Lunds Tekniska Högskola EDAN95
Institutionen för datavetenskap HT 2018

Tillämpad Maskininlärning
Applied Machine Learning
Tentamen 2019–05–02, 14.00–19.00, Sparta:B

You can give your answers in English or Swedish.
You are allowed to use a calculator.
You are welcome to use a combination of figures and text in your answers.
100 points, 50% needed for pass (preliminary).

1 Ensemble learning algorithms (JM), 5p

Multiple choice: Please answer each of the following five sub-questions by **writing down** the letter(s) corresponding to the correct answer(s). Note that a wrong answer may result in negative points, hence, it can give a negative result for the sub-question and thus cancel out positive points from other sub-questions, but the overall result for the question cannot become negative.

1. (1p) **Bootstrapping** is a technique of manipulating training data consisting of:
   (a) given a series of training set instances, training classifiers on them and averaging their output;
   (b) given a series of training set instances, training classifiers on them taking into account missclassified instances;
   (c) sampling instances with replacement.
2. (1p) **Boosting** is a technique of manipulating training data consisting of:
   (a) given a series of training set instances, training classifiers on them and averaging their output;
   (b) given a series of training set instances, training classifiers on them taking into account missclassified instances;
   (c) sampling instances with replacement.

3. (1p) **Bagging** is a technique of manipulating training data consisting of:
   (a) given a series of training set instances, training classifiers on them and averaging their output;
   (b) given a series of training set instances, training classifiers on them taking into account missclassified instances;
   (c) sampling instances with replacement.

4. (1p) **Ensemble learning** uses:
   (a) only decision trees as its building blocks;
   (b) only recurrent neural networks as its building blocks;
   (c) arbitrary classifiers as its building blocks.

5. (1p) **AdaBoost** is a boosting algorithm that got its name:
   (a) from Ada Lovelace, a computing science pioneer from 19th century;
   (b) from the term "Adaptive Boosting";
   (c) from the first names of its creators, Adam, Daniel and Ari.

2 k-Means (JM), 10p

Given the following data set: $\{(1.0, 1.0), (3.0, 4.0), (5.0, 7.0), (1.5, 2.0), (3.5, 5.0), (3.0, 1.0), (4.5, 5.0), (3.5, 4.5)\}$ cluster it into `three` clusters using k-means algorithm. You may begin with the clusters initialized to the first three points in the data set (but feel free to use some other seed that you deem useful). Explain what you do.

3 Decision trees (JM), 10p

Given the set of instances shown in Table 1, representing mobile phone usage (low, medium or high) by some individuals, build a decision tree properly classifying all the instances. Your trees should minimize Gini impurity index in every node (a.k.a. CART algorithm).

The **Gini impurity index** may be defined as follows: it is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity index can be computed by summing the probability p_i of an item with label i being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

Income	Age	Education	Marital Status	Usage
Low	Old	University	Married	Low
Medium	Young	College	Single	Medium
Low	Old	University	Married	Low
High	Young	University	Single	High
Low	Old	University	Married	Low
High	Young	College	Single	Medium
Medium	Young	College	Married	Medium
Medium	Old	High School	Single	Low
High	Old	University	Single	High
Low	Old	High School	Married	Low
Medium	Young	College	Married	Medium
Medium	Old	High School	Single	Low
High	Old	University	Single	High
Low	Old	High School	Married	Low
Medium	Young	College	Married	Medium

Table 1: Mobile phone usage.

Thus Gini impurity index G for a set of items with M classes may be found as follows. Suppose $i \in \{1, 2, \ldots, M\}$, and let p_i be the fraction of items labeled with class i in the set.

$$G = \sum_{i=1}^{M} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{M} p_i(1-p_i) = \sum_{i=1}^{M} p_i(p_i - p_i^2) = \sum_{i=1}^{M} p_i - \sum_{i=1}^{M} p_i^2 = 1 - \sum_{i=1}^{M} p_i^2$$

Further questions on following pages

4 Neural networks (PN), 35p

In this task, you will build an elementary recurrent neural network to classify (categorize) texts. You will use the Keras API.

4.1 Word Encoding

In this section, you will explain some key concepts of word encoding. You will examine two techniques: One-hot encoding and embeddings.

One-Hot Encoding.

1. Describe conceptually what is one-hot encoding for categorical values.
2. Using the two documents:

   D1: Chrysler plans new investments in Latin America.
   D2: Chrysler plans major investments in Mexico.

   give the list of unique words. You will sort this list alphabetically and you will set the words in lower case.
3. What is the dimension of the vectors needed to represent a word in this corpus;
4. Give the representation of the words *plans* and *latin*. You will use the same indices as in the sorted list. Note that there is one vector per word.

Programming. You will now write a Python program to convert your corpus into one-hot encoded vectors. You will suppose that this corpus is available as a list of lists as for example:

```
corpus = [['Chrysler', 'plans', 'new', 'investments', 'in', '
  'Latin', 'America', '.'],
['Chrysler', 'plans', 'major', 'investments', 'in', 'Mexico', '.']]
```

1. You will collect and sort a list of unique words from this list of lists;
2. You will associate each unique word with an index (an integer);
3. You will create a one-hot vector for each unique word. Note that there is one vector per word;
4. You will convert the corpus into lists of one-hot vectors.

Word Embeddings.

1. One-hot encoding is not usable for large corpora. Describe why;
2. In addition, one-hot encoding has poor semantic properties. Describe why;
3. Describe concretely what is a word embedding. You can think of the Glove embeddings that you used in the assignments;
4. Give an idea on how to obtain the embeddings.

4.2 Building a Network

In this exercise, you will build a recurrent neural network to categorize texts from the Reuters corpus. This corpus consists of business newswires, where each text is assigned a category from the business domain.

Recurrent networks.

1. Describe what is a simple recurrent neural network and how it is different from a feed-forward architecture;
2. Describe what is a long short-term memory network (LSTM).

Preprocessing the data. To read the dataset, Keras provides a `reuters.load_data()` function that loads the newswires. You will use this code:

```
from keras.datasets import reuters
from keras.preprocessing import sequence
max_features = 10000
maxlen = 500
print('Loading data...')
(input_train, y_train), (input_test, y_test) = reuters.load_data(
     num_words=max_features)
print(len(input_train), 'train sequences')
print(len(input_test), 'test sequences')
print('Pad sequences (samples x time)')
input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
input_test = sequence.pad_sequences(input_test, maxlen=maxlen)
print('input_train shape:', input_train.shape)
print('input_test shape:', input_test.shape)
```

The content of the newswires is stored in the `input_train` variable and the categories in `y_train`. `input_train[0]` contains the first text and `y_train[0]`, the category of the first text, a number. The content of `input_train[0]` is not the text itself, but a list of integers corresponding to word indices. We will only use the most frequent words up to a certain number, here 10,000. This number is stored in `max_features`. The other words will be marked as unknown. `maxlen` is the maximum length of a sequence. If a newswire is longer, it will be truncated. Similarly, `(input_test, y_test)` contains the test set.

1. Why do we need to set a maximum for the length of a sequence? Would it be possible to have a different strategy?
2. Describe what is the purpose of `sequence.pad_sequences()`;
3. What will be the output of this function;
4. As loss, we will use categorical crossentropy. To have compatible targets, convert `y_train` and `y_test` into one-hot vectors. You will use the Keras built-in `to_categorical` function:

   ```
   from keras.utils.np_utils import to_categorical
   ```

Building a network. You will now design a simple architecture to carry out the categorization.

1. Build a sequential feedforward network consisting of:
   - An embedding layer. The length of the embedded vectors will be of 32;
   - A bidirectional LSTM layer, where the number of units will be of 32;
   - A dense layer.

   You will use the `Sequential()`, `Embedded()`, `Bidirectional()`, `LSTM()`, and `Dense()` classes.
2. Compile your network. You will select an optimizer as well as a loss;
3. Fit your model;
4. Evaluate it using the appropriate evaluation function.

4.3 Precision and Recall

Table 2 shows the confusion matrix resulting from a classification experiment. In this exercise, you will clarify how to read such a matrix and compute the precision and recall for the three classes. Just write the fractions, for instance 1/3 and not 0.33.

1. Describe how to interpret the matrix in Table 2;

True\Predicted	Class 1	Class 2	Class 3	Class 4	Precision	Recall
Class 1	40	10	20	10		
Class 2	15	40	10	5		
Class 3	15	5	60	30		
Class 4	10	5	30	80		

Table 2: A confusion matrix

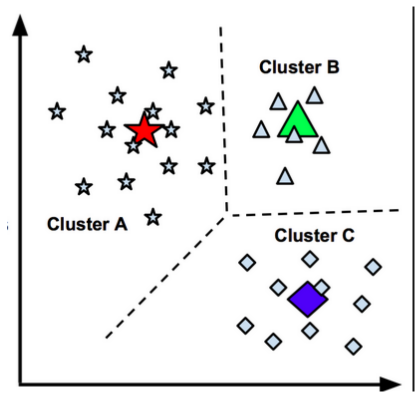
2. What would be a perfect matrix? Give the corresponding values for Table 2;
3. For a given class, the recall is defined as the true positives divided by the sum of the true positives and false negatives. For Class 1, how many true positives are there (correctly classified as Class 1)? And false negatives (wrongly classified as not being in Class 1)?
4. For the four classes, compute the recalls. (Write them in the form of fractions);
5. For a given class, the precision is defined as the true positives divided by the true and false positives. For Class 1, how many true positives are there? And how many false positives (wrongly classified as being in Class 1)?
6. For the four classes, compute the precision.
7. Define what the F1 score is. (Just define it, do not compute it).

Further questions on following pages

5 Bayesian Learning / Classifiers (VK), 20p

1. Consider the joint probability $P(\bar{x}, y)$, where $\bar{x}$ is an observation and y is a class label. What is the *Bayesian Theorem*? What part in that theorem
   - is the likelihood?
   - is the prior?

2. Imagine you have three classes (here: red/star, green/triangle, blue/square), let μ_r, μ_g, μ_b be the means of the three classes, and let $\bar{x}$ be a new observation. Using a Maximum-Likelihood approach, we aim to find the class to which $\bar{x}$ belongs with maximal probability. Please formalise this Maximum-Likelihood formulation either with a mathematical notation or through pseudo-code.



3. One can identify the class membership of $\bar{x}$ by looking for the μ that *minimises* $\|\mu - \bar{x}\|_2$. In case of the maximum Likelihood, we want to *maximise*. Please specify the maximum likelihood while assuming a Gaussian probability distribution.

4. Note that in this 2D example the Gaussian will use a 2D covariance matrix.
   - How does this covariance look like if we assume the two dimensions to be independent (*Naïve Bayesian*)? Please provide the mathematical proof.
   - How does this covariance look like if we assume statistical dependency between the two dimensions?
   - Is it valid to assume statistical independence? What are the advantages / disadvantages of doing so?

5. What is IID, and why is it important? Please provide an example.

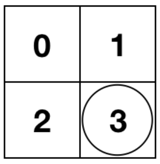
6 MDPs / Reinforcement Learning (ET), 20p

Consider an agent A, that would at any given point in time be in a state s, in which it can apply an action a. This will lead to it changing into a new state s' and receiving a reward r according to the transition function $\delta(s, a) \rightarrow s'$ and the reward function $r(s, a) \rightarrow \mathbb{R}$.

1. What is described by
   - the function $\pi(s) \rightarrow a$?
   - the function $U^{\pi}(s) \rightarrow \mathbb{R}$?

2. Note down and explain Bellman's equation, i.e., the equation that puts a given π and $U^{\pi}(s)$ in relation to each other! What is the equation's relationship with the algorithm called *Value Iteration*?

3. Assume a scenario with an agent in a grid world of 2x2=4 states according to the figure below. State 3 in the bottom right corner denotes the goal state, giving a reward of 1 when being reached from any of the two adjacent states, all other actions from any state produce 0 as reward. The agent can only move straight, i.e., there are four possible actions (UP=0, RIGHT=1, DOWN=2, LEFT=3), and an attempted move towards a wall will result in not moving.
   Note down the transition and reward functions, as well as the optimal π^* (explain briefly what is meant by optimal)!
   Compute the corresponding optimal $U^{\pi^*}(s)$ for states 0, 1, and 2 using *Value Iteration* for two iterations with a discount factor γ of 0.95!



4. In the lectures on Reinforcement Learning also an algorithm called *Policy Iteration* was introduced. What does it do in comparison to *Value Iteration*? Illustrate your explanation with a mathematical formulation or program code (pseudocode is sufficient)!

5. Straight forward *Policy Iteration* as assumed here only works based on given reward and transition functions (i.e., with a fully observable problem). What can you do in the more general case of only getting rewards and new states based on applying actions? Give one example of another learning algorithm discussed in the lectures and explain its principle idea!