10-601 Machine Learning, Fall 2009: Homework 4

Due: Wednesday, October 28th, 10:30 am

Instructions There are 2 questions on this assignment worth a total of 100 points. Please hand in a hard copy at the beginning of the class. Please print your code and attach it to the write-up. Refer to the webpage for policies regarding collaboration, due dates, and extensions.

1 Naive Bayes and Logistic Regression [35 points]

1.1 Redundant feature [15 points]

Consider the classification task with Boolean labels, Y, taking values T or F, and three Boolean features, X_1, X_2 and X_3. The features X_1 and X_2 are conditionally independent given Y and while the feature X_3 is a copy of X_2 (i.e., X_3=X_2 always). The conditional probabilities are given by:

$$\begin{aligned} \mathbf{P}(X_1 = T \,|\, Y = T) &= p \\ \mathbf{P}(X_1 = T \,|\, Y = F) &= 1 - p \\ \mathbf{P}(X_2 = F \,|\, Y = T) &= q \\ \mathbf{P}(X_2 = F \,|\, Y = F) &= 1 - q \\ \mathbf{P}(Y = T) = \mathbf{P}(Y = F) &= 0.5 \end{aligned}$$

You are given a test example with the feature values: $X_1 = T$ and $X_2 = X_3 = F$. We would like to classify this example by predicting the value of Y.

1. Show that if the Naive Bayes assumption is used, that is features are conditionally independent of each other given class labels Y, the decision rule for classifying the test example as $Y = T$ given its feature values is: $p \geq \frac{(1-q)^2}{q^2+(1-q)^2}$

2. What would be the optimal decision rule? (The optimal decision rule is the rule when we do not use the Naive Bayes assumption, but use only the given conditional independence assumptions.) Express this decision rule in terms of p and q.

3. Plot the two decision rules you derived in 1.1.1 and 1.1.2. The x-axis should be q and y-axis should be p both varying in [0,1]. Indicate for which regions of the graph the test example is classified as $Y = T$ and $Y = F$. Show on the graph (shade and label the areas) where the Naive Bayes decision rule makes mistakes relative to the optimal decision rule.

1.2 Equivalence of NB and LR [10 points]

In section 3 of the Mitchell et al. reading, http://www.cs.cmu.edu/%7Etom/mlbook/NBayesLogReg.pdf, it is shown that if Y is Boolean and $X = \langle X_1 \ldots X_n \rangle$ is a vector of continuous variables, the form of $\mathbf{P}(Y \,|\, X)$

entailed by the assumptions of a Gaussian Naive Bayes (GNB) classifier is precisely the form used by Logistic Regression (LR) with appropriate parameters W. In particular:

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

and

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

Now consider the case where Y is Boolean and $X = \langle X_1 \dots X_n \rangle$ is a vector of binary variables instead of continuous variables. Prove for this case also $\mathbf{P}(Y \mid X)$ follows the same form (and hence that Logistic Regression is also the discriminative counterpart to a Naive Bayes generative classifier over Boolean features).

Hints:

1. Using simple notation will help you in the derivation. Since the X_i are Boolean variables, only one parameter is needed to define $\mathbf{P}(X_i \mid Y = y_k)$. Define $\theta_{i1} \equiv \mathbf{P}(X_i \mid Y = 1)$, in which case $\mathbf{P}(X_i = 0 \mid Y = 1) = 1 - \theta_{i1}$. Similarly, use θ_{i0} to denote $\mathbf{P}(X_i = 1 \mid Y = 0)$.

2. Notice with the above notation you can represent $\mathbf{P}(X_i \mid Y = 1)$ as follows

   $$\mathbf{P}(X_i \mid Y = 1) = (\theta_{i1})^{X_i}(1 - \theta_{i1})^{(1 - X_i)}$$

   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.

1.3 Relaxing the conditional independence assumption [10 points]

To capture interactions between features, the Logistic Regression model can be supplemented with extra terms. For example, a term can be added to capture a dependency between X_1 and X_2:

$$\mathbf{P}(Y = 1 \mid X) = \frac{1}{1 + \exp(w_0 + w_{1,2} X_1 X_2 + \sum_{i=1}^{n} w_i X_i)}$$

Similarly, the conditional independence assumptions made by Naive Bayes can be relaxed so that X_1 and X_2 are not assumed to be conditionally independent. In this case, we can write:

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(Y)\,\mathbf{P}(X_1, X_2 \mid Y) \prod_{i=3}^{n} \mathbf{P}(X_i \mid Y)}{\mathbf{P}(X)}$$

Prove that for this case, that $\mathbf{P}(Y \mid X)$ follows the same form as the logistic regression model supplemented with the extra term that captures the dependency between X_1 and X_2 (and hence that the supplemented Logistic Regression model is the discriminative counterpart to this generative classifier).

Hints:

1. Using simple notation will help here as well. You need more parameters than before to define $\mathbf{P}(X_1, X_2 \mid Y)$. Define $\beta_{ijk} \equiv \mathbf{P}(X_1 = i, X_2 = j \mid Y = k)$.

2. The above notation can be used to represent $\mathbf{P}(X_1, X_2 \mid Y = k)$ as follows

   $$\mathbf{P}(X_1, X_2 \mid Y = k) = (\beta_{11k})^{X_1 X_2}(\beta_{10k})^{X_1(1 - X_2)}(\beta_{01k})^{(1 - X_1)X_2}(\beta_{00k})^{(1 - X_1)(1 - X_2)}$$

2 Hot Eigenfaces for Gaussian Naive Bayes [65 Points]

In this problem you will have the opportunity to apply PCA to a moderately sized collection of images scraped from the *Hot-or-Not*[1] website and then use a Naive Bayes classifier to learn to classify facial attractiveness.

[1] The images were collected by Ryan White from the Hot-or-Not website http://www.hotornot.com/. To learn more about how the data was collected, get the color images, and try your algorithms on the male face data, go to http://www.ryanmwhite.com/research/tr_hot.html.

For this problem we have selected only female faces because they are more consistently labeled and are typically more amenable to basic classifiers. The data for this problem is available at http://www.cs.cmu.edu/~ggordon/10601/hws/hw4/faces.mat. The *faces.mat* Matlab file contains the following data:

h: The height of each image.

w: The width of each image.

tr and te: These structs contain the training and test data respectively. The fields in these structs are:

n: The number of images.

hot: A vector with `n` elements. The entry `hot(i)` is 1 if the i^{th} image is hot and 0 otherwise.

images: An `n` by `w * h` matrix where each row corresponds to a separate image. To view the j^{th} image in matlab you would use the following code:

```
load('faces.mat');
figure(1);
j = 3;
% Reshape the row vector into an 86 by 86 pixel image
img = reshape(tr.images(j, :), h, w);
% Show the image
imagesc(img);
% Set the plot to gray-scale
colormap('gray');
```

Use only the data in the `tr` struct for all training and validation. The data in `te` will only be used in the last part to evaluate your classifier. We will explicitly say when to use `te`. Don't use `te` until we tell you.

2.1 Basic Analysis [15 Points]

We will begin with some basic data analysis to become more familiar with the data.

1. What fraction of the faces are hot?
2. Using `subplot(1,2,i)` plot both the average hot and not hot faces side by side. To compute the average face simply use the `mean` function on the respective class of face vectors (i.e., `mean(tr.images(tr.hot,:))`).
3. Based on the average faces, does it seem like their may be sufficient difference to build a classifier for facial attractiveness? Provide a *very* brief explanation.
4. Compute the covariance between the pixels (i.e., `cov(tr.images)`). Plot the distribution of the values for the off-diagonal terms. Plot the distribution of the diagonal (variance) terms. (That is, make histograms of the given sets of numbers.)
5. Based on the above observations, what can you say about the dependence/independence of pixels in these images? Give a *brief* justification why neighboring pixels might be correlated.

2.2 Singular Value Decomposition [15 Points]

For this question you will need to use the matlab `svds` function to compute the top 100 eigenvectors. Please attach the code you write for this section to your solution set.

1. Compute the average face `xbar` using the `mean` function. You will need `xbar` throughout the rest of this problem. Center the faces data by subtracting `xbar` from each row of `tr.images` storing the result in `Xcenter`. Run `svds` on `Xcenter` (i.e., `[U, svalues, efaces] = svds(Xcenter, 100);`). The columns of `efaces` are the eigenfaces and the diagonal entries of `svalues` are the singular values. Please include the code you used for the past few steps.

2. Plot the singular values (i.e., `plot(diag(svalues))`).

3. Do the singular values decrease rapidly?

4. What does this suggest about the approximation quality obtained by representing faces as a linear combination of the top few eigenfaces?

5. Plot the top 10 eigenfaces (preferably using `subplot(2,5,i)` to save paper).

6. For two of the eigenfaces provide a *brief* interpretation of what aspect of the images they might encode (i.e., "lighting", "eyes", ...).

2.3 Dimensionality Reduction [15 Points]

We will now use the eigenfaces computed in the previous part to do dimensionality reduction. We will see how we can transform 7396 dimensional vectors into 100 dimensional vectors while preserving most of the original information. You should not call `svds` again (this will be very slow); instead, use your existing eigenfaces from the previous question.

1. Create a function to project a collection of faces onto a fixed number (`dim`) of eigenfaces. To project the matrix `X` of images (as row vectors) into the low dimensional linear space spanned by the eigenfaces use the following:

   ```
   %>  Z = (X - repmat(xbar, n, 1)) * efaces(1:dim,:)';
   ```

   where `efaces` are the eigenfaces as row vectors. The `n` by `dim` matrix `Z` contains the low dimensional representation of each image. Provide the code for this function.

2. Create a function to map points in the low dimensional space back to the original space. You may want to use the following piece of code:

   ```
   %>   Xapprox = (Z * efaces(1:dim, :)) + repmat(xbar, n, 1);
   ```

   Provide the code for this function.

3. Project the first 10 faces onto the first 10, 50, and 100 eigenfaces and then compute `Xapprox`. Using a 4 row by 10 column grid (use `\subplot(4,10,i)`), plot:

   Row 1: the original images.

   Row 2: the `Xapprox` approximation using the first `dim=100` eigenfaces.

   Row 3: the `Xapprox` approximation using the first `dim=50` eigenfaces.

   Row 4: the `Xapprox` approximation using the first `dim=10` eigenfaces.

4. From the above plot identify characteristics or features that are lost in the low dimensional representations. For example, can you express glasses or mouth gestures in the low dimensional subspaces?

2.4 Gaussian Naive Bayes [20 Points]

Ultimately, beauty is in the eye of the beholder. In the case of this question the beholder is a simple Gaussian Naive Bayes classifier. Here we will use a Gaussian Naive Bayes classifier with separate variance terms for each feature and class. Thus, the joint probability can be written as:

$$\mathbf{P}\left(Z_1, \ldots, Z_{\text{dim}}, Y \mid \theta, \mu, \sigma\right) \propto \theta^Y (1-\theta)^{(1-Y)} \prod_{i=1}^{\text{dim}} \exp\left(-\frac{(Z_i - \mu_{i,Y})^2}{2\sigma_{i,Y}^2}\right)$$

1. Project all the training images into the `dim=100` linear space using the above function. Compute the covariance matrix for the `dim=100` dimensional representation `Z` and plot the distribution of the diagonal and off diagonal terms.

2. If you assume that the rows of `Z` are Gaussian, what can you say about the independence of the "features" of `Z`? How might this be helpful when using a Gaussian Naive Bayes classifier?

3. Write a function that given the data `Z` and `Y = hot` computes the parameters θ and the 2 by `dim` parameter matrices μ and σ for the Gaussian Naive Bayes classifier. Provide the code in your solution.

4. Write a function that given the data `Z` and the parameters θ, μ, and σ predicts the probability that $Y = 1$ (i.e., the face is hot). Provide the code in your solution.

5. We now will train a Gaussian Naive Bayes classifier to predict whether a face is hot using the low dimensional feature space. To determine the best value for `dim` we will use cross validation on `tr`. *Do not* use the `te` data. For each `dim` from 1 to 100 do 100 random splits of `tr` into $\frac{4}{5}$ training and $\frac{1}{5}$ test. For efficiency, you may use the original eigenfaces and xbar computed from the earlier sections. You may want to use the following fragment of code:

```
split = round(4/5 * tr.n);
trials = 100;
for p = 1:100
    % Compute the projection
    X = project(tr.images, efaces, xbar, p);
    y = tr.hot;
    for t = 1:trials
        % Do Random Split
        ind = randperm(tr.n);
        Xtr = X(ind(1:split), :);          ytr = y(ind(1:split));
        Xval = X(ind((split+1):end),:);    yval = y(ind((split+1):end));
        % Train the Gaussian Naive Bayes classifier
        % Predict the y values for the heldout data
        % Compute the classification error
    end
end
```

6. Plot the average cross validation prediction accuracy as a function of the number of principal components.

7. What number of components maximizes the average cross validation prediction accuracy?

8. You may now use the test data `te`. Using the optimal number of components project the faces from `te` into the low dimensional space. Then using the Gaussian Naive Bayes model trained on all of `tr` compute the prediction accuracy. Compare the prediction accuracy for the Gaussian Naive Bayes classifier to the prediction accuracy for the simple classifier that believes all faces are hot.