CSCI 1950-F Homework 5: Logistic Regression

Brown University, Spring 2011

Homework due at 12:00pm on March 11, 2011

Question 1

In this question, we consider a continuous estimation problem in which the input x and response variable y are both real numbers. Their joint probability density function $p(x, y)$ is constant within the closed, shaded region shown in Figure 1, and zero elsewhere.

a) For any estimator $\hat{y}(x)$, consider the quadratic loss function $L(\hat{y}(x), y) = (\hat{y}(x) - y)^2$. Determine the estimator that minimizes the following posterior expected loss:

$$\mathrm{E}[L(\hat{y}(x), y) \mid x] = \int_{-\infty}^{\infty} L(\hat{y}(x), y)\, p(y \mid x)\, dy$$

Simplify the form of your answer as much as possible.

b) Consider how the estimator $\hat{y}(x)$ from part (a) behaves for three particular input variables: $x = -1.5$, $x = 0$, and $x = 1.5$. In each case, is $\hat{y}(x)$ also a *maximum a posteriori* (MAP) estimator? Why or why not?

c) Suppose that rather than knowing the true density function $p(x, y)$, we instead have N training examples $\{(x_i, y_i)\}_{i=1}^N$, where (x_i, y_i) are independent samples from $p(x, y)$. Consider the following linear regression model:

$$p(y \mid \boldsymbol{w}, x, \beta) = \mathrm{Normal}(y \mid \boldsymbol{w}^T \phi(x), \beta^{-1})$$

Suppose that we choose $\phi(x) = [1, x]^T$ as our basis or feature functions, and estimate $\hat{\boldsymbol{w}}$ via maximum likelihood. As $N \to \infty$, will the expected loss of the prediction function $\hat{y}_{\hat{\boldsymbol{w}}}(x) = \hat{\boldsymbol{w}}^T \phi(x)$ be lower than, higher than, or equal to that of the estimator from part (a)? Justify your answer.

d) Consider again the linear regression model from part (c), but with alternative features $\phi'(x) = [1, |x|]^T$. As $N \to \infty$, will the expected loss of $\hat{y}'_{\hat{\boldsymbol{w}}}(x) = \hat{\boldsymbol{w}}^T \phi'(x)$ be lower than, higher than, or equal to that of the estimator from part (a)? Justify your answer.

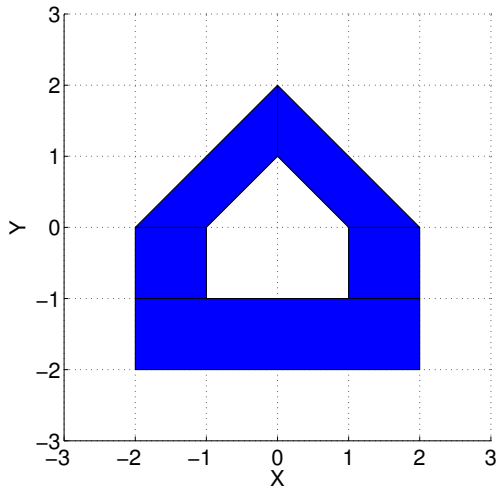

Figure 1: Joint probability density function $p(x, y)$ for question 1. The density is constant (uniform) in the shaded region, and zero elsewhere.

Question 2:

This problem investigates multinomial logistic regression classifiers. Let $y \in \{1, \ldots, C\}$ denote the discrete class label we want to predict, x the input or conditioning variables, and $\phi(x) \in \mathbb{R}^m$ a fixed (possibly non-linear) feature function. One form of the multinomial logistic regression model is then

$$p(y = c \mid x, \boldsymbol{w}) = \frac{\exp(w_c^T \phi(x))}{\sum_{c'=1}^{C} \exp(w_{c'}^T \phi(x))},$$

where $w_c \in \mathbb{R}^m$ are the feature weights associated with class c, and $\boldsymbol{w} \in \mathbb{R}^{Cm}$ is a vector concatenating the weights for all classes.

This parameterization is not identifiable: coordinated translations of the weight vectors w_c for different classes can produce an identical model. To avoid this ambiguity and improve robustness, we place a zero-mean, diagonal-covariance Gaussian prior on the weight vector:

$$p(\boldsymbol{w}) \propto \exp\left(-\frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w}\right)$$

Here, α is a tunable parameter that controls the degree of regularization.

a) *Give an expression for the regularized negative log conditional likelihood of a generic data set $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, ignoring any terms and factors that do not depend on $\boldsymbol{w}$. Also give an expression for the derivative of the regularized negative log conditional likelihood with respect to a particular feature weight w_{ck}. A detailed derivation is not necessary, but be sure your notation is clear.*

We now examine the Nursery data set from homework 1, available at

`/course/cs195f/asgn/naive_bayes/handout/nursery/nursery.mat`.

You should divide this data into equal-sized training, development and testing data sets as follows (the `reset` ensures that we'll all use the same training/validation/test split):

```
load('/course/cs195f/asgn/naive_bayes/handout/nursery/nursery.mat');
reset(RandStream.getDefaultStream)
data = data(randperm(size(data,1)),:);
train = data(1:size(data,1)/3,:);
dev = data(size(data,1)/3+1:2*size(data,1)/3,:);
test = data(2*size(data,1)/3+1:end,:);
```

In this data set, each conditioning variable x_i is in fact a vector with $m = 8$ discrete components or attributes, $x_{ij} \in \mathcal{X}_j$, $j = 1, \ldots 8$. You should encode these via the following feature map $\phi(x_i) \in \mathbb{R}^m$, where $m = 1 + \sum_{j=1}^{8} |\mathcal{X}_j|$:

- The first component should be a constant bias or offset term, $\phi_1(x_i) = 1$.
- For each $j = 1, \ldots, 8$, include $|\mathcal{X}_j|$ binary features. Set these so that for each training example x_i, exactly one is on (corresponding to the discrete value of x_{ij}) and the remainder off.

Given this data and the objective from part (a), we will use the `minFunc` function from the pmtk3 Matlab toolbox (http://code.google.com/p/pmtk3/) to find the weight vector $\boldsymbol{w}$ that minimizes the regularized negative log conditional likelihood on the training data. You should provide the optimizer with the gradient of the regularized negative log conditional likelihood, and write your *own* function to compute the objective and its gradient. We suggest calling this function as follows:

```
options.MaxIter = 500;  options.numDiff = 0;  options.Display = 'iter';
argminw = minFunc(myfun, w0, options);
```

Here, `w0` is an initial guess at the weight vector (a vector of all zeros is fine). `myfun` is a function that you've written that takes a weight vector `w` as its argument and returns a pair of values: the regularized negative log conditional likelihood, and a vector of its derivatives. You may find it useful to pass parameters to `myfun` by using a *function handle*:

```
argminw = minFunc(@(u) myfun(u, e1, e2,...), w0, options);
```

where `e1`, `e2`, etc., are variables whose values you want to pass as arguments to `myfun`.

For all of the following questions, do all calculations for 30 logarithmically-spaced values of α between 10^{-6} and 10 as follows:

```
alpha = logspace(-6,1,30);
```

There's some sample code to get you started at `/course/cs195f/asgn/logistic`.

b) Plot the negative log conditional likelihood of the `train` *data as a function of* α *when training and evaluating on* `train`*.*

c) Plot of the accuracy of the classifier as a function of α *when training and evaluating on* `train`*.*

d) Plot the negative log conditional likelihood of the `dev` *data as a function of* α*. That is, for each value of* α *estimate the feature weight* $\boldsymbol{w}$ *from the* `train` *data, and then calculate the negative log conditional likelihood of the* `dev` *data for that value of* $\boldsymbol{w}$*. Find a value of* α *which minimizes the negative log conditional likelihood of the* `dev` *data.*

e) For the weight vectors $\boldsymbol{w}$ *corresponding to the value of* α *identified in part (d), evaluate the corresponding classifier on* `test` *and report your results.*

f) Train a naive Bayes classifier on this `train` *data. Use the Bayesian parameter estimator, with a uniform prior, from homework 1. What is its performance on* `test`*? Compare to the logistic regression model.*

Question 3:

This question uses synthetic data to compare the properties of logistic regression and linear regression for classification. Each data item has two continuous features $x \in \mathbb{R}^2$ and is labeled as one of either $K = 2$ or $K = 3$ classes. The data are stored as .mat files here:

`/course/cs195f/asgn/logistic/toy`

Your linear regression code from homework 4 can be adapted for classification as follows. For K classes, each response is encoded as a row vector $Y_i = [Y_{i1}, \ldots, Y_{iK}]$, where $Y_{ik} = 1$ for an example of class k, and zero otherwise. For N data samples we define the $N \times K$ matrix Y as a matrix of 0's and 1's, with each row having a single 1. We fit a linear regression model to each of the columns of Y as follows:

$$\hat{Y} = \Phi(\Phi^T\Phi)^{-1}\Phi^T Y$$

Here, Φ is the $N \times 3$ model matrix of corresponding to the feature function $\phi(x_i) = [1, x_{i1}, x_{i2}]^T$, i.e. the raw 2D input data augmented by a constant bias feature. The weights corresponding to the least squares prediction above equal

$$\hat{W} = (\Phi^T\Phi)^{-1}\Phi^T Y$$

Here, $\hat{W}$ is a $3 \times K$ matrix where the k^{th} column $\hat{w}_k$ represents the linear regression fit for class k. Finally, we can use this linear regression model to classify a new observation as

$$\hat{y}(x) = \arg\max_k \ \phi(x)^T \hat{w}_k$$

The supplied function `plotClassifier` can be used to visualize decision boundaries.

We compare the performance of this linear least squares classifier to a multinomial logistic regression classifier, both using the same features. To fit multinomial logistic regression models, use your implementation from question 2, with a small but nonzero regularization constant $\alpha = 10^{-8}$ to ensure identifiability.

a) *The first dataset contains two classes which lie in well-separated clouds. Implement the least squares classifier described above. Estimate weights from training data, and plot the learned decision boundary together with the training points. If implemented correctly, your test accuracy should be 100%. Is this the case?*

b) *The second dataset contains three classes with means arranged in a triangular pattern. Train a least squares classifier as above, as well as a multinomial logistic regression classifier using the same features. Plot the training decision boundaries for both classifiers, and report test accuracy for each. Explain any performance differences.*

c) *The third dataset contains three classes with means arranged in a straight line. Train a least squares classifier as above, as well as a multinomial logistic regression classifier using the same features. Plot the training decision boundaries for both classifiers, and report test accuracy for each. Explain any performance differences.*