error function. Using the symbols you introduced in (a), write down the derivative of the error function with respect to an input to hidden weight. (1p)

d) Some gradient descent methods (such as: momentum terms; "Rprop"; "Adam") include information about earlier gradients to hopefully improve weight updates. Describe one such algorithm in more detail than this brief introduction. (1p)

Answers:

2a Input to hidden weights ω_k, hidden node bias weight ω_0, input to output weights $\tilde{\omega}_k$, output node bias $\tilde{\omega}_0$, and hidden to output weight $\hat{\omega}$. The mapping is then

$$y = \varphi_o(\hat{\omega}\varphi_h(\sum_{k=1} x_k\omega_k + \omega_0) + \sum_{k=1} x_k\tilde{\omega}_k + \tilde{\omega}_0)$$

where x_k are the inputs.

2b Mean squared error function, $E = \frac{1}{N}\sum_n (y_n - d_n)^2$, is used for regression problems (with continuous outputs).

Cross-entropy error function, $E = \frac{1}{N}\sum_n [d_n \ln(y_n) + (1 - d_n)\ln(1 - y_n)]$, is used for binary classification problems with 0/1 target values.

2c We can write

$$\frac{\partial E}{\partial \omega_k} = \sum_n \frac{\partial E}{\partial h_n}\frac{\partial h_n}{\partial \omega_k}$$

where $h_n = \hat{\omega}\varphi_h(\sum_{k=1} x_{nk}\omega_k + \omega_0) + \sum_{k=1} x_{nk}\tilde{\omega}_k + \tilde{\omega}_0$. We can evaluate $\frac{\partial h_n}{\partial \omega_k}$,

$$\frac{\partial h_n}{\partial \omega_k} = \hat{\omega}\varphi_h'(\sum_{k'=1} x_{nk'}\omega_k + \omega_0)x_{nk}$$

2d The **momentum term** is adding a part of the previous to the current one. In a plateau-like region, when several updates in a row have the same sign, it will increase the minimization process. **Rprop** introduces individual dynamical learning rates for each weight according to, $\Delta\omega_k = -\eta_k \mathrm{sgn}(g_k)$, where η_k are updated to some defined schedule. **Adam** is based on a running average of the gradient, just as momentum, but also by a running average of the square of derivatives. Adam has three hyper-parameters η, β_1 and β_2.

3.

a) What is over-training and why can it be a problem in machine learning? (2p)

b) "Early stopping", "L2 (weight decay) regularization" and "drop-out" can be used to reduce over-training. Describe each of them and explain why they can prevent over-training. (3p)

Answers:

3a Overtraining is associated with the problem of a machine learning model becoming to specialized on the training data and cannot generalize to another dataset within the same domain. There are many things that can cause overtraining, such as the presence of noise in the data, small training dataset, too flexible machine learning model, too many irrelevant input features. Overtraining can be detected by estimating the generalization performance during training. The main concern with overtraining is the fact that the trained model will not perform well on new data.

3b **Early stopping** will monitoring the performance on a validation dataset during training and simply stop the training when there are signs that the validation performance has reached an optimum. **L2** regularization will add a term to the loss function that is the sum of (a selection of) squared weights in the network. This will prevent weights from growing to large values during training. Large weights are often needed for the network to overtrain, hence L2 regularization will prevent overtraining. **Drop-out** is a technique that randomly removes hidden nodes (or input nodes) during training. A new random set of nodes is selected for every pattern during the back-propagation procedure. The hyper-parameter for drop-out is the probability that a hidden node is "dropped out" for a given input pattern. The drop-out method prevents hidden nodes to "collaborate" in order to focus on details in the training dataset (overtraining).

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.

a) We can define hyper-parameters to be all "settings" (except the weights) that define the network and how to train it. Assume you should construct a MLP for a binary classification task. List different kinds of hyper-parameters when defining and training your MLP. (0.5 points per hyper-parameter, max 2p)

b) When doing model selection we typically need three different data sets. Let's call them D1, D2 and D3. Explain how to use the different datasets to find your optimal network and estimate the generalization performance. (3p)

Answers:

4a Number of hidden nodes in each hidden layer, number of layers, choice of regularization method, the different parameters for the regularization parameters (L2, dropout etc), possible use of skip layer connections, choice of activation function for the hidden layers, learning rate, choice of optimization method.

4b Here is a common usage of these three datasets: D1 (Training). Use this for training the weights in your network, given a complete set of hyper-parameters. D2 (Model selection). Make different choices of hyper-parameters, train on D1 and select the optimal set of hyper-parameters that gives optimal performance on D2. Combine D1 and D2 to one dataset and train a final model for the selected set hyper-parameters. D3 (Test). Evaluate this model on D3. Note: This procedure will not work if you are using early stop, where you always need D2 to stop the training process on D1.

5.

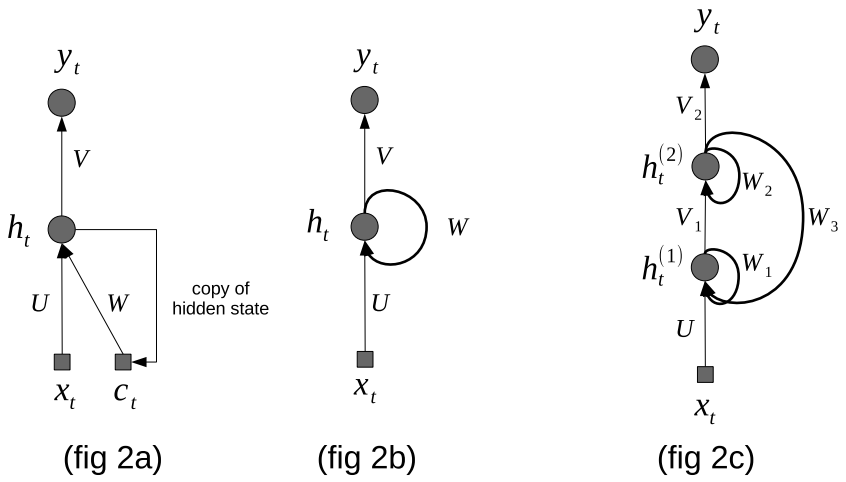

Figure 2:

a) Figure 2 (2a and 2b) shows two simple recurrent networks. Fig. 2a is the Elman type (ET) and fig. 2b is the general type (GT). 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. The ET network works as follows:

$$\begin{aligned} y(t) &= g_{\mathrm{o}}\left(Vh(t)\right) \\ h(t) &= g_{\mathrm{h}}\left(Ux(t) + Wc(t)\right) \\ c(t) &= h(t-1) \end{aligned}$$

And the GT network have these update equations:

$$\begin{aligned} y(t) &= g_{\mathrm{o}}\left(Vh(t)\right) \\ h(t) &= g_{\mathrm{h}}\left(Ux(t) + Wh(t-1)\right) \end{aligned}$$

Although they appear to be same there is a difference between the two networks when evaluating the derivative,

$$\frac{\partial h(t)}{\partial W}$$

Explain the difference between ET and GT when evaluating this derivative (2p)

b) Figure 2 (2c) shows another recurrent network with two hidden nodes. Again the input sequence $x_1, x_2, ..., x_T$ is producing an output sequence $y_1, y_2, ..., y_T$. Unfold this network two time steps, i.e. show the unfolded network for inputs x_1, x_2, x_3 (mark all weights in the unfolded network with the labels $U, V_1, V_2, W_1, W_2, W_3$). (2p)

c) The GT network type can be used to model sequence data. When dealing with sequences with long time dependencies it can however be tricky to train the unfolded GT network because of the vanishing gradient problem. What is that? (1p)

Answers:

5a For the Elman network we assume that $c(t)$ does not depend on W, hence the evaluation of $\frac{\partial h(t)}{\partial W}$ becomes easy. For the GT type we treat the derivative in a "proper" way, which means that we need to evaluate $\frac{\partial h(t)}{\partial W}$ through out the unfolded network.

5b See figure 3.

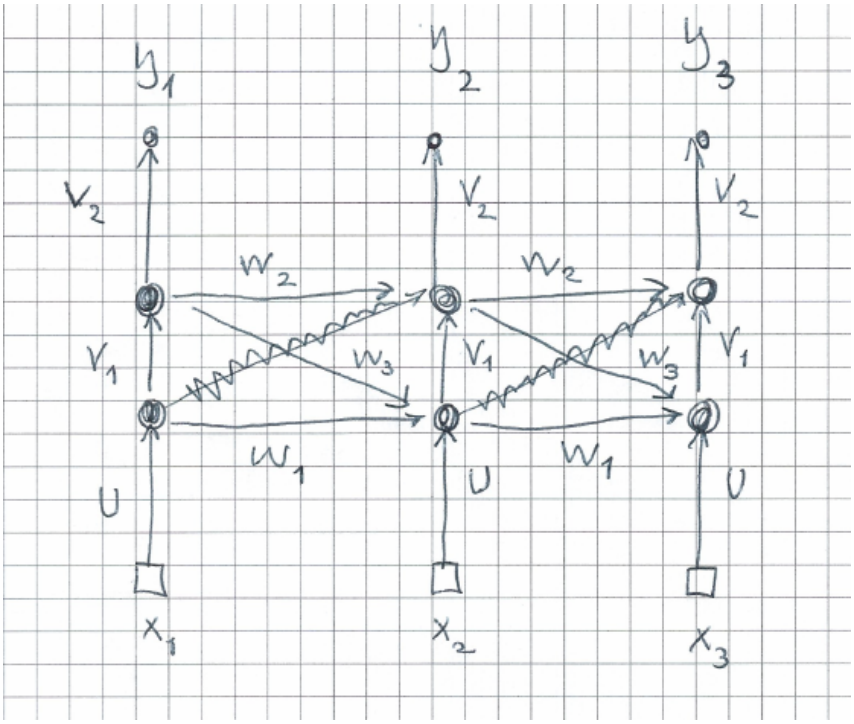

Figure 3:

5c The unfolded network has as many hidden layers as the length of the sequence. When computing the gradient through this (deep) network we end up multiplying many term. If these terms are < 1 we can get very small (or vanishing) gradients.

6. Autoencoder: Figure 4 shows an example of an autoencoder.

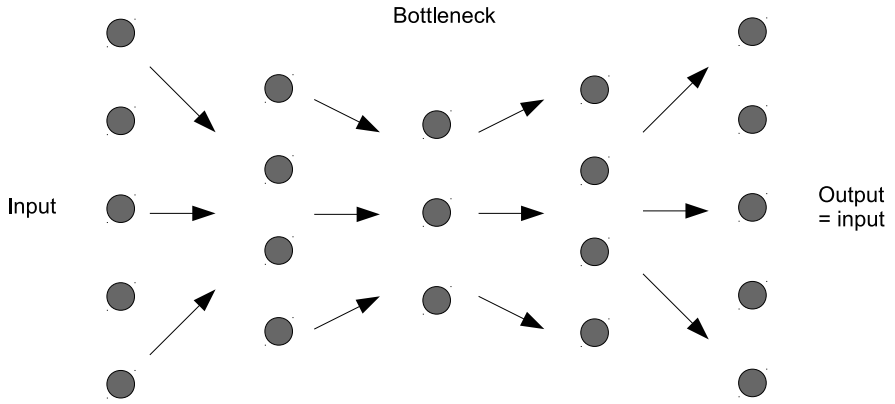

Figure 4:

a) Describe one usage of the autoencoder? (1p)

b) Assume that all input data for the autoencoder are binary (0/1) values. Write down a suatable error function for this autoencoder and explain why this error function. (2p).

CNN:

a) You have the following setup for a binary image classification task using CNNs. Image size: 11x11. One convolutional layer: 1 kernel, size 5x5 (stride = 2, no zero-padding) + max-pooling 2x2 (stride = 1). Dense layer: 3 hidden nodes, Output layer: 1 single output node. How many weights (including bias weights) does this CNN have? (2p)

Answers:

6a The autoencoder can be used to find an efficient representation of the input data. It can therefore be used in a semi-supervised learning setting. We can learn an efficient representation of the data using a large unlabeled dataset, followed by a supervised learning step, using this new data representation, with a possibly smaller labeled dataset.

Other usages can be outlier detection, where the reconstruction error will be large for cases that does not "fit" the latent space representation of the data. Such cases can be interpreted as outliers.

A third usage can be imputation of missing data, where the autoencoder can learn to reconstruct missing inputs.

6b If we have binary (0/1) values they most likely represent a categorical feature with two possibilities. Here we know that the most appropriate error function the cross-entropy. We must however treat all binary input variables independently of each other. Hence the following error function for the autoencoder

$$E = -\frac{1}{N}\sum_n \sum_k \left[x_{nk} \ln(y_k(\mathbf{x}_n)) + (1 - x_{nk}) \ln\left(1 - y_k(\mathbf{x}_n)\right)\right]$$

where $y_k(\mathbf{x}_n)$ is the k:th output from the autoencoder and $\mathbf{x}_n$ is the n:th input pattern.

6c (CNN) The 5x5 kernel gives $5 * 5 + 1 = 26$ weights. After the convolution process we have a 4x4 filtered image. Applying 2x2 max pooling with stride 1, gives as an 3x3 image. There are no weights in connection with max pooling. The 3x3 image is flattened to a 9 input - 3 hidden nodes - 1 output MLP. This networks has $(9 + 1) * 3 + (3 + 1) * 1 = 30 + 4 = 34$ weights. In total 60 weights.

7. Remember the Hopfield model (see figure 5)!

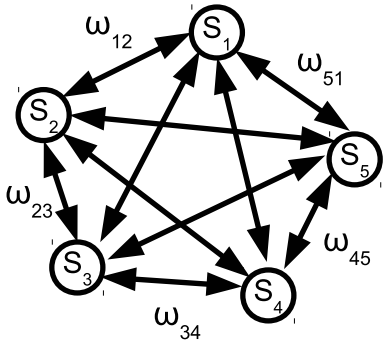

Figure 5:

It had the following properties,

- The nodes s_i in the network are binary with $s_i \in \{-1, 1\}$.
- We have symmetrical weights, i.e. $w_{ij} = w_{ji} \ \ \forall\, i, j$.
- No self feedback weights, i.e. $w_{ii} = 0 \ \ \forall\, i$.

Assume the nodes are updated by the following equations,

$$s_i = \text{sgn}(h_i) \tag{1}$$

$$\text{sgn}(x) = \begin{cases} 1 & \text{if} \quad x \geq 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{2}$$

where $h_i = \sum_j w_{ij} s_j$. For the Hopfield model we can define an *energy* function,

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j$$

The idea of calling E an energy function is that it should decrease when updating the Hopfield model.

This is now your task, show that $\Delta E \leq 0$ when the nodes of the Hopfield model are updated according to eqn. 1 (5p).

Hint: You can assume serial updating of the nodes, meaning that you could analyze what happens to E when a given node e.g. s_k is updated to the value s'_k given by eqn. 1.

Answer: We had that

$$s_i = \text{sgn}(\sum_j w_{ij} s_j)$$

Let us look at one node s_k and let s'_k be the new value (after the update).

Before the update

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j = -\frac{1}{2} \sum_{i \neq k} \sum_{j \neq k} w_{ij} s_i s_j - s_k \sum_{i \neq k} w_{ik} s_i$$

and

$$E' = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j = -\frac{1}{2} \sum_{i \neq k} \sum_{j \neq k} w_{ij} s_i s_j - s'_k \sum_{i \neq k} w_{ik} s_i$$

The change of energy ΔE is

$$\Delta E = E' - E = s'_k \sum_{i \neq k} w_{ik} s_i + s_k \sum_{i \neq k} w_{ik} s_i = (s_k - s'_k) \sum_{i \neq k} w_{ik} s_i$$

Now if $s'_k = s_k$ (no change after update), then trivially $\Delta E = 0$.
If $s'_k = -s_k$ (the only other possibility), then

$$\Delta E = 2s_k \sum_{i \neq k} w_{ik} s_i = 2s_k \sum_i w_{ik} s_i \text{ since } w_{ii} = 0$$

Now according to the update rule $\sum_i w_{ik} s_i$ must have the same sign as s'_k. We can then wright

$$\Delta E = 2s_k A$$

where A and s_k have opposite sign. This leads to

$$\Delta E < 0$$

Hence, $\Delta E \leq 0$ when updating the Hopfield model. This means that we will always reach a local minima of E when updating the model.

Good Luck!
Mattias & Patrik