CS 5350/6350: Machine Learining Fall 2017

Homework 5

Handed out: Thursday November 16th, 2017
Due date: Thursday November 30th, 2017

General Instructions

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 10 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- The homework is due by midnight of the due date. Please submit the homework on Canvas.

1 Logistic Regression

We looked Maximum A Posteriori (MAP) learning of the logisitic regression classifier in class. In particular, we showed that learning the classifier is equivalent to the following optimization problem:

$$\min_{\mathbf{w}} \left\{ \sum_{i=1}^{m} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w} \right\}$$

In this question, you will derive the stochastic gradient descent algorithm for the logistic regression classifier.

1. [5 points] What is the derivative of the function $g(\mathbf{w}) = \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$ with respect to the weight vector?
2. [5 points] The inner most step in the SGD algorihtm is the gradient update where we use a single example instead of the entire dataset to compute the gradient. Write down the objective where the entire dataset is composed of a single example, say $(\mathbf{x}_i, y_i)$. Derive the gradient with respect to the weight vector.

3. [10 points] Write down the pseudo code for the stochastic gradient algorithm using the gradient from previous part.

   Hint: The answer to this question will be an algorithm that is similar to the SGD based learner we developed in the class for SVMs.

2 Experiments

For this question, you will have to implement and compare six different learning strategies: SVM, logistic regression (from your answer to the previous question), the naive Bayes classifier, bagging and two ensembles over decision trees.

2.1 The task and data

This homework explores the problem of plagiarism. That is, looking at overlapping words and phrases in two pieces of text, we seek to predict whether they were written by the same person.

In order to study this, we have constructed a dataset from political speeches over the last 200 years. The instances we want to classify are pairs of speeches and the label is $+1$ if they were delivered by the same person, and -1 otherwise. To help with your experiments, we have extracted word, bigram and trigram features from the text and converted the features into the familiar liblinear format that we used in the previous homeworks. The data directory (provided as the compressed file `data.zip`) contains the following files:

1. `speeches.train.liblinear`: The full training set, with 2818 examples.
2. `speeches.test.liblinear`: The test set, with 940 examples.
3. To help with cross-validation, we have split the training set into five parts `training00.data` - `training04.data` in the folder `CVSplits`.

2.2 Implementation Notes

Each algorithm has different hyper-parameters, as described below. Use 5-fold cross-validation to identify the best hyper-parameters as you did in the previous homework.

Be careful: the dimensionality of this dataset is very large, you may want to use a different strategy to represent your feature vectors. One approach is to store inputs as sparse vectors rather than dense vectors by only storing non-zero elements – for example, instead of storing the vector $[1.1, 2, 0, 0, 1.3, 0, 0, 0]$ as an array, we can keep the non-zero entries in memory as a map from from indexes to values, namely $\{0 : 1.1, 1 : 2, 4 : 1.3\}$.

2.3 Algorithms to Compare

1. [15 points] **Support Vector Machine**

   Implement simple SGD version of SVM as described in the class. You need to use sub-gradient instead of the gradient.

Hyper-parameters:

(a) Initial learning rate: $\gamma_0 \in \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$

(b) The regularization/loss tradeoff parameter: $C \in \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$

2. [15 points] **Logistic regression**

Implement the Logistic Regression learner based on your algorithm in the Question 1.

Hyper-parameters:

(a) Initial learning rate: $\gamma_0 \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$

(b) Tradeoff: $\sigma^2 \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4\}$

3. [15 points] **Naive Bayes**

Implement the simple Naive Bayes learner. You need to count the features to get the likelihoods one at a time. To get the prior, you will need to count the number of examples in each class.

For every feature x_i, you should estimate its likelihood of taking a value for a given label y (which is either $+$ or $-$) as:

$$P(x_i|y) = \frac{Count(x_i, y) + \lambda}{Count(y) + S_i\lambda}.$$

Here, S_i is the number of all possible values that x_i can take in the data. (In the data provided, each feature is binary, which should simplify your implementation a lot.)

The hyper-parameter λ is a smoothing term. In example we saw in class, we set $\lambda = 1$. But, in this experiment, you should choose the best λ based on cross-validation.

Hyper-parameter: Smoothing term: $\lambda \in \{2, 1.5, 1.0, 0.5\}$

4. [15 points] **Bagged Forests**

In class we have learned how the bagging algorithms work. In this setting, you are going to build a bagging algorithm based on depth-limited decision trees learned using the ID3 algorithm.

Given the dataset, you need to build 1000 decision trees that are limited to depth = 3. For each decision tree, you need to sample 100 examples with replacement from training set and use this subset to train your decision tree. After you get 1000 trees, for a new example, the prediction will be label that gets the larger vote the vote among these trees.

5. [10 points] **SVM over trees**

Use the 1000 decision trees in the last question to predict the label for each example in the training and test set. Then for each example, there will be 1000 predictions. Instead of simple voting over the predictions of these trees, we would like to use SVM to combine these predictions for this question. Specifically, after growing the 1000 trees, you should construct a new dataset consisting of transformed features. The features transformation $\phi(\mathbf{x})$ is defined using 1000 trees as follows:

$$\phi(x) = [\text{tree}_1(x), \text{tree}_2(x), \ldots, \text{tree}_{1000}(x)]$$

In other words, you will build an 1000 dimensional vector consisting of the prediction (1 or -1) of each tree that you created. Thus, you have a learned feature transformation. Now, train an SVM (using your learner from the first part of this question) on this transformed data.

Hyper-parameters:

(a) Learning rate $\gamma \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$

(b) Tradeoff $C \in \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$

(c) Depth: $d \in \{50, 100, 150, 200\}$

6. [10 points] **Logistic regression over trees**

   Use the logistic regression instead of SVM on the same dataset generated in last question.

   Hyper-parameters:

   (a) Initial learning rate: $\gamma_0 \in \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$

   (b) Tradeoff: $\sigma^2 \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4\}$

   (c) Depth: $d \in \{50, 100, 150, 200\}$

7. [10 points] **Extra Credit**

   For the dataset in this assignment training decision trees with depth 200 using ID3 takes a very very long time. Explain **why** it takes so long in 3 sentences or fewer. Be precise; the extra-credit is all-or-nothing.

2.4 What to report

1. For each algorithm above, briefly describe the design decisions that you have made in your implementation. (E.g, what programming language, how do you represent the vectors, trees, etc.)

2. Report the best hyper-parameters and accuracy on training set and test set. Please fill Table 1.

Experiment Submission Guidelines

1. The report should detail your experiments. For each step, explain in no more than a paragraph or so how your implementation works.

	Best hyper-parameters	Average cross-validation accuracy	Training accuracy	Test Accuracy
SVM				
Logistic regression				
Naive Bayes				
Bagged Forests				
SVM over trees				
Logistic regression over trees				

Table 1: Result table

2. *Your code should run on the CADE machines.* You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.

   You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.

3. Please do not hand in binary files! We will *not* grade binary submissions.