Machine Learning 10-701 September 18, 2013
http://alex.smola.org/teaching/cmu2013-10-701x/ due on October 2, 2013
Carnegie Mellon University

Homework 1

START HERE: Instructions

- The homework is due at 10:30am on October 2, 2013. Anything that is received after that time will not be considered.
- Answers to everything but the coding question will be submitted electronically (e.g. as a PDF or handwritten and scanned). We'll give exact instructions on the email list soon; in the meantime, make sure you prepare the answers to each question separately.
- Please follow the instruction for code submission in problem 4 correctly.
- The handout for questions 3 and 4 is at http://alex.smola.org/teaching/cmu2013-10-701x/assignments/assignment_1_handout.tar.
- Collaboration on solving the homework is allowed (after you have thought about the problems on your own). However, when you do collaborate, you should list your collaborators! You might also have gotten some inspiration from resources (books or online etc...). This might be OK only after you have tried to solve the problem, and couldn't. In such a case, you should cite your resources.
- If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution.

1 Probability Review [Ahmed; 20pts]

1.1 Why just 2 variables ? Let's go for 3 [6 pts]

Let A, B and C be three discrete random variables, use Bayes rule to show:

1. Conditional Bayes rule: $P(A|B,C) = \frac{P(A,B|C)}{P(B|C)}$.
2. Conditional marginalization: $P(A|C) = \sum_B P(A,B|C)$.
3. Chain rule: $P(A|C) = \sum_B P(A|B,C)P(B|C)$.

1.2 Evaluating Test Results [8 pts]

A movie recommendation website is adopting a new recommendation algorithm (A2). Before total adoption, the developers conduct an A/B test where the old algorithm (A1) is used for some randomly chosen transactions for the sake of comparison. A successful transaction is a one where one of the recommended movies is clicked on. Table 1 shows the results of the test.

	A1	A2
# successful	6000	2150
# failed	1700	500

Table 1: A/B Test Results

- If a transaction succeeds, what is the probability that it has been handled by A2?
- Do you recommend using A2 instead of A1? Why?
- A test engineer claims that if a transaction succeeds with A1, there is 70% chance it will succeed with A2. Show that this claim cannot be true. (Assume that errors in estimating probabilities from test results are negligible.)

Homework 1

Please justify your answers with probabilistic arguments and show your calculations.

1.3 Monty Hall Problem [6pts]

In a game show, you are faced with three doors. Behind one door is a car; behind the others, goats. The door to keep the car behind is chosen uniformly at random. You pick a door and the host, who knows what's behind the doors, opens another door which has a goat. (If you pick the door with the car the host will choose one of the other doors with equal probabilities). You are then given the opportunity to stick with the door you picked or switch to the other closed door.

At first glance, it seems that both options are equally likely to win, since the door to keep the car behind was chosen uniformly at random. We will show that switching actually doubles your chance to win (well, assuming you prefer a car over a goat).

Let $carX$, $chooseX$ and $openX$ denote that the car is behind door X, that you choose door X and that the host opens door X respectively. Assume, without loss of generality, that you choose door 1 and the host opens door 2. Use conditional Bayes rule to show that

$$\frac{P(car3|choose1, open2)}{P(car1|choose1, open2)} = 2$$

[Hint: This looks like the Bayes classification rule, but since we don't know the choice strategy, you will want to keep expanding by conditioning on $choose1$.]

2 Regression [Leila; 20pts]

This question is meant to refresh or improve your knowledge of Linear Regression, Ridge Regression, MLE and MAP estimation.

2.1 Linear Regression [10pts]

Consider a linear model of the form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1}$$

Where $\mathbf{y}$ is an $N \times 1$ vector of observed output, $\mathbf{X}$ is an $N \times p$ matrix of covariates of interest, $\boldsymbol{\beta}$ is the $p \times 1$ vector of parameters and $\boldsymbol{\epsilon}$ is an $N \times 1$ vector of noise. The noise terms ϵ_i are sampled from $\mathcal{N}(0, \sigma^2)$.

Finding the solution for this problem is done by minimizing the sum of squares:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \tag{2}$$

Which can also be written as:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{3}$$

1. We will being by showing that the MLE solution to this problem is the same as the least squares solution. First, write the likelihood of the data, i.e. $P(\mathbf{y}|\boldsymbol{\beta})$.

   Hint: The y_is are IID, so $P(\mathbf{y}|\boldsymbol{\beta}) = \prod_i P(y_i|\boldsymbol{\beta})$

2. To find the maximum likelihood solution, we can find the solution that maximizes the log of the likelihood function. Why can we do this step? Show that this problem becomes the same as solving equation (3).

3. Find the value of $\boldsymbol{\beta}$ that maximizes the log likelihood.

   Hint: You can use this link to help you with the calculus: http://www.cs.nyu.edu/~roweis/notes/matrixid.pdf. Remember, we are optimizing with respect to β.

2.2 Ridge Regression [10pts]

In order to prevent an overfit of the data, we can add an L2 penalty on the magnitude of the parameters that we are learning, to prevent them from growing arbitrarily. The Ridge Regression problem is formulated as:

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{4}$$

where $\lambda > 0$ is the regularization parameter.

1. Find the Ridge solution.

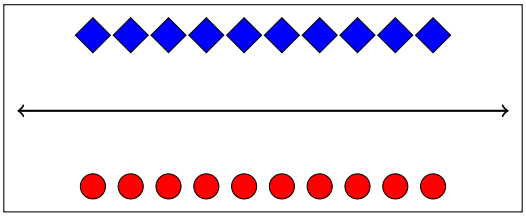
2. Now we will think about the problem in another way. Let $\boldsymbol{\beta} \sim \mathcal{N}(0, \lambda^{-1}\mathbf{I}_p)$ be a prior on the parameter vector $\boldsymbol{\beta}$. $\mathbf{I}_p$ is an identity matrix of size p. We will now find the maximum a posteriori estimate (MAP) of $\boldsymbol{\beta}$. This is achieved by finding the mode of the posterior distribution of $\boldsymbol{\beta}$. For simplicity, take $\sigma^2 = 1$.

   Write the posterior distribution for $\boldsymbol{\beta}$. Show that finding the MAP estimate is the same as solving equation (4).

3. Think of two potential problems with your solution for part 2.1. How does Ridge Regression fix them?

3 Classification [Dougal; 20 pts]

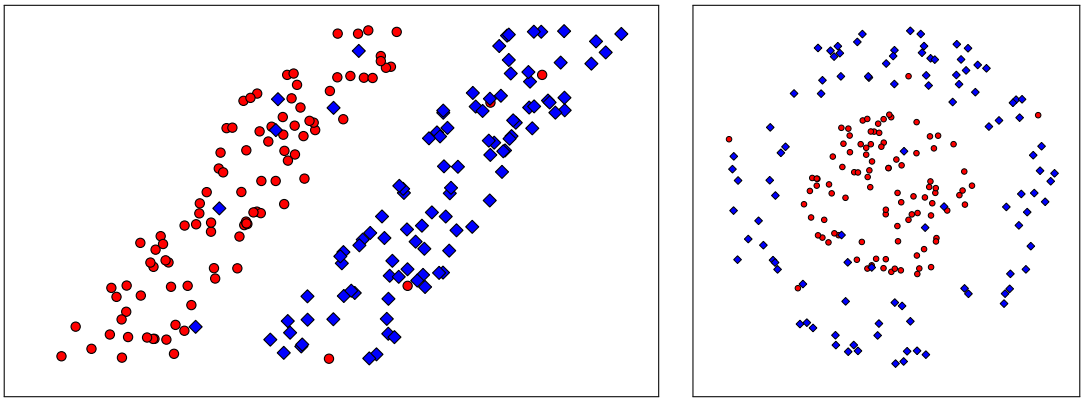
One of the most important aspects of a classifier is the type of functions that it can learn. A typical classifier will partition the input space into regions which are assigned positive values and regions which are assigned negative values; we call the boundary between those regions the decision boundary. For example, here is the decision boundary for a linear classifier on a trivial two-dimensional dataset, where blue diamonds represent one class and red circles the other:



Homework 1

3.1 Drawing decision boundaries [8 pts]

For both of the two-dimensional datasets below, sketch a decision boundary that each of the following classifiers might come up with and briefly describe the important features of the boundary: what made it choose that boundary, how good of a classifier it is, and anything else noteworthy. (We're not looking for a super-precise drawing here as long as you have it approximately right and mention any important aspects of why it looks that way.) Classifiers to use: (a) logistic regression or perceptron (say which one you chose), (b) Gaussian Naive Bayes (with separate variances as in 4.3), (c) 1-nearest-neighbor, and (d) 10-nearest-neighbor. Each dataset contains 200 total points, 100 from each of the two well-separated clusters; however, a few labels have been flipped due to noisy observation. Note that we're talking about applying each classifier to the raw features plotted here as Cartesian coordinates, not any nonlinear transformations such as operating in polar coordinates.



The images are available in the handout for this assignment, named `q3-1.png` and `q3-2.png`, for you to draw on more than once.

3.2 Defeating classifiers [12 pts]

Your friend just heard about this whole "machine learning" thing and read a few Wikipedia pages. They ask you, "Isn't this whole problem solved? Just use one of these easy classifiers for everything!" Rather than even bringing up that there's more to machine learning than classification of real-valued vectors, concerns about scalability, or anything else, you decide to show them wrong in style: by coming up with a "nice" dataset that their suggested classifier will completely fail on.

Draw and briefly describe three two-dimensional datasets with an obvious (to a human) decision boundary on which each of the following classifiers will do no better than chance. For our purposes here, a dataset means $n \geq 10$ points, with about half of them positive and half negative, and a clear way to make a similar dataset with any larger n; the classifier should get no more than approximately 50% accuracy for any n. (Somewhere around 51% accuracy for large n is fine, even if the improvement over chance is consistent rather than being due to random variation; 60% is not.) Classifiers to defeat: (a) logistic regression or perceptron, (b) Gaussian Naive Bayes (with separate variances as before), and (c) 1-nearest neighbor.

Hint: We showed in class that as the number of i.i.d. samples from a distribution increases to infinity, the error rate of the 1-NN classifier becomes no worse than twice that of the best possible classifier. That said, part (c) is still possible.

4 Coding Competition [Carlton; 40 pts + 5 Bonus]

This is a coding question. In this question you will implement the *k-Nearest Neighbor* and *Naive Bayes* algorithms in Octave. Octave is a free scientific programming language, with syntax (almost) identical to that of Matlab. Installation instructions can be found on the Octave website as linked above. If you've never used Octave or Matlab before, you may wish to check out this tutorial or this one.

For each sub question you will be given a single function signature, and asked to write a single Octave function which satisfies the signature.

This problem is automatically graded using the CMU Autolab system. The code which you write will be executed remotely against a suite of tests, and the results used to automatically assign you a grade. In order for your code to execute correctly on our servers you should avoid using libraries beyond the *basic* octave libraries.

The Autolab interface for this course is at https://autolab.cs.cmu.edu/10701-f13. You can sign in using your andrew credentials. You should make sure to edit your account information and choose a nickname/handle. This handle will be used to display your results for the challenge question on the class leaderboard.

- **Data** All questions will use the following datastructures: $XTrain \in \mathbb{R}^{n \times f}$ is a matrix of training data, where each row is a training point, and each column is a feature. $XTest \in \mathbb{R}^{m \times f}$ is a matrix of test data, where each row is a test point, and each column is a feature. $yTrain \in \{1, ..., c\}^{n \times 1}$ is a vector of training labels. $yTest \in \{1, ..., c\}^{m \times 1}$ is a (hidden) vector of test labels.
- **Submission Instructions** We have provided you with a single folder containing each of the functions you need to complete. *Do not modify the structure of this directory or rename these files*. Complete each of these functions, then compress this directory as a tar file and submit to autolab online. You may submit as many times as you like (up to the due date).

4.1 Logspace Arithmetic

When working with very small and very large numbers it is useful to work in logspace to avoid numerical precision issues. In logspace we keep track of the logs of numbers, instead of the numbers themselves. So in logspace e^2 is represented as 2, and e^3 is represented as 3. The challenge is to add and multiply these numbers *while remaining in logspace*. Note that if we exponentiate our numbers at any point in the calculation it complete defeats the purpose of working in log space. Hint: Alex has an excellent post on his blog about this topic:

1. **Logspace Multiplication [5 pts]**
   Complete the function $logProd(x)$ which takes as input a vector of numbers in logspace, and returns the product of these numbers in logspace. E.g. $logProd(x) = log(\prod_i \exp(x_i))$.
2. **Logspace Addition [5 pts]**
   Complete the function $logSum(x)$ which takes as input a vector of numbers in logspace, and returns the sum of these numbers in logspace. E.g. $logSum(x) = log(\sum_i \exp(x_i))$.

4.2 k Nearest Neighbor Classification

In this question you will implement the k Nearest Neighbor Classification algorithm. You can (and should) test your implementation locally using the *XTrain* and *yTrain* data provided. This is a real dataset called *Iris* from the UCI machine learning repository

1. **k Nearest Neighbor [5 pts]**
   Complete the function $knn(XTrain, XTest, k)$ which returns a $m \times k$ matrix D, where $D_{i,j}$ is the index of the jth nearest neighbor in $XTrain$ of the ith row of $XTest$.

2. **kNN Classification [5 pts]**
   Complete the function $nearestNeighborClassify(XTrain, XTest, yTrain, k)$ which returns a $m \times 1$ vector c of predicted class values, where c_i is the predicted class for the ith row of $XTrain$.

4.3 Naive Bayes

In this question you will implement the Gaussian Naive Bayes Classification algorithm. As a reminder, in the Naive Bayes algorithm we calculate $p(c|f) = p(f|c)p(c) = p(c)\prod_i p(f_i|c)$. In Gaussian Naive Bayes we learn a one-dimensional Gaussian for each feature in each class, i.e. $p(f_i|c) = N(f_i; \mu_{i,c}, \sigma^2_{i,c})$, where $\mu_{i,c}$ is the mean of feature f_i for those instances in class c, and $\sigma^2_{i,c}$ is the variance of feature f_i for instances in class c. Once again you should test your implementation locally using the *XTrain* and *yTrain* data provided.

1. **Prior [5 pts]**
   Complete the function $prior(yTrain)$ which returns a $c \times 1$ vector y, where y_i is the prior probability of class i.

2. **Likelihood [5 pts]**
   Complete the function $likelihood(XTrain, yTrain)$ which returns $[M, V]$. M is an $m \times c$ matrix where $M_{i,j}$ is the conditional mean of feature i given class j. V is an $m \times c$ matrix where $V_{i,j}$ is the conditional variance of feature i given class j.

3. **Naive Bayes Classifier [5 pts]**
   Complete the function $naiveBayesClassify(XTrain, XTest, yTrain)$ which returns a $m \times 1$ vector c of predicted class values, where c_i is the predicted class for the ith row of $XTrain$.

4.4 CHALLENGE: Non-Euclidean Distance [5 pts + 5 bonus]

This is a challenge question, and is much, much more difficult than previous questions. Furthermore this question is a Competition! There is a class leaderboard for this question on the autolab website, where students are identified by their chosen nickname/handle. Bonus points will be awarded to students who have top 10 classification accuracy on the class leaderboard.

In this question you will design a custom distance metric for use with the k Nearest Neighbor algorithm.

As the name suggests, the k Nearest Neighbor algorithm requires us to determine the k closest points in the training set for each point in the test set. However, which points are closest depends on which notion of distance we use.

Your task is to design a custom distance metric which gives good classification accuracy when used with the kNN classification algorithm on a new dataset. You are given two data files: `XTrainChallenge.mat` and `yTrainChallenge.mat`. Complete the function $myDist(XTrain, XTest)$ which returns an $m \times n$ matrix D, where $D_{i,j}$ is the distance between the ith element of $XTest$ and the j'th element of $XTrain$. We will use your distance function and the k nearest neighbor algorithm (with $k = 5$) to classify our hidden test set.

HINTS: Searching on google for "distance metrics" will return a dizzying array of different approaches, each best suited to slightly different data. There are the basic distance metrics L1, L2, hamming etc. There are psuedometrics such as KL divergence. There are approaches which attempt to align the data such as dynamic time warping. There are approaches which first transform the domain (Laplace Transform, Fourier

---

Transform, Cosine Transform) then apply a basic distance metric. I would strongly suggest you begin by plotting the data (using `plot(XTrain(1,:))`) to get a feel for what it looks like, then consider what sort of data it might be, and what distance metrics suit this type of data.

Furthermore this is real data, and real data is dirty! You may want to consider normalizing the data prior to computing distances via centering etc.