SMAI Assignment 3 Report

Problem 1

Code

```
import numpy as np
import csv
import pdb
from collections import defaultdict

def nb_train(train_data,plabel,nlabel):
	num_attributes = len(train_data[0]) - 1
	pdict = [defaultdict(float) for x in range(num_attributes)]
	ndict = [defaultdict(float) for x in range(num_attributes)]
	plist = [0]*num_attributes
	nlist = [0]*num_attributes
	num_psample = 0
	num_nsample = 0
	for row in train_data:
	# count number of positive and negative labels
	class_label = row[-1]
	if class_label == plabel:
		num_psample += 1
		cdictlist = pdict
		clist = plist
	elif class_label == nlabel:
		num_nsample += 1
		cdictlist = ndict
		clist = nlist

	# compute feature freq and attr freq for each class
	feat_vec = row[:-1]
	ind = -1
	for feat in feat_vec:
		ind += 1
		if feat == '?':
			continue
		cdictlist[ind][feat] += 1
		clist[ind] += 1
```

```
	# compute priors
	priors = [np.log(float(num_psample)/(num_psample + num_nsample)),
np.log(float(num_nsample)/(num_psample + num_nsample))]
	# take log of probabilities
	for i in range(num_attributes):
	for feat in pdict[i]:
		if pdict[i][feat]!=0:
			pdict[i][feat] = np.log(pdict[i][feat]/plist[i])
		else:
			pdict[i][feat] = np.log(0.000000001)
	for feat in ndict[i]:
		if ndict[i][feat]!=0:
			ndict[i][feat] = np.log(ndict[i][feat]/nlist[i])
		else:
			ndict[i][feat] = np.log(0.000000001)
	return [priors, pdict, ndict]

def nb_predict(priors, pdict, ndict, test_data, plabel, nlabel):
	classified = 0
	misclassified = 0
	ans = 0

	for row in test_data:
	 # compute posterior probability for each test sample against
each class
	ground_truth = row[-1]
	feat_vec = row[:-1]
	ind = -1
	pcumulative = priors[0]
	ncumulative = priors[1]
	for feat in feat_vec:
		ind += 1
		if feat=='?':
			continue
		pcumulative += pdict[ind][feat]
		ncumulative += ndict[ind][feat]
	# predict label based on posterior probabilities
	if pcumulative > ncumulative:
		predicted = plabel
```

```
        else:
            predicted = nlabel
        # check against ground truth
        if predicted == ground_truth:
            classified += 1
        else:
            misclassified += 1
    print "Accuracy",float(classified)/(classified+misclassified),"
over ", classified + misclassified, " samples."
    return classified, misclassified

total_classified = 0
total_misclassified = 0
# process data
raw_data = csv.reader(open("census-income.data"))
data0 = []
for i in list(raw_data):
    data0.append([item.strip() for item in i])
data = np.array(data0)
data = data[:,[2,3,4,41]]
np.random.shuffle(data)
data = data[:10000]
acc = []
for i in range(10):
    np.random.shuffle(data)
    train_data = data[0:5000]
    test_data = data[5001:10000]
    [priors, pdict, ndict] = nb_train(train_data,'50000+.','-
50000.')
    [classified, misclassified] = nb_predict(priors, pdict, ndict,
test_data, '50000+.','- 50000.')
    total_classified += classified
    total_misclassified += misclassified
    acc.append(float(classified)/(classified+misclassified))
acc = np.array(acc)
mean_acc =
float(total_classified)/(total_classified+total_misclassified)
sd = np.std(acc)
print "Mean Accuracy",mean_acc,"Standard Deviation",sd
```

Observations

```
joycode@nelovo:~/sem5/smai/Assign2/census$ python p1.py
Accuracy 0.813162632527  over  4999  samples.
Accuracy 0.823364672935  over  4999  samples.
Accuracy 0.828965793159  over  4999  samples.
Accuracy 0.831566313263  over  4999  samples.
Accuracy 0.831766353271  over  4999  samples.
Accuracy 0.806561312262  over  4999  samples.
Accuracy 0.831966393279  over  4999  samples.
Accuracy 0.806361272254  over  4999  samples.
Accuracy 0.801960392078  over  4999  samples.
Accuracy 0.835767153431  over  4999  samples.
Mean Accuracy 0.821144228846 Standard Deviation 0.0121713947933
```

- Ties are resolved by giving default class to class B
- Two things can be done for missing features:
  - Omit the records with missing values.
  - Omit only the missing attributes,where attributes are the values the feature can take. They are not considered while multiplying the probabilities,but are taken into account while general division.

Problem 2

Bayesian Parameter Estimation for Univariate Density Functions.

We know that class posterior probability can be estimated as:

$$p(w_i|x, D_i) = \frac{p(x|w_i, D_i)\, P(w_i)}{\sum_{j=1}^{c} p(x|w_j, D_j) P(w_j)}$$

where w_i : class
D_i : data samples for that class

fixing w_i; only 1 class, remove w_i

$$p(x|D) = \int_\theta p(x|\theta)\, p(\theta|D)\, d\theta$$. Let $\hat{\theta}$ is our optimal parameter.. (say mean).

$$p(x|D) \approx p(x|\hat{\theta}) \quad \text{(where } p(\theta|D) \text{ peaks).}$$

$$p(\theta|D) = \frac{p(D|\theta)\, p(\theta)}{\int_\theta p(D|\theta)\, p(\theta)\, d\theta}$$

We can think of this as following: Let the distribution $p(x|w_i, D)$ is a gaussian, with parameters μ & σ. Then the parameter 'μ' also varies like a Gaussian

$$p(\mu|D) = \frac{p(D|\mu)\, p(\mu)}{\int_\mu p(D|\mu)\, p(\mu)\, d\mu} \quad -①$$

$p(\mu)$: prior: also a gaussian.

$p(x_k|\mu) :\sim N(\mu, \sigma^2)$
$p(\mu) :\sim N(\mu_0, \sigma_0^2)$.

Consider ①

$$p(\mu|D) = \frac{p(D|\mu)\,p(\mu)}{\int p(D|\mu)\,p(\mu)\,d\mu}$$

$$= \alpha \prod_{k=1}^{n} p(x_k|\mu)\,p(\mu)$$: as all data points are independent.

α ↳ Constant normalizing term of Denominator

As $p(x_k|\mu) \sim N(\mu, \sigma^2)$.

$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_k-\mu)^2}{2\sigma^2}}$$

$$\prod_{k=1}^{n} p(x_k|\mu) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\sum_{k=1}^{n} \frac{(x_k-\mu)^2}{2\sigma^2}}$$

$$p(\mu) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}$$

$$\therefore p(\mu|D) = \alpha'' \exp\left(-\frac{1}{2}\left[\sum_{k=1}^{n} \frac{(x_k-\mu)^2}{2\sigma^2} + \frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right]\right)$$

$$= \alpha'' \exp\left(-\frac{1}{2}\left[\left(\frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}\right)\mu^2 - 2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n} x_k + \frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right)$$

$$\sigma_n^2 = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2 + \sigma^2}$$

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right) \text{mean}_{\text{samples}} + \left(\frac{\sigma^2}{n\sigma_0^2+\sigma^2}\right) \mu_0$$

Bayesian Parameter Estimation for Multivariate Density Functions:

Modifying the above result for a multivariate density function, $\bar{x}$: d dimension input, n such inputs

$$p(w_i \mid \bar{x}, D_i) = \frac{p(\bar{x} \mid w_i, D_i)\, P(w_i)}{\sum_{j=1}^{c} p(x \mid w_j, D_j)\, P(w_j)}$$

$$p(\bar{x} \mid D) = \int p(x \mid \bar{\theta})\, p(\bar{\theta} \mid D)$$

where $\bar{\theta}$ is a vector of (multiple) parameters

approximating,

$$p(\bar{x} \mid D) \approx p(x \mid \bar{\theta})$$

$$p(\bar{\theta} \mid D) = \frac{p(D \mid \bar{\theta})\, p(\bar{\theta})}{\int_{\theta} p(D \mid \bar{\theta})\, p(\bar{\theta})\, d\theta}$$

$$p(D \mid \bar{\theta}) = \prod_{k=1}^{n} p(x_k \mid \bar{\theta})$$

$$\therefore p(\bar{\theta} \mid D) = \alpha \prod_{k=1}^{n} p(x_k \mid \bar{\theta})\, p(\bar{\theta})$$

Also,

$$p(\bar{x}_k \mid \bar{\theta}) = \frac{1}{(2\pi)^{d/2} \det(\Sigma)} \exp\left(-\frac{(x_k - \bar{\theta})\, \Sigma^{-1} (x_k - \theta)^T}{2} \right) \sim \mathcal{N}(\bar{\theta}, \Sigma)$$

$$p(\bar{\theta}) = \frac{1}{(2\pi)^{d/2} \det(\Sigma_0)} \exp\left(-\frac{1}{2} (\theta - \bar{\theta}_0)\, \Sigma_0^{-1} (\theta - \bar{\theta}_0)^T \right)$$

$$\sim \mathcal{N}(\bar{\theta}_0, \Sigma_0)$$

$\therefore p(\bar{\theta}|D) = \alpha'' \exp(-\frac{1}{2}$

$$\prod_{k=1}^{n} p(x_k|\bar{\theta}) = \frac{1}{\left((2\pi)^{\frac{d}{2}}\right)^{3} \det(\Sigma)^{n}} \exp\left(-\sum_{k=1}^{n} \frac{(\bar{x}_k-\bar{\theta})\,\Sigma^{-1}\,(\bar{x}_k-\bar{\theta})^{T}}{2} \right).$$

$$\therefore p(\bar{\theta}|D) = \alpha'' \exp\left(-\frac{1}{2}\sum_{k=1}^{n} (\bar{x}_k-\bar{\theta})\,\Sigma^{-1}(\bar{x}_k-\bar{\theta})^{T} - \frac{1}{2}(\bar{\theta}-\bar{\theta}_0)\,\Sigma_0^{-1}(\bar{\theta}-\bar{\theta}_0)^{T} \right)$$

$$= \alpha'' \exp\left(-\frac{1}{2}\left(\underbrace{\sum_{k=1}^{n} (\bar{x}_k-\bar{\theta})\,\Sigma^{-1}(\bar{x}_k-\bar{\theta})^{T}}_{A} + \underbrace{(\bar{\theta}-\bar{\theta}_0)\,\Sigma_0^{-1}(\bar{\theta}-\bar{\theta}_0)^{T}}_{B} \right)\right)$$

Consider A

$$= \underbrace{\sum_{k=1}^{n}(\bar{x}_k\,\Sigma^{-1}\bar{x}_k^{T})}_{c} - \underbrace{\sum_{k=1}^{n}(\bar{x}_k\,\Sigma^{-1}\bar{\theta}^{T} - \bar{\theta}\,\Sigma^{-1}\bar{x}_k^{T})}_{bx} + \underbrace{\bar{\theta} n \Sigma^{-1}\bar{\theta}^{T}}_{ax^2}$$

Consider B

$$= \underbrace{\bar{\theta}\,\Sigma_0^{-1}\bar{\theta}^{T}}_{ax^2} \underbrace{- \bar{\theta}\,\Sigma_0^{-1}\bar{\theta}_0^{T} - \bar{\theta}_0\,\Sigma_0^{-1}\bar{\theta}^{T}}_{bx} + \underbrace{\bar{\theta}_0\,\Sigma_0^{-1}\bar{\theta}_0^{T}}_{c}$$

Combining A and B, we and completing the square in quadratic form, we obtain the final result as

$$\mu_n = \Sigma_0\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\hat{\mu}_n + \frac{1}{n}\Sigma\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\bar{\theta}_0$$

$$\Sigma_n = \Sigma_0\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\frac{1}{n}\Sigma,$$

where: $\hat{\mu}_n$ = mean of all samples

n = # samples.

Problem 3

Code

```
clear;
clc;

fid = fopen('DOROTHEA/dorothea_train.data');
np = 800;
nvar = 100000;
X = zeros(nvar,np);
for i=1:np
      tline = fgetl(fid);
      ind = int32(str2double(strsplit(tline)));
      ind = ind(1:end-1);
      X(ind,i) = 1;
end
fclose(fid);
Y = load('DOROTHEA/dorothea_train.labels');

%PCA
k = 100;
X = X - mean(X,2)*ones(1,np);
K = X' * X;
[U,L] = eig(K);
[~,ind] = sort(diag(L),'descend');
PC = X * U(:,1:k);
X = PC' * X;

%LDA
% p = randperm(nvar, 10000);
% X = X(p,:);
% W1 = find(Y == -1);
% W2 = find(Y == 1);
% X_1 = X(:,W1);
% X_2 = X(:,W2);
% Xc_1 = X_1 - mean(X_1,2) * ones(1,size(W1,1));
% Xc_2 = X_2 - mean(X_2,2) * ones(1,size(W2,1));
% disp('done');
```

```
% Sw = Xc_1 * Xc_1' + Xc_2 * Xc_2';
% X = Sw \ (mean(X_1,2) - mean(X_2,2));

C = cvpartition(Y,'KFold',8);

meanacc = 0;

for epo=1:8

      tradat = X(:,training(C,epo));
      tralab = Y(training(C,epo));

      tesdat = X(:,test(C,epo));
      teslab = Y(test(C,epo));

      uniqlab = unique(tralab);
      nclass = length(uniqlab);
      nvar = size(X,1);
      ntest = length(teslab);

      for i=1:nclass
      ftralab(i) = sum(double(tralab==uniqlab(i)))/length(tralab);
      end

      for i=1:nclass
      tradat_i = tradat(:,(tralab==uniqlab(i)));
      tradat_mean(:,i) = mean(tradat_i,2);
      tradat_std(:,i) = std(tradat_i,0,2);
      end

      for i=1:ntest
      fteslab =
normcdf(ones(nclass,1)*tesdat(:,i)',tradat_mean',tradat_std');
      P(i,:) = ftralab.*prod(fteslab,2)';
      end

      [pv0,id]=max(P,[],2);
      for i=1:length(id)
      pv(i,1)=uniqlab(id(i));
```

```
        end

        acc = sum(pv == teslab)/length(pv);
        meanacc = meanacc + acc;

end

disp(['accuracy = ',num2str(meanacc*(100/8)),'%'])
```

Observations

- **Used k-fold cross validation on sample data; k = 8 i.e. test = 100, train = 700**
- **PCA k = 100, accuracy = 60.625%**
- **PCA k = 500, accuracy = 90.25%**
- **PCA k = 800, accuracy = accuracy = 90.25%**