CS6140 machine Learning HW3: Generative Models, PCA

Problem 1: GDA [50 points]

Perform *Gaussian Discriminant Analysis* (GDA) on the spambase data with k-fold cross validation.

For each fold, use 1 fold for testing and k-1 folds for training.

During the analysis, you should test both of the following covariance assumptions:

- Single covariance model (all the training data)
- Class conditional covariance model (estimated separately for each class)

In the context of GDA, the former is called *Linear Discriminant Analysis* while the latter is called *Quadratic Discriminant Analysis*.

Based on the training and testing performance, does it appear the data are normally distributed?

Problem 2: Naive Bayes [50 points]

Create a set of *Naive Bayes classifiers* for detecting e-mail spam and test the classifiers on the Spambase dataset via 10-fold cross validation.

2.1: Build the classifier(s)

Create three distinct Naive Bayes classifiers by varying the likelihood distribution. In particular, try:

- Modeling the likelihood using a Bernoulli distribution
- Modeling the likelihood using a Gaussian distribution
- Modeling the likelihood non-parametrically

For real-valued features, a Bernoulli likelihood may be obtained by thresholding against a scalar-valued statistic, such as the sample mean μ. Similarly, a Gaussian likelihood function can be obtained by estimating the [class conditional] expected value μ and variance σ. For the non-parametric setting, model the feature distribution either with a kernel density estimate (KDE) or a histogram.

Bernoulli example: Consider some threshold $\mu \in \mathbb{R}$. To compute the conditional probability of a feature f_i, compute the fraction by which f_i is above or below μ over all the data within its class. In other words, for feature f_i with expected value μ_i, estimate:

$$P(f_i \le \mu_i \mid \text{spam}) \text{ and } P(f_i > \mu_i \mid \text{spam})$$

$$P(f_i \le \mu_i \mid \text{non-spam}) \text{ and } P(f_i > \mu_i \mid \text{non-spam})$$

and use these estimated values in your Naive Bayes predictor.

For all the models mentioned, you may want to consider using some kind of additive smoothing technique as a regularizer.

Evaluate the performance of the classifiers using the following three performance summaries.

Context: In many situations, false positive (Type I) and false negative (Type II) errors incur different costs. In spam filtering, for example, a false positive is a legitimate e-mail that is misclassified as spam (and perhaps automatically redirected to a “spam folder” or, worse, auto-deleted) while a *false negative* is a spam message that is misclassified as legitimate (and sent to one’s inbox).

When using Naive Bayes, one can easily make such trade-offs. For example, in the usual Bayes formulation, with $\mathbf{x}$ the data vector and y the class variable, one would predict “spam” if:

$$P(y = \text{spam}|\mathbf{x}) > P(y = \text{non-spam}|\mathbf{x})$$

or equivalently, in a log-odds formulation,

$$\ln(P(y = \text{spam}|\mathbf{x})/P(y = \text{non-spam}|\mathbf{x})) > 0$$

However, note that one could choose to classify an e-mail as spam for any threshold τ, i.e.:

$$\ln(P(y = \text{spam}|\mathbf{x})/P(y = \text{non-spam}|\mathbf{x})) > \tau$$

Larger values of τ reduce the number of spam classifications, reducing false positives at the expense of more false negatives. Negative values of τ have the converse effect.

Most users are willing to accept some false negative examples so long as very few legitimate e-mails are misclassified. Given your classifiers and their ROC curves, what value of τ would you choose to deploy in a real email spam filter?

Problem 3: EM on generated data [50 points]

Implement your own E, M steps and run the EM algorithm with random initial parameter values and see if you recover the true parameters of the underlying generative model. The math operations (Gaussian(x) etc) can be done via library calls.

- The file 2gaussian.txt was generated from a mixture of two 2D Gaussians with the parameters below.

$$\mu_1 = [3,3], \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}, \quad n_1 = 2000$$

$$\mu_1 = [7,4], \quad \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad n_1 = 4000$$

- The file 3gaussian.txt, generated using a mixture of three Gaussians.

$$\mu_1 = [3,3], \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}, \quad n_1 = 2000$$

$$\mu_2 = [7,4], \quad \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad n_2 = 3000$$

$$\mu_3 = [5,7], \quad \Sigma_3 = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix}, \quad n_3 = 5000$$

Verify your findings against the true parameters via your choice of summary (e.g. absolute parameters differences, contour plots, etc.)

PROBLEM 4 Bayes Math [30p GR ONLY]

A. Prove that

$$P(A|B,C) = \frac{P(B|A,C) \cdot P(A|C)}{P(B|C)}$$

B. You are given a coin which you know is either fair or double-headed. You believe that the a priori odds of it being fair are F to 1; i.e., you believe that the a priori probability of the coin being fair is F/(F+1) . You now begin to flip the coin in order to learn more. Obviously, if you ever see a tail you know immediately that the coin is fair. As a function of F, how many heads in a row would you need to see before becoming convinced that there is a better than even chance that the coin is double-headed?

PROBLEM 5 - EM with mixture of binomial distributions [30p GR-ONLY]

Cheng's note summarizing E an M steps for this problem.

A. Generate mixture data of 2 coins (flips). Pick a p,r,pi as in the EM example discussed in class (or in notes). Say p=.75, r=.4, pi=.8, but you should try this for several sets of values. Generate the outcome of the coin experiment by first picking a coin (pi probability for first coin, 1-pi probability for the second coin), then flip that coin K times (use K=10) with probability of head (p if first coin is picked, r if the second coin is picked) and finally write down a 1 if the head is seen, 0 for the tail. Repeat this M=100 times or more; so your outcome is a stream of M sequences of K flips : (1001110001; 0001110001; 1010100101 etc). You have to implement the generative logic yourself, but all math calls including rand() can be done via libraries

B. Modify the E, M steps in Pb4 to work with Binomial distributions and infer parameters from data. Now using the stream of 1 and 0 observed, recover p,r,pi using the EM algorithm; K is known in advance. Report in a table the parameter values found by comparison with the ones used to generate data; try several sets of (p,r,pi). Here are some useful notes, and other readings (1 , 2 , 3 , 4) for the coin mixture.

C). Repeat A) and B) with T coins instead of two. You will need more mixture parameters.

Problem 6[30 points]: Kernel PCA features with a Linear Classifier

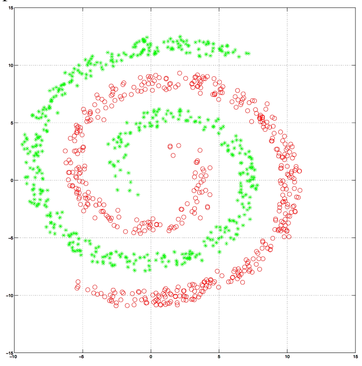
Dataset: 1000 2-dim datapoints TwoSpirals
Dataset: 1000 2-dim datapoints ThreeCircles

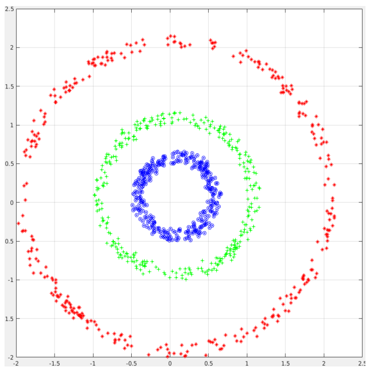
A) First, train a Linear/Logistic Regression (library, logistic if data labels are categories) and confirm that it doesnt work , i.e. it has a high classification error or high Root Mean Squared Error.
B) Run KernelPCA with Gaussian Kernel to obtain a representation of T features. For reference these steps we demoed in class (Matlab):

```
%get pairwise squared euclidian distance
X2 = dot(X,X,2);
DIST_euclid = bsxfun(@plus, X2, X2') - 2 * X * X';
% get a kernel matrix NxN
sigma = 3;
K = exp(-DIST_euclid/sigma);
%normalize the Kernel to correspond to zero-mean
U = ones(N)/ N ;
Kn = K - U*K -K*U + U*K*U ;
% obtain kernel eignevalues, vectors; then sort them with largest eig first
[V,D] = eig(Kn,'vector') ;
[D,sorteig] = sort(D,'descend') ;
V = V(:, sorteig);
% get the projection matrix
```

```
XG = Kn*V;
%get first 3 dimensions
X3G = XG(:,1:3);
%get first 20 dimensions
X20G = XG(:,1:20);
%get first 100 dimensions
X100G = XG(:,1:100);
```

C) Retrain the regression algorithm on the same data kernelized / dual form. How large T needs to be to get good performance?





PROBLEM 7 Naive Bayes with Gaussian Mixtures [Optional, no credit]

Rerun the Naive Bayes classifier on Spam Data. For each feature (1-dim) use a mixture of K Gaussians as your model; run the EM to estimate the 3*K parameters for each feature : mean1, var1, w1; mean2,var2, w2;… meanK,varK,wK; constrained by w1+w2 +…+wK=1. (You would need a separate mixture for positive and negative data, for each feature). We observed best results for K=9.

Testing: for each testing point, apply the Naive Bayes classifier as before: take the log-odd of product of probabilities over features mixtures (and the prior), separately for positive side and negative side; use the overall probability ratio as an output score, and compute the AUC for testing set. Do this for a 10-fold cross validation setup. Is the overall 10-fold average AUC better than before, when each feature model was a single Gaussian?

PROBLEM 8 [Optional, no credit]

a. Somebody tosses a fair coin and if the result is heads, you get nothing, otherwise you get \$5. How much would you be pay to play this game? What if the win is \$500 instead of \$5?

b. Suppose you play instead the following game:

- At the beginning of each game you pay an entry fee of \$100. A coin is tossed until a head appears, counting n = the number of tosses it took to see the first head.
- Your reward is 2^n (that is: if a head appears first at the 4th toss, you get \$16).

Would you be willing to play this game (why)?

c. Lets assume you answered “yes” at part b (if you did not, you need to fix your math on expected values). What is the probability that you make a profit in one game? How about in two games?

d)**(difficult)** After about how many games (estimate) the probability of making a profit overall is bigger than 50% ?

PROBLEM 9 [Optional, no credit]

DHS CH2, Pb 43

PROBLEM 10 part 2 [Optional, no credit]]

a. DHS CH2, Pb 45

PROBLEM 11 [Optional, no credit]

b. DHS CH2, Pb 44

PROBLEM 12 [Optional, no credit; requires independent study of Hidden Markov Models, DHS ch 3.10]

DHS Pb 3.50 (page 154-155) The standard method for calculating the probability of a sequence in a given HMM is to use the forward probabilities αi(t).

References

0. Good references on GDA: https://www.eecs189.org/static/notes/n18.pdf
1. Naive Bayes overview using different notation than most: http://www.cs.columbia.edu/~mcollins/em.pdf
2. Short & limited but well-written Naive Bayes implementation in pure NumPy: https://www.python-engineer.com/courses/mlfromscratch/05_naivebayes/