Last Name ..... First Name.....

Artificial Neural Networks: Exam
2 July 2018

- Write your name in legible letters on top of this page.
- The exam lasts 160 min.
- Write **all** your answers in a legible way on the exam (no extra sheets).
- No documentation is allowed (no textbook, no handwritten notes, no sheets of paper)
- No calculator is allowed.
- Put your bag with your computer and smart phone in front of the room, below the blackboard.
- Have your student card displayed before you on your desk.
- **Check that your exam has 14 pages**

Evaluation:

1. ...../15 pts
2. ..... / 9 pts
3. ..... / 6 pts
4. ..... / 4 pts
5. ..... / 5 pts
6. ..... / 6 pts

Total: ..... / 45 pts

Definitions

The symbol η is reserved for the learning rate.

Throughout the exam, a data base for supervised learning will be denoted by $(\mathbf{x}^\mu, \mathbf{t}^\mu)$ with $1 \leq \mu \leq P$ where $\mathbf{t}^\mu$ is the target output. The actual output of the artificial neural network will be denoted by $\hat{\mathbf{y}}$ in general and $\hat{\mathbf{y}}^\mu$ to denote the output in response to input pattern μ. Bold face symbols refer to vectors, normal face to a single component or a single input/output.

The total input to a unit i is denoted by $a_i = \sum_j w_{ij} x_j$ with weights w_{ij} and the output of that same unit by $g(a_i)$. The function g is called the gain function. Sometimes the threshold is made explicit by a symbol ϑ; sometimes it is implicit in the weights.

In the context of reinforcement learning the symbol a refers to an action; the symbols r and R to a reward; the symbol s to a discrete state; and the symbol γ to a discount rate. If the input space is continuous then inputs are also written as $\mathbf{x}$.

How to give answers

The first part contains 15 Yes-No question. For each question, you have **three possibilities: Tick yes, or no, or nothing.** Every correct answer gives one positive point, every wrong answer one negative point, and no answer no point. If the final count (with this procedure) across 15 questions is below zero, we give zero points. With this procedure random guesses are subtracted and 15 correct answers give 15 points.

The remaining parts involve calculations. Please write the answers in the space provided for that purpose.

We also provide some free space for calculations. We will not look at these for grading. You can also use colored paper for side calculations.

1 15 Yes-No questions (15 points; explanation on page to the left)

1. Backprop is much more than just an efficiently implemented chain rule of calculus

   [ ] Yes or No [ ]

2. For a simple network with a single output and no hidden layer, the 'perceptron' update rule can be written as $\Delta\mathbf{w} = \beta(\hat{y}^\mu)[t^\mu - \hat{y}^\mu]\mathbf{x}^\mu$ with a suitable choice of the function $\beta(\hat{y}^\mu)$

   [ ] Yes or No [ ]

3. A cross-entropy error function for multi-class output together with one-hot coding of the target (unit k codes for class k) and softmax function for the output guarantees that the value of the output unit k is equal to the posterior probability $P(C_k|\mathbf{x})$, once the optimization algorithm has converged to the minimum.

   [ ] Yes or No [ ]

4. For a simple network with a single output and no hidden layer, the 'stochastic gradient descent' update rule on a quadratic error function can be written as $\Delta\mathbf{w} = \beta(\hat{y}^\mu)[t^\mu - \hat{y}^\mu]\mathbf{x}^\mu$ with a suitable choice of the function $\beta(\hat{y}^\mu)$

   [ ] Yes or No [ ]

5. The prediction for future data with your artificial neural network will alwyas be best if past data is perfectly fit.

   [ ] Yes or No [ ]

6. For a multi-attribute classification problem where the target vector $\mathbf{t}^\mu$ has elements zero and one, you should – in a probabilistic setting – best use multiple output units, each with a standard sigmoidal output function.

   [ ] Yes or No [ ]

7. Training data for video frame prediction is cheap because no human labeling is needed.

   [ ] Yes or No [ ]

8. One batch update step of the backprop algorithm scales with order $O(K^2P)$ if we assume a network with N_0 input units, $N_1 = K$ neurons in the first hidden layer, $N_2 = K$ neurons in the second hidden layer and N_0 outputs where $N_1 = N_2 = K$ are arbitrary, but much larger than $N_0 = N_{out}$.

   [ ] Yes or No [ ]

9. In a 3-step SARSA algorithm you follow the sequence s_t, a_t, r_t; $s_{t+1}, a_{t+1}, r_{t+1}$; $s_{t+2}, a_{t+2}, r_{t+2}$; $s_{t+3}, a_{t+3}, r_{t+3}$; ... . Once you have reached state s_{t+3} you update $Q(s_t, a_t)$ by an amount
$\Delta Q(s_t, a_t) = \eta \left[r_t + \gamma r_{t+1} + \gamma^2 Q(s_{t+2}, a_{t+2}) - Q(s_t, a_t) \right]$

[ ] Yes or No [ ]

10. For a simple network with a single output and no hidden layer, the 'stochastic gradient descent' update rule on a 'crossentropy' error function can be written as $\Delta \mathbf{w} = \beta(\hat{y}^\mu)[t^\mu - \hat{y}^\mu]\mathbf{x}^\mu$ with a suitable choice of the function $\beta(\hat{y}^\mu)$

[ ] Yes or No [ ]

11. You use an error function E that depends on parameters $\mathbf{w}$, minimize the error, and evaluate on the validation set which yields E^*. You then switch to a cost function $C(\mathbf{w}) = E(\mathbf{w}) + \lambda|\mathbf{w}|$ where the absolute value signs denote the L2-norm and $\lambda > 1$, mininize the cost, and evaluate on the validation set which yields C^*. Since λ is positive and absolute values are positive, C^* is always larger than E^*.

[ ] Yes or No [ ]

12. You received a large data base with $P = 100000$ entries and train a deep neural network on it. Optimization of the network parameters with a maximum likelihood approach implies that you will find the parameters that actually have generated the data.

[ ] Yes or No [ ]

13. Forward propagation with Rectified Linear Units works well because a large fraction of the paths connecting a given input unit to a given output unit are activated so that we get a highly distributed coding scheme.

[ ] Yes or No [ ]

14. A update step of 'gradient descent with Nesterov momentum' needs as many gradient evaluation steps as an update step of 'gradient descent with standard momentum.'

[ ] Yes or No [ ]

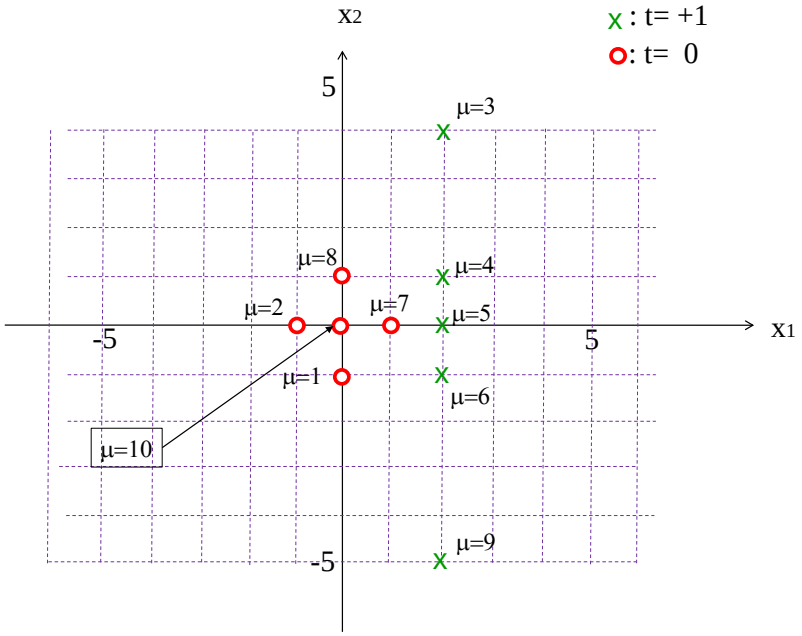
15. Deep Learning performs better than other algorithms on all problems, if we use a sufficient number of hidden layers and a sufficient number of units per hidden layer,

[ ] Yes or No [ ]

Number of correct answers: ...../15
Number of wrong answers: ...../15 number of points: ...../ 15

2 Simple Perceptron (9 points)

You use a simple perceptron with two inputs (x_1, x_2), no hidden layer, one binary output $y \in \{0, 1\}$ with a threshold ϑ and weights w_1 and w_2 to classify the following ten input points.



where the cross symbol means positive target $t = +1$ and the open circle symbol means negative target $t = 0$.

(a) Find (by hand, by intuition or using an algorithm) a solution of the problem. Write down the parameters of your solution

$w_1 =$ .....

$w_2 =$ .....

$\vartheta =$ .....

number of points: ...../ 2

(b) If there is more than one solution to the classification problem, give a second one here:

$w_1 =$ .....

$w_2 =$ .....

$\vartheta =$ .....

number of points: ...../ 1

(c) Write down the general formula of one update step in the perceptron algorithm for the parameters w_1, w_2, ϑ.

$\Delta w_1 =$ .....

.....

$\Delta w_2 =$ .....

.....

$\Delta \vartheta =$ .....

.....

number of points: ...../ 1

(d) Initialize your algorithm with parameters $w_1(0) = 3, w_2(0) = 2, \vartheta(0) = 0$. You apply patterns sequentially, starting with $\mu = 1$, then $\mu = 2$, $\mu = 3$ ... .

(d1) When does the first change of parameters occur?

First change induced by application of pattern number $\mu =$ .....

(d2) What are the new values of the parameters? [If your algorithm uses a learning rate η, then please set $\eta = 1$.]

$w_1(1) =$ .....

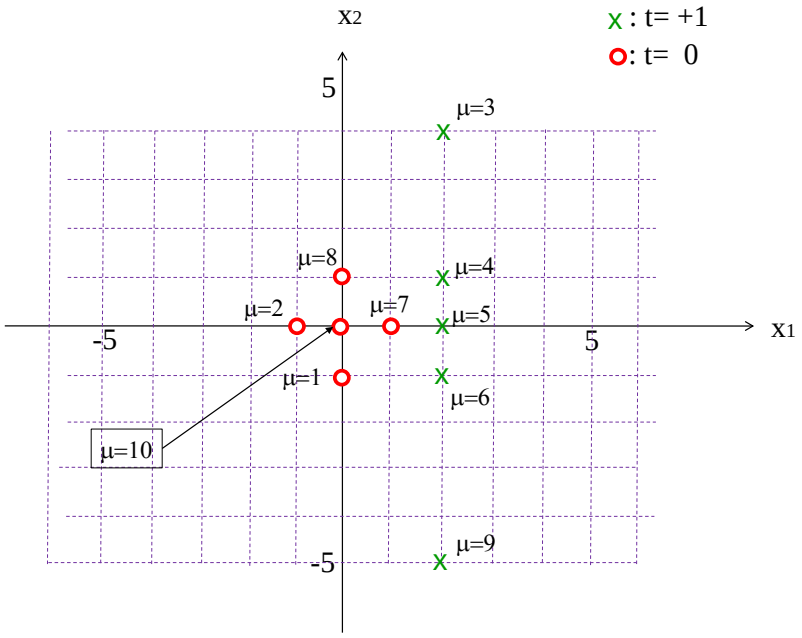
$w_2(1) =$ .....

$\vartheta(1) =$ .....

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.

(e) Draw the separating hyperplane BEFORE AND AFTER the first update step in the graph below.



number of points: ...../ 2

(f) Evaluate now the next update step and draw the resulting hyperplane in the above figure.

$w_1(2) =$ .....

$w_2(2) =$ .....

$\vartheta(2) =$ .....

number of points: ...../ 1

(g) How did the 'distance from origin' of the separating hyperplane evolve?

'distance from origin' before the **first** update step = ...

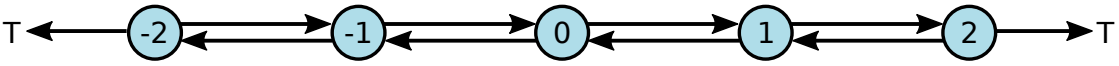
'distance from origin' after the **first** update step = ...

'distance from origin' after the **second** update step = ...

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.

3 RL and TD algorithms (6 points)



An agent moves in an environment with discrete states: There are five nonterminal states ($s \in \{-2, -1, 0, 1, 2\}$) with two actions in each state ('Left' and 'Right'), where 'Left' results in a transition into state $s_t - 1$ and 'Right' results in a transition into state $s_t + 1$ (represented by the arrows).

Each episode terminates when the agent moves Left from state -2 (with a reward of $r = -1$), and likewise terminates when the agent moves Right from state 2 (with a reward of $r = +1$). All other actions result in a reward of 0.

The agent explores the state space and updates Q–values using 1–step SARSA.

In order to debug your algorithm you have chosen a specific initialization:

Q–values for all actions moving to the right are initialized to $Q(\cdot, \text{Right}) = 0$.

Q–values for all actions moving to the left are initialized to $Q(\cdot, \text{Left}) = 3$.

(a) Write down the update formula for 1–step SARSA using a discount factor γ and a learning rate η.

.....

number of points: ...../ 1

(b) The first two episodes are as follows:

(1) (0, Left), (-1, Left), (-2, Left)

(2) (0, Right), (1, Right), (2, Right)

Using learning rate $\eta = 1$, determine the resulting Q–values in terms of γ after these two episodes.

Write down those Q-values that are different compared to initialization here:

.....

.....

.....

number of points: ...../ 2

(c) The agent continues now for two further episodes (always $\eta = 1$)

(3) (0, Right), (1, Right), (2, Right)

(4) (0, Right), (1, Right), (2, Left), (1, Left), (0, Left), (-1, Left), (-2, Left)

Write down those Q-values that are different after episode 4 compared to initialization here:

.....

.....

.....

.....

.....

number of points: ...../ 2

(d) Assume that, after the first step of episode (4), the agent updated Q(0, Right) according to standard **Q–learning** with $\eta = 1$ instead of SARSA.

What is Q(0, Right) in this case?

.....

.....

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.

4 Minima and Saddle points (4 points)

You use a multi-layer feedforward network with two hidden layers for a classification task $t^\mu \in \{0, 1\}$. The input data $\mathbf{x}$ is five-dimensional. The data base contains $P = 10000$ examples $1 \leq \mu \leq P$. For 5 000 of these we have $t^\mu = +1$ for the other 5 000 we have $t^\mu = 0$.

You use $N_1 = 100$ units in the first hidden layer and $N_2 = 50$ units in the second layer and a single output neuron

(a) Suppose that, by using one of the standard optimization methods for classification in artificial neural networks, you have found a solution. Performance on the validation set indicates that this is an excellent solution. You have checked that this solution makes use of all neurons because all units have different weight vectors.

Your friend says: 'I am sure that there is a second solution that is just as good as the one you found.' Is she right? Give an argument

Yes/No because .....

.....

number of points: ...../ 1

(b) How many solutions that are completely equivalent are there roughly, based on permutation arguments?

.....

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.

(c) You now use, for the same data base, a smaller network with only one hidden layer with 50 units.

You have have access to a smart optimization method to find a minimum. It gives you back a set of weights in the minimum together with the guarantee that no minimum has a lower value than this one. The method also informs you that there are in total $M - 1$ minima that are equivalent to the one given by the algorithm.

(c1) Based on permutation arguments, do you expect that M is smaller or larger than 80 000? Give an argument

.....

.....

number of points: ...../ 1

(c2) Your friend argues that the number of saddle points in the cost function is larger than M.

Is she right? Give an argument

Yes/No because .....

.....

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.

5 BackProp with Gaussian units in the first layer (5 points)

Our data base $(\mathbf{x}^\mu, \mathbf{t}^\mu)$ has N-dimensional input and N-dimensional target output. We have a network with two hidden layers.

You use a quadratic error function for each pattern μ

$$E(\mu) = \sum_{m=1}^{N} [t_m^\mu - \hat{y}_m^\mu]^2 \tag{1}$$

where $\hat{y}_m^\mu = \sum_{k=1}^{K} w_{mk}^{(3)} x_k^{(2)}$.

The second hidden layer has normal sigmoidal units with gain function $g(a)$, while the first hidden layer contains Gaussian basis functions. Thresholds are implicit (by adding an extra unit) and will not be treated explicitly.

Thus

$$x_k^{(2)} = g[\sum_{j=1}^{J} w_{kj}^{(2)} \exp[-0.5(\mathbf{x}^\mu - \mathbf{c}_j)^2] \tag{2}$$

where $\mathbf{c}_j$ is the center of the Gaussian of unit j.

Our aim is to calculate the derivative of the error function with respect to the parameters c_{ji}, that is, component i of Gaussian unit j.

A direct calculation with the chain rule yields that the derivative with respect to c_{45} is

$$\frac{dE(\mu)}{dc_{45}} = \sum_{m=1}^{N} [t_m^\mu - \hat{y}_m^\mu] \sum_{k=1}^{K} w_{mk}^{(3)} g'(a_k) w_{k4}^{(2)} (x_5^\mu - c_{45}) \exp[-0.5(\mathbf{x}^\mu - \mathbf{c}_j)^2] \tag{3}$$

Reorder the terms of the gradient calculation for arbitrary c_{ij} so as to arrive at an efficient backpropagation algorithm with a forward pass and a backward pass and an update step for the parameters c_{ij}. Summarize your results in pseudo-code using the layout on the next page:

Free space for your calculations, do not use to write down answers.

Pseudocode (for one pattern)

(0) Initialization

Parameters $w_{ij}^{(n)}$ and c_{ij} are initialized at small values $[\epsilon, \epsilon]$.

Pattern $\mathbf{x}^{\mu}$ is applied at the input

(a) Forward pass

.....

.....

.....

.....

.....

.....

number of points: ...../ 2

(b) Backward pass

.....

.....

.....

.....

.....

.....

number of points: ...../ 2

(c) Update of parameters c_{ij}

.....

.....

number of points: ...../ 1

6 Policy gradient (6 points)

An agent moves in a two-dimensional arena. In each trial, it starts (with probability $P(\mathbf{x}_n)$) from one of 400 possible initial states $\mathbf{x}_n$ with $1 \leq n \leq 400$. In each initial state it has four possible actions a_k with $k \in \{1, 2, 3, 4\}$. The actions are implemented by an artificial neural network with four output neurons corresponding to actions a_1, a_2, a_3, and a_4 with 1-hot coding. If $a_k = 1$ action a_k is taken.

The probability of taking action a_i is given by the softmax function

$$\pi(a_i|\mathbf{x}; W) = \frac{\exp[\sum_{k=1}^{2500} w_{ik} y_k]}{\sum_j \exp[\sum_{k=1}^{2500} w_{jk} y_k]} \tag{4}$$

where W is the matrix of weights w_{ik} and $y_k = f(\mathbf{x} - \mathbf{c}_k)$. Here $\mathbf{x} \in \mathbb{R}^2$ is the input signal that contains the state information and $\mathbf{c}_k$ is the center of one of 2500 Gaussian functions that fully cover the two-dimensional input space.

For an action a_i in state $\mathbf{x}_n$, the agent receives a reward $R(a_i, \mathbf{x}_n)$ and then the agent is reinitialized in a new state.

(a) Calculate the gradient of the mean reward

$\langle R \rangle = \sum_{n=1}^{400} \sum_{i=1}^{4} R(a_i, \mathbf{x}_n) \pi(a_i|\mathbf{x}_n; W) P(\mathbf{x}_n)$ with respect to the weight w_{25}.

$\frac{d}{dw_{25}} \langle R \rangle =$

.....

number of points: ...../ 3

(b) Use the result from (a) to write down a batch update rule for the weight w_{25}.

$\Delta w_{25} =$

.....

number of points: ...../ 1

(c) Go from the batch rule for weight w_{25} in (b) to an 'online rule' for an arbitrary weight w_{ij} that would allow the agent to update the weight after each action.

$\Delta w_{ij} =$

.....

number of points: ...../ 2