Project 4: Classification

Problem 1 - Decision Trees

Use the ID3 algorithm for the following question.

1. Build a decision tree from the given tennis dataset. You should build a tree to predict PlayTennis, based on the other attributes (but, do not use the Day attribute in your tree.). Show all of your work, calculations, and decisions as you build the tree.

   What is the classification accuracy?

2. Is it possible to produce some set of correct training examples that will get the algorihtm to include the attribute Temperature in the learned tree, even though the true target concept is independent of Temperature? if no, explain. If yes, give such a set.

3. Now, build a tree using only examples D1–D7. What is the classification accuracy for the training set? what is the accuracy for the test set (examples D8–D14)? explain why you think these are the results.

4. In this case, and others, there are only a few labelled examples available for training (that is, no additional data is available for testing or validation). Suggest a concrete pruning strategy, that can be readily embedded in the algorithm, to avoid over fitting. Explain why you think this strategy should work.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Table 1: The play tennis dataset

Problem 2 - Neural Networks

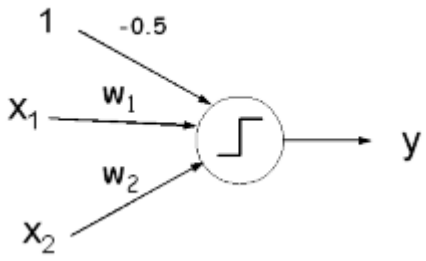
Given is the following single neuron perceptron. In this one-layer perceptron model, the neuron calculates a weighted sum of inputs. Then, it applies a threshold to the result: if the sum is larger than zero, the output is 1. Otherwise, the output is zero.

Consider the following examples, where Z is the desired output (indeed, this is the OR function).

In this question, you will apply the Perceptron update algorithm to automatically learn the network's weights, so that it classifies correctly all the training examples. The algorithm is simple:

Iterate through the training examples, one by one (if the last example was used, and the algorithm hasn't converged yet, start again from the first example, and so forth.).

For every example i:



X_1	X_2	Z
0	0	0
0	1	1
1	0	1
1	1	1

- Calculate the net's output Y_i.
- Multiply the error ($Z_i - Y_i$) by the learning rate η. Add this correction to any weight for which the input in the example was non-zero.

  That is, if for the current example i $X_1 = 1$, then update $W_1^0 \rightarrow W_1 + \eta(Z_i - Y_i)$, etc.
- If the network outputs the correct result for all of the training set examples, terminate.

Our questions:

X_1	X_2	W_1	W_2	Z	Y	Error	W_1	W_2
0	0	0.1	0.3					
0	1							
1	0							
1	1							
0	0							
0	1							
1	0							
1	1							
...								

Table 2: Results format

1. Apply the algorithm for the given training examples. Use learning rate $\eta = 0.2$. Assign the weights the initial values $W_1 = 0.1$, $W_2 = 0.3$.

   Give your results as specified in Table 1.

   You should expect to getting the final weights within only a few passes over the training examples.

2. The perceptron training algorithm is in fact a simple gradient descent update. In this question, you will derive this algorithm.

   The approach for training a perceptron here is to minimize a squared error function.

   - Give the definition of a squared error function, E, in terms of W_1, W_2, X_{i1} ,X_{i2} and Z_i.
   - Each weight should now be updated by taking a small step in the opposite direction of its gradient (so as to minimize the error):

   $$W^0 = W - \eta \nabla E(W)$$

   Show how this translates into the algorithm that you applied in the previous question.

3. In practice, the training example may be noisy. Suppose that there are contradicting examples in the training set: for example, an additional example, where $X_1 = 1$, $X_2 = 1$, $Z = 0$. How do you think this will affect the algorithm's behavior? (you are welcome to go ahead and try).

Problem 3 - Naive Bayes and KNN

In this assignment you will build classifiers that should distinguish between valid email messages and spam.

You are given a labelled dataset, included in the file. The dataset is split into two zipped files: train and test. Each file will unpack into a directory that includes two kinds of email messages (one per file): SPAM (file names starting with sp) and MAIL (file names starting with numbers).

You will build classifiers using the train data only. The classifier performance should be evaluated using the test data only.

We will represent every message as a bag of words, as detailed below.

1. Following is some relevant facts that you may want to use in building a Naive Bayes (NB) classifier.

   As a Bayesian classifier, it computes the following:

$$Pr(SPAM \mid message) = \frac{Pr(SPAM)\,Pr(message \mid SPAM)}{Pr(message)} \sim Pr(SPAM)\,Pr(message \mid SPAM)$$

$$Pr(MAIL \mid message) = \frac{Pr(MAIL)\,Pr(message \mid MAIL)}{Pr(message)} \sim Pr(MAIL)\,Pr(message \mid MAIL)$$

   Your classifier will write out "SPAM" if Pr(SPAM|message) > Pr(MAIL|message) and "MAIL" otherwise.

   How to compute those quantities? Pr(SPAM) and Pr(MAIL) should be easy given the training data. The other two terms, Pr(message|SPAM) and Pr(message|MAIL), are where the "naive" part of Naive-Bayes comes in.

   Let us represent a message as a sequence of n words, w_1, w_2...,w_n (where w_i may be equal to w_j). That is, we consider as features the occurrences of the message words.

The Naive-Bayes model assumes independence between the features. Thus, it computes the probability of a message conditioning on SPAM and MAIL as a product of the feature values probabilities, given SPAM and MAIL. So we have:

Pr(message|SPAM) = Pr(w_1(message)|SPAM)×Pr(w_2(message)|SPAM)...×Pr(w_n(message)|SPAM)

Pr(message|MAIL) = Pr(w_1(message)|MAIL)×Pr(w_2(message)|MAIL)...×Pr(w_n(message)|MAIL)

What are the probabilities *Pr(w_i(message)|SPAM), Pr(w_i(message)|MAIL)*?

These can be readily obtained from the training data as well. For that, you should count the occurrences of words in the training data for each class. For example, suppose that in the SPAM training messages, *Count('cheap') = 200, Count("now") = 50*, etc.; and *Count(allwords) = 10000*. Then, *Pr("cheap"|SPAM) =* $\frac{200}{10000}$ *= 0.02* .

However, there is a small fix to this simple ratio. While you obtain word counts from the data available for training, the messages in the test set may contain some words that did not occur in the training set (these are Out-Of-Vocabulary words, or OOV). In these cases *Pr(w_k(message)|SPAM)* would be zero, meaning that *Pr(message|SPAM)* be zero as well. To avoid that, you may need to smooth your probability estimates.

The most simple smoothing mechanism is Laplace's law or add-one. Here, you simply add 1 to the count of each word (including the OOV token); you have to adjust the total count, as well, in order to produce a probability distribution:

$$P(w \mid CLASS) = \frac{Count(w \mid CLASS) + 1}{N(CLASS) + V}$$

The vocabulary size is the count of all the unique words that were observed in both the training and the test datasets. Per the previous example, suppose V = 1000. Then,

$$P("cheap" \mid SPAM) = \frac{Count("cheap" \mid SPAM) + \lambda}{N(SPAM) + V} = \frac{200 + 1}{10000 + 1000}$$

What is the probability of an unseen word in this case? (it's the same, only that *Count(OOV)=0*)

Finally, multiplying many small probabilities, you may run into underflow issues. To avoid that, the product $\prod Q(w_i \mid CLASS)$ can be exchanged with $\sum \log(w_i \mid CLASS)$.

- Write code that trains a NB classifier, as described. Your code should:
  a. train a model based on the training data
  b. output the accuracy rates per the training and test sets, where

$$Accuracy = \frac{\#\ -\ of\ -\ correctly\ -\ classified\ -\ messages}{messages}$$

- Train a NB using random subsets of 20%,40%,60%,80%, and 100% of the training data. Report the corresponding accuracies per the train (that is, the net portion of training data that was used) and test

data. Since results vary depending on the particular random set used, report average results over 5 runs. Give your conclusions.

- Implement cross validation as follows. Split the training set randomly into 10 portions, of the same size. Train a model 10 times - where in every iteration one of the training data portions serves as the test set, and the model is trained on the rest of the data. Report the accuracy per individual test portion, and the overall cross-validation accuracy. Do you find the cross validation results reliable, comparing to evaluating accuracy using a separate test set?

2. Write code that implements a KNN classifier, for the same problem. For that, you will represent every message as a vector of word counts. The vector length is the vocabulary size (per the training data). Naturally, most of its entries would equal zero. For example, suppose that V=1000, and that i = 52 is the index of the word "cheap". If message m has "cheap" appearing 7 times in it, then its representing vector will have 7 in the 52nd index.

   For every message in the test set, calculate the similarity to each of the messages in the training set using a **cosine similarity measure**. Consider the top k most similar messages, to determine the test message class. The cosine similarity measure is calculated as follows:

$$Sim(A,B) = \frac{AB}{|A||B|} = \frac{x_{1A}x_{1B} + x_{2A}x_{2B} + \cdots}{\sqrt{(x_{1A}^2 + x_{2A}^2 + \cdots)}\sqrt{(x_{1B}^2 + x_{2B}^2 + \cdots)}}$$

   where x_{iA} is the count of the word of index i in document A.

   - Write the code for KNN classification.
   - Train the classifier using K = 1, K = 3, K = 5, K = 19. What are the corresponding accuracies on the test data?

3. Now, as a pre-processing step, allow to choose the top T features (words). We recommend that you use information gain as your criteria. Consider only those words that appear more than 50 times in the training data. Note that you need not select the features in a hierarchical manner.

   What are the top 10 most predictive words found by your measure?

   What is the test accuracy, using NB and KNN models (assign K=3) that are structured using all of the training data, for T = 20, T = 50, T = 100, T = 200, T = 500? Explain your results. What are the effects of feature selection on each of these algorithms?

4. Suggest your own model, including new features, which you think may be useful for this problem. One example of a feature is the length (in bytes) of a message. Another feature is the number of words. Or the number of occurrences of the letter Q. You can come up with features very easily. Use the KNN or the NB model.

   Did you succeed to improve performance on the test set (comparing to previous results)? provide your features, type of model (NB or KNN) and results. If you managed to improve performance you will get 20 points for this question. Otherwise, you will get some of these points, depending on how we like your model... (we will try to be generous). Also, we may publish your model, if it does substantially well. (But, hey, remember that you are not allowed to look at the test set!)

Problem 4 – Performance Evaluation of classifiers for espam dataset

Consider a case study that involves the performance evaluation of the following classifiers on the espam dataset:

Decision Tree algorithm CART, **Multivariable Adaptive Regression Splines** (MARS), **neural network** with 3 hidden nodes, **logistic regression** (GLM) Using the 20 strongest variables, **Naïve Bayers**, an ordinary **multiple regression,** and kNN algorithm. Display the performance of the classifiers in Spam Email Detection – gains Chart.