CS 6923: Machine Learning

Spring 2017

Homework 4

Submit on NYU Classes by Tues. Nov. 14 at 10:00 a.m. You may work together with one other person on this homework. If you do that, hand in JUST ONE homework for the two of you, with both of your names on it. You may *discuss* this homework with other students but YOU MAY NOT SHARE WRITTEN ANSWERS OR CODE WITH ANYONE BUT YOUR PARTNER.

IMPORTANT SUBMISSION INSTRUCTIONS: Please submit your solutions in 3 separate files: one file for your written answers to Part I, one file for your written answers/output for the questions in Part II, and one file with your code (a zip file if your code requires more than one file).

Part I: Written Exercises

1. Suppose logistic regression is run on a dataset $\mathcal{X} = \{(x^t, r^t)\}_{t=1}^N$, to train a hypothesis $h(x) = \frac{1}{1+e^{-(w^T x + w_0)}}$. Assume there are two classes, C_1 and C_0, and each r^t is either 1 or 0, designating whether x^t is in C_1 or C_0. As usual, $h(x)$ is an estimate of the probability that x is in Class C_1.

   Let $V(h)$ denote the error function we defined for logistic regression:

$$V(h) = \sum_t \left[-r^t \log y^t - (1 - r^t)\log(1 - y^t)\right]$$

   This is sometimes called the logistic loss or the cross-entropy loss. It is also sometimes called the log loss (see below).

   (a) For $j \in \{0, 1\}$, let $p_{t,j}$ be the estimate, according to h, of the probability that x^t is in Class C_j. So, $p_{t,1} = h(x^t)$ and $p_{t,0} = 1 - h(x^t)$. We can now rewrite $V(h)$ as follows:

$$V(h) = \left[\sum_{t:r^t=1} -\log p_{t,1}\right] + \left[\sum_{t:r^t=0} -\log p_{t,0}\right]$$

   In this expression, if j is the correct class for x^t, then h is being "charged" the value $-\log p_{t,j}$. This is why this error function is sometimes called the "log loss".

   Graph the function $-\log p$ as p goes from 0 to 1. What is the value of $-\log p$ at $p = 1$? At $p = 1/2$?

   (b) Consider the following table for a dataset $\mathcal{X} = \{(x^t, r^t)\}_{t=1}^N$, showing the given labels r^t of the examples, and the predictions y^t made by a hypothesis h:

t	r^t	y^t
1	1	0.90
2	1	0.83
3	0	0.01
4	0	0.04

   Calculate $V(h)$ for this dataset. Show your work.

2. Suppose $V(h)$ is an error function that measures the error of hypothesis h on the training data. If we are trying to avoid overfitting, we may not just want to find the h that minimizes $V(h)$. We may also want to try to find an h that is not too "complex".

   To accomplish this, we may define another function, $R(h)$, that measures the "complexity" of h. Then we look for a hypothesis that minimizes a new loss function,

$$V(h) + \lambda R(h)$$

   where λ is a constant (parameter) that we can choose. The term $\lambda R(h)$ is called a "penalty" term that penalizes hypotheses that are complex. If we set $\lambda = 0$, then we are just using the standard loss function. If we set λ to be a large number, then we are tolerating more errors on the training data in order to have a simpler hypothesis.

   The approach of adding a penalty like this is called "regularization".

   (a) A hypothesis that is overfitting in logistic regression will often have weights that have large absolute value. One common way of defining $R(h)$, for a logistic regression hypothesis h defined by parameters w, w_0 (where $w = (w_1, \ldots, w_d)$), is as follows:

$$R(h) = w_1^2 + \ldots + w_d^2$$

   (Note that this does not include w_0.)
   Using this $R(h)$, define a new regularized loss function for logistic regression:

$$L(h) = V(h) + \lambda R(h)$$

   where $V(h)$ is the usual logistic loss, and λ is a constant that we will choose later (e.g. $\lambda = 3$).
   The update rule for gradient descent for logistic regression, using the logistic loss function, is as follows:
   $w_j = w_j + \sum_t (r^t - y^t) x_j^t$ for $j = 1, \ldots, d$
   $w_0 = w_0 + \sum_t (r^t - y^t)$
   where $y = \frac{1}{e^{-(w^T x + w_0)}}$.
   What is the update rule for logistic regression if we instead use the loss function $L(h)$? Your answer should include the parameter λ.

   (b) In discussing logistic regression in class, we showed that the logistic loss function is equal to $-\log P(\mathcal{X}|h)$. Therefore, the hypothesis h that minimizes $V(h)$ is the ML hypothesis.
   Suppose $V(h)$ is the logistic loss function, and we define $R(h)$ slightly differently than we did above, by including w_0. Specifically, let's define

$$R_1(h) = w_0^2 + w_1^2 + \ldots + w_d^2$$

   Consider a prior distribution on h, where the weights $w_0, \ldots, w_d$ defining h are independent, and each $w_i \sim \mathcal{N}(0, 1)$. In this case, the MAP hypothesis h is the one that minimizes

$$V(h) + cR_1(h)$$

   for some particular value of c. Which value?

3. In this exercise you will get practice with performing gradient descent in one variable.

   Consider the function

$$f(x) = 2x^4 - 2x^3 - 12x^2 + 6$$

   Graph this function for x in the interval [-4,4]. You will see that it has two minima in that range, a local minimum and a global minimum. These are the only minima of the function.

   (a) What is the value of x at the local minimum and at the global minimum? Find the answer to this question like either by hand with calculus or using matlab or other software.

   (b) Suppose we apply gradient descent to this function, starting with $x = -4$. To do this, we will need to update x using the update rule

   $x = x + -\eta * f'(x)$

   where f' is the derivative of f, and η is the "step size".

   Write a small program implementating gradient descent for this function. Setting $x = -4$ and $\eta = 0.001$, run gradient descent for 5 iterations (that is, do the update 5 times). Report the values of x and $f(x)$ at the start and after each of the first 5 iterations.

   Run the gradient descent again, starting with x=-4, for 1000 iterations. Report the last 5 values of x and f(x).

   Has the value of x converged? Has the gradient descent found a minimum? Is it the global or the local minimum?

   You do NOT have to hand in your code.

   (c) Repeat the previous exercise, but this time, start with x=4.

   (d) Setting $x = -4$ and $\eta = 0.01$, run gradient descent for 1000 iterations. As in the previous two exercises, report the initial values of x and f(x), the next 5 values of x and f(x), and the last 5 values of x and f(x). Compare the results obtained this time to the results obtained above for $x = -4$ and $\eta = 0.001$. What happened?

   (e) Setting $x = -4$ and $\eta = 0.1$, run gradient descent for 100 iterations. What happened?

4. The entropy of a labeled dataset S is defined as follows:

$$Entropy(S) = -\sum_{l \in L} \frac{N_l}{N} \log_2 \frac{N_l}{N}$$

   where L is the set of class labels for the examples in S (e.g., + or -), N is the number of examples in S, and N_l is the number of examples in S that have label l. (In lecture, we discussed this definition for the case where there are 2 classes. The above definition is the generalization to k class problems, for $k \geq 2$).

   Let x_i be a discrete-valued attribute of the examples in dataset S, and let V be the set of possible values of attribute x_i. For $v \in V$, let S_v be the subset of examples in S satisfying $x_i = v$.

   Consider a classification problem with categorical attributes. Applied to such a problem, the decision tree algorithm discussed in class builds a decision tree where each node is labeled by an attribute x_i. When building the tree, at each node, the algorithm chooses the decision (i.e., the x_i) that minimizes

$$\sum_{v \in V} \frac{|S_v|}{|S|} Entropy(S_v)$$

This is equivalent to saying that it chooses the x_i that maximizes the following quantity, which is sometimes called the Information Gain of x_i (with respect to S)

$$\text{Information-Gain}(S) = \mathit{Entropy}(S) - \sum_{v \in V} \frac{|S_v|}{|S|} \mathit{Entropy}(S_v)$$

It can be shown that Information-Gain(S) is always greater than or equal to 0. Recall that we consider $0 \log_2 0$ to be equal to 0.

(a) Answer the following question without using a calculator. Which dataset has higher entropy: A dataset with 5 positive examples and 7 negative examples, or a dataset with 3 positive examples and 6 negative examples?

(b) Calculate the Information Gain of x_3 in the following dataset. Show your work.

	x_1	x_2	x_3	r
$x^{(1)}$	T	F	F	-
$x^{(2)}$	F	T	F	+
$x^{(3)}$	T	T	F	-
$x^{(4)}$	T	F	T	+
$x^{(5)}$	F	F	T	-
$x^{(6)}$	F	F	F	-

(c) What is the entropy of a labeled dataset S, with z possible labels, if each label appears equally often among the examples in the dataset?

(d) Information gain has many nice properties, and is a reasonable quantity to use when choosing attributes to place in decision tree nodes. However, it has some drawbacks.

When used with categorical attributes, it is said that Information Gain favors attributes with many possible values. In other words, an attribute with, e.g., 100 possible values will usually have a higher information gain than an attribute with only 2 possible values, even if it is not better in terms of helping to classify examples.

As an extreme case, consider two attributes used in classifying medical patients. One attribute has 2 values, male or female. Another attribute has 26^7 possible values, each associated with a unique 7-letter PIN assigned to a patient. For some reason, we are treating this as a categorical attribute (sometimes called a nominal attribute), rather than as a continuous attribute. Why is the second attribute likely to have higher gain than the first? Considering that our goal is to have a tree with good prediction accuracy, why wouldn't we want to place the second attribute into a node in our decision tree?

Part II: Programming Exercises

In this assignment we will work on a text categorization task. We will use the well-known 20-newsgroups dataset. You should download the dataset from the following url: `http://qwone.com/~jason/20Newsgroups/20news-bydate-matlab.tgz`

The examples in this dataset are short documents (postings) about different topics. Each posting is labeled by topic, and there are 20 topics. The documents have been preprocessed, and a vocabulary of size 61,188 was extracted. Each "term" in vocabulary is one attribute, so there are 61,188 attributes. The vocabulary is in the following file: `http://qwone.com/~jason/20Newsgroups/vocabulary.txt`

The attributes are numbered in the order given in this file, starting with 1 (so attribute 1 is "archive").

The training documents are represented in the file train.data. The representation has the following form: For each document (example) docIdx, there is a line in the file for every term appearing in the document. The format of the line is

```
docIdx termIdx count
```

where count is the number of occurences of termIdx in document docIdx. If a term does not appear in a document, then there will be no corresponding line in train.data. Therefore, the count in every line is always at least 1.

The labels of the training documents are in a separate file, train.label. Line j of train.label has the label of document number j. The labels are given as numbers. The mapping from label (newsgroup) names to label numbers is given in the file train.map.

The test data is similarly given in test.data, test.label, and test.map (test.map is the same as train.map).

More information about the dataset can be found here: `http://qwone.com/~jason/20Newsgroups/`

Note: You may want to store the training data (and testing data) in a matrix of dimension equal to (#documents) x (#terms). The entry in row docIdx and column termIdx should be equal to the number of occurences of termIdx in docIdx. Most of the entries in this matrix will be 0. Because this matrix is very large, you may want to store it in sprase format (available in both MATLAB and Python). A sparse matrix is a compact way of representing a matrix that has a lot of entries equal to 0. The training (and test file) is in a format that makes the data easy to load into a sparse matrix.

5. Here we consider a binary classification problem for the above dataset, where we need to classify each article as either belonging to a Science (sci) newsgroup, or as not belonging to a science newsgroup. In effect, we are treating classes 12, 13, 14, and 15 (sci.crypt, sci.electronics, sci.med, sci.space) as a single class, and all the classes as another single class. Let Sci be the positive class, and the set of other newsgroups be the negative class.

   Hand in your code for the following questions. Include a README file and/or python notebook if it is not obvious how you compiled and ran your code.

   (a) In this step, we will do feature selection using Information Gain to choose 100 of the features. Using just the training data, "score" the features by computing their Information Gain (relative to the positive and negative classes specified above). Find the 100 features with the highest Information Gain. Output the top 5 features, together with their Information Gain.

   (b) Using only the 100 features with the highest information gain, run logistic regression *with no regularization* on the training data to train a logistic classifier. Implement logistic regression using gradient descent, using the update rules from class (and the textbook).

   You will probably have to experiment with different learning rates η until you get one that works well. You do NOT have to show the experimentation, your code can just use the learning rate that worked best.

   Output the value of η.

   (c) Test the logistic classifier that was learned using the training data, by using it to classify the examples in the test set. In doing testing, classify an example x as positive if the logistic classifer estimates that $P[+|x] \geq 1/2$.

   Output the resulting confusion matrix and report the accuracy attained.

(d) In text categorization, it is common to report other statistics that can be derived from the confusion matrix: Recall and Precision.

Recall is another name for the true positive rate. Precision is equal to TP/(TP + FP), where TP is the number of true positive examples (correct label = predicted label = +), and FP is the number of false positive examples (correct label = - and predicted label = +).

Output the recall and precision from the confusion matrix you just computed.

(e) The reason for reporting Recall and Precision can be explained as follows. Suppose we are only interested in reading documents in the + class (in our problem, the science documents). A classifier is used to label the documents + or -, and only those labeled + are given to us to read.

Recall measures the percentage of documents actually in the positive class which are given to us to read. *Precision* measures the percentage of documents that we are given to read that would actually interest us (i.e., that are actually in the positive class). If all examples are predicted to be +, then recall is 100% but precision is probably very low. If we're very conservative about predicting the + label, and only predict + for a few documents, then we may get 100% precision, but our accuracy will be low. So while we would like both precision and recall to be 100%, there is generally a tradeoff.

When you use a logistic classifier, one way to vary precision and recall is to change the threshold for predicting +. That is, instead of predicting + if the classifier estimates $P[+|x] \geq 1/2$, you can predict + if the classifer estimates $P[+|x] \geq \gamma$, for some other fraction γ.

Suppose we want to have at least 97% recall. Compute the maximum value of γ that would achieve that recall. Using that value of γ, compute the precision. (This is the best precision that could be obtained with at least 97% recall.) What is the value of γ and what precision was obtained?

(f) Consider the weight vector that was learned by your logistic classifier. Output the 3 attributes with the highest magnitude weights. (The magnitude of a weight is its absolute value.)

For each of these attributes, also output the associated weight.

Are any of the weights negative? Intuitively, does it make sense that these are the words with the highest magnitude weights?

(g) There are many things you could do in order to try and achieve higher accuracy (and better precision/recall pairs) for this problem. For example, you could change the feature selection method, or the number of features. You could use regularization in the logistic classifier.

You could also use other representations of the documents, as is often done in text categorization. Instead of using the raw counts as the attribute values, you could use TF-IDF values for the features and/or normalize each document vector so that the vector has length 1. [For more information about this and other techniques used in text categorization, see, for example, the following article: `http://www.scholarpedia.org/article/Text_categorization`.]

Try one of two of these approaches, or you may use approaches of your own choosing. Output the resulting recall, precision, and accuracy for $\gamma = 0.5$. Also output the best precision that could be obtained with at least 97% recall.

Also, describe what you did and why you thought it might work well. Did it work well? If not, why do you think it did not work well?