CIS 520, Machine Learning, Fall 2015: Assignment 7
Due: Mon, Nov 16, 2015. 11:59pm, PDF to Canvas
[100 points]

Arpit Panwar

Collaborators:

Sruthi Nair

Instructions. Please write up your responses to the following problems clearly and concisely. We encourage you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Canvas. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the homework to understand the problem and reach a solution in groups up to size **four students.** However, *each student must write down the solution independently, and without referring to written notes from the joint session.* **In addition, each student must write on the problem set the names of the people with whom you collaborated.** You must understand the solution well enough in order to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

1 Sensational EM [20 points]

Suppose we have a robot with a single unreliable range sensor. For example, if the robot is standing 3 meters away from the nearest obstacle, we might get readings like this:

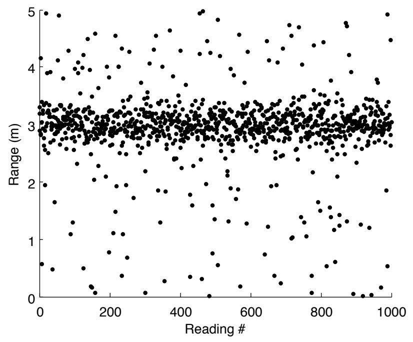


Figure 1: Range readings from sensor

If the sensor fails when obtaining a reading, it returns a value with uniform probability in the range $[0, 5]$. Otherwise, the sensor returns a value distributed according to $N(\mu, \sigma^2)$, where σ^2 is the variance of the sensor readings and μ is the actual distance to the nearest obstacle. We will assume that σ^2 is specified by the manufacturer of the sensor, and our goal is to determine μ given a set of readings $x_1, \ldots, x_n$.

The difficulty is that π_0, the failure rate of the sensor, is unknown ahead of time, and we want to avoid biasing our estimate of μ with completely meaningless samples where the sensor failed. Therefore, we model a sensor reading x_i according to the outcome of a latent variable z_i indicating whether or not the sensor failed:

$$p(z_i = 0) = \pi_0, \qquad p(x_i \mid z_i = 1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(x_i - \mu)^2\right\}, \qquad p(x_i \mid z_i = 0) = \frac{1}{5}\mathbf{1}(0 \leq x_i \leq 5)$$

This is a *mixture model* with two components: a Gaussian with mean μ and variance σ^2, and a uniform over $[0, 5]$.

1. **[4 points]** Using the equations defined above, write out the expression for the *marginal log likelihood* of a dataset $x_1, \ldots, x_n$ given the parameters (this is what maximum likelihood is maximizing). Your solution should be in terms of π_0, σ^2, μ, and the x_i. *Hint: Remember to marginalize over the z_i's.* Write your final answer as: $\log p(x_1, \ldots, x_n \mid \mu, \sigma^2, \pi_0) =$ expression.

$$logP\,(D|\pi\mu\sigma) = \sum_i log \sum_k P_\pi(Z_i = k)P_{\mu\sigma}(X_i|Z_i = k)$$

$$= \sum_i log \sum_{k=1}^{2} P(Z_i = k)P(X_i|Z_i = k)$$

$$= \sum_i log \left(\pi_0 * \frac{1}{5} + (1-\pi_0)\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-1}{2\sigma^2}(x_i-\mu)^2}\right)$$

$$= \sum_i log \left(\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-1}{2\sigma^2}(x_i-\mu)^2} + \pi_0(\frac{1}{5} - \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-1}{2\sigma^2}(x_i-\mu)^2})\right)$$

2. **[4 points]** (E-Step) For fixed μ, π_0, and σ^2, compute the posterior probabilities $q(z_i = 0 \mid x_i) = p(z_i = 0 \mid x, \mu, \pi_0, \sigma^2)$. Hint: Remember $P(B \mid A) = P(A \mid B)P(B)/\sum_B P(A \mid B)P(B)$. Your solution should be in terms of π_0, σ^2, μ, and the x_i. Write your final answer as: $q(z_i = 0 \mid x_i) =$ expression.

$$P(Z_i = 0|x, \mu, \pi_0, \sigma^2) = \frac{P(x_1, x_2 \ldots |z_i = 0)P(Z_i = 0)}{\sum_k P(x_1, x_2 \ldots |z_i = k)P(Z_i = k)}$$

$$= \frac{\frac{1}{5}\pi_0}{\frac{1}{\sqrt{2\pi}\sigma}e^{\left(\frac{-1}{2\sigma^2}(x_i-\mu)^2\right)}(1-\pi_0) + \frac{1}{5}\pi_0}$$

3. **[4 points]** (M-step for μ) Given the estimates of the posterior $q(z_i = 0 \mid x_i)$ and fixed π_0 and σ^2, now write down the update of μ. Your solution can include any of the following terms: $q(z_i = 0 \mid x_i), \pi_0, \sigma^2$ and the x_i. Write your final answer as: $\mu =$ expression.

$$\mu_k = \sum_i \frac{p(Z_i = k|X_i)x_i}{\sum_i p(Z_i = k|x_i)}$$

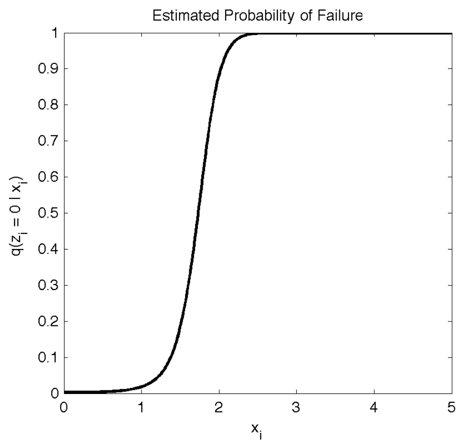
$$= \sum_i \frac{p(Z_i = 1|X_i)x_i}{\sum_i p(Z_i = 1|x_i)}$$

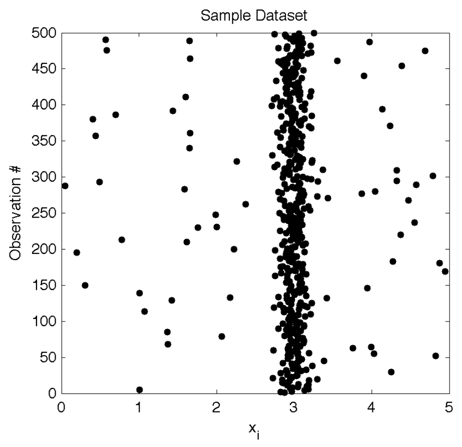
$$= \sum_i \frac{(1 - q(z_i = 0 \mid x_i))\,x_i}{\sum_i (1 - q(z_i = 0 \mid x_i))}$$

4. **[4 points]** (M-step for π_0) Given the estimates of the posterior $q(z_i = 0 \mid x_i)$, updated μ, and fixed σ^2, now write down the update for π_0. Your solution can include any of the following terms: $q(z_i = 0 \mid x_i), \mu, \sigma^2$ and the x_i. Write your final answer as: $\pi_0 =$ expression.

$$\pi_k = \frac{1}{n}\sum_i P(z_i = k|x_i)$$
$$\pi_0 = \frac{1}{n}\sum_i P(Z_i = 0|x_i)$$
$$= \frac{1}{n}\sum_i q(z_i = 0 \mid x_i)$$

5. **[4 points]** Suppose we will now apply our EM procedure to the sample dataset below. If we initialize $\mu = 0$ and $\pi_0 = 1/100$ (with parameter $\sigma^2 = 1/2$) we get the following distribution for $q(z_i = 0 \mid x_i)$ as a function of the observation x_i:





Note that the plot of the data has been rotated to align with the plot of $q(z_i = 0 \mid x_i)$. In practice, this initialization leads to μ incorrectly converging to a value close to zero on this dataset.

Briefly explain (2-3 sentences) why we might expect this failure to happen, given the above $q(z_i = 0 \mid x_i)$ distribution. *Hint: Look at where the majority of points lie with respect to* $q(z_i = 0 \mid x_i)$. *How will this affect the update for* π_0 *and* μ*?*

For a large number of values our mean reaches 0 and the π_k remains constant thus we see the distribution we are seeing

2 Pretty Crazy Awesome PCA [28 points]

2.1 Classy Axes

1. **[5 points]** For the graph in Figure 2, sketch the 2 principal components that would be found by PCA. Label them as "1" and "2" to indicate which is the first, and which is the second axis. You don't have to precisely calculate the equations for these axes; you should be able to figure out which directions are the principal components without needing complicated formulas and draw lines in approximately those directions. Remember that PCA ignores class labels.

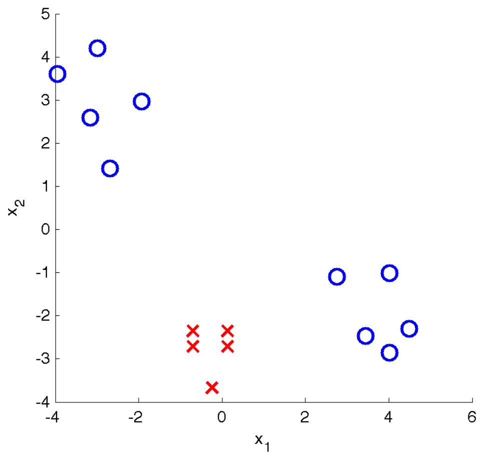


Figure 2: Graph of Mystery.

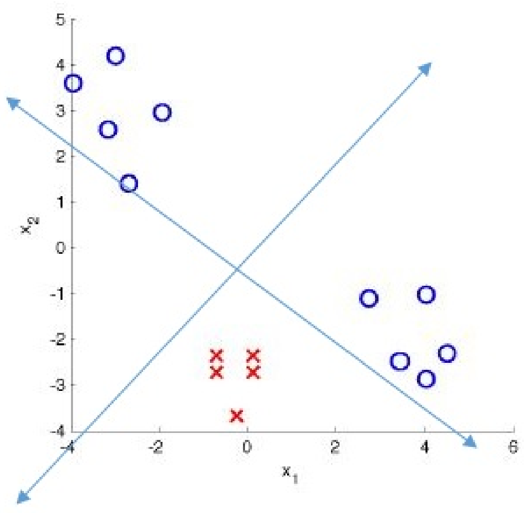


Figure 3: Principal Components

2. **[2 points]** If we project the data onto a single principal component, which component will allow a decision stump to perfectly separate the data: the first, the second, or both?

   Both

3. **[2 points]** Suppose instead we transform the data using a non-linear function of only one of the original axes: either $\phi(\mathbf{x}) = x_1^2$ or $\phi(\mathbf{x}) = x_2^2$. Which will allow a decision stump to perfectly separate the data?

   $\phi(\mathbf{x}) = x_1^2$ perfectly separates the data

2.2 Like a good neighbor, PCA is there

You showed in your first homework that nearest-neighbor classification can suffer when many irrelevant features are present. Here you will show that PCA can solve this problem by analyzing the covariance matrix of the features. Suppose we have two classes, $y = 1, 2$, and $P(\mathbf{x} \mid y)$ is Gaussian. For simplicity assume for all Gaussians below that the variance σ^2 is the same. Reminder: If $X \sim N(\mu, \sigma^2)$, then $\mathbf{E}[X] = \mu$, $Var[X] = \sigma^2$, and $\mathbf{E}[X^2] = \mu^2 + \sigma^2$. Also, recall that expectation is linear and it obeys the following four properties:

$$\begin{aligned}
\mathbf{E}[X + c] &= \mathbf{E}[X] + c \quad \text{for any constant } c \\
\mathbf{E}[X + Y] &= \mathbf{E}[X] + \mathbf{E}[Y] \\
\mathbf{E}[aX] &= a\mathbf{E}[X] \quad \text{for any constant } a \\
\mathbf{E}[X] &= \mathbf{E}_Y[\mathbf{E}_X[X \mid Y]] \quad \text{(iterated expectation)}
\end{aligned}$$

Using law of total variance we have

$$\begin{aligned}
Var(X) &= \mathbf{E}[Var(X|Y)] + Var[E(X|Y)] \\
\mathbf{E}[Var(X|Y)] &= P(Y=1)Var(X|Y=1) + P(Y=2)Var(X|Y=2) \\
&= \frac{1}{2}\sigma^2 + \frac{1}{2}\sigma^2 = \sigma^2
\end{aligned}$$

$$\begin{aligned}
Var(E[X|Y]) &= \frac{1}{2}(\mathbf{E}[X|Y=1])^2 + \frac{1}{2}(\mathbf{E}[X|Y=2])^2 - \left(\frac{1}{2}E[X|Y=1] + \frac{1}{2}E[X|Y=2]\right)^2 \\
&= 1
\end{aligned}$$

$$Var(X) = \sigma^2 + 1$$

1. **[4 points]** Suppose our data is generated according to a Gaussian mixture: first Y is sampled from the set of classes $\{1, 2\}$ with equal probability ($\frac{1}{2}$) of each class being selected, and then a *scalar* X is sampled according to the Gaussian $P(X \mid Y)$. Let 1 be the mean of class 1 and -1 be the mean of class 2. What is the variance of X? Your answer should be in terms of σ^2.

   The variance will remain the same since the feature is uninformative, it will have no impact on the variance
   $Var = \sigma^2 + 1$

2. **[3 points]** Now suppose a data point X' is generated according to same process as above, but the feature is uninformative, so the mean of class 1 and 2 are both zero. What is the variance of X'? Your answer should be in terms of σ^2.

   As we saw in part 1, for uninformative feature, since the mean is 0, the variance will be $\sigma^2 + 1$

3. **[3 points]** Suppose now we generate X as in part 1 and X' as in part 2 from a single Y. What is the *covariance* between X and X'? Recall that covariance is defined as $\mathbf{E}[(X - \mathbf{E}_X[X])(X' - \mathbf{E}_{X'}[X'])]$.

$$
\begin{aligned}
&\mathbf{E}[(X - \mathbf{E}_X[X])(X' - \mathbf{E}_{X'}[X'])] \\
&\qquad = \mathbf{E}\left[XX' - \mathbf{E}_X[X]X' - X\mathbf{E}_{X'} + \mathbf{E}_X[X']\mathbf{E}_{X'}[X']\right] \\
&\qquad\quad We\ have\ \mathbf{E}[X] = \mu_1 = 0 \\
&\qquad\quad E[X'] = \mu_2 = 0 \\
&\qquad = \mathbf{E}[XX'] \\
&\qquad = \mathbf{E}[X]\mathbf{E}[X'] \\
&\qquad = 0
\end{aligned}
$$

4. **[4 points]** Suppose we generate a dataset $(\mathbf{x}_1, y_1) \dots, (\mathbf{x}_n, y_n)$ by generating one informative feature X_1 as in part 1 and m non-informative features $X_2, \dots, X_{m+1}$ as in part 2 for each example. (An example 2D dataset is shown in Figure 4.) What are the eigenvalues of the covariance matrix of the data? Assume that $n \to \infty$, so that the covariances are exactly the true covariances that you computed in the previous questions. *Hint: The eigenvalues of a diagonal matrix are the diagonal entries of the matrix, and the eigenvectors are the standard bases* $e_1 = [1\ \ 0\ \ 0\ \ \dots\ \ 0]^\top$*, etc.*

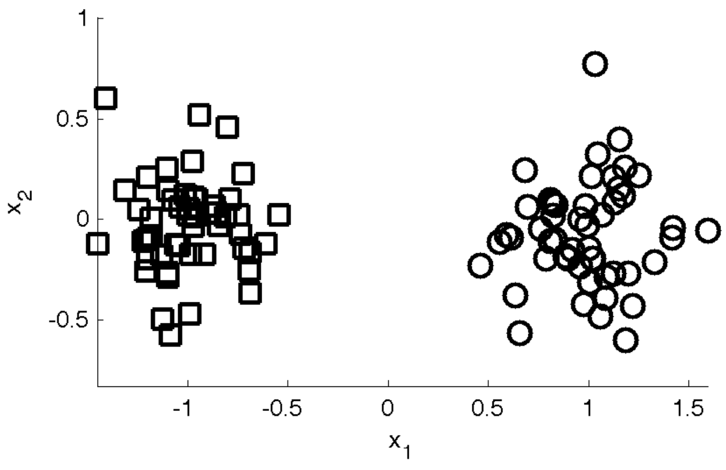


Figure 4: Sample dataset with only one informative dimension: x_1.

Since only the first feature is informative the covariance of the first feature matters

5. **[3 points]** Suppose we use a nearest-neighbor classifier on our data using the *first* principal component only. Briefly explain (2-3 sentences) why the ratio of intra-class to inter-class distances in the space defined by the first principal component will increase, decrease, or not change as m increases (as more uninformative features are added to the data).

No change

6. **[2 points]** Based on your previous answer, how will running PCA (and keeping only 1 component) before using K-NN affect the performance of K-NN as m increases? That is, will it increase, decrease, or have no effect on the accuracy of K-NN?

Increases the accuracy

3 Principal Component Analysis [30 points]

Task description. Here we will consider a dataset for optical character recognition (OCR). The dataset we provide consists of a sequence of words, one character per row.[1] The very first character of each word was capitalized in the original data and has been omitted for simplicity. The dataset is located in `data/ocr_data.mat`. The format of the data is described in `data/data_format.txt`, so please read that file before continuing.

In this problem you will use PCA to reduce the training data (the 0/1 pixel data of the training set) from 64 dimensions to 20 dimensions. To answer the questions you need to write code in Matlab to process the dataset. *You* **are** *allowed to use built-in matlab function for PCA, K-means, etc. for this problem.* Note, there is *no* automatic grader for this problem and you do not need to submit your code for grading.

1. **[5 points]** Using the top 2 PCA dimensions, display all the test letters "x" and "y", using crosses and circles respectively. An example of a plot of "x" vs "w" is shown in Figure 7 below.

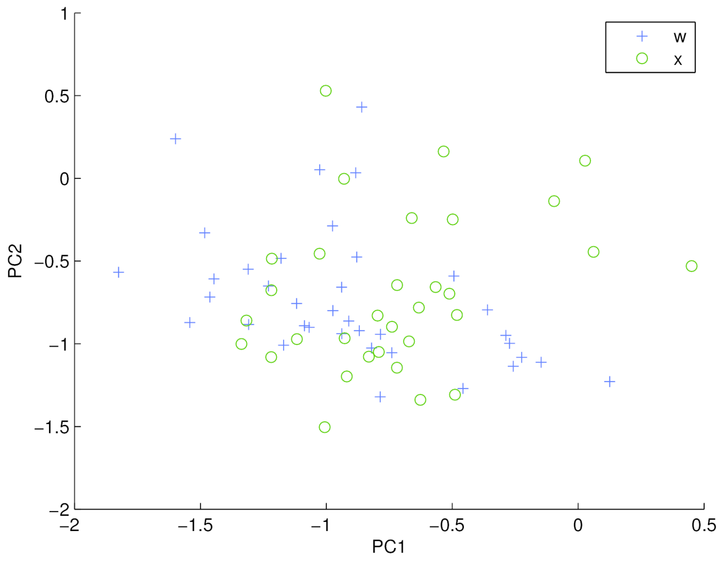


Figure 5: Plot of 'x'-'w' characters from top 2 PCA dimensions

[1]This dataset is a modified version of the dataset at `http://www.seas.upenn.edu/~taskar/ocr/`, which has been subsampled and slightly altered to simplify the processing.

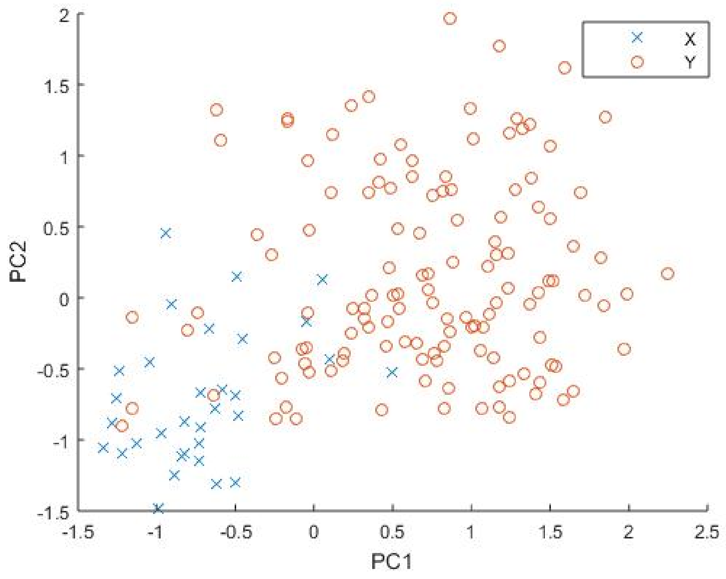


Figure 6: Plot of 'x'-'y' characters from top 2 PCA dimensions

2. **[5 points]** Plot the average (in sample) reconstruction error $||X - \hat{X}||_F$ of the digits as a function of the number of principle components that are included. How many principle components are required to get 90% reconstruction accuracy? (I.e., to explain 90% of the variance: $||X - \hat{X}||_F / ||X - \bar{X}||_F = 0.9$ where $\bar{X}|$ is a matrix, every row of which is the average x – the average image. Reminder: be sure to subtract that mean from X before you find the principle components, and to add it back in for your predictions.

   We get 90

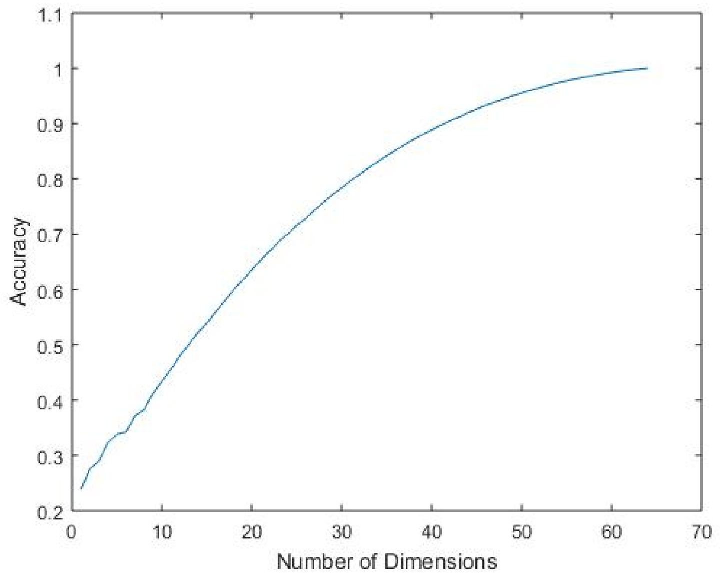


Figure 7: Error versus number of PCA dimensions dimensions

3. **[10 points]** Learn a Gaussian Naive Bayes classifier using MLE on the PCA-ed data and report accuracy on the test set using 5, 10 and 20 top dimensions. You should be learning a mean and variance for each class and each dimension. So 26 parameters for class priors, and 26 times (5,10 or 20) means plus 26 times (5,10, or 20) variances. In addition to reporting accuracies, write a brief description (3-4 sentences) of how these results compare to the performance of Naive Bayes on the original data. **Please include your source code in your write-up, but do not include built-in matlab functions.**

   Source Code:

$$load('./data/ocr_data.mat');$$
$$pooledLetters = [trainset.letter; testset.letter];$$
$$pooledPixels = [trainset.pixels; testset.pixels];$$
$$[coeff, score, latent] = pca(pooledPixels);$$
$$pooleddata_5 = score(:, 1:5);$$
$$pooleddata_10 = score(:, 1:10);$$
$$pooleddata_20 = score(:, 1:20);$$
$$trainsize = size(trainset.pixels, 1);$$
$$pooledSize = size(pooledPixels, 1);$$
$$traindata_5 = pooleddata_5(1 : trainsize, :);$$
$$testdata_5 = pooleddata_5(trainsize + 1 : pooledSize, :);$$
$$traindata_10 = pooleddata_10(1 : trainsize, :);$$
$$testdata_10 = pooleddata_10(trainsize + 1 : pooledSize, :);$$
$$traindata_20 = pooleddata_20(1 : trainsize, :);$$
$$testdata_20 = pooleddata_20(trainsize + 1 : pooledSize, :);$$
$$model_5 = fitcnb(traindata_5, trainset.letter);$$
$$model_10 = fitcnb(traindata_10, trainset.letter);$$
$$model_20 = fitcnb(traindata_20, trainset.letter);$$
$$[label_5, Posterior_5, Cost_5] = predict(model_5, testdata_5);$$
$$[label_10, Posterior_10, Cost_10] = predict(model_10, testdata_10);$$
$$[label_20, Posterior_20, Cost_20] = predict(model_20, testdata_20);$$
$$error_5 = find(label_5 \ = testset.letter);$$
$$error_10 = find(label_10 \ = testset.letter);$$
$$error_20 = find(label_20 \ = testset.letter);$$
$$errorval_5 = size(error_5, 1)/size(testset.letter, 1);$$
$$errorval_10 = size(error_10, 1)/size(testset.letter, 1);$$
$$errorval_20 = size(error_20, 1)/size(testset.letter, 1);$$
$$origModel = fitcnb(trainset.pixels, trainset.letter,' Distribution',' mvmn');$$
$$[label_orig, post_orig, cost_orig] = predict(origModel, testset.pixels);$$
$$errorOrig = find(label_orig \ = testset.letter);$$
$$errorValOrig = size(errorOrig, 1)/size(testset.letter, 1);$$

The error for Naive Bayes classifier when run on original data is more than when run on the PCA'ed data as we are including only the most important features.

Accuracy for Original Data = 0.5434
Accuracy for Naive Bayes with data with 5 dimensions = 0.4366
Accuracy for Naive Bayes with data with 10 dimensions = 0.6013
Accuracy for Naive Bayes with data with 20 dimensions = 0.6630

4. **[10 points]** Learn a k-means classifier on the samePCA-ed data and report accuracy on the test set using 5, 10 and 20 top dimensions. In addition to reporting accuracies, write a brief description (3-4 sentences) of how these results compare to the performance of The Gaussian Naive Bayes classifier. **Please include your source code in your write-up, but do not include built-in matlab functions.**

$$load('./data/ocr_data.mat');$$
$$pooledLetters = [trainset.letter; testset.letter];$$
$$pooledPixels = [trainset.pixels; testset.pixels];$$
$$[coeff, score, latent] = pca(pooledPixels);$$
$$pooleddata_5 = score(:, 1 : 5);$$
$$pooleddata_{10} = score(:, 1 : 10);$$
$$pooleddata_{20} = score(:, 1 : 20);$$
$$trainsize = size(trainset.pixels, 1);$$
$$pooledSize = size(pooledPixels, 1);$$
$$traindata_5 = pooleddata_5(1 : trainsize, :);$$
$$testdata_5 = pooleddata_5(trainsize + 1 : pooledSize, :);$$
$$traindata_{10} = pooleddata_{10}(1 : trainsize, :);$$
$$testdata_{10} = pooleddata_{10}(trainsize + 1 : pooledSize, :);$$
$$traindata_{20} = pooleddata_{20}(1 : trainsize, :);$$
$$testdata_{20} = pooleddata_{20}(trainsize + 1 : pooledSize, :);$$
$$[idx_5, C_5] = kmeans(traindata_5, 153,' MaxIter', 300);$$
$$[idx_{10}, C_{10}] = kmeans(traindata_{10}, 153,' MaxIter', 300);$$
$$[idx_{20}, C_{20}] = kmeans(traindata_{20}, 153,' MaxIter', 300);$$
$$clusterMap_5 = zeros(153, 1);$$
$$clusterMap_{10} = zeros(153, 1);$$
$$clusterMap_{20} = zeros(153, 1);$$
$$fori = 1 : 153$$
$$tempClusterVal = idx_5(:, 1) == i;$$
$$vals = trainset.letter(tempClusterVal);$$
$$clusterMap_5(i) = mode(vals);$$
$$tempClusterVal = idx_{10}(:, 1) == i;$$
$$vals = trainset.letter(tempClusterVal);$$
$$clusterMap_{10}(i) = mode(vals);$$
$$tempClusterVal = find(idx_{20}(:, 1) == i);$$
$$vals = trainset.letter(tempClusterVal);$$
$$clusterMap_{20}(i) = mode(vals);$$
$$end$$
$$closest_5 = zeros(size(testdata_5, 1), 1);$$
$$closest_{10} = zeros(size(testdata_{10}, 1), 1);$$
$$closest_{20} = zeros(size(testdata_{20}, 1), 1);$$
$$fori = 1 : size(testdata_5, 1)$$
$$distances_5 = sqrt(sum(bsxfun(@minus, C_5, testdata_5(i, :)).^2, 2));$$
$$closest_5(i) = clusterMap_5(distances_5 == min(distances_5));$$
$$distances_10 = sqrt(sum(bsxfun(@minus, C_10, testdata_10(i, :)).^2, 2));$$
$$closest_10(i) = clusterMap_10(distances_10 == min(distances_10));$$
$$distances_20 = sqrt(sum(bsxfun(@minus, C_20, testdata_20(i, :)).^2, 2));$$
$$closest_20(i) = clusterMap_20(distances_20 == min(distances_20));$$
$$end$$
$$error_5 = find(closest_5 \ = testset.letter);$$

$$error_10 = find(closest_10 \ = testset.letter);$$
$$error_20 = find(closest_20 \ = testset.letter);$$
$$errorval_5 = size(error_5, 1)/size(testset.letter, 1);$$

Accuracy for K-Means with 5 clusters = 0.5177
Accuracy for K-Means with 10 clusters = 0.6437
Accuracy for K-Means with 20 clusters = 0.6571
Accuracy for K-Means with original data = 0.6507

Accuracy for K-Means for 20 clusters is better than original

4 K-Means [11 points]

Given a set of points $\mathbf{x}_1, \dots, \mathbf{x}_n$, recall that the objective of k-means clustering is to minimize within-class sum of squares

$$J(\mu, r) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} ||\mu_k - \mathbf{x}_i||_2^2$$

where $\mu_1, \dots, \mu_K$ are the centroids of the clusters and r_{ik} are binary variables indicating whether data point $\mathbf{x}_i$ belongs to cluster k.

The optimization is NP-hard. Instead, as described in class, a greedy iterative clustering algorithm is used to alternatively update μ_k and r_{ik} to optimize $J(\mu, r)$.:

- Initialize K cluster centroids at random
- Alternate until convergence:
  - Assignment step: Assign points to the nearest centroid

$$\arg\min_r J(\mu, r) \rightarrow r_{ik} = \mathbf{1}(k = \arg\min_{k'} ||\mu_{k'} - \mathbf{x}_i||_2^2)$$

  - Update step: Set the centroid to be the mean of the points assigned to it

$$\arg\min_\mu J(\mu, r) \rightarrow \mu_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

1. **[4 points]** The greedy iterative clustering algorithm converges when the assignments no longer change. Is the algorithm guaranteed to converge in a finite number of steps? Briefly explain why or why not. *Hint: Think about the total possible number of assignments and the trend of the objective function after each iteration.*

   The greedy algorithm converges in a finite number of steps as at every step we always assign a point to a cluster. Thus the solution will always converge as at every step we assign the point to the closest cluster.

2. **[7 points]** Consider the toy 2D dataset in the following figure. We set $K = 2$ in this problem. The * marker indicates the data point and the circle marker indicates the starting cluster centroids. Show the update step and assignment step for each iteration until the algorithm converges. You can use as many of the remaining figures provided as you need.

   Please note that the first update step is given as the initialization. For each iteration, use one figure to mark the updated centroids based on the assignment from last iteration, then circle the data point assignment based on the updated centroid. We expect you to understand the detailed procedure of k-means. You should be able to compute the centroids manually with a calculator and assign data points with geometric intuition. Please be sure to show the coordinates of each centroid in every iteration.

   What to hand in. You can

   - *scan: use your phone or a scanner to take the image with your circle and include it in the .pdf you hand in or*

- *use a pdf tool like adobe acrobat to write directly on the pdf or*
- *run a matlab program, and just hand in the outputs at each iteration with the centroids and the list of points in each cluster (but you should see graphically what is happening.)*

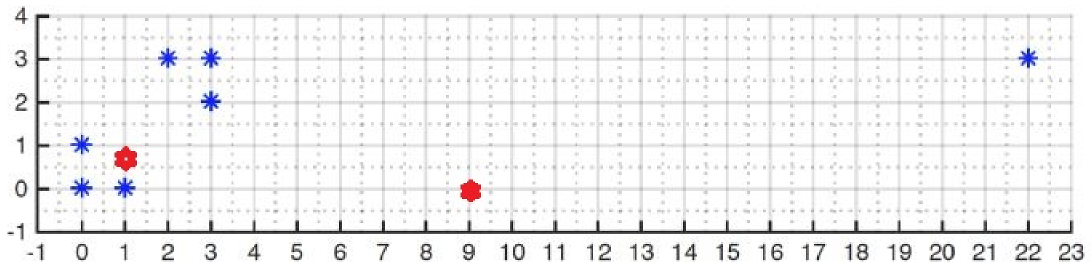


Figure 8: Figures for k-means

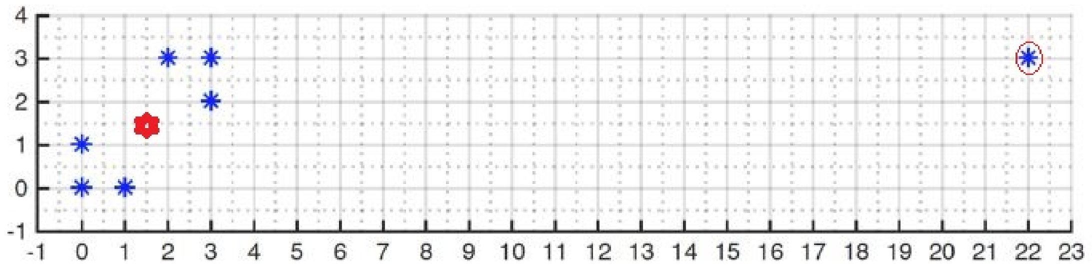


Figure 9: Figures for k-means

5 Semi-supervised learning [11 points]

In this exercise, we will use the same data set as in hw2 and hw5 for the task of handwritten digit recognition. Specifically, in this assignment, we will examine how neural net and PCA affect the performance of a model. We will also compare the performance of logistic regression and k-means on those data.

We will do 'semi-supervised learning' – use a large data set to learn dimensionality reduction, and then use the reduced dimension projection (PCA) or nonlinear transformation (auto-encoder) as features in supervised learning. Note that this will also be 'transductive learning' where the features vectors on the test set are also known at training time.

1. Train PCA and an auto-encoder on the big data (12,000 samples) (see **train.mat**, which has X_train 12,000*784 and Y_train 12,000 *1).
   Please use matlab function **pcacov** on the covariance matrix of X to train coefficients for each principle component. (see **pca_getpc.m** for details). For PCA, choose the number of principle components so as to keep 90% reconstruction accuracy. How many principle components do we need in this case? Use that many principle components.
   For the auto-encoder, the setup to train a model is basically the same as what we did in hw5, except that in this homework we change the hidden layer dimension from 100 to 200 (see **rbm.m** for details).

   Use num of principal components as 50.

2. Now we will do logistic regression on 3 different inputs:

   - The original features

- The PCA-ed data
- The auto-encoder outputs. To get the new features from auto-encoder, use the trained model in the question above, then plug in the original features into the auto-encoder.
  To generate new features from auto-encoder, please use the **newFeature_rbm** function in **hw7_kit**. Also note that after loading X_train and X_test, divide them by 255 so that each element in the matrices are less than 1. (See **test.m** for details)

You will be doing 10-way logistic regression (labels 1, ..., 9,10) and should use an L0 evaluation of the error (e.g. you get it right or get it wrong). A useful matlab function for this is **liblinear**
Usage:
for training **model = train(train_y, sparse(train_x),['-s 0','col'])**
for testing **[predicted_lable] = predict(test_y, sparse(test_x),['-q','col'])**
(see **logistic.m** for details).
Test your model on test data (load test.mat), what's the accuracy?
Compare the performance of the accuracy on test set of the three inputs. What's your observation?

Performance accuracy for original data = 0.8943
Performance accuracy for PCA'ed data = 0.9000
Performance accuracy for Auto-encoder data = 0.9466

The model with auto-encoder gives out the best data

3. Do the same thing using K-means, with k = 10, 25 respectively, but run the code on PCA-ed data and auto-encoder only. Test your model on test.mat, what's the accuracy?
Also, compare the performance of K-means with that of logistic regression.

Precision of Auto-encoder data with k-means for 10 and 25 = [0.6666;0.7651]
Precision of principal components with K-means for 10 and 25 = [0.5913;0.7493]

Precision of logistic regression is more than precision of kmeans