CS6140 Machine Learning

HW3 Generative Models

EM class notes, Other EM notes1, notes4, notes5

Multivariate Normal, Multivariate - mariginal and conditional, Correlation Coefficient

You have to write your own code, but it is ok to discuss or look up specific algorithmic/language code details. Before starting coding, make sure to understand :

- how 2-dim Gaussians work
- Naive Bayes independence of features, thus the product of probabilities
- languages like Python or Matlab or R can help immensely in dealing with multidimensional data, including the means and covariances in 2-dim.
- E and M steps for EM algorithm. You can look into existing EM code for help.

PROBLEM 1 GDA [50 points]

Run the Gaussian Discriminant Analysis on the spambase data. Use the k-folds from the previous problem (1 for testing, k-1 for training, for each fold)
Since you have 57 real value features, each of the 2gaussians (for + class and for - class) will have a mean vector with 57 components, and a they will have

- either a common (shared) covariance matrix size 57x57. This covariance is estimated from all training data (both classes)
- or two separate covariance 57x57 matrices (estimated separately for each class)

(you can use a Matlab or Python of Java built in function to estimated covariance matrices, but the estimator is easy to code up).
Looking at the training and testing performance, does it appear that the gaussian assumption (normal distributed data) holds for this particular dataset?

PROBLEM 2 Naive Bayes[50 points]

In this assignment, you will create a Naive Bayes classifier for detecting e-mail spam, and you will test your classifier on a publicly available spam dataset. You will experiment with three methods for modeling the distribution of features, and you will test your classifier using 10-fold cross-validation.

- **Preliminary:** Download the Spambase dataset available from the UCI Machine Learning Repository.

The Spambase data set consists of 4,601 e-mails, of which 1,813 are spam (39.4%). The data set archive contains a processed version of the e-mails wherein 57 real-valued features have been extracted and the spam/non-spam label has been assigned. You should work with this processed version of the data. The data set archive contains a description of the features extracted as well as some simple statistics over those features.

- Partition the data into 10 folds.

  To estimate the generalization (testing) error of your classifier, you will perform cross-validation. In *k*-fold cross-validation, one would ordinarily partition the data set randomly into *k* groups of roughly equal size and perform *k* experiments (the "folds") wherein a model is *trained* on *k*-1 of the groups and *tested* on the remaining group, where each group is used for testing exactly once. (A common value of *k* is 10.) The generalization error of the classifier is estimated by the *average* of the performance across all *k* folds.

  While one should perform cross-validation with random partitions, for consistency and comparability of your results, you should partition the data into 10 groups as follows: Consider the 4,601 data points in the order they appear in the processed data file. Each group will consist of every 10th data point in file order, i.e., Group 1 will consist of points {1,11,21,...}, Group 2 will consist of points {2,12,22,...}, ..., and Group 10 will consist of ponts {10,20,30,...}. Finally, Fold *k* will consist of *testing* on Group *k* a model obtained by *training* on the remaining *k*-1 groups.

- **Step 1:** Create three Naive Bayes classifiers by modeling the features in three different ways.

  The 57 features are real-valued, and one can model the feature distributions in simple and complex ways. You will explore the effect of this modeling on overall performance.

  1. Model the features as simple Bernoulli (Boolean) random variables. Consider a threshold such as the overall mean value of the feature (available in the Spambase documentation), and simply compute the fraction of the time that the feature value is above or below the overall mean value for each class. In other words, for feature f_i with overall mean value mu_i, estimate

     - Pr[f_i <= mu_i | spam]
     - Pr[f_i > mu_i | spam]
     - Pr[f_i <= mu_i | non-spam]
     - Pr[f_i > mu_i | non-spam]

     and use these estimated values in your Naive Bayes predictor, as appropriate. (I would suggest using Laplace or Laplace-like smoothing to avoid any issues with zero probabilities, and feel free to experiment with various theshold values, if you like.)

  2. Model the features as Gaussian random variables, estimating the class conditional mean and variance as appropriate. (Again, I would suggest smoothing your estimate of the variance, to avoid any zero variance issues.)

  3. Model the feature distribution via a histogram. Bucket the data into a number of *bins* and estimate the class conditional probabilities of each bin. For example, for any feature, consider the following values: min-value, mean-value, max-value, and the class conditional (spam/non-spam) mean values, one of which will presumably be *lower* than the overall mean, and one of which will be higher than the overall mean. Order these values numerically, and create four feature value bins as follows

- [min-value, low-mean-value]
- (low-mean-value, overall-mean-value]
- (overall-mean-value, high-mean-value]
- (high-mean-value, max-value]

and estimate the class conditional probabilities for each of these bins, in a manner similar to that used in the Bernoulli model above. (Again, use Laplace smoothing to avoid any issues with zero probabilities, and feel free to experiment with the number of bins and various theshold values, if you like.)

4. Use 9-bins histogram instead of four

- **Step 2:** Evaluate your results.

  1. *Error tables*: For each of your three classifiers, create a table with one row per fold showing your false positive, false negative, and overall error rates, and add one final row per table corresponding to the average error rates across all folds. For this problem, the false positive rate is the fraction of non-spam testing examples that are misclassified as spam, the false negative rate is the fraction of spam testing examples that are misclassified as non-spam, and the overall error rate is the fraction of overall examples that are misclassified.

  2. *ROC Curves:* In many situations, there is a different cost associated with false positive (Type I) and false negative (Type II) errors. In spam filtering, for example, a false positive error 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). Most users are willing to accept some false negative examples so long as very few legitimate e-mails are misclassified.

     When using Naive Bayes, one can easily make such trade-offs. For example, in the usual Bayes formulation, one would predict "spam" if

     $$\Pr[\text{spam} \mid \text{data}] > \Pr[\text{non-spam} \mid \text{data}]$$

     or equivalently, in a log-odds formulation,

     $$\ln (\Pr[\text{spam} \mid \text{data}] / \Pr[\text{non-spam} \mid \text{data}]) > 0.$$

     Note that one could choose to classify an e-mail as spam for any threshold tau

     $$\ln (\Pr[\text{spam} \mid \text{data}] / \Pr[\text{non-spam} \mid \text{data}]) > \text{tau}$$

     where positive values of tau effectively reduce the number of spam classifications, thus decreasing false positives at the expense of increasing false negatives, while negative values of tau have the opposite effect.

     One mechanism for visualizing this trade-off is through an ROC curve. (See, for example, the Wikipedia page on ROC curves.) An ROC curve effectively visualizes the true positive (detection) rate vs. the false positive (false alarm) rate for all possible thresholds. (The true positive rate is the fraction of spam messages above threshold; the false positive rate is the fraction of non-spam messages above threshold. Note that the true positive rate is 1 minus the false negative rate.)

One simple way to generate an ROC curve is to sort all *m* testing points by their log-odds, and for the top *k* testing points, for all *k* between 1 and *m*, calculate the true positive and false positive rates associated with the top *k* items; plot these points.

Create an ROC curve for each of your classifiers for Fold 1. Preferable draw all three curves on the same plot so that you can compare them.

3. *AUC:* An ROC curve visualizes the tradeoff between false positive rate and true positive rate (which is 1 minus the false negative rate) at various operating points. One measure for summarizing this data is to compute the *area under the ROC curve* or AUC. The AUC, as its name suggests, is simply the area under the ROC curve, which for a perfect curve is 1 and for a random predictor is 1/2. This area has an interesting probabilistic interpretation: it is the chance that the predictor will rank a randomly chosen positive example above a randomly chosen negative example.

   The AUC can be calculated fairly simply using the trapezoidal rule for estimating the area under a curve: If our *m* ROC data points are

   $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$

   in order, where

   $(x_1, y_1) = (0,0)$

   and

   $(x_m, y_m) = (1,1)$

   then the AUC is calculated as follows:

   $(1/2)\ \text{sum}_{k=2 \text{ to } m}\ (x_k - x_{k-1})\ (y_k + y_{k-1})$.

   Calculate the AUC of your three classifiers.

PROBLEM 3 EM algorithm[20 points]

Use this pdf of the EM code demoed in class. (Alternatively, you can print the code yourself from an editor). Annotate on the right the sections of the code with your explanation of what the code does. Submit as pdf.

PROBLEM 4 : EM on simple data [50 points]

A) The gaussian 2-dim data on file 2gaussian.txt has been generated using a mixture of two Gaussians, each 2-dim, with the parameters below. Run the EM algorithm with random initial values to recover the parameters.

```
mean_1 [3,3]); cov_1 = [[1,0],[0,3]]; n1=2000 points
mean_2 =[7,4]; cov_2 = [[1,0.5],[0.5,1]]; ; n2=4000 points
```

You should obtain a result visually like this (you dont necessarily have to plot it)

B) Same problem for 2-dim data on file 3gaussian.txt , generated using a mixture of three Gaussians. Verify your findings against the true parameters used generate the data below.

```
mean_1 = [3,3] ; cov_1 = [[1,0],[0,3]]; n1=2000
mean_2 = [7,4] ; cov_2 = [[1,0.5],[0.5,1]] ; n2=3000
mean_3 = [5,7] ; cov_3 = [[1,0.2],[0.2,1]]    ); n3=5000
```

PROBLEM 5 [20 points]

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 6 - [Extra Credit]

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

A.Generate mixture data (coin 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=1000 times or more; so your outcome is a stream of M sequences of K flips : (1001110001; 0001110001; 1010100101 etc)

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

PROBLEM 7 Naive Bayes with Gaussian Mixtures[Extra 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 [Extra 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 [Extra Credit]

DHS CH2, Pb 43

Let the components of the vector $\mathbf{x} = (x_1, ..., x_d)^t$ be binary valued (0 or 1) and $P(\omega_j)$ be the prior probability for the state of nature ω_j and $j = 1, ..., c$. Now define

$$p_{ij} = \text{Prob}(x_i = 1|\omega_j) \qquad \begin{array}{l} i = 1, ..., d \\ j = 1, ..., c, \end{array}$$

with the components of x_i being statistically independent for all $\mathbf{x}$ in ω_j.

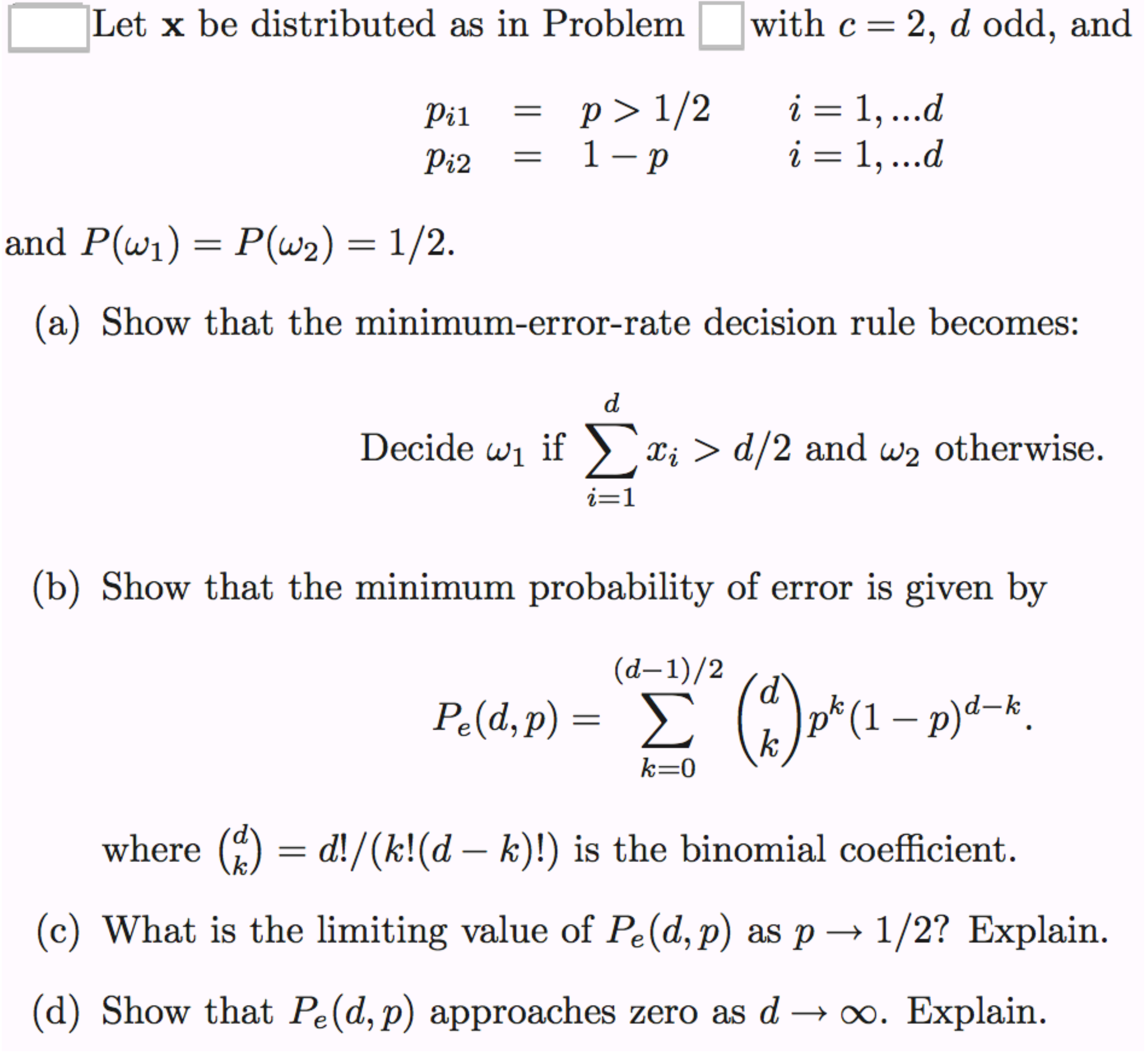
(a) Interpret in words the meaning of p_{ij}.

(b) Show that the minimum probability of error is achieved by the following decision rule: Decide ω_k if $g_k(\mathbf{x}) \geq g_j(\mathbf{x})$ for all j and k, where

$$g_j(\mathbf{x}) = \sum_{i=1}^{d} x_i \ln \frac{p_{ij}}{1 - p_{ij}} + \sum_{i=1}^{d} \ln (1 - p_{ij}) + \ln P(\omega_j).$$

PROBLEM 10 part 2 [Extra Credit]

a) DHS CH2, Pb 45



Let $\mathbf{x}$ be distributed as in Problem with $c = 2$, d odd, and

$$\begin{aligned} p_{i1} &= p > 1/2 & i &= 1, ...d \\ p_{i2} &= 1 - p & i &= 1, ...d \end{aligned}$$

and $P(\omega_1) = P(\omega_2) = 1/2$.

(a) Show that the minimum-error-rate decision rule becomes:

$$\text{Decide } \omega_1 \text{ if } \sum_{i=1}^{d} x_i > d/2 \text{ and } \omega_2 \text{ otherwise.}$$

(b) Show that the minimum probability of error is given by

$$P_e(d, p) = \sum_{k=0}^{(d-1)/2} \binom{d}{k} p^k (1-p)^{d-k}.$$

where $\binom{d}{k} = d!/(k!(d-k)!)$ is the binomial coefficient.

(c) What is the limiting value of $P_e(d, p)$ as $p \rightarrow 1/2$? Explain.

(d) Show that $P_e(d, p)$ approaches zero as $d \rightarrow \infty$. Explain.

PROBLEM 11 [Extra Credit]

b) DHS CH2, Pb 44

Let the components of the vector $\mathbf{x} = (x_1, ..., x_d)^t$ be ternary valued (1, 0 or -1), with

$$\begin{aligned} p_{ij} &= \text{Prob}(x_i = \ 1\ |\omega_j) \\ q_{ij} &= \text{Prob}(x_i = \ 0\ |\omega_j) \\ r_{ij} &= \text{Prob}(x_i = -1|\omega_j), \end{aligned}$$

and with the components of x_i being statistically independent for all $\mathbf{x}$ in ω_j.

(a) Show that a minimum probability of error decision rule can be derived that involves discriminant functions $g_j(\mathbf{x})$ that are quadratic function of the components x_i.

PROBLEM 12 [EXTRA 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 $\alpha_i(t)$.

(a) Show by a simple substitution that a symmetric method can be derived using the backward probabilities $\beta_i(t)$.

(b) Prove that one can get the probability by combining the forward and the backward probabilities at any place in the middle of the sequence. That is, show that

$$P(\boldsymbol{\omega}^{T'}) = \sum_{i=1}^{T'} \alpha_i(t)\beta_i(t),$$

where $\boldsymbol{\omega}^{T'}$ is a particular sequence of length $T' < T$.

(c) Show that your formula reduces to the known values at the beginning and end of the sequence.