CptS 570 Machine Learning, Fall 2021
Homework #2
Due Date: Oct 17. Deadlines are midnight PST.

NOTE 1: Please use a word processing software (e.g., Microsoft word or Latex) to write your answers. The rationale is that it is sometimes hard to read and understand the hand-written answers. Thanks for your understanding.

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.

1 Analytical Part (3 percent grade)

This part will be graded as a PASS or FAIL.

1. Suppose we have n_+ positive training examples and n_- negative training examples. Let C_+ be the center of the positive examples and C_- be the center of the negative examples, i.e., $C_+ = \frac{1}{n_+}\sum_{i:\ y_i=+1} x_i$ and $C_- = \frac{1}{n_-}\sum_{i:\ y_i=-1} x_i$. Consider a simple classifier called CLOSE that classifies a test example x by assigning it to the class whose center is closest.

   - Show that the decision boundary of the CLOSE classifier is a linear hyperplane of the form $sign(w \cdot x + b)$. Compute the values of w and b in terms of C_+ and C_-.
   - Recall that the weight vector can be written as a linear combination of all the training examples: $w = \sum_{i=1}^{n_+ + n_-} \alpha_i \cdot y_i \cdot x_i$. Compute the dual weights (α's). How many of the training examples are support vectors?

2. Suppose we use the following radial basis function (RBF) kernel: $K(x_i, x_j) = exp(-\frac{1}{2}\left\|x_i - x_j\right\|^2)$, which has some implicit unknown mapping $\phi(x)$.

   - Prove that the mapping $\phi(x)$ corresponding to RBF kernel has infinite dimensions.
   - Prove that for any two input examples x_i and x_j, the squared Euclidean distance of their corresponding points in the higher-dimensional space defined by ϕ is less than 2, i.e., $\left\|\phi(x_i) - \phi(x_j)\right\|^2 \leq 2$.

3. The decision boundary of a SVM with a kernel function (via implicit feature mapping $\phi(.)$) is defined as follows:
   $w \cdot \phi(x) + b = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b = f(x; \alpha, b)$
   , where w and b are parameters of the decision boundary in the feature space phi defined by the kernel function K, SV is the set of support vectors, and α_i is the dual weight of the i^{th} support vector.

   Let us assume that we use the radial basis function (RBF) kernel $K(x_i, x_j) = exp(-\frac{1}{2}\left\|x_i - x_j\right\|^2)$; also assume that the training examples are linearly separable in the feature space ϕ and SVM finds a decision boundary that perfectly separates the training examples.

   If we choose a testing example x_{far} that is far away from any training instance x_i (distance here is measured in the original feature space $\Re^d$). Prove that $f(x_{far}; \alpha, b) \approx b$.

4. The function $K(x_i, x_j) = -\langle x_i, x_j\rangle$ is a valid kernel. Prove or Disprove it.

5. You are provided with n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). The teacher gave you some additional information by specifying the costs for different mistakes C_+ and C_-, where C_+ and C_- stand for the cost of misclassifying a positive and negative example respectively.
   a. How will you modify the Soft-margin SVM formulation to be able to leverage this extra information? Please justify your answer.

6. You are provided with a set of n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). Suppose n is very large (say in the order of millions). In this case, standard SVM training algorithms will not scale due to large training set.
   Tom wants to devise a solution based on "Coarse-to-Fine" framework of problem solving. The basic idea is to cluster the training data; train a SVM classifier based on the clusters (coarse problem); refine the clusters as needed (fine problem); perform training on the finer problem; and repeat until convergence. Suppose we start with k_+ positive clusters and k_- negative clusters to begin with (a cluster is defined as a set of examples). Please specify the *mathematical formulation* (define all the variables used in your formulation) and concrete algorithm for each of the following steps to instantiate this idea:
   a) How to define the SVM training formulation for a given level of coarseness: a set of k_+ positive clusters and a set of k_- negative clusters?
   b) How to refine the clusters based on the resulting SVM classifier?
   c) What is the stopping criteria?
   Optional question: For what kind of problems will this solution fail?

7. You are provided with a set of n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n,)$, where x_i is the input example, y_i is the class label (+1 or -1). Suppose n is very large (say in the order of millions). In this case, online kernelized Perceptron algorithms will not scale if the number of allowed support vectors are *unbounded*.
   a) Suppose you have trained using kernelized Perceptron algorithm (without any bounds on support vectors) and got a set of support vectors SV. Tom wants to use this classifier for real-time prediction and cannot afford more than B kernel evaluations for each classification decision. Please give an algorithm to select B support vectors from SV. You need to motivate your design choices in order to convince Tom to use your solution.
   b) Tom wants to train using kernelized Perceptron algorithm, but wants to use at most B support vectors during the training process. Please modify the standard kernelized Perceptron training algorithm (from class slides) for this new setting. You need to motivate your design choices in order to convince Tom to use your solution.

2 Programming and Empirical Analysis Part (5 percent grade)

1. Empirical analysis question. You can use a publicly available SVM classifier implementation (e.g., scikit-learn) for SVM related experiments. scikit-learn (`http://scikit-learn.org/stable/modules/svm.html`).

   You will use the Fashion MNIST data (`https://github.com/zalandoresearch/fashion-mnist`). There is a fixed training and testing set. From training data, use first 80 percent for "training" and last 20 percent as "validation data".

   Each example is a 28x28 grayscale image, associated with a label from 10 classes: 0 T-shirt/top, 1 Trouser, 2 Pullover, 3 Dress, 4 Coat, 5 Sandal, 6 Shirt, 7 Sneaker, 8 Bag, 9 Ankle boot.

   You will use ONLY training data for training and testing data for evaluation.

   (a) Using a linear kernel, train the SVM on the training data for different values of C parameter: $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}$. Compute the training accuracy, validation accuracy, and testing accuracy for the SVM obtained with different values of the C parameter. Plot the training accuracy, validation accuracy, and testing accuracy as a function of C (C value on x-axis and Accuracy on y-axis) – one curve each for training, validation, and testing data. Also, plot the number of support vectors (if applicable for the SVM toolkit you are using) as a function of C. List your observations.

   (b) Select the best value of hyper-parameter C based on the accuracy on validation set and train a linear SVM on the *combined* set of training and validation examples. Compute the testing accuracy and the corresponding confusion matrix: a 10×10 matrix.

   (c) Repeat the experiment (a) with the best C value from (a) with polynomial kernel of degree 2, 3, and 4. Compare the training, validation, testing accuracies, and the number of support vectors for differnt kernels (linear, polynomial kernel of degree 2, polynomial kernel of degree 3, and polynomial kernel of degree 4). List your observations.

2. Programming question. You will implement the kernelized Perceptron training algorithm (discussed in the class) for *multi-class* classification.

   (a) You will use the Fashion MNIST data. Train the kernelized Perceptron classifier for 5 iterations with polynomial kernel (pick the best degree out of 2, 3, and 4 from the above experiment). Plot the number of mistakes as a function of training iterations. Compute the training, validation, and testing accuracy at the end of 5 iterations.

3. Programming question. "Breast Cancer" Classifier using Decision Trees. You will use the following dataset for this question: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`. You will use the first 70 percent examples for training, next 10 percent examples for validation, and last 20 percent examples for testing.

   (a) Implement the ID3 decision tree learning algorithm that we discussed in the class. The key step in the decision tree learning is choosing the next feature to split on. Implement the information gain heuristic for selecting the next feature. Please see lecture notes or `https://en.wikipedia.org/wiki/ID3_algorithm` for more details. Do the following to select candidate thresholds for continuous features: Sort all candidate values for feature f from training data. Suppose $f_1, f_2, \cdots, f_n$ is the sorted list. The candidate thresholds are chosen as $f_i + (f_{i+1} - f_i)/2$ for i=1 to n.

   (b) Run the decision tree construction algorithm on the training examples. Compute the accuracy on validation examples and testing examples.

(c) Implement the decision tree pruning algorithm discussed in the class (via validation data).

(d) Run the pruning algorithm on the decision tree constructed using training examples. Compute the accuracy on validation examples and testing examples. List your observations by comparing the performance of decision tree with and without pruning.

To debug and test your implementation, you can employ scikit-learn (`http://scikit-learn.org/stable/modules/tree.html`).

3 Instructions for Submission

Please follow the below instructions. If you do not follow them, your homework will not be graded. All submissions will be handled through Canvas.

PDF submission. One PDF file with both answers for analytical part (Part I) and empirical analysis questions with results/analysis (Part-II). Please label x-axis, y-axis, and name of the graphs appropriately. Please name this file as WWSUID-LASTNAME.pdf (e.g., 111222-Fern.pdf).

Code submission. You will submit one zip file for your code as per the instructions below. If your script and/or code does not execute, we will try to give some partial credit by looking at the overall code contents.

- Mention the programming language and version (e.g., Python 2.5) that you used.
- Submit one folder with name WSUID-LASTNAME.zip (e.g., 111222-Fern.zip) and include a README file.
- Include a script to run the code and it should be referred in the README file. Please make sure that your script runs on a standard linux machine.
- Don't submit the data folder. Assume there is a folder "data" with all the files.
- Output of your programs should be well-formatted in order to answer the empirical analysis questions.
- Structure your code meaningfully and add comments to make it readable.

If you have collaborated or discussed the homework with some student, please provide this information with all the relevant details. If we find that the code of two different students has traces of plagiarism, both students will be penalized and we will report the academic dishonesty case to graduate school (see https://communitystandards.wsu.edu/policies-and-reporting/academic-integrity-policy/). The bottom line is please DO NOT even think of going this route. It is very unpleasant to deal with these things for both faculty, TA, and students involved.

4 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.