Sample Machine Learning Exam Questions
With Answers

The exam will be **open book**: you can use Alpaydin's "Introduction to Machine Learning" as well as the lecture slides and any notes you've taken. You can use a calculator.
There are more questions in this sample than there will be in the exam: the exam will consist of 5 short questions and 4 more substantial questions, each consisting of 3–5 subquestions.

Good luck!

Answers in italics.

Questions

1. **Short answers, no justification required**

   (a) (**True** or **False**): The back-propagation algorithm learns a globally optimal neural network with hidden layers.
   False. Backprop learns a locally *optimal network*

   (b) (**True** or **False**): The error of a hypothesis measured over its training set provides a pessimistically biased estimate of the true error of the hypothesis (i.e., the real-world performance of the hypothesis would be better than the training set performance).
   False. The training error is optimistically biased since it's usually smaller than the true error

   (c) (**True** or **False**): Overfitting is more likely when the hypothesis space is small
   False. When the hypothesis space is small, it's less likely to find a hypothesis to fit the data very well,i.e., overfit.

   (d) (**True** or **False**): Because gradient descent is guaranteed to find a local optimum, it cannot overfit.
   False. Local or global optimum has nothing to do with overfitting.

   (e) (**True** or **False**): When training neural networks, the network weights are updated more often in incremental or stochastic gradient descent than in batch gradient descent.
   True. Stochastic gradient descent updates the weights after every pattern presentation. Batch learning only accumulates the weight changes until the whole batch of patterns has been presented once.

(f) (**True** or **False**): Because neural networks can only have real-valued inputs, they can only be used for regression problems.

False.

(g) (**True** or **False**): Gradient descent is guaranteed to find the global minimum.

False. It is guaranteed to find a local *minimum*

(h) (**True** or **False**): Implementing a k-nearest neighbour model requires a lot of memory because all training examples must be kept in memory.

True

(i) (**True** or **False**): Pruning in decision trees is a method to prevent overfitting.

True

(j) (**True** or **False**): Neural nets cannot model XOR.

False

(k) (**True** or **False**): Cross-validation can help determine the appropriate value for k in k-nearest neighbour.

True

(l) (**True** or **False**): Join-the-dots has a higher representational bias than linear regression.

False

(m) (**True** or **False**): Gradient descent is guaranteed to find a local minimum.

True

(n) (**True** or **False**): Any type of probability density estimator can serve as a basis for a MAP classification.

True

(o) (**True** or **False**): The k means algorithm can only differentiate between two clusters.

False

(p) (**True** or **False**): Support vector machines give a probability distribution over the possible labels given an input example.

False

2. **Decision Trees** Summertime is festival time: Pinkpop, Pukkelpop, Parkpop, to name but a few. To help you choose which of the acts to go see at the next festival you're visiting, you will develop a model based on the data in table 1.

Table 1: Band data

Style	Seen before	Set length	Like
Grunge	Yes	20	No
RnB	Yes	20	No
Grunge	No	20	No
Britpop	No	20	No
Britpop	No	20	Yes
Grunge	No	30	Yes
RnB	No	20	Yes
RnB	Yes	30	Yes

(a) What are the entropies $H(Like|Style = Britpop)$ and $H(Like|Set\ length = 30)$? Note that Alpaydin notates entropy as $\phi(p, 1-p)$ [1]

1 and 0

(b) Which attribute would the decision tree building algorithm choose to use for the root of the tree (assume no pruning)

Set length

(c) How does the standard Decision Tree algorithm (ID3) decide which attributes to put at the root of the tree?

It greedily selects attributes with the highest information gain. These are the attributes that it estimates to have the highest information gain. So it puts the attributes that it thinks give the most information about the class labels at the top.

(d) ID3 would result in the decision tree shown in Fig. 1. What would be this decision tree's training set error? Express your answer as the percentage of records that would be misclassied.

1 (record nr. 4) out of 8: 12.5%

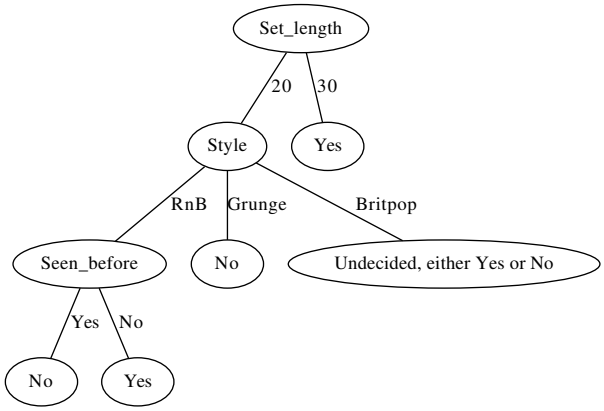
(e) Some implementations of the standard decision tree ID3 algorithm do not specify what to do if all attributes have zero information gain. Suggest (and motivate) how to proceed in this case. Use no more than three sentences.

The main thing is not to answer 'terminate'. The simplest option is randomly pick one. The motivation I gave in the lecture is that the XOR problem cannot be solved when terminating in this situation.

(f) Someone proposes a new scheme to prevent overfitting: she suggests to set a pre-defined maximum depth for the decision trees.

[1] If your calculator can't do log_2, use one of the following equations: $log_2(x) = 1.44 \cdot ln(x)$ or $log_2(x) = 3.32 \cdot log_{10}(x)$. If you didn't bring a calculator, you may give the answer as an expression.

Figure 1: The Generated Decision Tree



When the standard algorithm reaches this depth, it terminates. Could this help to prevent overfitting? Why (not)?

It could help, since it reduces the space of possible hypotheses (limits the expressivity of the decision trees).

3. **Bayesian Classifiers**

(a) Consider a naive Bayes classifier trained on the same dataset from Table 1. How would that predict `Like` given the input `Style = Grunge, Set length = 20`? Show your calculations.

$y_{predict} = maxarg(P(E = v|data)$.
For naive Bayes,
$P(Like = Yes|Style = Grunge, Set\ length = 20) = P(Like = Yes) \cdot P(Style = Grunge|Like = Yes) \cdot P(Set\ length = 20|Like = Yes) = \frac{4}{8} \cdot \frac{1}{4} \cdot \frac{2}{4} = 0.0625$
$P(Like = No|Style = Grunge, Set\ length = 20) = P(Like = No) \cdot P(Style = Grunge|Like = No) \cdot P(Set\ length = 20|Like = No) = \frac{4}{8} \cdot \frac{2}{4} \cdot \frac{4}{4} = 0.25$
So: $Like = No$

(b) Suppose that in a classification problem one of the possible outcomes is extremely uncommon. Would you prefer to use MAP or MLE in this case? Why?

MAP is the preferred option because it corrects for the a priori *probabilities, while MLE does not.*

(c) When building a naive Bayes classifier, we assume that the attributes are conditionally independent given the class label. What is the benefit of this assumption?

It makes it very simple to calculate joint probabilities because you can simply multiply the probabilities for the individual attributes. This allows fast calculation while using very little memory.

(d) For any given set of likelihood estimators, under which circumstances will the MAP and ML predictions be the same?

If the a priori probabilities for all possible outputs are the same: $\forall v \in \{possible\ values\ of\ y\} : P(y = v) = c$, with c some constant. Also known as 'equal priors.'

4. **Neural Nets and Regression** *(15 Points)*

(a) Does increasing the number of hidden nodes in a multilayer perceptron improve generalisation? Why (not)?

The short answer: increasing the number of hidden nodes increases its representational ability, which increases the risk of overtraining (learning all patterns by heart in the extreme case). However –the longer answer–, if the initial number of hidden nodes was too small to solve the problem at all, then increasing the number should improve generalisation. Up to the point where the number of nodes reaches the required number to represent the function, that is. Extra marks if you make this last observation

(b) When training perceptrons with gradient descent, one can add momentum: why and how does momentum help?

Momentum –as the name implies– adds momentum to the descent: the search doesn't stop at a minimum, but 'overshoots' it. This overshooting is hoped to get the descent out of local minima.

(c) What would be the consequence of setting momentum too high? Too low?

Too low: overshooting doesn't get us out of local minima; too high: we may overshoot the global minimum, possibly ending up in a local minimum further on, but at least it will take longer to converge as we go back and forth across the optimum (similar to setting learning rate too high).

(d) (Multiple choice) Back-Propagation (BackProp for short) is an algorithm for training multi-layered feedforward networks. How does it work?

A. Weights are replaced to match one training sample at a time

B. Weights are gradually changed to minimize the error over the training data

C. Nodes in the hidden layers are added or removed to improve generalisation

D. Margins are maximised by using non-linear kernels

Answer: B

5. **Instance-Based Learning**

   (a) The following data set (Fig. 2) shows classification data with two classes: plus and minus. Which of the two nearest neighbours classifiers has the largest Leave-One-Out Cross-validation error on this data set?

   i. 1-NN
   ii. 3-NN

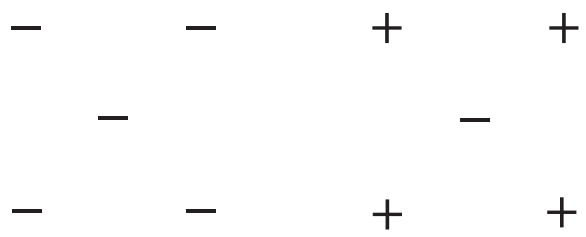


Figure 2: A classification data set

1-NN since 1-NN CV err: 5/10, 3-NN CV err: 1/10

   (b) Suppose that you want to use k-nearest neighbour on a data set that contains a non-numeric attribute 'weight' which can be *light*, *medium* or *heavy*. Could you re-code this attribute to be able to use it? How?

   (c) The same for an attribute 'material' which can have the values *aluminium* or *plastic*.

   (d) What is the role of a kernel function in instance-based learning and why does it allow instance-based learners to take all examples into account instead of only the k nearest neighbours?

   The kernel function focuses the average around the instance that is queried: instances at increasing distance have decreasing influence. Thus, the neighbourhood size can become infite: the kernel will make sure that far-away instances have little or no influence.

6. **Support Vector Machines**

   (a) Fig. 3 shows a data set with two real-valued inputs and one categorical output class. Positive points are shown as rectangles, negative points as triangles.

   Suppose that you are using a linear SVM with no provision for noise (i.e. a linear SVM that is trying to maximize its margin while ensuring that all datapoints are on their correct sides of

the margin). Draw three lines in the diagram showing the classification boundary and the two sides of the margin. Circle the support vectors.

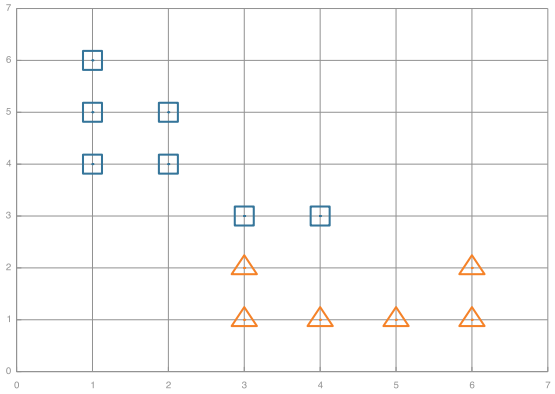


Figure 3: SVM data set

See Fig 4; the dotted lines indicate boundary, margins and support vectors.

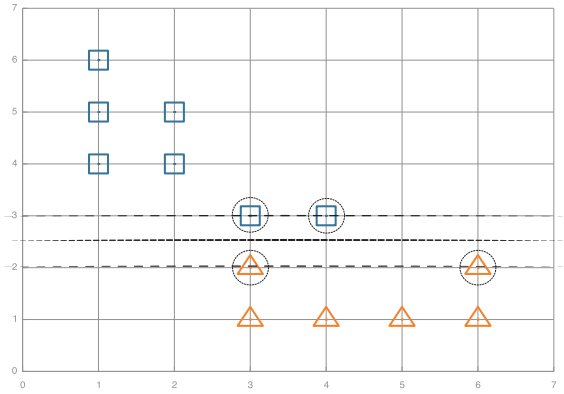


Figure 4: SVM data set with class boundary, margins and support vectors shown

7. **Hypothesis Comparison and Cross Validation**

(a) On a much larger data set than that in table 1, someone has developed a classifier that correctly classifies 700 out of 1000 test examples. Give an estimate of the true error of this classifier, and a 90% confidence interval around that estimate (you may

give your confidence interval in the form of an expression). [2]

The answer is the following (where $Z_{90} = 1.64$) : $\frac{300}{1000} \pm 1.64 \times \sqrt{\frac{0.3 \times (1-0.3)}{1000}} = 0.3 \pm 0.0024$

(b) In the above problem, if you increase the size of the test set by a factor of 10, assuming a similar error rate of 30%, what happens to the confidence interval?

It becomes smaller

(c) When tuning the parameters of a learning algorithm (e.g., the number of neighbours in kNN) to a given application, why is it necessary to use cross-validation or a test set?

8. **Understanding a novel learning method: Naive Bayes/Decision tree hybrids**

Suppose you were to augment a decision tree by adding a Naive Bayes classifier at each leaf. That is, consider learning a decision tree of depth k, (where k is smaller than the number of variables n), where each leaf contains not a class, but a Naive Bayes classifier, which is based on the $n - k$ variables not appearing on the path to that leaf.

(a) Briefly outline a learning algorithm for this representation, assuming that k is a fixed parameter.

The simplest idea was to grow a decision tree by the usual method to depth k. Then learn a NB model over the examples in each of the leaves at level k. More sophisticated versions of this method, eg. iterating one round of decision tree growth with one round of NB learning, were also acceptable.

(b) Will this result in having the same Naive Bayes classifier at each leaf? Why or why not? If not, how will they be different?

The NB classifiers learned at each leaf will be different because they will have been trained on different examples, based on the different paths the examples take through the tree and the corresponding conditions each example satisfies or doesnt.

(c) Briefly describe a plausible means of selecting a value for k.

Two common answers here: either a pruning-like method, based on information gain or something similar, or a cross validation method, ie, growing the tree for various values of k until performance on held-out data starts to fall. Whichever method you defined, it is essential to use held-out (i.e., not part of tha training set) data for cross-validation, eg. choosing k so as to maximize performance directly on training data was not acceptable.

[2] The table of z-values:

N%:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58