CSI 5325 Assignment 2

Greg Hamerly

Assigned: 2/6/2015; Due: 3/3/2015

Instructions

For this assignment, you should write your derivations in LaTeX. They should be concise and easy to follow, with appropriate English descriptions rather than just mathematics (i.e. explain what you are doing as necessary). The goal of the assignment is not just to answer the given questions correctly, but to explore the questions, get some answers, and explain your answers well.

Submit your assignment in two ways: hardcopy (in class) and by email (to hamerly@cs.baylor.edu). The email should contain a single attachment as a ZIP file. It should be named "lastname-xx.zip", where lastname is your last name, and xx is the number of the assignment.

Include any code you write with your electronic submission. Unless otherwise specified, write programs in Matlab (or Octave). Include any relevant figures and graphs (preferably plotted using Matlab / Octave) in your writeup.

Finally, please keep your submitted email attachments small. In particular, make sure you are only submitting things that are necessary (omit datasets I gave you, compiled programs, etc.). Also, try to keep your graphics small by using vector (rather than bitmap) formats (e.g. PDF or EPS rather than JPG or BMP). Vector graphics are generally smaller in size and better quality than bitmap.

1 Ridge regression (20 points)

For typical least-squares regression, the cost function is

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

for $X \in \mathbb{R}^{m\times n}$ and $y \in \mathbb{R}^m$. We minimized this function in matrix notation by taking the gradient, setting it equal to zero, and arriving at the normal equation:

$$X^T X\theta = X^T y$$

which led us to the least-squares estimate (also the MLE) of θ:

$$\theta = (X^T X)^{-1} X^T y$$

However, there are circumstances in which we the least-squares estimate is less desirable. One situation is when two features x_i and x_j are collinear ($x_i = \alpha x_j$), which can cause the estimates of θ_i and θ_j to grow without bound (as long as $\theta_j = -\alpha\theta_i$ the terms cancel when calculating $\theta^T x$).

An approach to fixing this problem is to penalize large components of θ, by adding a *regularization term* as follows:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2}\theta^T\theta$$

where λ is a parameter we must choose. Clearly, $\lambda = 0$ gives the standard least-squares cost function. This particular penalization method is known as *ridge regression*. We will talk more about regularization later in the semester.

(a) Using vector notation, find a closed-form solution for the θ that minimizes the ridge regression cost function.

(b) Implement ridge regression using the functions `hwk_ridge.m` and `hwk_zscore.m` on the prostate data provided. Show the effect of the parameter λ on three things:

- the values of the individual components of θ,
- the non-regularized cost function on held-out test data, and
- the value of $\theta^T\theta$.

Use three different graphs to show this, where each graph has λ as the ordinate. Try a wide range of values of λ. Describe what you observe. What conclusions can you draw from this experiment? Note that this dataset does not necessarily demonstrate the collinearity of features mentioned above; however, it is still interesting to explore with ridge regression.

2 Gaussian Discriminant Analysis (30 points)

Recall multivariate Gaussian Discriminant Analysis for classification, where we model the input data explicitly as:

$$p(x|y=0;\mu^0,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu^0)^T\Sigma^{-1}(x-\mu^0)\right)$$
$$p(x|y=1;\mu^1,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu^1)^T\Sigma^{-1}(x-\mu^1)\right)$$
$$p(y=1;\phi) = \phi$$

where Σ is a shared covariance matrix among all classes, and μ^0 and μ^1 are respectively the mean vectors for each class. For this problem, we will assume all vectors x, μ^0, μ^1 are $n \times 1$ column vectors.

(a) Prove that for a given dataset $(x^{(i)}, y^{(i)})$, $1 \le i \le m$, the MLEs for the parameters μ^0 and ϕ are

$$\hat{\mu^0} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$
$$\hat{\phi} = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)}=1\}$$

Note that the MLE for μ^1 has the identical form to that for μ^0, so we won't prove that explicitly. The MLE for Σ is $\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}-\mu^{y^{(i)}}\right)\left(x^{(i)}-\mu^{y^{(i)}}\right)^T$, but you don't have to prove or use this fact for this part of the problem.

Start with the joint likelihood $L(\mu^0, \mu^1, \Sigma, \phi)$, find the log-likelihood, and then take the partial derivatives with respect to each parameter to find the MLEs. It may be helpful to keep in mind these identities for matrix calculus on vectors x and a and matrix A:

$$\frac{\partial}{\partial x} x^T a = \frac{\partial}{\partial x} a^T x = a$$
$$\frac{\partial}{\partial x} x^T A = A^T$$
$$\frac{\partial}{\partial x} x^T A x = x^T (A + A^T) \qquad \text{(what if } A = A^T\text{?)}$$

(b) Prove that for two-class Gaussian Discriminant Analysis, with common covariance Σ, the posterior probability $p(y = 1|x; \mu^0, \mu^1, \phi, \Sigma)$ is a logistic sigmoid function. Recall the general form of the logistic sigmoid is $g(z) = 1/(1 + \exp(-z))$, so show that $p(y = 1|x; \mu^0, \mu^1, \phi, \Sigma) = g(\theta^T x)$ for some appropriately chosen θ. You may want to use Bayes' rule as a starting point. Also, recall that we typically define an intercept feature $x_0 = 1$.

(c) Implement Gaussian discriminant analysis in the files `hwk_gda.m` and `hwk_gda_predict.m` for use on the Vowel dataset. You'll implement the MLEs for ϕ, μ, and Σ, as well as an analysis of the prediction accuracy using a confusion matrix.

A confusion matrix C for a k-class problem is a $k \times k$ matrix where C_{ij} represents the number of times true class i was predicted as class j. Therefore, the number of correct predictions is along the diagonal, and all non-diagonal entries are counts of incorrect predictions.

Compare the accuracy you find on the training and test sets. How does the accuracy compare to the base rate? (The base rate is the accuracy you would achieve just by predicting the most frequently-occurring class all the time.)

3 Optimal margin classifier (10 points)

The optimal margin classifier is the one which maximizes the geometric margin separating examples from the two classes. This is defined as:

$$\min_{w,b} ||w||^2$$
$$\text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1 \text{ for all } i$$

where (w, b) are the parameters of the hyperplane. The weight vector w is orthogonal to the decision plane, and b is the bias term which allows the hyperplane to be away from the origin.

(a) Prove that w is orthogonal to the decision hyperplane (the hyperplane is the set of points $\{x|w^T x + b = 0\}$). Recall that two vectors u and v are orthogonal iff $u^T v = 0$.

(b) The equation $w^T x + b = 0$ forms a *single* hyperplane in the feature space of the variables x, with normal vector w. However, when thinking about minimizing the objective $w^T w$ subject to the m constraints $y^{(i)}(w^T x^{(i)} + b) \geq 1$, we said that each constraint forms a (different) hyperplane that separates the feasible and non-feasible space of solutions. Explain why each constraint's shape is a hyperplane, and why each of the m constraints makes a different hyperplane (when we have a single hyperplane for the decision boundary).

4 Naive Bayes and SVMs for Text Classification (20 points)

In this question you'll look into the Naive Bayes and Support Vector Machine algorithms for a spam classification problem. However, instead of implementing the algorithms yourself, you'll use a freely available machine learning library. There are many such libraries available, with different strengths and weaknesses, but for this problem you'll use the WEKA machine learning package, available at `http://www.cs.waikato.ac.nz/ml/weka/`. WEKA implements many standard machine learning algorithms, is written in Java, and has both a GUI and a command line interface. It is not the best library for very large-scale data sets, but it is very nice for playing around with many different algorithms on medium size problems.

You can download and install WEKA by following the instructions given on the website above. To use it from the command line, you first need to install a java runtime environment, then add the `weka.jar` file to your `CLASSPATH` environment variable. Finally, you can call WEKA using the command:

```
java <classifier> -t <training file> -T <test file>
```

For example, to run the Naive Bayes classifier (using the multinomial event model) on our provided spam data set by running the command:

```
java weka.classifiers.bayes.NaiveBayesMultinomial -t spam_train_1000.arff -T spam_test.arff
```

The spam classification dataset was provided courtesy of Christian Shelton (cshelton@cs.ucr.edu). Each example corresponds to a particular email, and each feature correspondes to a particular word. For privacy reasons we have removed the actual words themselves from the data set, and instead label the features generically as f1, f2, etc. However, the data set is from a real spam classification task, so the results demonstrate the performance of these algorithms on a real-world problem. There are actually several different training files, named `spam_train_50.arff`, `spam_train_100.arff`, etc. (the ".arff" format is the default format by WEKA), each containing the corresponding number of training examples. There is also a single test set `spam_test.arff`, which is a hold out set used for evaluating the classifier's performance.

(a) Run the `weka.classifiers.bayes.NaiveBayesMultinomial` classifier on the dataset and report the resulting error rates. Evaluate the performance of the classifier using each of the different training files (but each time using the same test file, spam_test.arff). Plot the error rate of the classifier versus the number of training examples.

(b) Repeat the previous part, but using the `weka.classifiers.functions.SMO` classifier, which implements the SMO algorithm to train an SVM. How does the performance of the SVM compare to that of Naive Bayes?

(c) Consider the issue of estimating the multinomial probability of seeing a word w by using a training set with m examples. If the training contains $b(w)$ occurrences of w, then the MLE is $P(w) = b(w)/m$. If $b(w) = 0$, then $P(w) = 0$. The Laplace-smoothed estimate will make sure this probability is not zero by calculating instead $P(w) = (b(w) + 1)/(m + d)$, where d is the number of unique words.

Thus, Laplace smoothing raises the probability of words that would have zero probability. How does Laplace smoothing affect the estimated probabilities of other words in the dictionary? Be mathematically precise.