

**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. Introductions to both PRTools5 and scikit-learn are given in Discussion 7 on Thursday, 2/21; also see the documents from that discussion, Intro\_to\_PRTools5.pdf and Intro\_to\_scikitlearn.pdf, posted on D2L.

1. 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. For these parts, use standardized data.*

- (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.

- (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:**

PRTTools5 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 performing run, pick one of the runs 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 your results from (d) and (e) on training data and on test data. That is, compare 2 features to 13 features in (d) for each dataset; likewise compare 2 features to 13 features for in (e) for each dataset. Also, compare (d) to (e) for 2 features for each dataset; and compare (d) to (e) for 13 features for each dataset.

*Parts (g)-(j) below use MSE (pseudo-inverse version) classification. Please also refer to the tips for PRTTools 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.
- (i) Compare your test-accuracy results of (g) and (h). Are they identical, similar, or quite different?
- (j) Compare and comment on your test-accuracy results of (h) and (e). Are they identical, similar, or quite different?

*Tip for PRTTools5 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 and Intro documents posted on D2L for more tips.

2. Note: in this problem you may do the plots by hand or by computer (your choice); but everything else (e.g., the quadratic mapping, finding decision regions and boundaries, etc.) is to be done by hand.

In a 2-class problem with 2 features, you are given the following training data:

$$S_1: (0,0)^T, (0,1)^T, (0,-1)^T$$

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

- (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) in the expanded feature space. [**Hint:** try plotting the points in  $(x_1^2, x_2^2)$  space.] Plot the boundary and decision regions (in  $(x_1^2, x_2^2)$  space), and give a complete weight vector  $\underline{w}'$  that correctly separates the prototypes in the expanded feature space. Also state the decision rules in the notation of the expanded feature space (i.e., in terms of the  $u_i$ ).
- (d) Map the decision boundary and rules/regions that you found in (c) back into the original feature space: that is, give the decision rules in this space (in terms of  $x_1$  and  $x_2$ ), and give an equation for the decision boundary (in terms of  $x_1$  and  $x_2$ ). Plot the data points, decision boundary, and show the decision regions, all in the original  $(x_1, x_2)$  feature space.