12 students wrote this exam. 1 passed with grade 5, 2 passed with grade 4, 4 passed with grade 3 and 5 failed. The top score was 35.

Commented version of the exam questions below.

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

Machine Learning written examination

Tuesday, August 22, 2015
14^{00} - 19^{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 around 15.30.

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

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:

- XOR = Exclusive OR (in Boolean logic)
- Multilayer perceptron = MLP = Feedforward network of (usually sigmoidal) summation units
- Back propagation = Backprop = Generalized delta rule
- RPROP = Resilient back propagation
- RBF(N) = Radial Basis Function (Network)
- SCL = Standard Competitive Learning = LVQ-I without a neighbourhood function
- SO(F)M = Self Organizing (Feature) Map
- GNG = Growing Neural Gas
- Population methods = search algorithms with multiple search points, including genetic algorithms (GA), genetic programming (GP), particle swarm optimization (PSO) and ant colonly optimization (ACO).

Good luck!

UPPSALA
UNIVERSITET

Supervised learning

1. A very common challenge in machine learning, which almost all learning algorithms must face, is to avoid *overtraining*. Explain the concept, and how *noise injection* may help prevent this!!.....(3+2=5)

   COMMENT: Overtraining (= overfitting) typically occurs when a learning system with too great representational power (for example a neural network with too many hidden nodes) is trained on too little data, or for too long. This increases the risk that the learning system will either learn the training set examples by heart, or create a more complex solution than necessary to fit the data. An overtrained system will not generalize well, i.e. it will perform badly on new data, not seen during training.

   Point reductions for some vague descriptions (for example being unclear when/why it occurs)

   Noise injection is to add a small amount of noise to the input values every time an input pattern is presented to the learning system, so that the patterns never look quite the same. It forces the learning system to find more general representations. Some students suggested (without penalty) that this is done by extending the training set with slightly modified copies of the existing patterns. That would work too, but maybe not as well since all patterns would still look the same every time they are presented.

   Point reductions for suggesting that the noise is added in the form of extra inputs, for suggesting that noise injection is to extend the set with new random patterns (how would you then decide the target values?), or not being clear that it's the <u>input</u> values which are modified, not the desired outputs (which would make the input-output mapping non-functional).

2. Define a binary perceptron (formally, i.e. write down the mathematical expressions for what it computes), and explain why a single binary perceptron cannot solve the XOR problem! .....(3+2=5)

   Comment:

$$y = \begin{cases} 1, S > 0 \\ 0, S \le 0 \end{cases} \quad S = \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=0}^{n} w_i x_i \quad \text{where} \begin{cases} n = \#inputs \\ x_0 = -1 \\ w_0 = \theta \end{cases}$$

   where y is the output, w_i is a weight, x_i is an input and θ is the threshold. Point reductions for incomplete definitions, or for describing some special case (e.g. only for two inputs which is not what "binary" means here), for having a sigmoidal activation function (not binary), or no activation function at all. Many students did not answer this first part of the question, in which a formal definition was explicitly asked for. A general description without any information on how the node computes its output value, received no points.

   Full credit for the second part required that the student illustrated the XOR problem as a classification problem, to show that the classes are not linearly separable, together with at least a statement that the perceptron forms a linear discriminant (line, or hyperplane). (I did not require proof of the latter though, since the question was to "explain" this, not to "show" or "prove".)

   It is not the node being binary (the step function) which makes the perceptron form a hyperplane. The shape of the discriminant is a consequence of the weighted sum (linear combination).

UPPSALA
UNIVERSITET

3. Describe the back propagation learning rule, with enough detail for a reader to be able to take your description and implement it! You may assume that the reader knows what neurons, weights, and multilayer perceptrons are, but not how they are trained..... (5)

   See handout (slides) from lecture 4.

   Point reductions for factual errors, but mostly for incomplete descriptions. The question formulation had a strong requirement on details here, which is very difficult to satisfy if you don't formulate the answer as an algorithm.

4. Explain the basic principle for manipulating step lengths in RPROP! .. (3)

   The idea is to look at the sign of the partial derivative, $\partial E/\partial w$. It if keeps its sign, it indicates that we are moving downhill and can increase speed (increase the step-length). If it changes sign we have probably overshot a minimum and should reduce it.

   Some point reductions for being unclear what the sign represents (it's that it flips which is interesting, it's not the case that positive is "good" or negative is "bad"), or for just stating general principles such as reducing the step length over time (which is a good principle but not what RPROP does).

5. Radial basis function networks belong to a family of algorithms sometimes called "Localized Learning Systems". What is the difference between RBFs and MLPs, which makes the RBFs "localized" and the MLPs more global? ..... (2)

   RBF hidden nodes (only the hidden nodes) compute a distance between the input vector and a weight vector, whereas MLP hidden nodes compute weighted sums. This makes the RBF discriminant a hypersphere (assuming Euclidean distance), which covers a local region, whereas MLP hidden nodes form hyperplanes which are infinite. Each hyperplane cuts the whole universe in half.

   RBFs feed this distance through a Gaussian function, which has a maximum for 0 and decreases when the distance increases. In other words, a RBF hidden node only responds with high values for patterns close to its region, and with very low values for 'outliers' far from that region. MLP hidden nodes feed their weighted sums through a sigmoid. The response only depends on the distance from the hyperplane, which is infinitely long, and the maximum/minimum values are for patterns infinitely far from it.

UPPSALA
UNIVERSITET

Unsupervised learning

6. Explain the *winner-takes-all* problem in standard competitive learning, and what can be done to avoid it (still using SCL)! .....(2+2=4)

   Competitive learning is to move the closest node (codebook vector) towards the latest input vector, to make it more likely to win also next time the same input vector is presented. The winner-takes-all problem occurs when a few nodes, in the extreme case only one node, wins all the time because they are closer than the other nodes to all the data. The other nodes will never win and therefore never move. At best this leads to underutilization of the network, at worst all the data is classified to the same class.

   The most common way to avoid this is to initialize the weight vectors by drawing vectors from the input data, so that each node is guaranteed to win for at least one input vector. Another way is to include the frequency of winning in the distance measure, so that a node that wins a lot will appear to be further away than it actually is.

   Introducing a neighbourhood function works as well, but is no longer SCL (by definition, see first page). Some point reductions also for not being clear why the winner wins and/or what happens when it does.

7. Self-organizing feature maps have a fixed neighbourhood topology, Growing Neural Gas has a dynamic neighbourhood and Standard Competitive Learning has no neighbourhood at all. Explain what this means in all three cases!. ..... (3)

   The neighbourhood topology decides which nodes in the network, if any, are moved together with the winner towards the input.

   In SCL there is no such neighbourhood, i.e. only the winner is moved.

   In SOFM, the neighbourhood is hard-coded as a grid and in that sense fixed, though the amount by how much a neighbour is moved towards the input may change over time. In other words, who is neighbour to whom is fixed, but the step length is not.

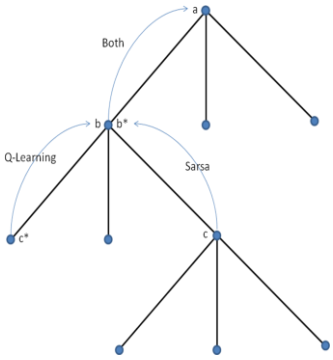
   In GNG, the neighbourhood is changing over time, by updating a graph. Nodes that strive for the same cluster may become neighbours, i.e. be connected in this graph if they were not already, and neighbouring nodes that strive for different clusters may have their neighbour status revoked, by cutting the edge between them in the graph.

   Point reductions for incomplete descriptions, most commonly not mentioning how the neighbourhood is defined (by a graph), what the effect is (that it decides which nodes to move), or implying that edges in GNG can only be removed (not created).

UPPSALA
UNIVERSITET

Reinforcement learning

8. Draw a state transition graph for a board game (or similar) being played by a reinforcement learning agent! Then use this graph to explain how Q-values are updated in Q-Learning, and how this is different from Sarsa! (The explanation is more important than remembering the exact update equations, though that probably helps)..... (5)



*See graph to the left (a similar example was discussed on lecture 8). The choices marked with * are the greedy choices (the ones with max Q-value). Q-Learning updates Q-values toward the max Q-value in the next state, i.e. it assumes that the agent will choose a greedy move there. Sarsa updates Q-values towards the Q-value of the chosen action (greedy or not). In the figure, in a, we chose the greedy action (b=b*) so there is no difference in that case. In b, however, we chose c which is not the greedy action (c*), so Q-Learning and Sarsa will update the Q-value of b towards different values.*

Point reductions for failing to illustrate the differences (no graph at all, graphs that do not illustrate state transitions, or linear ones without any choices to be made, for example). (Very few students seemed to know what a state transition graph is, which is surprising in itself but even more so since this is how the difference between Q-Learning and Sarsa was first explained on the course.) Point reductions also for claiming that the Q-learning is greedy, which it isn't, and/or for other misconceptions on what the Q-values are updated towards.

Common mistake which was not punished: Several students wrongly claimed that Q-values in Q-learning are monotonically increasing and that they can only decrease in Sarsa. This is not true. Q-values can decrease also in Q-learning, depending on (for example) their initial values (even if initialized to 0, the max Q-value in a state may be negative) or if the environment is non-stationary.

Population methods

9. Population methods, for example genetic algorithms and particle swarm optimization, usually don't follow, or depend on, gradients in the search space. Explain what this means and why this should be both an advantage (at least one, there are several) and a disadvantage! ..... (3)

To follow gradients in the search space is to use the slope of the objective function (its derivative) to decide in which direction to move, for example in the direction of the steepest descent. (Many students did not answer this part)

Pro: The most important advantage (required for full credit) of not following gradients is that it should reduce the risk to get stuck in local optima. If we always follow gradients, we will almost certainly get stuck in the closest local optimum.

Pro: Another advantage is that the objective function does not have to be differentiable. In classification, for example, we usually want to minimize the number of misclassifications, which is an integer and not differentiable so we have to express the objective function in some other way if the method requires gradients.

Pro: A third possible advantage is that it might be too time-consuming to compute gradients for each member of a large population. Population methods work best if the computations per member are as simple as possible.

Cons: The disadvantage of not following gradients is that it probably makes it more difficult to find the exact optimum when close to it. Fine-tuning is difficult if you throw away information on gradients.

Unjustified claims that one way is faster than the other did not receive points.

UPPSALA
UNIVERSITET

10. Genetic Programming

a) Describe the most common basic form of *recombination* (*crossover*) used in Genetic Programming! (Note, GP, not GA)… (3)

GP operates on parse trees, as illustrated in class. Crossover is to select a random a sub-tree of each parent and swap them.

Some point reductions for vagueness (illustrated examples would probably have helped). No points to students who did not heed the warning (within parenthesis, the question was about genetic programming, not genetic algorithms)

b) Given this crossover operator, can we be sure that the new individual encodes a legal (evaluable) expression? If not, is this a problem?..... (2)

No, we cannot. In fact, it is very likely that crossover produces non-functional offspring this way, unless we put constraints on how the subtrees are selected. Even a simple operation such as swapping two constants may lead to a division-by-zero error, for example.

In theory it works anyway, since the non-functional individuals are not likely to get a high fitness, so they will probably not be selected for the next generation anyway. But this can be very inefficient. If we want GP to perform better than random search, we must consider this issue.