Massachusetts Institute of Technology

6.867 Machine Learning, Fall 2006

Problem Set 1

Due Date: Thursday, Sep 28. (at 12:00 noon)
You may submit your solutions in class or in the box.

Section A: Background (Answering this section is optional; if you do submit answers to these questions, they will be graded but will be not scored i.e. your score for this assignment will not depend on this section.)

1. Write a MATLAB function `birthday_prob(n)` that computes the probability that, in a group of n people at least two share the same birthday (assume that $n \geq 2$). Using this function, compute the minimum size of the group such that this probability is atleast 0.5. Attach a printout of the code with your answer.

2. Let $X_1, X_2, X_3, \ldots, X_n$ be independent random variables, each with a Uniform distribution over $(0, 1)$:

   $$f(x) = \begin{cases} 1 & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

   Find (a) $E[\text{Max}(X_1, X_2, \ldots, X_n)]$ and (b) $E[\text{Min}(X_1, X_2, \ldots, X_n)]$.

3. The Flushing Meadows Open, a tennis tournament held every year, has a Men's Doubles event with 16 seeded pairs. Unfortunately, it is unseasonably cold and wet, and many players are down with flu. In particular, 4 of the seeded players are sick so that they and their partner can not play. What is the expected number of seeded pairs remaining in the tournament? Assume that any of the $\binom{32}{4}$ possible combinations of 4 seeded players is equally likely to be sick.

4. You are a contestant in the Monty Hall game show. In this game, there are three doors. Behind one door is a brand new Corvette; there are goats behind the other two doors. Monty Hall asks you to select a door; after you have chosen one, he opens one of the other two doors to reveal a goat behind it. You can now choose to stick with your original choice, or to switch to the remaining door.

   (a) What should you do? Will it make a difference if you switch?

   (b) Run a thousand trials of this experiment in MATLAB. Can you explain the results?

   (c) Bonky Hall, Monty's brother, starts his own game show in which there are four doors instead of three, but with the same setup otherwise. When you choose a door initially, Bonky also opens one door, revealing a goat behind it. In addition to your chosen door, there now remain two more closed doors. Should you switch to one of these? Why? If you decide to switch, which of the two doors should you choose?

5. (a) Let X be a Gaussian vector with

   $$E[X] = \begin{pmatrix} 10 \\ 5 \end{pmatrix} \quad cov(X) = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

   Write an expression for the pdf of X that does not use matrix notation, i.e., if $X = [x_1, x_2]$ write the joint pdf $P(x_1, x_2)$.

(b) Let A, B be $p \times q$ matrices and x be a random $q \times 1$ vector. Prove that

$$cov(Ax, Bx) = A\, cov(x)\, B^T$$

where $cov(u, v) = E[(u - E[u])(v - E[v])^T]$ is the cross-covariance matrix between random vectors u and v, while $cov(u) = E[(u - E[u])(u - E[u])^T]$ is the covariance matrix for u.

6. From our Linear Algebra notes...

Theorem 3.1.5: (Gram-Schmidt Process) Let $v_1, ..., v_k$ be an ordered, linearly independent set. Then, there exists an ordered set $u_1, ..., u_k$ of orthonormal vectors such that

$$\begin{aligned} \langle u_1 \rangle &= \langle v_1 \rangle \\ \langle u_1, u_2 \rangle &= \langle v_1, v_2 \rangle \\ &\vdots \\ \langle u_1, \ldots, u_k \rangle &= \langle v_1, \ldots, v_k \rangle. \end{aligned}$$

(Here, $\langle v_1, v_2, \ldots, v_k \rangle$ denotes the span of the vectors v_1, v_2, ..., and v_k; that is, the set of all vectors that can be expressed as a linear combination of the v_i's.)
PROOF: Set $u_1 = v_1 / \|v_1\|$. Then, $\|u_1\|^2 = 1$ and $\langle u_1 \rangle = \langle v_1 \rangle$. Set

$$u_2 = \frac{v_2 - (v_2^T u_1) u_1}{\|v_2 - (v_2^T u_1) u_1\|}.$$

Then,

$$u_2^T u_1 = \frac{v_2^T u_1 - (v_2^T u_1)(u_1^T u_1)}{\|v_2 - (v_2^T u_1) u_1\|} = 0,$$

and so $\{u_1, u_2\}$ is orthonormal, and $\langle u_1, u_2 \rangle = \langle v_1, v_2 \rangle$. Next, set:

$$u_3 = \frac{v_3 - (v_3^T u_2) u_2 - (v_3^T u_2) u_2}{\|v_3 - (v_3^T u_2) u_2 - (v_3^T u_2) u_2\|}.$$

Again, $u_3^T u_2 = u_3^T u_1 = 0$ and $\|u_3\|^2 = 1$. Continue in this way to obtain vectors $u_1, u_2, ..., u_k$ with the stated properties. ■

The procedure described in the proof is called the Gram-Schmidt Process. Using MATLAB, apply this procedure to the following set of (independent) vectors.

$$v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, v_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}, v_3 = \begin{bmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \\ 36 \end{bmatrix}, v_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

What happens if you do it in a different order (say, when you move v_4 to the front)?

Section B: Answering this section is compulsory; your score for this assignment will depend only on this section.

Some of the problems will require some MATLAB programming. On the course webpage, we have provided some data and scripts to get you started. These are provided as part of a single `.zip` archive. Once you have downloaded this archive, you can extract the individual files using the following command (on Linux):
`$unzip <FILE-NAME.zip>`
The archive also contains load scripts for you to load data into MATLAB. These scripts are named by the problem number. For example, to load the data for Problem 1(b), you should use the script `load_p1_b`.

1. Implement a Perceptron classifier in MATLAB. Start by implementing the following functions (you should attach a printout of your MATLAB code of these functions with your submission):

   - a function `perceptron_train(X, y)` where `X` and `y` are $n \times d$ and $n \times 1$ matrices respectively. This function trains a Perceptron classifier on a training set of n examples, each of which is a d-dimensional vector. The labels for the examples are in `y` and are 1 or -1. The function should return `[theta, k]`, the final classification vector and the number of updates performed, respectively. You may assume that the input data provided to your function is linearly separable.
   - a function `perceptron_test(theta, X_test, y_test)` where `theta` is the classification vector to be used. `X_test` and `y_test` are $m \times d$ and $m \times 1$ matrices respectively, corresponding to m test examples and their true labels. The function should return `test_err`, the fraction of test examples which were misclassified.

   For this problem, we have provided you two custom-created datasets. The dimension d of both the datasets is 2, for ease of plotting and visualization.

   (a) Load data using the `load_p1_a` script and train your Perceptron classifier on it. Using the function `perceptron_test`, ensure that your classifier makes no errors on the training data. What is the angle between `theta` and the vector $(1, 0)^T$? What is the number of updates k_a required before the Perceptron algorithm converges?

   (b) Repeat the above steps for data loaded from script `load_p1_b`. What is the angle between `theta` and the vector $(1, 0)^T$ now? What is the number of updates k_b now?

   (c) For parts (a) and (b), compute the geometric margins, γ^a_{geom} and γ^b_{geom}, of your classifiers with respect to their corresponding training datasets. Recall that the distance of a point x_t from the line $\theta^T x = 0$ is $|\frac{\theta^T x_t}{\|\theta\|}|$.

   (d) For parts (a) and (b), compute R_a and R_b, respectively. Recall that for any dataset $\mathcal{X}$, $R = \max\{\|x\| \mid x \in \mathcal{X}\}$.

   (e) Plot the data (as points in the X-Y plane) from part (a), along with decision boundary that your Perceptron classifier computed. Create another plot, this time using data from part (b) and the corresponding decision boundary. Your plots should clearly indicate the class of each point (e.g., by choosing different colors or symbols to mark the points from the two classes). We have a provided a MATLAB function `plot_points_and_classifier` which you may find useful.

2. Implement an SVM classifier in MATLAB, arranged like the Perceptron in problem 1, with functions `svm_train(X, y)` and `svm_test(theta, X_test, y_test)`. Again, include a printout of your code for these functions.

   HINT: Use the built-in quadratic program solver `quadprog(H, f, A, b)` which solves the quadratic program: $\min \frac{1}{2}x^T Hx + f^T x$ subject to the constraint $Ax \leq b$.

(a) Try the SVM on the two datasets from Problem 1. How different are the values of `theta` from values the Perceptron achieved? To do this comparison, should you compute the difference between two vectors or something else?

(b) For the decision boundaries computed by SVM, compute the corresponding geometric margins (as in Problem 1(c)). How do the margins achieved using the SVM compare with those achieved by using the Perceptron?

3. Now that you are familiar with SVMs, download this highly optimized version called SVM$_{light}$: `http://svmlight.joachims.org/`

   We will use this package to experiment with classifying images of handwritten digits. To simplify things, we will only be distinguishing between 0s and 1s.

   (a) Train a plain-vanilla SVM (i.e, no regularization) on `train-01-images.svm` (which is already in SVM$_{light}$'s format). What is the *training error*, i.e., the fraction of examples in the training data that are misclassified? From the set of misclassified images, pick one. Attach of plot of it with your solution. Why do you think the SVM fails to classify it correctly during training? Now apply the SVM to the test set `test-01-image.svm`; what is the *test error*, i.e., the fraction of test data that is misclassified?

   (b) Experiment with different values of the regularization term C (set using the `-c` flag). Start by guessing/estimating a range in which you think C should lie. Then choose the values of C (within that range) at which you will evaluate the performance of the SVM. You need not pick more than 10 such values, though you should feel free to pick as many (or as few!) as you want. For these values of C, plot the corresponding error. What value of C gives you the best training error on the dataset from part (a)? How does the test error for this choice of C compare with the test error you computed in part (a)? Could you optimize C so that it leads to the best *test* error, rather than the *training* error? Should you?

   (c) An adversary (we'll call him W) mislabeled 10% of the images in the original training set to produce `train-01-images-W.svm`. Using the same choices of C as in part (b), find the C that results in the best training error. What is the corresponding test error on `test-01-image.svm`?