Prof. Sergio A. Alvarez
St. Mary's Hall, S255
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CSCI 3345, Machine Learning
Fall 2019
Problem Set 3

1. LFD Problem 2.24, parts a-c. Part d is optional extra credit.

2. LFD problem 3.1. Include a source code listing in your writeup (attach the source file separately), as well as a plot that shows the data set that you are using. Use the matrix form of linear regression derived in chapter 3. Work in Python, using the matplotlib.pyplot.scatter function for plotting, and numpy.matrix for matrix operations, with A.dot(B) for matrix multiplication. Consult the appropriate documentation in all cases.

3. In the basic linear regression setup, we have a labeled data set $\{(x_i, y_i),\ i = 1, \cdots, N\}$, where each x_i is a row vector of non-target values, and y_i is the corresponding scalar target value – a single number. We collect the x_i together as the rows of a matrix, X. Likewise, we collect the target values into a column vector, y, so that the target value for the ith row of X is the value in the ith row of y. We now seek to find the values of a column vector of regression coefficients, c, that minimize the discrepancy between the vector of target values, y, and its linear approximation, Xc, associated with that particular choice of regression coefficients. The *best* choice of c will minimize the squared distance $\|Xc - y\|^2$. Since the squared distance can be written as the matrix product $f(c) = (Xc - y)^T(Xc - y)$, which is a quadratic function of c, we can find the best choice of c by setting the derivative of $f(c)$ equal to 0. In this task, you will calculate that derivative, step by step.

   (a) Using basic properties of matrix arithmetic (specify which ones), rewrite $f(c)$ as a sum of four matrix products with mixed signs. Explain.

   (b) One of the four additive terms in $f(c)$ is of the form $e(c) = Ac$, where A is a matrix. It's not hard to find the derivative of a linear term like this one. By definition, the derivative of $e(c)$ is a matrix, D, such that $e(c + s) = e(c) + Dc + o(s)$, where s represents a vector of small magnitude and the notation $o(s)$ represents a sublinear term, meaning that $\|o(s)\|/\|s\| \to 0$ as $s \to 0$. To find the derivative $e'(c)$, first calculate $e(c + s)$ exactly, and extract the derivative from the result. Work with the generic symbol A for the matrix here. *Explain.*

(c) Another term in $f(c)$ should be of the form $q(c) = c^T Qc$, where Q is a matrix. Find the derivative of $q(c)$ (using the generic Q notation for the matrix), by analyzing the expression for $q(c+s)$ as in the preceding part. *The answer won't match the book's notation exactly*, and that's ok. Use the fact that the transpose of a scalar is a scalar. You'll actually need a nonzero sub-linear term this time, by the way. Explain carefully.

(d) Put the preceding parts together to find the derivative of $f(c)$, using the fact that the derivative of a sum is the sum of the derivatives. We've only discussed three of the four terms, so you'll need to explain how to deal with the remaining one.

(e) It can be easier to work with the transpose of the derivative matrix of $f(c)$, known as its *gradient*, and denoted $\nabla f(c)$. The best regression coefficient vector makes the gradient vector equal to 0. Write the matrix equation that captures the zero gradient condition, based on your work above. The equation should be of the form $Ac = b$, for a suitable matrix, A, and column vector, b. Explain carefully.

4. The standard solution to the linear regression problem relies on invertibility of the matrix $X^T X$, where X is the matrix whose rows are the vectors of non-target values of the data examples (same notation as in the above task). Since not all matrices are invertible, that seems like a lot to ask. In this task, you will estimate the *probability* that $X^T X$ is invertible, for a randomly selected matrix, X. We'll make the reasonable assumption that the number of rows of X is strictly greater than the number of columns, which is the same as the assumption that there are strictly more data examples than there are non-target attributes. This is typically the case in linear regression. We'll refer to such matrices as being "tall". It's difficult to determine invertibility of real-valued matrices exactly when working with finite-precision arithmetic, as you will be doing. Therefore, as a proxy for invertibility of A, use the condition that the determinant of A is greater than the arbitrary threshold 10^{-10} in absolute value. Use `numpy.matrix` (with `A.dot(B)` for matrix multiplication, `A.transpose()` for transposition), `numpy.random`, `numpy.linalg.det`. Refer to the corresponding documentation.

   (a) Write a Python program that generates a large number (10^5, say) of random tall matrices and reports the fraction of those matrices that are invertible. That fraction is an estimate ("Monte Carlo estimate") of the probability of invertibility among tall matrices of the sizes considered. Let the number of rows be a random value between 2 and 10, and ensure that the number of columns is at least one less than the number of rows. Submit documented source code, and include the text of the source code in your main writeup, as well.

   (b) Run your program and report the estimated probability of invertibility. Provide screen shots. Discuss the plausibility of the invertibility assumption for $X^T X$ in light of your results.

5. Predictive performance of machine learning methods depends strongly on the quality of the data representation implicit in the input attributes. This phenomenon can be especially important when the hypothesis space has low complexity, which limits its expressive capability. For concreteness, you'll work with purely linear classifiers here, in which each hypothesis is of the form $h(x) = \text{sgn}(w^T x)$, where x is the column vector of predictive attributes and w is a column vector. The class predicted by h is determined solely by the sign of the quantity $w^T x$. The classes are $+1$ and -1.

   Data that are distributed normally (according to a Gaussian distribution for each class) are most naturally classified by quadratic boundaries in the attribute space (e.g., parabolas, circles, ellipses for 2-D data). Such decision boundaries cannot usually be captured by linear classifiers, except in a special case such as that of two identically distributed classes, in which the optimal decision boundary is the bisector of the line segment between the two class centers.

   In order to deal with more typical cases, it can be convenient to perform an appropriate nonlinear transformation of the input attributes prior to classification. Specifically, assume that the vectors of non-target attributes are two dimensional: $x = (x_1, x_2)^T$, and consider the following transformation of x into a new vector, z:

$$z = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)^T$$

   The new vector, z, is 6-dimensional, whereas the original vector, x, is 2-dimensional.

   (a) Write the equation of the circle of radius 3 and center at the point $(-5, 2)$ in the original x space. Do this in the original variables x_1, x_2, first, and then rewrite it as a *linear* expression in the transformed variables $z_1, \cdots, z_6$ (of the form $\sum_{i=1}^{6} c_i z_i$, with no additional "bias" term). Explain carefully.

   (b) Fit a `Perceptron` object from the `sklearn.linear_model` module to the labeled data in the file `sampleQuadData.txt`. Specify `fit_intercept=False` in the constructor method – the inclusion of the 1 constant as the first component of the transformed data, z_1, makes the bias (intercept) term unnecessary. Report the `score` obtained on this data set. Repeat the process, now using the data in the file `sampleQuadDataTransformed.txt`, which contains exactly the same set of data examples as in `sampleQuadData.txt`, but transformed as described in the statement, above. How do the results compare? Discuss, considering the corresponding *classification error rates.*

   (c) Take a closer look at the `Perceptron` instance for the transformed data, by examining its coefficients in the `_coef` data field. What is the approximate equation of the decision boundary in the space $(z_1, z_2, z_3, z_4, z_5, z_6)$? What is the corresponding equation in the original space (x_1, x_2)? Explain in detail.

   (d) Ignore the mixed $x_1 x_2$ term in the boundary equation from the preceding part in order to identify the geometric shape of the boundary (look up information about analytic geometry). State the specific shape name. Does this boundary shape seem like a good model of the available data in this task? Discuss.