69 students wrote this exam. 50 passed (72%), 20 with grade 5, 20 with grade 4, 10 with grade 3. 19 students failed. The top score was 38 and the average 25.4 (median 27).

Machine Learning written examination

Monday, April 3, 2017
8^{00} - 13^{00}

Allowed help material: Pen, paper and rubber, dictionary

Please, answer (in Swedish or English) the following questions to the best of your ability!

Please, only write on **ONE SIDE OF THE PAPER**!

Any assumptions made, which are not already part of the problem formulation, must be stated clearly in your answer unless is it explicitly stated below that you don't have to.

The maximum number of points is 40. To get the grade 3 (pass) a total of 20 points is required. The grade 4 requires 27 points and grade 5 requires 32 points.

I will drop in to answer questions sometime between 9.00 and 9.30.

In this exam, some concepts may be called by different names than the ones used in the book. Here is a list of useful synonyms and acronyms:

- Perceptron = summation unit = SU = conventional neuron
- Binary perceptron = perceptron with a binary step function as its activation function
- MLP = Multilayer perceptron = Feedforward neural network of (usually sigmoidal) summation units
- The XOR problem = special case of odd parity for 2-bit bit strings = Return 1 if the number of bits in the string is odd, 0 if it is even
- Back propagation = Backprop = Generalized delta rule
- RBF(N) = Radial Basis Function (Network)
- Cartesian coordinate (system) = (x,y) coordinate (system)
- SCL = Standard Competitive Learning = LVQ-I without a neighbourhood function
- SOFM = (Kohonen's) Self Organizing Feature Map
- EC = Evolutionary Computation (for example genetic algorithms)
- PSO = Particle Swarm Optimization

Olle's ants say hello to spring, and wish you good luck!
(Olle hopes that you don't need to rely on luck)

Institutionen för informationsteknologi
Olle Gällmo
Universitetsadjunkt

Adress:
Lägerhyddsvägen 2
Box 337
751 05 Uppsala

Telefon:
018 - 471 10 09

Telefax:
018 – 51 19 25

Hemsida:
user.it.uu.se/~crwth

Epost:
olle.gallmo@it.uu.se

Information Technology
Olle Gällmo
Lecturer

Address:
Lägerhyddsvägen 2
Box 337
SE-751 05 Uppsala
SWEDEN

Telephone:
+46 18 - 471 10 09

Telefax:
+46 18 – 51 19 25

Web site:
user.it.uu.se/~crwth

E-mail:
olle.gallmo@it.uu.se

UPPSALA
UNIVERSITET

1. For neural networks, how is the risk of *overtraining/overfitting* affected (increased or decreased) by:

   a) increased network size? (number of hidden nodes and/or layers)

   *COMMENT: The risk **increases** since the network will have more parameters than necessary to solve the problem, which is likely to lead to overfitting (= overtraining). Another way to describe it is that more parameters make it easier for the network to learn the patterns "by heart", instead of learning the underlying function.*

   b) increased training set size? (number of input-target examples)

   *COMMENT: The risk **decreases.** More patterns make it more difficult to learn the patterns by heart.*

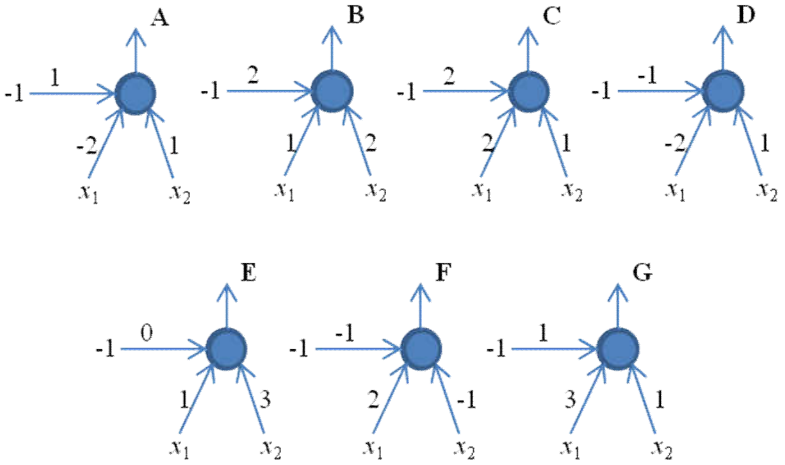
   c) increased training time? (number of passes through the training set)

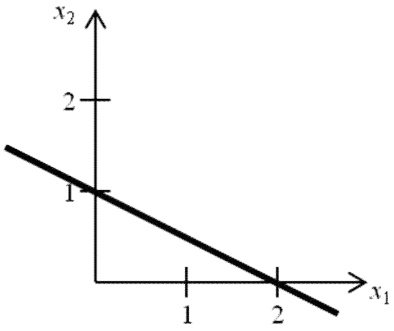
   *COMMENT: The risk **increases**. The network gets more time to overfit or to learn patterns by heart.*

   You only have to indicate whether the risk increases or decreases. No explanations are necessary. ..... (3)

UPPSALA
UNIVERSITET

2. A binary perceptron defines a hyperplane in the input space and responds with a 1 on one side of it, 0 on the other. Here are 7 binary perceptrons:





a) Since the nodes have two inputs, the input space is 2-dimensional and the hyperplane is therefore a line. One of the perceptrons above defines the hyperplane shown here. Which one? ..... (2)

*COMMENT: The correct answer is **B**.*

The line equation of a 2-input binary perceptron (given by setting the weighted sum to 0, since the node flips there, and solve for x_2) is:

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2}$$

This cuts the x_2 axis at θ/w_2. and the x_1 axis at θ/w_1. So, in this case, θ should be equal to w_2 and w_1 should be half of that. That's only true for perceptron B.

b) One of the perceptrons defines a hyperplane which crosses origo (0,0). Which one? ..... (2)

*COMMENT: The correct answer is **E**, the only perceptron given here which has a 0 threhold. That makes the hyperplane cut through origo.*

c) Two of the perceptrons define the same hyperplane. They only differ on which side the node would output a 1. Which pair? ..... (1)

*COMMENT: The correct answer is **A** and **F**. They have the same weights, but with opposite signs. This does not change the slope or position of the hyperplane, only on which side the node will respond with +1.*

For all three sub-questions here (a,b,c), you only have to identify the nodes asked for (by its letter name). No explanations are necessary.

UPPSALA
UNIVERSITET

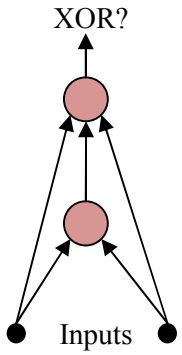
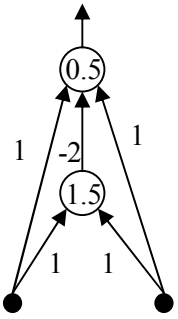


Figure for question 3: A 2-1-1 MLP to solve the XOR problem.

3. The most common architecture to solve the XOR problem is a 2-2-1 multilayer perceptron, that is to say a network with two inputs, two hidden nodes and 1 output. It is actually possible to solve the XOR problem with only one hidden node, if the output node also has direct connections to the inputs, bypassing the hidden node (see figure). Find a set of weights for this MLP which would solve the XOR problem! ..... (3)

   For simplicity, assume that the nodes are binary. Draw the network on your answer sheet and include the weight values – you don't have to explain how you computed them. Threshold/bias values can be written down in the nodes, but will be assumed to be thresholds (= subtracted).

   COMMENT: For example, make the output node an OR gate of the two network inputs and the hidden node an AND. Then connect the hidden node with the output node with a negative weight, so that the output will fire only if one of the inputs are 1, but not both. For example:



   Minor point reductions for minor errors or obscurities, for example if the thresholds were set exactly on the limit where the node flips. You then must know if the node flips at > or at >= 0.

   0 credit if the suggested solution was too far off (it should have been easy to test your solution). For some reason, some students came up with non-symmetric networks (weighing one input more than the other), which is a peculiar idea since since the problem is symmetric. There were also some suggested solutions where the top node had 0-weights to the inputs (only cared about the hidden node) which should be obvious can not work.

UPPSALA
UNIVERSITET

4. The update equation for the back propagation algorithm can, under certain assumptions discussed below, be written as:

$\Delta w_{ji} = \eta \delta_j x_i$, where

$\delta_j = \lambda y_j (1 - y_j)(d_j - y_j)$, or

$\delta_j = \lambda y_j (1 - y_j) \sum_k w_{kj} \delta_k$

a) One of the δ equations apply to the hidden layer(s), one applies to the output layer. Which is which? (*no explanation required*) ..... (1)

COMMENT: The first δ equation is for the output layer, the second (with the sum) for the hidden layer(s).

b) One assumption made in these equations is that the objective function to be minimized is the squared error. Which part or parts would change if we wanted to change that, to minimize something else? (*no explanation required*) ..... (2)

COMMENT: Only (d_j-y_j) in the equation for the output layer nodes, would change. The equation for the hidden layer is not affected (the hidden nodes are affected, indirectly, through the sum over k, but the equation as such is not).

c) Both equations assume that the nodes are sigmoidal (to be more precise, that the activation functions are *logistic*). Which part or parts would change if we used another activation function for the nodes? (*no explanation required*) ..... (2)

COMMENT: $\lambda y_j(1-y_j)$ would change in both equations where it occurs. This is the derivative of the logistic function, $y=1/(1+e^{-\lambda s})$, and λ controls its slope. Answers which did not include λ received full credit, since λ is very often set to 1 in practice.

UPPSALA
UNIVERSITET

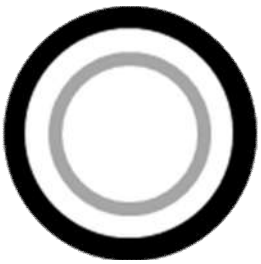


Figure for question 5: Two cirular regions.

5. MLP v.s. RBF

   a) Consider the classification problem depicted here to the left. The task is to separate the outer (black) circle from the inner (grey), in a Cartesian (x,y) coordinate system. Which architecture should be most suitable for this task (require the fewest number of nodes) – a multilayer perceptron or a radial basis function network? Why? (*explanation should not require more than a few sentences*)..... (3)

   COMMENT: RBFN should work best since it puts out hyperspheres in the input space, i.e. one node is sufficient to separate the two classes in this case. A MLP would have to form a circular(ish) region by combining hyperplanes, in this case at least three hidden nodes (= a triangle with extremely rounded corners), plus one output node.

   Some students had confused hyperspheres and Voronoi regions. Voronoi regions are not really applicable here. In contrast to competitive learning, there is no "winner" here or search for the closest node.

   Some point reductions for bad explanations and/or for not being clear (or being wrong) on what the discriminants do in the input space.

   b) Why is it relevant for the previous subquestion, that the coordinate system is Cartesian? (*explanation should not require more than a few sentences*)..... (2)

   COMMENT: Because it affects the shapes. For example, if we transformed the coordinate system to polar coordinates in this case, the problem would become trivial for a MLP, but more difficult for RBF. A single binary perceptron node would suffice (and it would in this case actually only require one of the two input values – the distance from origo).

   Some students claimed that MLP and/or RBFN are only defined for Cartesian space. This is not true. The network as such does not "know" anything about the coordinate system used. It just makes the problem more or less difficult because the shapes (of both data and discriminants) change.

   Some students seemed to think that Cartesian means 2D, and that this problem therefore is Cartesian by definition. Instead they explained how MLP and RBFs behave differently in higher dimensions, which is not what was asked for here.

   Some students seemed to think that the coordinate system is somehow connected to the activation function and that a problem in non-Cartesian space would no longer be differentiable.

UPPSALA
UNIVERSITET

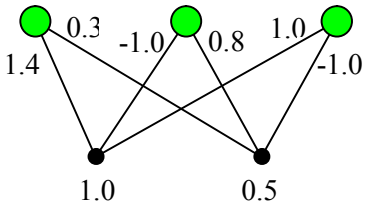


Figure for question 6: Competitive Learning

6. Consider the competitive learning network depicted here to the left. The three circles on top are the nodes. The numbers by the connections are the weights. The black circles at the bottom are the inputs and the numbers below are the current input values. What happens now:

   a) Which node wins, given this input, and why? (*explanation should not require more than a few sentences*) ..... (2)

   COMMENT: The leftmost node wins, since its weight vector, [1.4, 0.3], is closest (Euclidean distance) to the input vector, [1.0, 0.5]).

   Some students had the network compute weighted sums instead of distances, which in this case would identify the same winner, so that's OK, but in general the weights should be normalized first if you want to define the winner this way. Also, if you compute weighted sums, the winner is the node with the <u>greatest</u> weighted sum, not the smallest.

   b) What happens to the weights of the network if the standard competitive learning rule is applied to this situation with step length η=0.5? Write down the new set of weights and justify your answer (*the explanation should not require more than a few sentences*)... (3)

   COMMENT: The weight vector of the winner is moved halfway (since η=0.5) towards the input vector, so the new weight vector of that node is [1.2, 0.4]. The rest of the weights in the network are left unchanged in standard competitive learning.

   Surprisingly many students computed weight changes without using the input values (for example, just multiplying the weight with the step length). It should be obvious that no learning rule could work if it does not care about the input values.

   Some students had the right idea, but did some minor mathematical error (sign errors for example) which moves the node <u>away</u> from the input. The mistakes may have been minor, but it should have been obvious in those cases that the result was wrong (further away from the input).

UPPSALA
UNIVERSITET

7. What is the purpose of the neighbourhood function used in self organizing feature maps? Please explain how the neighbourhood function is used, what it typically looks like (as a function), and why it is needed (what we want to achieve)!..... (3)

COMMENT: In SOFM, the nodes are connected in a neighbourhood structure (usually a 2D grid). Instead of just moving the winner (the weight vector which is closest to the input vector), as in competitive learning, we move all weight vectors towards the input, but to a degree which depends on how far that node is from the winner in the network structure.

This is done by defining a neighbourhood function f(j,k) where j is the index of the node to move and k is the index of the winner node. This function should have a max=1 for the winner itself (j=k) and then decrease with distance between j and k. A Gaussian function, for example. This neighbourhood function is used as a multiplier in the update formula:

$$\Delta w_{ji} = \eta f(j,k)(x_i - w_{ji})$$

This is what makes self organizing feature maps a <u>map</u>, i.e. topologically preserving, so that inputs close to each other in the input space will also activate areas close to each other on the grid. In other words, this is what makes the 2D map approximate the density function in the high-dimensional input space.

Most common reasons for point reductions:

- *no discussion on topological preservation (or something to that effect). Despite several warnings during the course, a great majority of the students claimed that the task of the neighbourhood function is to avoid the winner-takes-all scenario of competitive learning. It does indeed reduce this risk, but it is a side-effect, not the reason.*
- *not being explicit about the distance measure for the neighbourhood function, being a distance between nodes in the structure, not in the input space.*
- *not showing or arguing how the function is applied (to the update formula, or in words)*

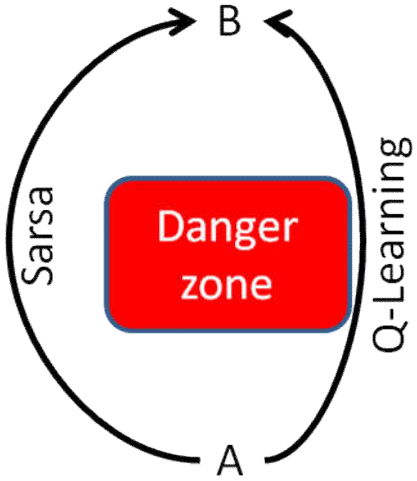
Some students had confused the neighbourhood structure (the grid – deciding who is neighbour to whom) with the neigbourhood function (deciding how much to move them).

UPPSALA
UNIVERSITET

8. Consider a navigation problem (sketched below) where we want a reinforcement learning agent to find the shortest path from **A** to **B**. There is a dangerous area between **A** and **B** (a region of states where the agent would be punished or possibly even killed if it enters), so the agent must find a path around it, to avoid the area.

   If we train both a Q-learning agent and a Sarsa agent on this problem, with constant and non-zero control parameters, the Q-Learning agent would typically converge to a shorter route (closer to the danger zone) than Sarsa, which would seem to be more careful and converge to a slightly longer route around the dangerous area. Why? ..... (4)

   (If you can't answer the specific question, you can still get partial credit by describing the difference between Q-Learning and Sarsa in other ways)



COMMENT: Q-Learning updates the Q-value Q(s,a) for state s and action a, based on the maximum Q-value in the next state, s'. It does not matter if the agent selected another action there, due to exploration, the maximum Q-value in state s' will still be used to update Q(s,a). So even if the agent explores from s' and therefore enters the danger zone, Q(s,a) will not be affected.

Sarsa, on the other hand, updates Q(s,a) based on the action actually selected in s', so if that was an exploratory move into the danger zone, Q(s,a) will be directly affected by that. This will in effect push down the expected value for Q(s,a) which may make it worth while to take a longer route around the danger zone.

[Actually, it is not certain that the expected value of Q(s,a) will be pushed down sufficiently to make a detour worth while. It also depends on the exploration rate and how strongly we reward taking the shortest route. For example, in how large the positive reward for reaching the goal is, compared to the cost of getting there. I did not expect students to find that weakness in the problem formulation, and as far as I could tell, no one did]

Minor point reductions for simply not describing this very well (not being convincing). For example not being clear that the interesting part is what happens to the values in state s, not in state s'. Major point reductions for claiming that Q-Learning is greedy, i.e. does not explore at all.

Some students claimed that Q-values in Q-learning can only grow (i.e. that r+max(Q(s',a'))-Q(s,a) is always positive). This is not true in general (though it was for the lab on this course). It assumes that the environment is stationary, and it depends on the initial Q-values and on the reward strategy. The maximum reward may very well be a negative value, for example. This mistake did in most cases not affect grading though.

UPPSALA
UNIVERSITET

9. It has been claimed in some course books that that one important difference between particle swarm optimization and evolutionary computation is that PSO has memory (the particles remember their personal bests), while EC does not. This is not strictly true – EC may also have such a memory. What is that concept in EC which corresponds to the memory in PSO? (*explanation should not require more than a few sentences*) ..... (3)

   COMMENT: Elitism – to guarantee that the best individual(s) in the population are copied/reproduced as they are, unaltered, to the next generation. This way we know that the best solution is preserved.

   Partial credit (2p) to descriptions which in effect are elitism, but which did not mention the name, or vice versa (mentioning the name but not decribing it well enough).

   Reproduction in itself is not the answer (though partial credit was given for it, 1p) since it depends on which individual is selected (probabilistically). The point with elitism is that reproduction of the best individual(s) is guaranteed and therefore a reliable memory of the best solution found so far.

   The genotype is not the answer either. This corresponds to the position of the particle in PSO, i.e. the position in the search space where it is right now (x in PSO, not p). In a way, this is also a form of memory, but it is very short-term, since we move/jump around all the time.

   Fitness is not a memory either, it's an evaluation. It's the position we should remember, not the evaluation of that position. There is no use remembering how much gold we found at a treasure site if we don't remember where it was.

UPPSALA
UNIVERSITET

10. The velocity update equation of the particle swarm optimization method *lbest*, in its basic form, can we written as:

$$v_{i,d}(t) = v_{i,d}(t-1) + \mathrm{U}(0,\phi_1) * \left(p_{i,d} - x_{i,d}(t-1)\right) + \mathrm{U}(0,\phi_2) * \left(p_{l,d} - x_{i,d}(t-1)\right)$$

where $v_{i,d}(t)$ is the velocity of particle i in dimension d at time t, $x_{i,d}(t)$ is the position of particle i in dimension d at time t, and $\mathrm{U}(0,\phi_1)$ and $\mathrm{U}(0,\phi_2)$ are random numbers drawn from a uniform distribution in the range $[0,\phi_1]$ and $[0,\phi_2]$, respectively.

a) Explain the variables $p_{i,d}$ and $p_{l,d}$! ..... (2)

COMMENT: $p_{i,d}$ is the best position found so far by this particle (i), $p_{l,d}$ is the best position found so far by any particle among this particle's neighbours. This neighbhourhood is fixed and defined by a graph, for example a ring structure or a hypercube.

Point reductions (very common) for not explaining the neighbourhood (at least that it is structural and not topological)

b) ϕ_1 and ϕ_2 are constants, usually recommended to be set close to 2. What's the intuition behind this recommendation? Why 2? ..... (2)

COMMENT: Since the average is then close to 1. ϕ_1 and ϕ_2 controls the step length towards the personal best and the local best, respectively. Setting the value close to 2 means that the average value will be close to 1, i.e. the search will be centered (roughly) around the best positions found so far. There is usually no point to go back the exact best positions again, but to search around them seems reasonable.

Some students seemed to think that the question was about the PSO extension called constriction (which imposes conditions on these two parameters – the sum $\phi_1 + \phi_2$ must be greater than 4 if constriction is used).

Some students only described the intuition behind the parameters as such (how they are weighing the cognitive/social components against each other), but missed the main point: Why the values should be close to 2.

(*explanations should not require more than a few sentences for each sub-question*)