CS 534: Machine Learning
Homework 4

(Due Nov 14th at 11:59 PM on Gradescope)

Submission Instructions:

The homework must be submitted electronically on Gradescope as a single submission. Each question must be tagged appropriately (you need to set which page the problem is on), otherwise you may lose credit. The answer must also be explicit, if you are exporting an iPython notebook, you will need to add remarks to denote your final answer for the question.

The code can be done in Python, R, or MATLAB. The use of any other programming language must be confirmed with Huan (TA). Source code must be submitted separately on Canvas (the deadline is 15 minutes after the Gradescope submission) and zipped into a single file. There is no need to attach the data, and you can assume the data will be in the same directory as the code. The source code should be executable and there should be no explicit path names for the data files (`joyce/CS534/HW4/train.csv` is an example of hard-coding).

1. **(5 + 5 = 10 pts) Kernel Methods**

   (a) Assuming that $\mathbf{x} = [x_1, x_2], \mathbf{z} = [z_1, z_2]$ (i.e., both vectors are two-dimensional) and $\beta > 0$, show that the following is a kernel:

   $$k_\beta(\mathbf{x}, \mathbf{z}) = (1 + \beta \mathbf{x} \cdot \mathbf{z})^2 - 1$$

   Do so by demonstrating a feature mapping $\phi(\mathbf{x})$ such that $k_\beta(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{z})$.

   (b) One way to construct kernels is to build them from simpler ones. Assuming $k_1(\mathbf{x}, \mathbf{z})$ and $k_2(\mathbf{x}, \mathbf{z})$ are kernels, then one can show that so are these:

   i. (scaling) $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) f(\mathbf{z}) k_1(\mathbf{x}, \mathbf{z})$ for any function $f(\mathbf{x}) \in \mathbb{R}$

   ii. (sum) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$

   iii. (product) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})\ k_2(\mathbf{x}, \mathbf{z})$

   Using the above rules and the fact that $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$ show that the following is also a kernel:

   $$\left(1 + (\frac{\mathbf{x}}{||\mathbf{x}||_2})^\top (\frac{\mathbf{z}}{||\mathbf{z}||_2})\right)^3$$

2. **(15 + 7 + 4 + 4 = 30 pts) Pseudo-Random Forest to Predict Loan Defaults**

   We will be using the Loan Defaults dataset from Homework #3 `loan_default.csv`. You will implement an adaptation of the random forest to detect loan defaults. Instead of randomly sampling features for each node of each tree in your forest, you will choose a random subspace (i.e., limit the attributes to consider) that the tree will be created on. This allows you to use existing decision trees without having to build your own[1].

   (a) Build the adaptation of the random forest using your favorite language. You will want to make sure your forest supports the following parameters:

   - nest: the number of trees in the forest

[1]Pick your favorite splitting criteria and stick with it. Empirically, splitting criteria has a much smaller effect than the other parameters.

- maxDepth: the maximum depth of each tree
- minSamplesLeaf: the minimum number of samples per leaf node
- q: the number of features for each tree

In addition to predicting the output, your forest should be able to provide the out-of-bag (OOB) sample accuracy, and feature importance.

(b) For the number of trees (nest) in the range of 5-25, find the best parameters for the max depth, minimum sample leaf, and the number of features to consider for each tree. Justify your selection with supporting evidence (i.e., plots).

(c) Using your optimal parameters, how well does your version of the random forest perform on the test data? How does this compare to the OOB sample accuracy?

(d) What are the most important features from your model?

3. **(2 + 3 + 15 + 5 + 10 + 3 + 2 = 40 pts) Spam Detection via Perceptron**

We have provided a new e-mail spam dataset `spamAssassin.data`, which is a subset of the SpamAssassin Public Corpus (see `https://spamassassin.apache.org/old/publiccorpus/`). Here is a sample email that contains a URL, an email address (at the end), numbers and dollar amounts.

```
> Anyone knows how much it costs to host a web portal ?
> Well, it depends on how many visitors youre expecting. This can be anywhere
from less than 10 bucks a month to a couple of $100. You should checkout
http://www.rackspace.com/ or perhaps Amazon EC2 if youre running something big...

To unsubscribe yourself from this mailing list,
send an email to: groupnameunsubscribe@egroups.com
```

We have already implemented the following email preprocessing steps: lower-casing; removal of HTML tags; normalization of URLs, e-mail addresses, and numbers. In addition, words have been reduced to their stemmed form. For example, "discount", "discounts", "discounted" and "discounting" are all replaced with "discount". Finally, we removed all non-words and punctuation. The result of these preprocessing steps on the same email is shown below:

```
anyon know how much it cost to host a web portal well it depend on how mani visitor
your expect thi can be anywher from less than number buck a month to a coupl of
dollarnumb you should checkout httpaddr or perhap amazon ecnumb if your run someth
big to unsubscrib yourself from thi mail list send an email to emailaddr
```

For this problem, you will implement the Perceptron algorithm and apply it to the problem of e-mail spam classification. You'll be comparing two different variants of the algorithm as well as the number of epochs through the data.

(a) What is your model assessment strategy? Justify your validation methodology. (You may want to read the rest of this problem before you proceed to understand what your tasks will be).

(b) Build a vocabulary list using only the e-mail training set by finding all words that occur across the training set. Ignore all words that appear in fewer than 30 e-mails of the e-mail training set – this is both a means of preventing overfitting and of improving scalability. For each email, transform it into a feature vector $\mathbf{x}$ where the ith entry, x_i, is 1 if the ith word in the vocabulary occurs in the email, and 0 otherwise.

(c) Implement the perceptron algorithm. You'll want at least two functions:

- Train the model that uses the examples provided to the function that returns the final classification vector, $\mathbf{w}$, with the number of updates (mistakes) performed, and the number of passes (epochs) through the data. The algorithm will terminate either when all the points are correctly classified, or the max number of epochs have been reached. For the corner case of $\mathbf{w} \cdot \mathbf{x} = 0$, predict the $+1$ class.
- Test new data given the classification vector $\mathbf{w}$, and new data points. The function should return the test error.

(d) Train the perceptron using your training set. Plot the training and estimated generalization error as a function of the maximum number of epochs. What is the optimal algorithm and parameter? How many mistakes are made before the algorithm terminates if you wanted to classify all the training points properly?

(e) Implement the averaged perceptron algorithm, which is the same as your current implementation but which, rather than returning the final weight vector, returns the average of all weight vectors considered during the algorithm (including examples where no mistake was made). Averaging reduces the variance between the different vectors, and is a powerful means of preventing the learning algorithm from overfitting (serving as a type of regularization). Plot the training and estimated generalization error as a function of the maximum number of epochs for the averaged perceptron.

(f) What is your final or "optimal" algorithm? In other words, train the model with as much data as you can possibly with the optimal algorithm + hyperparameter (maximum number of epochs) values. What is the expected predictive performance of this model?

(g) Using the vocabulary list together with the parameters learned in the previous question, output the 15 words with the most positive weights. What are they? Which 15 words have the most negative weights?

4. **(2+3+4+3+4+4= 20 points) Loan Default Prediction with SVM**

We will be using the loan dataset and evaluating the model on AUC, F_1 score, and F_2 score. Note that SVM with polynomial and RBF kernels can be quite expensive – you might want to use a beefier machine to do this.

(a) Why would you potentially be interested in optimizing for F_2 score compared to F_1 score in the context of finding individuals that are likely to default?

(b) Train a linear SVM. How did you find the optimal parameter (C)?

(c) Train a polynomial kernel where the parameter values are chosen using Grid Search cross-validation (hint: think about using `GridSearchCV` in sklearn).

(d) Train a polynomial kernel where the parameter values are chosen using Random Search cross-validation (hint: think about using `RandomizedSearchCV` in sklearn) with 1/4 of the total number of parameter settings of the previous part. For example, if GridSearch tries 12 different settings in total, RandomizedSearch would try 3. How does the parameters from GridSearch and RandomizedSearch perform on the test dataset? Comment on the trade-off between computational run-time and the predictive performance of the model between the two cross-validation techniques.

(e) Build an RBF-kernel SVM. How did you choose the optimal parameter(s) for your model?

(f) Report the evaluation metrics for the training and test set for all of the above. How do they (the 4 SVM models) compare in terms of the 3 different metrics? How does it compare against the decision tree, bootstrap aggregating of linear models (problem 3 of homework 3), and the pseudo random forest from problem 2 in terms of computation time, predictive accuracy, and ease of interpretation of the resulting model?