Homework 2, due Tuesday March 1 COMS 4721 Spring 2016

⋆⋆⋆ Do two of { Problem 1, Problem 2, Problem 3 }, and also Problem 4. ⋆⋆⋆

Problem 1 (Margins; 10 points)**.** The Perceptron convergence theorem is often stated in the following way.

> If $R := \max_{(\boldsymbol{x},y)\in S} \|\boldsymbol{x}\|_2$, and $\boldsymbol{u}_\star \in \mathbb{R}^d$ satisfies $\|\boldsymbol{u}_\star\|_2 = 1$ and
>
> $$y\langle \boldsymbol{u}_\star, \boldsymbol{x}\rangle \;\geq\; \gamma \quad \text{for all } (\boldsymbol{x}, y) \in S$$
>
> for some $\gamma > 0$, then Perceptron halts after at most $(R/\gamma)^2$ iterations.

Explain why this theorem is the same as what was presented in lecture. Specifically, let $\boldsymbol{w}_\star$ be the vector from the Perceptron convergence theorem as given in the lecture, with length $\|\boldsymbol{w}_\star\|_2$ as small as possible. And let $\boldsymbol{u}_\star$ be the vector from the present version of the Perceptron convergence theorem such that γ is as large as possible. What is the relationship between $\boldsymbol{w}_\star$ and $\boldsymbol{u}_\star$, and between $\|\boldsymbol{w}_\star\|_2$ and γ? What is the shortest distance from a data point $\boldsymbol{x}$ from S to the (homogeneous) hyperplane with normal vector $\boldsymbol{w}_\star$? Give succinct (but precise) explanations for your answers.

Problem 2 (Features; 10 points)**.** It is common to pre-preprocess the feature vectors in $\mathbb{R}^d$ before passing them to a learning algorithm. Two simple ways to pre-process are as follows.

- **Centering**: Subtract the mean $\hat{\boldsymbol{\mu}} := \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \boldsymbol{x}$ (of the training data) from every feature vector:
$$\boldsymbol{x} \;\mapsto\; \boldsymbol{x} - \hat{\boldsymbol{\mu}}\,.$$
- **Standardization**: Perform centering, and then divide every feature by the per-feature standard deviation $\hat{\sigma}_i = \sqrt{\frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S}(x_i - \hat{\mu}_i)^2}$:
$$(x_1, x_2, \ldots, x_d) \;\mapsto\; \left(\frac{x_1 - \hat{\mu}_1}{\hat{\sigma}_1}, \frac{x_2 - \hat{\mu}_2}{\hat{\sigma}_2}, \ldots, \frac{x_d - \hat{\mu}_d}{\hat{\sigma}_d}\right).$$

(The same transformations should be applied to *all* feature vectors you encounter, including any future test points.)

For each of the following learning algorithms, and each of the above pre-processing transformations, explain whether or not each of the transformation can affect the resulting learned classifier.

(a) The classifier based on the generative model where class conditional distributions are multivariate Gaussian distributions with a fixed covariance equal to the identity matrix $\boldsymbol{I}$. Assume MLE is used for parameter estimation.

(b) The 1-NN classifier using Euclidean distance.

(c) The greedy decision tree learning algorithm with axis-aligned splits. (For concreteness, assume Gini index is used as the uncertainty measure, and the algorithm stops after 20 leaf nodes.)

(d) Empirical Risk Minimization: the (intractable) algorithm that finds the linear classifier (both the weight vector and threshold) that has the smallest training error rate.

You should assume the following: (i) the per-feature standard deviations are never zero; and (ii) there are never any "ties" whenever you do an arg max or an arg min. Also ignore computational and numerical precision issues.

Problem 3 (Covariance matrices; 10 points)**.** Let $\boldsymbol{X}$ be a mean-zero random vector in $\mathbb{R}^d$ (so $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{0}$). Let $\boldsymbol{\Sigma} := \mathbb{E}(\boldsymbol{X}\boldsymbol{X}^\top)$ be the covariance matrix of $\boldsymbol{X}$, and suppose its eigenvalues are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$. Let $\sigma^2 > 0$ be a positive number.

(a) What are the eigenvalues of $\boldsymbol{\Sigma} + \sigma^2 \boldsymbol{I}$?

(b) What are the eigenvalues of $(\boldsymbol{\Sigma} + \sigma^2 \boldsymbol{I})^{-1}$?

(c) Suppose $\boldsymbol{R} \in \mathbb{R}^{d \times d}$ is an invertible symmetric matrix that satisfies $\boldsymbol{R}\boldsymbol{R} = \boldsymbol{\Sigma} + \sigma^2 \boldsymbol{I}$. Let $\boldsymbol{v}_1 \in \mathbb{R}^d$ be an eigenvector of $\boldsymbol{\Sigma}$ corresponding to its largest eigenvalue λ_1. What is the variance of the random variable $\boldsymbol{v}_1^\top \boldsymbol{R}^{-1} \boldsymbol{X}$? Explain your answer succinctly (but precisely).

Hint: The answers can be given in terms of σ^2 and the eigenvalues of $\boldsymbol{\Sigma}$.

Problem 4 (Linear/quadratc classifiers; 40 points)**.** Download the spam data set `spam_fixed.mat` from Courseworks. This is the data set described in the *ESL* text for a binary classification problem of predicting whether an e-mail is spam or not. The training data and test data are in the usual format. You can read about the original features in *ESL* (Chapter 9.1, pages 300–301); in this version, the features are (approximately) standardized.

Write a MATLAB script or Python script that, using only the training data, tries out six different methods for learning linear/quadratic classifiers, and ultimately selects (via ten-fold cross validation) and uses one of these methods. (Think of the learning method itself as a "hyperparameter"!) The six methods to try are the following.

1. Averaged-Perceptron with 64 passes through the data.

   Averaged-Perceptron is like Voted-Perceptron (which uses Online Perceptron), except instead of forming the final classifier as a majority vote over the various linear classifiers, you simply form a single linear classifier by averaging the weight vectors and thresholds of the various linear classifiers. Important: Before each pass of Online Perceptron, you should randomly shuffle the order of the training examples.

2. Logistic regression model with MLE for parameter estimation.

3. Generative model classifier where class conditional distributions are multivariate Gaussian distributions *with shared covariance matrix for all classes.* Use MLE for parameter estimation.

4. Same as above, except arbitrary Gaussians (i.e., each class with its own covariance matrix).

5&6. Averaged-Perceptron and logistic regression as above, with feature map $\boldsymbol{\phi}\colon \mathbb{R}^{57} \to \mathbb{R}^{1710}$ given by $\boldsymbol{\phi}(\boldsymbol{x}) := (x_1, x_2, \ldots, x_{57}, x_1^2, x_2^2, \ldots, x_{57}^2, x_1x_2, x_1x_3, \ldots, x_1x_{57}, \ldots, x_{56}x_{57})$. No need to use the kernel trick here.

Note that we want to learn linear classifiers that are possibly non-homogeneous: so you should learn a weight vector $\boldsymbol{w}$ in $\mathbb{R}^{57}$ (or $\mathbb{R}^{1710}$) *and* a threshold value $t \in \mathbb{R}$.

You should write your own code for implementing Online Perceptron, Averaged-Perceptron, and cross-validation (put these code in separate files). The code (and the driver script that runs everything) should be easy-to-read, commented as necessary. For the other learning methods, you can use existing library implementations, but you are responsible for the correctness of these implementations (as per the specifications from the course lectures).

Report the cross-validation error rates for all methods, the training error rate of the classifier learned by the selected method (and state which method was chosen), and the test error rate for the learned classifier. Submit all codes & scripts you write yourself, and give references to any existing software you use for the other learning algorithms.