EE 525X / Spring 2018

Homework 2

Please scan and upload your assignments on or before February 10, 2018.

- You are encouraged to discuss ideas and collaborate with each other; but
- you must *clearly* acknowledge your collaborator, and
- you must compose your own writeup and/or code independently, and
- you must combine all derivations, code, and results as a single PDF to be uploaded on BB.
- Maximum score: 60 points

---

1. **(5 points)** Recall that a function of the form

$$f(x) = g(\langle w, x \rangle + b),$$

where g is a function that maps real numbers to class labels is called a *linear threshold function* (LTF). LTFs can be used to express a wide range of functions, some that are unexpected. For example, if the features of the data vector x are binary (0/1), the function $f(x) = x_1$ OR x_2 OR $\ldots$ OR x_d can be represented as an LTF:

$$f(x) = \text{Heaviside}(x_1 + x_2 + \ldots + x_d - 1).$$

Derive LTF representations of the following functions. Assume that data is represented by binary d-dimensional vectors, $x \in \{0, 1\}^d$.

   a. $f(x)$ is a conjunction (AND) of the subset of features indexed by $i_1, \ldots, i_k$.

   b. $f(x) = \text{MajorityBit}(x)$.

2. **(10 points)** In class, we discussed how to represent XOR-like functions using quadratic features, since standard perceptrons are insufficient for this task. However, here we show that XOR-type functions can indeed be simulated using *multi-layer* networks of perceptrons. This example shows a glimpse of the expressive power of "deep neural networks": merely increasing the depth from 1 to 2 layers can help reproduce highly nonlinear decision boundaries.

   a. Consider a standard two-bit XOR function, where we have 2-dimensional inputs $x_1, x_2 = \pm 1$, and output $y = x_1(XOR)x_2 = \begin{cases} -1 & \text{if} \quad x_1 = x_2 \\ 1 & \text{otherwise} \end{cases}$.

   Geometrically argue why a single perceptron cannot be used to simulate the above function.

   b. Graphically depict, and write down the equation for, the optimal decision region for the following logical functions:
      (i) $x_1(AND)(NOT(x_2))$
      (ii) $(NOT(x_1))(AND)x_2$
      (iii) $x_1(OR)x_2$ Make note of the weights learned corresponding to the optimal decision boundary for each function.

   c. Using the above information, simulate a multi-layer perceptron *network* for the XOR operation with the learned weights from Part (b).

3. **(10 points)** Suppose we are given real-valued scalar data (i.e., $d = 1$) belonging to one of two classes. We are given a set of three data samples with negative labels, $X_- = \{0, 1, -1\}$, and a set of three data samples with positive labels, $X_+ = \{-3, 3, -2\}$. Our goal is to build a classifier for this dataset. We will show that kernel methods are particularly useful in this case.

   a. Argue that no perfect linear separator in the original space can exist.

   b. Argue that if the data is mapped via the two-dimensional feature mapping $\phi(u) = (u, u^2)$, then a perfect linear separator exists.

   c. Given two (scalar) data points x_1 and x_2, write down the explicit form of the kernel inner product for this new feature space in terms of x_1 and x_2.

   d. Explicitly calculate the maximum-margin separating hyperplane in the kernel feature space. (It should be a straight line since the feature space is two-dimensional; provide the equation for this line.) Calculate the value of its margin. Use the mapping ϕ to plot the data points in the new feature space. Draw the separating hyperplane, and mark the closest points nearest to the hyperplane.

4. **(10 points)** Assuming a classification application with n data points in d dimensions where $n = d = 10^{10}$, *rank* the following algorithms in decreasing order of (i) training times; (ii) testing times. Explicitly state these running times in terms of n and d.

   a. Perceptron.

   b. Nearest neighbors.

   c. Kernel perceptron with a polynomial kernel of order 4.

   d. Kernel perceptron with gaussian kernel.

   e. Linear support vector machines.

5. **(15 points)** We are given a sample of Italian wine. We can narrow its origin down to one of 3 possible regions – Veneto, Tuscany, and Piedmont – but no further. Fortunately, (i) we have several sample wines from these regions, and (ii) we know how to build a classifier from samples. This problem details how to build such a wine classification method.

   a. Download the data from http://archive.ics.uci.edu/ml/datasets/Wine. Go through the ReadMe file carefully for instructions.

   b. Randomly choose all but 5 samples from each class as the training data points. Test the performance of a k-NN classifier with $k = 1$. Record how many data points (out of 15) were wrongly classified. Repeat this random choice several times to calculate the probability of error.

   c. How does your above answer change if k increases? Plot/tabulate the probability of error as a function of k (up to $k = 9$).

   d. Staring at the data a bit, we realize that the "scale" of the numbers is a bit strange (some feature values are much larger than others.) To fix this, we compute the mean μ and standard deviation σ of each feature, and normalize each observation x as $\frac{x-\mu}{\sigma}$. Repeat the above NN classification with the transformed data, and show that your classification performance improves. (Such a transformation is called a *z-score* in statistics.)

6. **(10 points)** In this problem, we will implement the Perceptron algorithm on synthetic training data.

    a. Suppose that the data dimension d equals 2. Generate two classes of data points with 100 points each, by sampling from Gaussian distributions centered at $(0.5, 0.5)$ and $(-0.5, -0.5)$. Choose the variance of the Gaussian to be small enough so that the data points are sufficiently well separated. Plot the data points on the 2D plane to confirm that this is the case.

    b. Implement the Perceptron algorithm as discussed in class. Choose the initial weights to be zero and the maximum number of epochs as $T = 100$, and the learning rate $\alpha = 1$. How quickly does your implementation converge?

    c. Now, repeat the above experiment with a second synthetic dataset; this time, increase the variance of the Gaussians such that the generated data points from different classes now overlap. What happens to the behavior of the algorithm? Does it converge? Show that classification regions obtained at the end of T epochs.

7. **(Optional)**: how much time did you spend on this assignment?