18 students wrote this exam. 2 passed with grade 4, 4 passed with grade 3 and 12 failed.

Commented version of the exam questions below.

Machine Learning written examination

Tuesday, June 13, 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
- RBF(N) = Radial Basis Function (Network)
- SCL = Standard Competitive Learning = LVQ-I without a neighbourhood function
- GNG = Growing Neural Gas
- EC = Evolutionary Computation (for example genetic algorithms)
- PSO = Particle Swarm Optimization
- ACO = Ant Colony Optimization
- AS = Ant System (the basic form of ACO discussed on this course)

There are fewer main questions on this exam compared to the previous one, but more connected sub-questions. It also contains more (sub)questions where you are expected to explain things. However, in all cases, there is a short explanation in a few sentences which would give full credit. Depending on how <u>you</u> explain things the length may vary, but if you need more than one page you are probably on the wrong track.

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

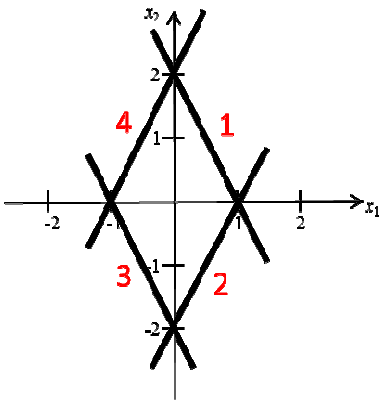
Telephone:
+46 18 - 471 10 09

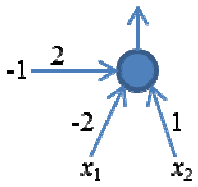
Telefax:
+46 18 – 51 19 25

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

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

UPPSALA
UNIVERSITET





Figures for question 1.

1. A binary perceptron defines a hyperplane in the input space and responds with a +1 on one side of it, 0 on the other. The figure to the left illustrates four hyperplanes, together forming the shape of a diamond. Below that, a binary perceptron which defines one of them. The hyperplanes are enumerated 1-4 and they cut the x_1-axis at -1 and +1 and the x_2-axis at -2 and +2.

   a) Which of the four hyperplanes is defined by the perceptron below? *(include a test to show that your answer is correct)* ..... (3)

   COMMENT: The perceptron defines hyperplane 4.

   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, the x_2 axis at 2 and the x_1 axis at -1. Some point reductions for non-convincing arguments or lack of a test.

   b) On which side of the hyperplane does the perceptron in the figure respond with a +1? *(include a test to show that your answer is correct)* ..... (1)

   COMMENT: The node reponds with +1 on the outside (of the diamond), i.e. with 0 on the origo-side. Easiest tested for origo. With both inputs equal to 0 the weighted sum is -2 (due to the threshold) and therefore the node's output is 0 on that side.

   c) What is the simplest way to adjust the weights of this perceptron so that it responds with a +1 on the other side? ..... (2)

   COMMENT: By negating all weights (including the threshold). Some students negated just some of the weights, not all of them. This would move the hyperplane – we want to flip it, not move it.

   1p deduction if only the input weights were negated, or only the threshold (having the same effect, moving the hyperplane to the position of hyperplane 2). 2p reduction if only one of the input weights were negated (in effect moving the hyperplane to the position of hyperplane 1 or 3).

UPPSALA
UNIVERSITET

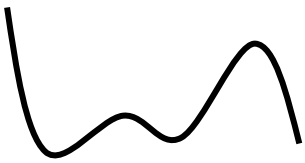
2. Multilayer perceptrons

a) Neural networks used for function approximation usually should have linear outputs. Why?..... (2)

COMMENT: Because we usually do not know the range of the target function, and even if we do it is unlikely that the range happens to be in the range of the usual sigmoid [0,1]. A function approximator should be able to output any value. This does not reduce the network's computational abilities. It is the hidden layer which must be non-linear, not the output layer.

This was intended to be an easy question, but for some reason it turned out not to be – only one student got full credit. Some students' answers were either trivially true (more or less given by the question), or not true at all (claiming that linear outputs are necessary to make them differentiable for example).

b) How many hidden nodes would be required (at least), for a conventional multilayer perceptron (with one hidden layer) to do a good approximation of the function below? Why?..... (3)



COMMENT: Five, since the function has five monotonic regions. The hidden layer of a multilayer perceptron consists of sigmoidal nodes and sigmoids are monotonic. We therefore usually need at least as many hidden nodes as there are monotonic regions in the target function. Partial credit (2p) for counting the number of inflection points instead (4 in this case).

Points deducted for arguments based on hyperplanes. They are not relevant here – this is function approximation, not classification. Function approximation does not work by trying to fit hyperplanes to the function (though in this case, that kind of reasoning would lead to the same result).

UPPSALA
UNIVERSITET

c) What is the likely effect on this approximation if you have too many hidden nodes?..... (2)

COMMENT: Too many hidden nodes is likely to lead to overfitting and the approximation to oscillate between the training set data points, i.e. solutions where there are more motonic regions than necessary to fit the data.

Point reductions for vague explanations or just talking about overfitting without saying anyting about what it means for the shape of the approximation.

d) One way to reduce this effect is to restrict the freedom of the hidden layer nodes. Several such methods have been discussed on this course, for example *weight decay*. What is weight decay?..... (2)

COMMENT: Weight decay is to let each weight strive for 0. A simple way to implement this: After updating a weight w, update it again using $w_{new}=(1-\varepsilon)w_{old}$*, where* $\varepsilon=[0,1[$ *is the decay rate.*

Weight decay can be used as a pruning technique (cutting connections with weights ending up close to 0), but it restricts the freedom of the hidden layer even if we don't cut connections. (It's also good for numerical reasons)

The general concept here is called regularization. Other regularization methods mentioned or discussed on this course include Early stopping, Dropout, Noise injection, Multitask-learning, and Lagrangian optimization (adding constraint terms to the objective function).

Point reductions for vague or incomplete explanations, or for misconceptions on what weight decay does. Some students seem to have confused weight decay with the idea of reducing the learning rate (η*) over time.*

UPPSALA
UNIVERSITET

3. Radial basis function (RBF) networks

   a) What is computed by a hidden node in a RBF network? (in the feed-forward phase) *(a description in words is fine as long as it contains all relevant aspects)* ..... (2)

   COMMENT: RBF hidden nodes compute the distance between the input vector $\boldsymbol{x}$ *and the weight vector* $\boldsymbol{w}$ *and feed that through a Gaussian activation function (or similar shape), thus producing a node value which is at max (1) for* $\boldsymbol{x}=\boldsymbol{w}$ *and decreasing with distance from* $\boldsymbol{w}$ *(for a Gaussian, assymptotically towards 0).*

   There is no threshold weight, but on the other hand there must be a parameter which defines the width of the activation function (for a Gaussian, the standard deviation, σ*). The comment in the question on "all relevant aspects" was intended as a hint not to forget the widths.*

   b) How are RBF networks usually trained? ..... (3)

   COMMENT: The two layers are trained differently. The output layer is just a layer of regular perceptrons and as such can be trained by for example the delta-rule.

   The hidden layer positions ($\boldsymbol{x}$*), are usually trained by competitive learning or K-means. Their widths (*σ*) are usually set to a constant value, computed after the positions have been found, for example to the average distance between a node and its closest neighbour (in weight space).*

4. Unsupervised learning

   a) Standard competitive learning and K-Means are closely related. How? ..... (2)

   COMMENT: They are equivalent, if standard competitive learning is trained by epoch/batch learning. (If pattern learning is used, standard competitive learning becomes more stochastic, due to the random order of patterns presentations.)

   b) What would happen to the network weights if you trained a self organizing feature map on random data (i.e. data drawn from a uniform distribution)? ..... (2)

   COMMENT: They would also be uniformly distributed. The point of self organizing feature maps is to preserve topology, i.e. to preserve statistical distributions.

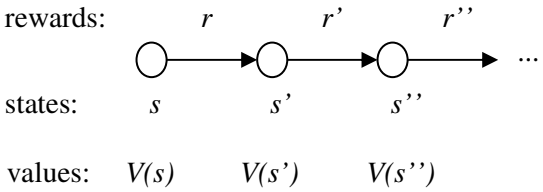
   Some students claimed that the weights would become equal, i.e. converge to the same spot. That would not preserve topology.

UPPSALA
UNIVERSITET

c) In Growing Neural Gas, new nodes are inserted between the node with the greatest accumulated error, and the node among its current neighbours with the greatest error. How is this error defined?..... (2)

COMMENT: The error is a (discounted) accumulated distance the node has moved around as a winner. A node which moves around a lot, when it wins, is likely to need help covering the data.

5. Most reinforcement learning algorithms are variants of an even more general concept called *temporal difference learning*. Use the state transition graph below to explain temporal difference learning



a) Define V(s) recursively in terms of the following rewards and values! ..... (2)

COMMENT: $V(s) = r + \gamma V(s')$, where γ is a discount factor in the range [0,1[.

This is a relation, not an update rule. Learning rates do not apply here. Nor do exploration rates (ε) which some students had confused with the discount factor.

b) Show how values are updated, using the TD(0) update rule! ..... (2)

COMMENT: They are updated proportionally (the learning rate, η, is the proportionality constant) to the TD-error, which is the difference between the two sides of the equation from the previous question. The update rule then becomes: $V(s) := V(s) + \eta[r + \gamma V(s') - V(s)]$.

c) In Q-Learning and Sarsa, values are associated to state-action pairs instead of just states, as in TD(0). A common way to make sure that Q-Learning/Sarsa agents explore is to use the ε-greedy algorithm. What is ε-greedy?..... (2)

COMMENT: ε-greedy is perhaps the simplest way to implement stochastic action selection. With probability ε, the agent explores (selects a random action). Otherwise (= with probability 1-ε) it exploits what it knows, i.e. takes the action that is currently believed to be best, i.e. the one with the highest Q-value.

Some students in effect just repeated what was given in the question – that ε-greedy is for exploration. The question here was how ε-greedy does this. Some point reductions also for not

stating how actions are selected (randomly) when ε-greedy decides to explore.

6. Population methods (evolutionary computing and swarm intelligence)

   a) How does the choice of using fitness or rank selection affect the risk of premature convergence in evolutionary computing, and why? . (3)

   COMMENT: Fitness selection may lead to greater risk of premature convergence, than rank selection.

   In Fitness selection, individuals are selected proporationally to their fitness value. If one individual has a much greater fitness then the others, that individual will be very likely to be selected and may therefore quickly dominate the population (in effect pulling all other individuals to it = premature convergence).

   In rank selection we instead select proportionally to rank in a list sorted after fitness. The best individual still has the greatest selection probability, but not that much greater than number two in the list (even if there is a big difference in actual fitness).

   Some students misunderstood what rank selection does, or failed to compare it to fitness selection, just explaining how both may lead to premature convergence.

   b) Which design choices in particle swarm optimization affect the risk of premature convergence, and how?..... (3)

   COMMENT: I should have asked for the 'most important' choices. Of course there may be others than than the one I had intended for students to discuss here, but the most important are:

   1. *The neighbourhood structure (how sparse/dense it is). The sparser the structure is, the less risk of premature convergence, since the particles will not affect each other directly. Therefore gbest (which is the densest form of lbest) is more likely to cause problems than lbest using a ring structure, for example.*
   2. *The weight parameters, θ_1 and θ_2, in the update formula, which control the balance of the cognitive and social components in the update formula. If the social component is too strong, the population will be more likely to converge prematurely.*
   3. *Population size (larger is better), (Not required for full credit)*

   Some students mentioned Vmax as well, but not with sufficiently good arguments on how it would affect premature convergence.

UPPSALA
UNIVERSITET

c) Which design choices in Ant System (the basic form of ant colony optimization discussed on this course) affect the risk of premature convergence, and how? ..... (2)

COMMENT: The intended answer here was the choice of the parameters α and β, which control the balance between following pheromone trails v.s. acting on local information in the update formula. It's the same argument as for the balance of the cognitive/social components in PSO. If the "social component" (which here corresponds to the pheromone trails) is to strong it is more likely to converge prematurely.

Population size does not have the same effect here, and if it does it may actually affect the risk both ways (but I did not expect students to note that). Same thing with evaporation rate – for normal value is should not affect this much, but in extreme cases it may (no evaporation rate at all for example, which may saturate the paths).

Full credit only required a good discussion on α and β.