

CptS 570 Machine Learning, Fall 2018

Homework #3

Due Date: Tue, Nov 12 midnight via Blackboard

NOTE 1: Please use a word processing software (e.g., Microsoft word or Latex) to write your answers and submit a printed copy to me at the beginning of class on Oct 23. The rationale is that it is sometimes hard to read and understand the hand-written answers.

NOTE 2: Please ensure that all the graphs are appropriately labeled (x-axis, y-axis, and each curve). The caption or heading of each graph should be informative and self-contained.

Analytical Part (3 Percent)

1. Suppose $x = (x_1, x_2, \dots, x_d)$ and $z = (z_1, z_2, \dots, z_d)$ be any two points in a high-dimensional space (i.e., d is very large).
 - a. Try to prove the following, where the right-hand side quantity represent the standard Euclidean distance.

$$\left(\frac{1}{\sqrt{d}} \sum_{i=1}^d x_i - \frac{1}{\sqrt{d}} \sum_{i=1}^d z_i \right)^2 \leq \sum_{i=1}^d (x_i - z_i)^2 \quad (1)$$

Hint: Use Jensen's inequality – If X is a random variable and f is a convex function, then $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$.

- b. We know that the computation of nearest neighbors is very expensive in the high-dimensional space. Discuss how we can make use of the above property to make the nearest neighbors computation efficient?
2. We briefly discussed in the class about Locality Sensitive Hashing (LSH) algorithm to make the nearest neighbor classifier efficient. Please read the following paper and briefly summarize the key ideas as you understood:
Alexandr Andoni, Piotr Indyk: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of ACM 51(1): 117-122 (2008) <http://people.csail.mit.edu/indyk/p117-andoni.pdf>
 3. We know that we can convert any decision tree into a set of if-then rules, where there is one rule per leaf node. Suppose you are given a set of rules $R = \{r_1, r_2, \dots, r_k\}$, where r_i corresponds to the i^{th} rule. Is it possible to convert the rule set R into an equivalent decision tree? Explain your construction or give a counterexample.
 4. Please read the following paper and briefly summarize the key ideas as you understood (You can skip the proofs, but it is important to understand the main results):

Andrew Y. Ng, Michael I. Jordan: On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. NIPS 2001: 841-848 <http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf>

5. Naive Bayes vs. Logistic Regression

- Let us assume that the training data satisfies the Naive Bayes assumption (i.e., features are independent given the class label). As the training data approaches infinity, which classifier will produce better results, Naive Bayes or Logistic Regression? Please explain your reasoning.
- Let us assume that the training data does **NOT** satisfy the Naive Bayes assumption. As the training data approaches infinity, which classifier will produce better results, Naive Bayes or Logistic Regression? Please explain your reasoning.
- Can we compute $P(X)$ from the learned parameters of a Naive Bayes classifier? Please explain your reasoning.
- Can we compute $P(X)$ from the learned parameters of a Logistic Regression classifier? Please explain your reasoning.

6. Please read the following paper and briefly summarize the key ideas as you understood:

Thomas G. Dietterich (1995) Overfitting and under-computing in machine learning. *Computing Surveys*, 27(3), 326-327.

<http://www.cs.orst.edu/~tgdp/publications/cs95.ps.gz>

7. We need to perform statistical tests to compare the performance of two learning algorithms on a given learning task. Please read the following paper and briefly summarize the key ideas as you understood:

Thomas G. Dietterich: Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms. *Neural Computation* 10(7): 1895-1923 (1998) <http://sci2s.ugr.es/keel/pdf/algorithm/articulo/dietterich1998.pdf>

Programming and Empirical Analysis (5 Percent)

1. (40 points) Fortune Cookie Classifier¹

You will build a binary fortune cookie classifier. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future (class 1) and messages that just contain a wise saying (class 0). For example,

“Never go in against a Sicilian when death is on the line” would be a message in class 0.

“You will get an A in Machine learning class” would be a message in class 1.

Files Provided There are three sets of files. All words in these files are lower case and punctuation has been removed.

1) The training data:

traindata.txt: This is the training data consisting of fortune cookie messages.

trainlabels.txt: This file contains the class labels for the training data.

2) The testing data:

testdata.txt: This is the testing data consisting of fortune cookie messages.

testlabels.txt: This file contains the class labels for the testing data.

3) A list of stopwords: stoplist.txt

There are two steps: the pre-processing step and the classification step. In the pre-processing step, you will convert fortune cookie messages into features to be used by your classifier. You will be using a bag of words representation. The following steps outline the process involved:

¹Thanks to Weng-Keen Wong and his advisor Andrew Moore for sharing the data.

Form the vocabulary. The vocabulary consists of the set of all the words that are in the training data with stop words removed (stop words are common, uninformative words such as “a” and “the” that are listed in the file `stoplist.txt`). The vocabulary will now be the features of your training data. Keep the vocabulary in alphabetical order to help you with debugging.

Now, convert the training data into a set of features. Let M be the size of your vocabulary. For each fortune cookie message, you will convert it into a feature vector of size M . Each slot in that feature vector takes the value of 0 or 1. For these M slots, if the i th slot is 1, it means that the i th word in the vocabulary is present in the fortune cookie message; otherwise, if it is 0, then the i th word is not present in the message. Most of these feature vector slots will be 0. Since you are keeping the vocabulary in alphabetical order, the first feature will be the first word alphabetically in the vocabulary.

Implement the Naive Bayes Classifier (with Laplace Smoothing) and run it on the training data. Compute the training and testing accuracy.

To debug and test your implementation, you can employ Weka (`weka.classifiers.bayes.NaiveBayes`): <http://www.cs.waikato.ac.nz/ml/weka/downloading.html> or scikit-learn (http://scikit-learn.org/stable/modules/naive_bayes.html)

2. **(60 points)** Convolutional Neural Networks (CNNs) for solving image classification task. You will train a CNN on Fashion MNIST data. The network architecture contains 4 CNN layers followed by one pooling layer and a final fully connected layer. The basic architecture (in sequential order) will be as follows:

First CNN layer: input channels - 1, output channels - 8, kernel size = 5, padding = 2, stride = 2 followed by ReLU operation

Second CNN layer: input channels - 8, output channels - 16, kernel size = 3, padding = 1, stride = 2 followed by ReLU operation

Third CNN layer: input channels - 16, output channels - 32, kernel size = 3, padding = 1, stride = 2 followed by ReLU operation

Fourth CNN layer: input channels - 32, output channels - 32, kernel size = 3, padding = 1, stride = 2 followed by ReLU operation

one “Average” pooling layer (`nn.AdaptiveAvgPool2d(1)` would work in PyTorch)

Fully connected layer (`nn.Linear` in PyTorch) - determine the number of input features from previous CNN layers. This can be done easily by hand. The number of output features will be equal to number of classes, i.e., 10. If you want help, you can use the direct formula given on this page: <http://cs231n.github.io/convolutional-networks/>.

This will be a straightforward extension from the code discussed in the demo session. Plot the training and testing accuracy as a function of at least 10 epochs. You could use a smaller sized dataset if compute power is a hurdle. A good choice would be 50 percent of the training set and 10 percent of the testing set. Please make sure you have equal ratio of all classes in the dataset. You can try all tips mentioned in the demo session for solving this task. Optionally, it will be a good idea to try adding other training techniques to see the maximum accuracy possible. Some of them include batch normalization, data augmentation, using other optimizers like ADAM etc.

Instructions for Code Submission and Output Format.

Please follow the below instructions. It will help us in grading your programming part of the homework. Please submit the zip file on Blackboard.

- Supported programming languages: Python, Java, C++
- Store all the relevant files in a folder and submit the corresponding zipfile named after your student-id, e.g., 114513209.zip
- This folder should have a script file named

```
run_code.sh
```

Executing this script should do all the necessary steps required for executing the code including compiling, linking, and execution

- Assume relative file paths in your code. Some examples:

```
‘./filename.txt’ or ‘../hw2/filename.txt’
```

- The output of your program should be dumped in a file named “output.txt”
- Make sure the output.txt file is dumped when you execute the script

```
run_code.sh
```

- Zip the entire folder and submit it as

```
<student_id>.zip
```

Grading Rubric

Each question in the students work will be assigned a letter grade of either A,B,C,D, or F by the Instructor and TAs. This five-point (discrete) scale is described as follows:

- **A) Exemplary (=100%).**
Solution presented solves the problem stated correctly and meets all requirements of the problem.
Solution is clearly presented.
Assumptions made are reasonable and are explicitly stated in the solution.
Solution represents an elegant and effective way to solve the problem and is not overly complicated than is necessary.
- **B) Capable (=75%).**
Solution is mostly correct, satisfying most of the above criteria under the exemplary category, but contains some minor pitfalls, errors/flaws or limitations.
- **C) Needs Improvement (=50%).**
Solution demonstrates a viable approach toward solving the problem but contains some major pitfalls, errors/flaws or limitations.

- **D) Unsatisfactory (=25%)**

Critical elements of the solution are missing or significantly flawed.

Solution does not demonstrate sufficient understanding of the problem and/or any reasonable directions to solve the problem.

- **F) Not attempted (=0%)**

No solution provided.

The points on a given homework question will be equal to the percentage assigned (given by the letter grades shown above) multiplied by the maximum number of possible points worth for that question. For example, if a question is worth 6 points and the answer is awarded a *B* grade, then that implies 4.5 points out of 6.