Assignment Four-Five
ECE 4950

- Provide credit to **any sources** other than the course staff that helped you solve the problems. This includes **all students** you talked to regarding the problems.
- You can look up definitions/basics online (e.g., wikipedia, stack-exchange, etc)
- **The due date is 4/28/2017, 23.59.59 eastern time**.
- Submission rules are the same as previous assignments.

Problem 1 (10 points). Recall cross-validation error. The training set T is divided into two sets, and after training on one of them, the error on the remaining is the *cross-validation error.* For a T with n points, the leave-one-out error is the cross-validation error when the validation set has size **one**, and training is done with the remaining $n-1$ examples.

1. Design a two dimensional ($k = 2$) data set T with two classes (labels) and $n = 4$ such that:
   - the leave-one-out error of 1-nearest neighbor algorithm is zero, regardless of which data point is chosen for cross-validation, and
   - the perceptron algorithm, *when trained on entire T*, does not converge.
2. Design a two dimensional ($k = 2$) data set T with two classes (labels) and $n = 4$ such that:
   - the leave-one-out error of 1-nearest neighbor algorithm is not zero, regardless of which data point is chosen for cross-validation, and
   - the perceptron algorithm, *when trained on entire T*, converges.

Please show the training points on a two dimensional grid, clearly marking the classes as $\times$, and $\circ$.

Problem 2. (25 points). Recall that SVM's obtain *non-linear* decision boundaries by mapping the feature vectors $\bar{X}$ to a possibly high dimensional space via a function $\phi(\bar{X})$, and then finding a linear decision boundary in the new space.

We also saw that to implement SVM, it suffices to know the kernel function $K(\bar{X}, \bar{Y}) = \phi(\bar{X}) \cdot \phi(\bar{Y})$, without even explicitly specifying the function ϕ. Mercer's theorem states that a function K is a kernel function if and only if for any n vectors, $\bar{X}(1), \ldots, \bar{X}(n)$, and **any** real numbers $c_1, \ldots, c_n$, $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\bar{X}(i), \bar{X}(j)) \geq 0$.

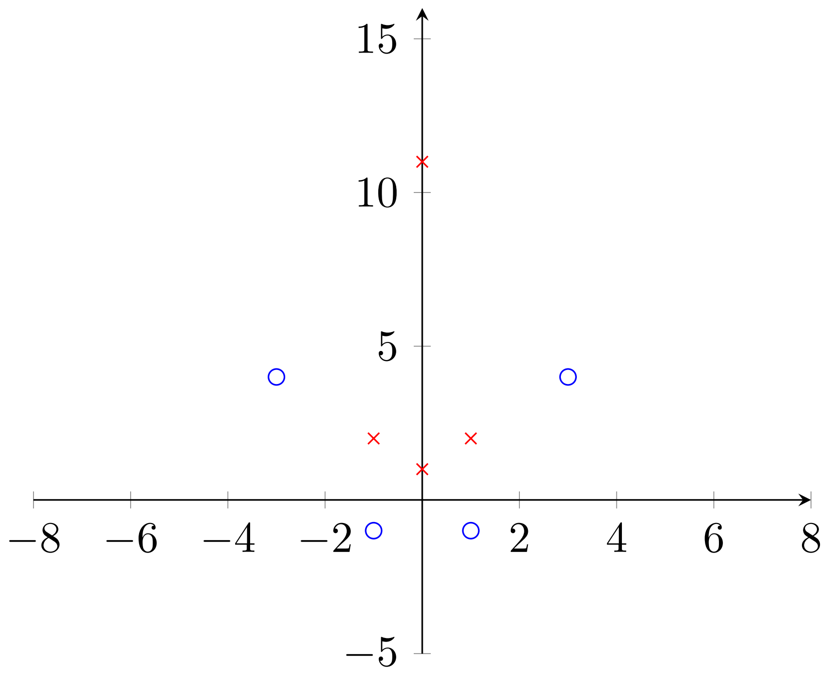
1. Prove the following half of Mercer's theorem: If K is a kernel then $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\bar{X}(i), \bar{X}(j)) \geq 0$. (Hint: Use $K(\bar{X}, \bar{Y}) = \phi(\bar{X}) \cdot \phi(\bar{Y})$).

2. Design a function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that is *not* a kernel.

3. Suppose $k = 2$, namely the original features are of the form $\bar{X} = [\bar{X}_1, \bar{X}_2]$. Show that $K(\bar{X}, \bar{Y}) = (1 + \bar{X} \cdot \bar{Y})^2$ is a kernel function. This is called as quadratic kernel. (Hint: One way to do this is to find $\phi : \mathbb{R}^2 \to \mathbb{R}^m$ (for some m) such that $\phi(\bar{X}) \cdot \phi(\bar{Y}) = (1 + \bar{X} \cdot \bar{Y})^2$).

4. Consider the training examples $\langle[0,1],1\rangle, \langle[1,2],1\rangle, \langle[-1,2],1\rangle, \langle[0,11],1\rangle, \langle[3,4],-1\rangle, \langle[-3,4],-1\rangle, \langle[1-1],-1\rangle, \langle[-1,-1],-1\rangle$. We have plotted the data points below.

   - Is the data **linearly classifiable** in the original 2-d space? If yes, please come up with *any* linear decision boundary that separates the data. If no, please explain why.
   - Is the data linearly classifiable in the feature space corresponding to the quadratic kernel. If yes, please come up with *any* linear decision boundary that separates the data. If no, please explain why.



Problem 3. (10 points). Suppose AdaBoost is run on n training examples, and suppose on each round that the weighted training error ε_t of the tth weak hypothesis is at most $\frac{1}{2} - \gamma$, for some number $\gamma > 0$. Show that after $T > \frac{\ln n}{2\gamma^2}$ rounds of AdaBoost the final combined classifier has zero training error!

Problem 4. (15 points). Recall bagging. Starting from a training set of size n, we created m bootstrap training sets $T_1, \ldots, T_m$, each of size n each by sampling with replacement from T.

1. For a bootstrap sample T_i, what is the expected fraction of the training set that does not appear at all in T_i? As $n \to \infty$, what does this fraction approach?

2. Let $m > 2 \ln n$, and $n \to \infty$. Show that the expected number of training examples in T that appear in at least one T_i is more than $n - 1$.

Problem 5. (40 points). You will implement new classification algorithms over the wdbc dataset with 400 training and 169 test examples. Please download it again if you do not have it.

In the class I mentioned that my svm code is taking ages to run. If you open the dataset, you might notice that some features have much larger values, and variances than others. We discussed in the class how this might affect the performance of nearest neighbor classifiers. Turns out that my svm code was slow due to the same reason. Normalization is the process of scaling the features to have values in *similar* ranges, and could help improve the accuracy of various algorithms. Do the following normalization to the dataset:

- For each of the 30 features, find the largest value that appears for it in the training set (`X-trn-400.csv`).
- Divide each feature value in the training and test set with the largest value obtained for the feature in the training set.

Call this the normalized data.

***k*-Nearest Neighbors (*k*-NN).** Please do the following.

1. Implement k-NN classification algorithm for $k = 1, 3, 5, 7, 9, 11, 13, 15$ for both unnormalized and normalized datasets.
2. In a single figure, plot the test accuracy (percentage) of k-NN classifiers on both datasets as a function of k.
3. Which dataset has higher accuracy? What is a potential reason for this?
4. The default distance measure used in k-NN is the ℓ_2 (euclidean) norm (which is what you should have implemented in the previous parts). **Only** for the normalized dataset, implement k-nearest neighbor classification algorithm for $k = 1, 3, 5, 7, 9, 11, 13, 15$ using the ℓ_1 norm as the distance measure. In a single figure, plot the accuracy of using the ℓ_2, and the ℓ_1 norm on the normalized dataset as a function of k.

   What is the best accuracy and corresponding k for the ℓ_2 and the ℓ_1 norm?

Support Vector Machines (SVM). Unless otherwise mentioned, use the normalized dataset for SVM questions.

1. Run a linear SVM for the penalty parameter $C = .0001, .01, .1, 1, 10, 100, 1000, 10000, 100000$, and plot the accuracy as a function of $\log C$ (semi-log plot).
2. Repeat the above for polynomial kernels of degrees 2, 3, 4, 5.
3. Explain the accuracy curve obtained (overfitting, underfitting etc) *for linear SVM.*
4. Run SVM with the Gaussian (RBF) kernel. Choose a few values of `gamma` and error penalty C that you like and show the accuracy you obtain for these. Tweak the values of γ and C to obtain the highest test accuracy you can obtain. What is the value of this highest test accuracy?
5. Try running SVM with a polynomial kernel of degree **three** on the unnormalized data. Kill the program if it does not complete within 1800 seconds. Why do you think the unnormalized data set takes longer to converge?

Bagging and Random Forests. A random forest classifier is essentially a bagging scheme that: (i) obtains a bunch of decision trees from the bootstrap samples, and then (ii) decides on the final label via majority.

1. For $m = 1, 2, 3, 4, 5, 6, 7$, implement a random forest classifier that takes m bootstrap samples, and plot the accuracy on the test data as a function of m.

2. Fix a value of m you like (between 7 and 11 (inclusive)). For `min_samples_leaf` $= 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$ plot test accuracy as a function of `min_samples_leaf`.

Boosting. AdaBoost is a method of combining weak classifiers to make strong classifiers.

1. Implement AdaBoost using decision stumps (trees with depth 1), for the number of weak classifiers (equals number of iterations) equal to $5, 10, 25, 50, 100, 150, 200$.

2. Repeat the previous part with trees of depth 10.

3. Why do you think the performance drops? Please explain.

Please summarize in no more than 100 words your thoughts on the algorithms, the normalization, and any other thoughts you had in the implementation.