UNIVERSITY OF SOUTHAMPTON COMP3008W1

SEMESTER 2 EXAMINATION 2009/2010

MACHINE LEARNING

Duration: 120 mins

Answer all parts of the question in section A (20 marks) and TWO *questions from section B (25 marks each)*

This examination is worth 70%. The coursework was worth 30%.

University approved calculators MAY be used.

 Number of pages 11

Section A

Question 1

(a) Explain what is meant by generalisation error and describe how it is estimated.

Test of very basic knowledge that is central to the whole course.

The generalisation error is the error of a learning machine on unseen data. It is measured using a validation set (i.e. a set of data which is not used in training). In a classification problem this would usually be the fraction of the validation set the learning machine failed on. For a regression task it would be something like the root mean squared error of the validation set.

(2 marks)

(b) Give a Bayesian interpretation for minimising the sum of the mean squared error plus a regularisation term.

The mean squared error can be thought of a log-likelihood given a Gaussian model for the errors, while the regularisation term can be viewed as a log-prior. Thus, minimising the mean squared error together with a regularisation term can been seen as calculating the maximum a posteriori solution.

(3 marks)

(c) Show that a MLP using linear nodes is no more powerful than a linear perceptron.

A simple proof students have seen in the course.

Assuming we have a two layer MLP. The prediction from any MLP using linear nodes with input x will be of the form

$$y = \sum_i w_i^2 \sum_j w_{i,j}^1 x_j.$$

Changing the order of summation we have

$$y = \sum_j x_j \sum_i w_{i,j}^1 w_i^2 = \sum_j x_j \bar{w}_j$$

where $\bar{w}_j = \sum_i w^1_{i,j} w^2_i$, but this is just the form of a single layer linear perceptron. To generalise to more layers we note that the first layer can be reduced to a single layer. We can carry on this reduction so that a linear network with any number of layers reduces to a single layer.

(5 marks)

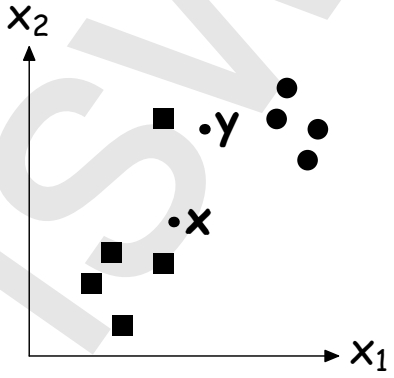
(d) Describe what K-fold cross-validation is and explain what its purpose is.

In K-fold cross-validation the data set is divided into K partitions. One partition is used for estimating the generalisation error and the other $K-1$ partitions is used for training. This is repeated K times using a different partition as the test set. Each time a new partition is used.

The purpose is to get an accurate estimate for the generalisation error using limited data.

(3 marks)

(e) The dataset below consists of two classes, squares and circles.



Give the classification of the points $\boldsymbol{x}$ and $\boldsymbol{y}$ produced by a K-Nearest Neighbours (KNN) algorithm with $K = 1$ and $K = 3$. Explain why increasing K acts like a regulariser?

Test student practical and theoretical understanding of KNNs.

The classification is given by

K	$\boldsymbol{x}$	$\boldsymbol{y}$
1	■	■
3	■	●

Increasing K reduces the influence of outliers and smooths the decision boundary. In the example above, using $K = 3$ (arguably) gives a more reliable classification for the point y.

(4 marks)

(f) Explain why normalising data can be important in the context of KNN.

Test understanding of data handling and how KNN works.

Normalising data will generally change the contribution of different features in determining the distance between feature vectors. Often the size of the feature vectors may reflect only an arbitrary choice of units. Normalising the feature vectors make each feature contribute roughly equally which can improve the results.

(3 marks)

Section B

Question 2 You are given a dataset $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)|k = 1, 2, 3, 4\}$ where

k	$\boldsymbol{x}_k$	y_k
1	(0,0)	0
2	(1,0)	1
3	(0,1)	1
4	(1,1)	0

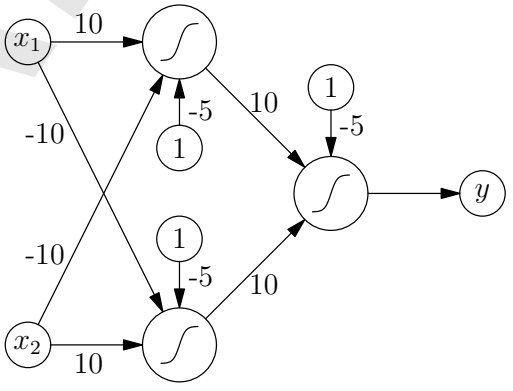
(a) Explain why a perceptron is not capable of correctly classifying this dataset.

Easy bookwork question

This data set is the classic XOR problem. It is not linearly separable (i.e. there is no line that separates the two classes of data). Perceptrons can only classify linear separable problems.

(3 marks)

(b) The diagram below shows a MLP with two input nodes, two nodes in the hidden layer and an output node. The additional nodes, labelled 1, are pseudo inputs for implementing a threshold for each node. The weights connecting the nodes are shown on the connecting lines. The output of the nodes is equal to $g(V) = 1/(1 + \mathrm{e}^{-V})$ where V is the weighted sum of the inputs.



TURN OVER

Show that this MLP will accurately classify the data above.

This is a "problem solving" question. At least it tests the student's understanding.

This MLP computes the function

$$y = g\big(10\, g(10x_1 - 10x_2 - 5) + 10\, g(-10x_1 + 10x_2 - 5) - 5\big)\,.$$

- **For $\boldsymbol{x}_1 = (0, 0)$ we get $y = g(10g(-5) + 10g(-5) - 5)$. But $g(-5) = 1/(1 + \mathrm{e}^5) = 0.0067 \approx 0$, Thus $y \approx g(-5) \approx 0$.**
- **We get exactly the same output for $\boldsymbol{x}_4 = (1, 1)$.**
- **For $\boldsymbol{x}_2 = (1, 0)$ we get $y = g(10g(5) + 10g(-15) - 5)$. But $g(-15) \approx 0$ and $g(5) = 1/(1 + \mathrm{e}^{-5}) = 0.9933 \approx 1$ thus $y \approx g(5) \approx 1$.**
- **By symmetry we get exactly the same result for $\boldsymbol{x}_3 = (0, 1)$.**

(15 marks)

(c) The problem shown is a parity problem in two dimensions. Explain why high dimensional parity problems are hard for MLPs to learn.

This test theoretical understanding

In principle, a large enough MLP should be able to solve parity as MLPs are universal approximators. However, parity is extremely linearly non-separable (high order). A separate hyperplane would be required to correctly classify every point. The MLP would therefore have to have an exponential (in the dimension of the input) number of hidden units. Learning such a function would be extremely difficult even if we trained on all possible input patterns (an exponentially large number). Generalisation would be almost impossible since changing a single variable in the input changes the categorisation.

(7 marks)

Question 3

(a) Describe the steps in Principal Component Analysis.

This is straight from the notes but non-trivial.

Given a data set $\{(\boldsymbol{x}_k, t_k) | k = 1, \ldots, P\}$ we compute the average vector $\boldsymbol{\mu}$

$$\boldsymbol{\mu} = \frac{1}{P} \sum_{k=1}^{P} \boldsymbol{x}_k$$

and the covariance matrix

$$\mathbf{C} = \frac{1}{P-1} \sum_{k=1}^{P} (\boldsymbol{x}_k - \boldsymbol{\mu}) (\boldsymbol{x}_k - \boldsymbol{\mu})^{\mathsf{T}}.$$

The covariance matrix will be a positive semi-definite matrix. We compute the eigenvalues λ_i and corresponding eigenvectors $\boldsymbol{v}_i$ of the covariant matrix C. We now choose a subset of the eigenvectors with the largest eigenvalues $\{\boldsymbol{v}_i | \lambda_i > \epsilon\}$. These eigenvectors our the principal components. We then project all inputs into a subspace spanned by the principal components. That is, we construct the matrix consisting of our n principal components

$$\mathbf{P} = \begin{pmatrix} \boldsymbol{v}_1^{\mathsf{T}} \\ \boldsymbol{v}_2^{\mathsf{T}} \\ \vdots \\ \boldsymbol{v}_n^{\mathsf{T}} \end{pmatrix}$$

and project a input pattern x into a lower dimensional vector

$$\boldsymbol{z} = \mathbf{P} (\boldsymbol{x} - \boldsymbol{\mu}).$$

We choose ϵ so that the principal components describe the majority of the variation in the inputs.

(7 marks)

(b) Explain the benefits of performing PCA as a preprocessing stage for supervised learning.

Tests integration of material from different parts of the course.

PCA is used to reduce the dimensionality of the input vectors. This means we can use much simpler learning machines (with few free parameters). Simpler machines are likely to give better generalisation performance.

TURN OVER

(3 marks)

(c) Explain why performing PCA on relatively small dimensional feature vectors may be advantageous despite throwing away information.

Test understanding of generalisation theory (integrating knowledge).

By reducing the dimensionality of the input features we reduce the tendency of a learning machine to over-fit that data. That is, we reduce the capacity of the learning machine. PCA throws away the data which is least informative in terms of reconstructing the original features. Hopefully, this information is also least informative in predicting the target label.

(5 marks)

(d) Explain why conventional PCA cannot be used with large dimensional images and explain how PCA can be carried out.

Test theoretical understanding of working in dual spaces. This has been covered in class.

Large images will easily have of order 1 million pixels. In conventional PCA this would require constructing a $n \times n$ matrix where n is of the order of 1 million and finding its eigen-decomposition. This is not possible. However, we can instead construct the matrix D with components $D_{k,k'} = (\boldsymbol{x}_k - \boldsymbol{\mu})^\mathsf{T}(\boldsymbol{x}_{k'} - \boldsymbol{\mu})$ this will have the same eigenvalue structure as the covariance matrix. Furthermore the eigenvectors of the covariance matrix can be obtained from the eigenvectors of D. This can be viewed as working in a subspace of the full image space spanned by the feature vectors.

(7 marks)

(e) Describe the connection between how PCA works on large images and the kernel trick used in SVMs.

Test ability to integrate knowledge from different areas.

In the kernel trick data is mapped into a very high-dimensional space. To exploit with this data SVMs work in the dual space spanned by the patterns. This is the same as the PCA described above.

(3 marks)

Question 4

(a) Describe the similarities and differences between multi-layer perceptrons (MLPs), radial basis function networks (RBFs) and support vector machines (SVMs).

Test non-mathematical knowledge.

Many ways to answer. The points I am looking for are

- **All three techniques based of the perceptron. In MLPs the earlier layers are perceptrons, in RBFs they are radial basis functions and in SVMs they are the features corresponding to the eigenvalues of a kernel.**
- **All three can be used for regression of classification.**
- **MLPs trained using back-propagation of errors. They have a non-unique solution. Complexity depends on number of hidden nodes. Liable to over-fit the training data. Often use *ad hoc* methods such as early stopping to prevent over-fitting. Can have many output nodes.**
- **RBFs typically use unsupervised learning to choose the centres for the input layer. The labelled data used to train the final layer (a perceptron). Training is fast. Can have many output nodes. Often use regulariser on the output layer. The solution found is unique.**
- **SVMs use a kernel function to perform a mapping into a very high dimensional feature space. An optimally stable perceptron is used in the feature space. This controls the capacity of the learning machine reducing the problem of over-fitting. The learning algorithm uses quadratic optimisation. The computation complexity grows as the number of training patterns cubed. For very large datasets SVMs can become impractical. The solution found is unique.**

(10 marks)

(b) Why are regularisation terms added to the error function?

(Test understanding of generalisation theory.)

Complicated learning machines are often necessary to learn difficult data-sets. However, given a limited amount of data, they are liable to

over-fit the training data. A regularisation term biases the machine to find "simpler" solutions which are more likely to represent the underlying rule being learned and so will have better generalisation performance.

(5 marks)

(c) A weight decay term has the form $\lambda \sum_i w_i^2$. Show how adding such a term modifies the update rule for the weights and hence explain why it is known as a weight decay term.

(Test ability to manipulate mathematical formula necessary for understanding machine learning.)

We typically train a learning machine by minimising an error term and a regularisation term

$$E(\boldsymbol{w}) = \sum_{k=1}^{P} \left(F(\boldsymbol{x}_k; \boldsymbol{w}) - y_k\right)^2 + \lambda \sum_i w_i^2$$

were we have a (multi-)set of data $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^P$ where $\boldsymbol{x}_k$ are the input features and y_k are the targets. $F(\boldsymbol{x}_k; \boldsymbol{w})$ is the output of the learning machine. To minimise the weights we move in a direction which minimises the error (i.e. in the direction of the negative gradient)

$$\begin{aligned}\boldsymbol{\nabla} E(\boldsymbol{w}) &= 2 \sum_{k=1}^{P} \boldsymbol{\nabla} F(\boldsymbol{x}_k; \boldsymbol{w}) \left(F(\boldsymbol{x}_k; \boldsymbol{w}) - y_k\right) + 2\,\lambda\,\boldsymbol{w} \\ &= 2 \sum_{k=1}^{P} \boldsymbol{\nabla} F(\boldsymbol{x}_k; \boldsymbol{w}) \delta_k + 2\,\lambda\,\boldsymbol{w}\end{aligned}$$

where $\delta_k = F(\boldsymbol{x}_k; \boldsymbol{w}) - y_k$ is the error between the predicted and true result. Thus we update the weights

$$\begin{aligned}\Delta \boldsymbol{w} = \boldsymbol{w}' - \boldsymbol{w} &= -\eta\,\boldsymbol{\nabla} E(\boldsymbol{w}) \\ &= -2\,\eta \sum_{k=1}^{P} \boldsymbol{\nabla} F(\boldsymbol{x}_k; \boldsymbol{w})\,\delta_k - 2\,\eta\,\lambda\,\boldsymbol{w}\end{aligned}$$

where w' are the updated weights and η is a learning weight. Solving for w' we find

$$\boldsymbol{w}' = (1 - 2\,\eta\,\lambda)\,\boldsymbol{w} - 2\,\eta \sum_{k=1}^{P} \boldsymbol{\nabla} F(\boldsymbol{x}_k; \boldsymbol{w})\,\delta_k.$$

Thus, at each learning step the previously learned weights are reduced by a factor $1 - 2\,\eta\,\lambda$, hence the term weight decay.

(10 marks)

END OF PAPER