10-701/15-781 Machine Learning, Fall 2005

Assignment 2 DRAFT 2
Out: 9/27/05 **Due: beginning of class 10/06/05**

If you have questions, please contact Mike Stilman `<robot+ta@cmu.edu>`.

Linear Regression

1. (Noise in Linear Regression) [25 pts] Linear regression is applied when we assume that there is an underlying linear function $\mathbf{f}(x) = \mathbf{w}x + E$ that is generating data. Here, E is a random variable representing noise.

   In class you saw that selecting $\mathbf{w}$ to minimize the sum of squares of residual error yields the Maximum Likelihood $\mathbf{w}$. To prove this, we assumed that E is independent, normally distributed and of constant variance.

   Your classmate has some trouble believing that noise is normally distributed. He claims that the noise distribution should be bounded by some region $[-a, a]$ where p(E) is 0 outside these bounds and strictly greater than 0 within the bounds.

   (a) Lets look at a simple bounded distribution of noise:

   $$p(E) = \begin{cases} 1 & -\frac{1}{2} \leq E \leq \frac{1}{2} \\ 0 & \end{cases}$$

   In this case, least squares (L_2) regression will not necessarily maximize the likelihood of the data. Demonstrate an example where L_2 regression finds a $\mathbf{w}$ that is not MLE. (i.e. Present a univariate linear function and data points that do not violate the uniform error distribution. Then show that L_2 regression results in a suboptimal model.)

   (b) Your classmate accepts that uniform distributions are not necessarily a good model for noise. The bigger question, however, remains unresolved: Is there a bounded distribution for noise such that least squares regression always finds the MLE $\mathbf{w}$? Either define such a distribution and prove that L_2 regression must find the MLE $\mathbf{w}$ (for any valid linear function/data points), or prove that no such distribution exists. (Visual arguments are acceptable, but they must be supported by clear and correct logical reasoning.)

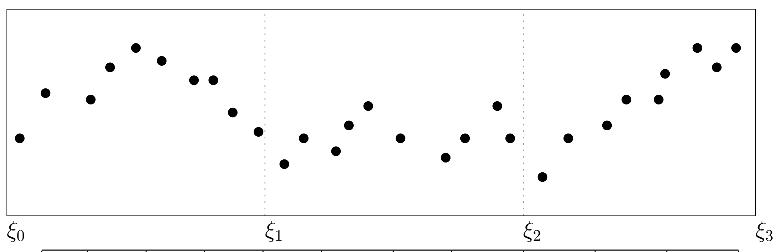
2. (Interpolating with Regression) [20 pts] One fun application of regression is interpolation. Basic linear regression fits a line to points. By introducing basis functions we were able to model polynomials.

   Some functions, however, are quite complex and could require very high order polynomials to achieve accurate modeling. Often it is better to split these functions into segments at some number of points that we call knots.

   In the following exercises you may use these functions to simplify your notation:

   $$I(a, b, x) = \begin{cases} 1 & a \leq x < b \\ 0 & otherwise \end{cases} \qquad G(x) = \begin{cases} 1 & 0 < x \\ 0 & otherwise \end{cases}$$

(a) We have split the following data-set into three evenly spaced regions at knots ξ_1 and ξ_2 (where $\xi_{i+1} - \xi_i = 160$). The data set represents some function $y = \mathbf{f}(x) + E$. Suppose we applied linear regression to each segment separately. Draw a rough sketch of the resulting model function.



x	8	24	52	64	80	96	116	128	140	156
y	48	76	72	92	104	96	84	84	68	52

x	184	204	212	224	244	272	284	304	300
y	48	40	56	68	48	36	48	68	52

x	344	348	372	384	404	408	428	432	452
y	20	48	56	68	72	68	104	88	104

(b) Your model should be discontinuous. It is possible to achieve the same model with a single linear regression of the form:

$$y' = \sum_{i=1}^{m} \beta_i h_i(x)$$

Find a set of six (6) basis functions, $h_i(x)$, such that their linear combination can represent any piecewise linear function with discontinuities at the knots ξ. Find the parameters β_i that satisfy the maximum likelihood model for the given data.

(c) Generally, we do want our model y' to be continuous. Find a set of four (4) basis functions, $h_i(x)$, such that $y' = \sum_{i=1}^{m} \beta_i h_i(x)$ can represent any continuous piecewise linear function that has discontinuous derivatives only at the knots. Estimate β_i and formulate a new y' that minimizes the squared residual error.

Theoretical Analysis of Logistic Regression and Naive Bayes

[20 pts] In class and in Tom's draft chapter handout we showed that when Y is Boolean and $X = (X_1 \ldots X_n)$ is a vector of continuous variables, then the assumptions of the Gaussian Naive Bayes classifier imply that $P(Y|X)$ is given by the logistic function with appropriate parameters W. In particular:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

and

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Consider instead the case where Y is Boolean and $X = (X_1 \ldots X_n)$ is a vector of *Boolean* variables. Prove for this case also that $P(Y|X)$ follows this same form (and hence that logistic regression is also the discriminative counterpart to a Naive Bayes generative classifier over Boolean features).

Hints:

1. Simple notation will help. Since the X_i are Boolean variables, you need only one parameter to define $P(X_i|Y = y_k)$. Define $\theta_{i1} \equiv P(X_i = 1|Y = 1)$, in which case $P(X_i = 0|Y = 1) = (1 - \theta_{i1})$. Similarly, use θ_{i0} to denote $P(X_i = 1|Y = 0)$.
2. Notice with the above notation you can represent $P(X_i|Y = 1)$ as follows
$$P(X_i|Y = 1) = \quad \theta_{i1}^{X_i} \quad (1 - \theta_{i1})^{(1-X_i)}$$
Note when $X_i = 1$ the second term is equal to 1 because its exponent is zero. Similarly, when $X_i = 0$ the first term is equal to 1 because its exponent is zero.

Programming Generative and Discriminative Classifiers

Discriminative classifiers learn the parameters of $P(Y|X)$ directly, whereas generative classifiers instead learn the parameters of $P(X|Y)$ and $P(Y)$.

In this exercise you are asked to implement and compare both types of classifiers. You may use any programming language you like (Matlab, C++, C, Java...). *All programming must be done from first principles. You are only permitted to use existing tools for simple linear algebra such as matrix multiplication/inversion. Do NOT use any toolkit that performs machine learning functions.*

For this assignment, please submit all answers and any plots that are requested in the following questions. Also, *print out* and clearly label any code you wrote for this assignment and append it to the back of your submission. We do not require comments, however the clarity of your code and explanations will affect how much partial credit we can give. We encourage you to discuss the questions, but you must write/submit your own code and your own answers.

The provided data has two real variables X_1, X_2 and the boolean variable Y representing a class. Each line in the data files represents a data point (X_1, X_2, Y).

Make a 2D plot (X_1, X_2) of "test-1", using different symbols for data points that belong to different classes.

1. (Naive Bayes) [9 pts] You may notice that given the class (Y), the data follows a bi-variate normal distribution. For Naive Bayes, we make the assumption that X_1 is conditionally independent of X_2 given the class.
   (a) Write the equation for the probability of some data point coming from class $Y = 1$ i.e. $P(Y = 1|X_1 = x_1, X_2 = x_2)$.
   (b) Write a program that estimates the means and variances of the Gaussians, as well as P(Y) for each class. This program should yield all the information necessary to complete your equation in (a). Run this program on "train-1."

(c) Part of your program should estimate the two normal distributions that maximize the probability of the data. Submit one plot that shows the mean and the two standard deviation iso-contour for each Gaussian.

(d) Write a program that uses the parameters learned from "train-1" to classify the points in "test-1". Report your test error.

2. (Logistic Regression) [9 pts] Now, let us not make the conditional independence assumption and classify with Logistic Regression. For this exercise, use gradient descent as presented in (Ch. 3.2) of Tom's handout.

   (a) Write a program that optimizes the weights $w0, w1$ and $w2$ to construct a logistic regression model of the data. In your program, set the step size $\eta = 1 \times 10^{-6}$ and fix the number of iterations to 20000. (Perform exactly this number of iterations whether or not gradient descent converges).

   (b) Train your program on "train-1" and report the learned weights.

   (c) Write a program that uses the linear regression model to predict the class of the data based on observed (X_1, X_2). Test it on "test-1". Report your test error.

3. (Comparisons) [15 pts] Let us call Logistic Regression LR and Naive Bayes NB. So far, you have used the entire training set to train these classifiers. Suppose that less data was available. Limit your classifier to training on subsets of the provided training data. (e.g. 500, 1000, 1500 ... 10000.) Retrain LR and NB on subsets of "train-1" data and observe their performance on "test-1". To reduce needless variance in your experiment, be sure to use the same training data subsets (not just random subsets of the same size) to train both LR and FB.

   (a) Construct a single plot showing the accuracy of each classifier as it depends on the number of training examples.

   (b) What do you notice about the relationship between classifier accuracy and the number of available data points?

   (c) Make a short list of observations about LR and NB that you have made throughout this exercise. Using knowledge from class/experience in programming and understanding of the assumptions briefly describe the causes for the differences you observe. (1-2 sentences per observation).

4. (Full Bayes) [2 pts - Only if you have free time] Without making the conditional independence assumption, it is still possible to train a Bayesian classifier. Suppose that X_1 is not conditionally independent of X_2 given the class (i.e. you can't assume $p(X_1|X_2, Y) = p(X_1|Y)$).

   (a) Write the equation for the probability of some data point coming from class $Y = 1$ i.e. $P(Y = 1|X_1 = x_1, X_2 = x_2)$.

   (b) Duplicate and modify your Naive Bayes classifier to reflect this changed formula. Train your classifier on "train-1." It should learn $P(Y)$, the means and the full covariance matrices for the Gaussians.

   (c) Plot the mean and 2 SD iso-contour for each Gaussian. How does your plot compare to Naive Bayes?

   (d) Test your classifier on "test-1." What do you notice?