EE 559 **Homework Week 7** Posted: Wed., 2/21/2018
Jenkins Due: Fri., 3/2/2018, 11:59 AM*

Notes:

1. If you use Matlab for computer problems in this class, starting with this assignment you will need the toolbox PRTools5. You can read about it and download it at http://37steps.com . Be sure you download version 5 (not version 4).
2. If you use Python for computer problems in this class, starting with this assignment you will need the library scikit-learn. More info can be found at: http://scikit-learn.org/stable/install.html .
3. An introduction to both PRTools5 and scikit-learn are given in Discussion 7 on Thursday, 2/22.
4. * Solutions to this homework will be posted 12 hours after the assignment due date (because of the upcoming midterm exam).

1. Suppose you set up a training algorithm to use a modified MSE criterion:

$$J(\underline{w}) = \frac{1}{N}\left\|\underline{\underline{X}}\,\underline{w} - \underline{b}\right\|_2^2 + \lambda\left\|\underline{w}\right\|_2^2$$

in which the purpose of the new term is to prefer small $\left\|\underline{w}\right\|_2$ if $\lambda > 0$.

(a) Find $\nabla_{\underline{w}} J(\underline{w})$ using gradient relations.

(b) Find the optimal $\underline{w} = \hat{\underline{w}}$ by solving $\nabla_{\underline{w}} J(\underline{w}) = \underline{0}$. Compare your result to the pseudoinverse solution.

2. In this problem, you are asked to compare the classification performance of perceptron (sequential gradient descent or similar version) and MSE (pseudoinverse version) classifiers on the wine dataset. This problem is designed to be used with the functions provided by the PRTools5 toolbox (if using Matlab), or the named functions from scikit-learn (if using Python).

For this problem, use the wine dataset files provided in the HW2 folder.

(a) State clearly whether you are using PRTools5 and Matlab, or scikit-learn and Python.

(b) Store a copy of the unnormalized data (as provided), and also a standardized version of the data, for your use. Standardized means each feature is normalized to 0 mean and unit variance. Note that the normalizing factors should be calculated from the training data only (why?), and then applied to both the training data and test data.

For this part, report on the mean and standard deviation of each feature of the unnormalized training data, and answer the "why?" question above.

Hint: For part (b), you may either code it yourself, or use available functions: scikit-learn's function sklearn.preprocessing.StandardScaler, or Matlab's function zscore().

Parts (c)-(f) below use perceptron.

(c) For the perceptron classifier (perlc in PRTools5 or sklearn.linear_model.Perceptron in Python), answer the following questions (by looking at the documentation, comments, or the code, as needed):

(i) What is the default initial weight vector?

(ii) What is the halting condition? If the solution weight vector (which would correctly classify all training data points) is not reached, what is the backup halting condition?

Hints for (ii): (1) For both PRTools and scikit-learn, this may require some digging into the code to answer. Note that for PRTools, the backup halting condition is not just the runtime. (2) If you prefer, you can skip this for now and answer most of the parts below without the answer to (c)(ii); then come back to finish this. (Some of the parts that involve comparing and explaining may depend on the answer to (c)(ii).)

(d) This part will be done twice: once using only the first 2 features, and then again using all 13 features.

Apply the perceptron learning algorithm to the training data, using the one vs. rest method. Report the resulting 3 weight vectors and the classification accuracy of your classifier on both the training set and the test set.

Tips:

PRTools5 users: Note that perlc first augments and standardizes the data by default, so you can input the unnormalized unaugmented data. Also, one vs. rest is the default method. testc will give classification error rate. You can retrieve weight vectors using getWeightsFromPrmapping.m (provided in the HW7 folder).

Scikit-learn users: You can extract the weight vectors using model.coef (for nonaugmented $\underline{w}$) and model.intercept (for w_0).

(e) This part should also be done twice (for the first 2 features and for all 13 features). Run the perceptron of part (d) 100 times, with randomly chosen starting weight vectors each time. Pick the run that has the best performance on the training set. (If there is a tie for the best set of 3 weight vectors, pick one of the sets at random.) Report the final 3 weight vectors, and the classification accuracy on both the training set and the test set.

(f) Compare and comment on (explain) your test-accuracy results from (d) and (e). (Compare 2 features to 13 features for each; and, compare (d) to (e).)

Parts (g)-(j) below use MSE (pseudo-inverse version) classification. Please also refer to the tips for PRTools implementation and for scikit-learn implementation below.

(g) For this part use unnormalized data. Run the pseudoinverse classifier, and report the classification accuracy on the test data, for the first 2 features and for all 13 features.

(h) Repeat part (g) except using standardized data. **Hint:** the results might not be what you expected.

(i) Compare and comment on (explain) your accuracy results of (g) and (h).

(j) Compare and comment on (explain) your accuracy results of (h) and (e).

Tip for PRTools5 implementation of pseudoinverse classifier

Use fisherc.

Tips for scikit-learn implementation of pseudoinverse classifier

Use sklearn.linear_model.LinearRegression. This regression function can be used for 2-class classification.

> Use non-reflected data points.

> Refer to Discussion 7 for more tips.

3. In a 2-class problem with 2 features, you are given the following prototypes:

$$S_1: \ (0,0)^T, (0,1)^T, (0,-1)^T$$
$$S_2: \ (-2,0)^T, (-1,0)^T, (0,2)^T, (0,-2)^T, (1,0)^T, (2,0)^T$$

(a) Plot the points in 2D (non-augmented) feature space. Are they linearly separable?

The rest of this problem deals with using a phi machine approach to get a nonlinear classifier. Use a quadratic polynomial mapping, and order the components of your mapped vectors $\underline{u}$ the same as we did in lecture.

(b) List the points [as $(D'+1)$-tuples] in expanded feature space.

(c) Find a decision boundary (by hand is fine) in the expanded feature space. [**Hint**: try plotting the points in $\left(x_1^2, x_2^2\right)$ space.] Plot the boundary and decision regions [in $\left(x_1^2, x_2^2\right)$ space], and give a complete weight vector $\underline{w}'$ that correctly separates the prototypes in the expanded feature space.

(d) Map the decision boundary and regions that you found in (c) back into the original feature space; plot them. Give an equation for the decision boundary in this space (that is, in terms of x_1 and x_2).