CIS 520, Machine Learning, Fall 2018: Assignment 6
Due: Sunday, November 11th, 11:59pm
[100 points]

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 Gradescope and SELECT PAGES when uploading to Gradescope. We will not accept paper copies of the homework.**

Note, there is no automatic grader for this homework but you do need to submit your code for grading. Please add your MATLAB code snippets inline with your solutions. We encourage you to use `mcode` or `listings` packages for LaTeX. Code submitted separately will not be graded.

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 **two 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. It is OK to submit the exact same code used by the partner who you listed in this HW submission.** 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.)

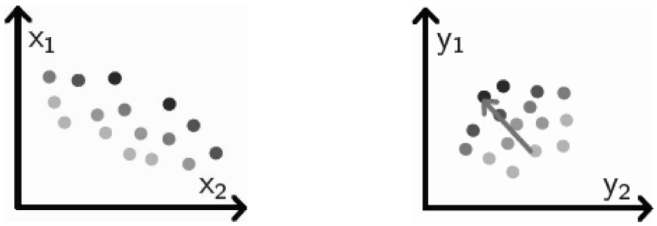
1 CCA [13 points]

Canonical correlation analysis (CCA) is a method similar to PCA, but instead of finding the directions of maximum variance (or minimum reconstruction error) within a matrix, it handles the situation that each data point (i.e., each observation) has two representations (i.e., two sets of features or "views"), e.g., a web page can be represented by the text on that page, and can also be represented by other pages linked to that page.

As a simple example, assume each data point has two representations x and y, each of which is a 2-dimensional feature vector, i.e., $x = [x_1; x_2]^T$ and $y = [y_1; y_2]^T$. Note that in general x and y can have different numbers of features in them. Given a set of data points, CCA finds a pair of projection directions $(u; v)$ to maximize the sample correlation $corr[(u^T x), (v^T y)]$ along the directions u and v. In other words, after we project the $x's$ onto u and the $y's$ onto v, the two projected representations $u^T x$ and $v^T y$ should be maximally correlated. Intuitively, data points with large values in one projected direction should also have large values in the other projected direction.

1. **[5 points]** Consider the data points shown in the figure below. The data x and y are paired – each point in the left figure corresponds to a specific point in the right figure and vice versa, since these two

points are two representations (views) of the same object. Different objects are shown in different gray scales in the two figures (so you should be able to approximately figure out how points are paired). In the right figure we've given one CCA projection direction v, draw the other CCA projection direction u in the left figure. (Problem adapted from Yi Zhang at CMU.)



2. **[8 points]** The formal solution for CCA is that the projections U and V are the (largest) left and right singular vectors of $Z = (X^TX)^{-1/2}X^TY(Y^TY)^{-1/2}$ where X and Y are the two "views" of the data, each containing n observations. Using the breast cancer data in `breast_cancer.mat`, define `X` = `X_train` and `Y` = `Y_train` and compute the single largest left and right singular vectors for the matrix Z as described above. These are the "canonical vectors" . Here, X is of dimension $n \times p_x$, Y of dimension $n \times p_y$, U of dimension $p_x \times k$, and V^T of dimension $p_y \times k$. Since we keep only one (largest) component, $k = 1$ in this case.

   (a) What is the correlation between XU and YV^T? This asks how correlated the two different estimates of the 'hidden state' from the two views are. CCA tries to make this number as big as possible. Note that in this problem we are looking at the "degenerate" case which the y view has only a single feature.

   (b) How does the above compare to the correlation between $\widehat{y}$ and y where $\widehat{y}$ is this the result of estimating y using only a single principal component, i.e. principal components regression (PCR) with a single component.

Stated Points:	13
Section Points:	13
Total Points So Far:	113
CORRECT TOTAL	

2 EM Algorithm with Red and Blue Coins [24 points]

Your friend has two coins: a red coin and a blue coin, with biases p_r and p_b, respectively (i.e. the red coin comes up heads with probability p_r, and the blue coin does so with probability p_b). She also has an inherent preference π for the red coin. She conducts a sequence of m coin tosses: for each toss, she first picks either the red coin with probability π or the blue coin with probability $1 - \pi$, and then tosses the corresponding coin; the process for each toss is carried out independently of all other tosses. You don't know which coin was used on each toss; all you are told are the outcomes of the m tosses (heads or tails). In particular, for each toss i, define a random variable X_i as

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th toss results in heads} \\ 0 & \text{otherwise.} \end{cases}$$

Then the data you see are the values $x_1, \ldots, x_m$ taken by these m random variables. Based on this data, you want to estimate the parameters $\theta = (\pi, p_r, p_b)$. To help with this, for each toss i, define a latent

(unobserved) random variable Z_i as follows:

$$Z_i = \begin{cases} 1 & \text{if the } i\text{-th toss used the red coin} \\ 0 & \text{otherwise.} \end{cases}$$

1. **[4 points]** Let X be a random variable denoting the outcome of a coin toss according to the process described above, and let Z be the corresponding latent random variable indicating which coin was used, also as described above (both X and Z take values in $\{0, 1\}$ as above). Write an expression for the joint distribution of X and Z. Give your answer in the form

$$p(x, z;\, \theta) = \underline{\qquad\qquad\qquad\qquad\qquad\qquad}\,.$$

2. **[6 points]** Write an expression for the complete-data log-likelihood, $\ln \mathcal{L}_c(\theta) = \sum_{i=1}^{m} \ln p(x_i, z_i;\, \theta)$.

3. **[6 points]** Suppose you knew the values z_i taken by the latent variables Z_i. What would be the maximum-likelihood parameter estimates $\widehat{\theta}$? Give expressions for $\widehat{\pi}$, $\widehat{p}_r$, and $\widehat{p}_b$ (in terms of x_i and z_i).

4. **[4 points]** In the absence of knowledge of z_i, one possibility for estimating θ is to use the EM algorithm. Recall that the algorithm starts with some initial parameter estimates θ^0, and then on each iteration t, performs an E-step followed by an M-step. Let θ^t denote the parameter estimates at the start of iteration t. In the E-step, for each toss i, the algorithm requires computing the posterior distribution of the latent variable Z_i under the current parameters θ^t. Calculate the posterior probability $\mathbf{P}(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$.

5. **[4 points]** For each toss i, denote the posterior probability computed in part (d) above by γ_i^t (so that $\gamma_i^t = \mathbf{P}(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$). Then the expected complete-data log-likelihood with respect to these posterior distributions is

$$\sum_{i=1}^{m} \Big(\gamma_i^t \cdot \ln p(x_i, 1;\, \theta) + (1 - \gamma_i^t) \cdot \ln p(x_i, 0;\, \theta) \Big)\,.$$

   The M-step of the EM algorithm requires finding parameters θ^{t+1} that maximize this expected complete-data log-likelihood. Determine the updated parameters θ^{t+1}. Give expressions for π^{t+1}, p_r^{t+1}, and p_b^{t+1} (in terms of x_i and γ_i^t).

3 Gaussian Mixture Models and the EM Algorithm [25 points]

Write a piece of MATLAB code to implement the EM algorithm for learning a Gaussian mixture model (GMM) given a data set $\mathbf{X}$, where $\mathbf{X}$ is an $m \times d$ matrix (m instances, each of dimension d), and a target number of Gaussians K: your program should take the training data $\mathbf{X}$ and the target number of Gaussians K as input and should output estimated parameters of a mixture of K Gaussians (specifically, it should output mixing coefficients $\widehat{} \in \Delta_K$, mean vectors $\widehat{}_1, \ldots, \widehat{}_K \in \mathbb{R}^d$, and covariance matrices $\widehat{}_1, \ldots, \widehat{}_K \in \mathbb{R}^{d \times d}$). For this problem, you are provided a 2-dimensional data set generated from a mixture of (3) Gaussians. The data set is divided into training and test sets; the training set has 480 data points and the test set has 120 data points. The goal is to learn a GMM from the training data; the quality of the learned GMM will be measured via its log-likelihood on the test data.

EM initialization and stopping criterion: Initialize the mixing coefficients to a uniform distribution, and each of the covariance matrices to be the $d \times d$ identity matrix: ${}^0 = (1/K, \ldots, 1/K)^\top$, and ${}_1^0 = \ldots = {}_K^0 = \mathbf{I}_d$. Initializations for the means will be provided to you (if you like, you can write your program to take the mean initialization as an additional input). For the stopping criterion, on each iteration t, keep track of

the (incomplete-data) log-likelihood on the training data, $\sum_{i=1}^{m} \ln p(x_i; \theta^t)$; stop when either the change in log-likelihood, $\sum_{i=1}^{m} \ln p(x_i; \theta^{t+1}) - \sum_{i=1}^{m} \ln p(x_i; \theta^t)$, becomes smaller than 10^{-6}, or when the number of iterations reaches 1000 (whichever occurs earlier; here θ^t denotes the parameter estimates on iteration t).

1. **[15 points] Known K.** For this part, assume you are given $K = 3$.

   (a) **Learning curve.** Use your implementation of EM for GMMs to learn a mixture of 3 Gaussians from increasing fractions of the training data (10% of the training data, then 20% of the training data, then 30% and so on upto 100%). You must use the subsets provided in the folder `TrainSubsets`); in each case, to initialize the means of the Gaussians, use the initializations provided in the folder `MeanInitialization` (see the `README.txt` file for more details). In each case, calculate the **normalized** (incomplete-data) log-likelihood of the learned model on the training data, as well as the normalized log-likelihood on the test data (here normalizing means dividing the log-likelihood by the number of data points); you can use the provided file `compute_nllh.m` to compute the normalized log-likelihood. In a single plot, give curves showing the normalized train and test log-likelihoods (on the y-axis) as a function of the fraction of data used for training (on the x-axis). (Note: the training log-likelihood should be calculated only on the subset of examples used for training, not on all the training examples available in the given data set.)

   (b) **Analysis of learned models.** For each of the learned models (corresponding to the different subsets of the training data), show a plot of the Gaussians in the learned GMM overlaid on the test data (you can use the code in `sample_plot.m` to help with the plotting). What do you observe? For the final model (learned from 100% of the training data), also write down all the parameters of the learned GMM, together with the (normalized) train and test log-likelihoods.

2. **[10 points] Unknown K.** Now suppose you do not know the right number of Gaussians in the mixture model to be learned. Select the number of Gaussians K from the range $\{1, \ldots, 5\}$ using 5-fold cross-validation on the full training data (use the folds provided in the folder `CrossValidation`). For each K, to initialize the means of the K Gaussians, use the initializations provided in the folder `MeanInitialization`. In a single plot, draw 3 curves showing the (normalized) train, test, and cross-validation log-likelihoods (on the y-axis) as a function of number of Gaussians K (on the x-axis); again, you can use the provided file `compute_nllh.m` to compute the normalized log-likelihood. Write down the chosen value of K (with the highest cross-validation log-likelihood).

4 K-Means [23 points]

Given a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, 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 $\mu_1, \ldots, \mu_K$ 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, the following 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. **[3 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.*

2. **[20 points]** Consider the toy 2D dataset in the following Figure 1 and Figure 2. We set $K = 3$ in this problem. The $*$ marker indicates the data points and the colored markers (blue circle, green triangle and red square) indicate the 3 starting cluster centroids. Notice that we have a slight different initiation from part (a) to part (b), but it might converge to very different results. For both cases, show the update step and assignment step for each iteration until the algorithm converges. You can use as many of the blank figures (Figure 3) 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 indicate 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
   - *write on pdf:* use a pdf tool like adobe acrobat to write directly on the pdf, or
   - *use MATLAB:* 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).

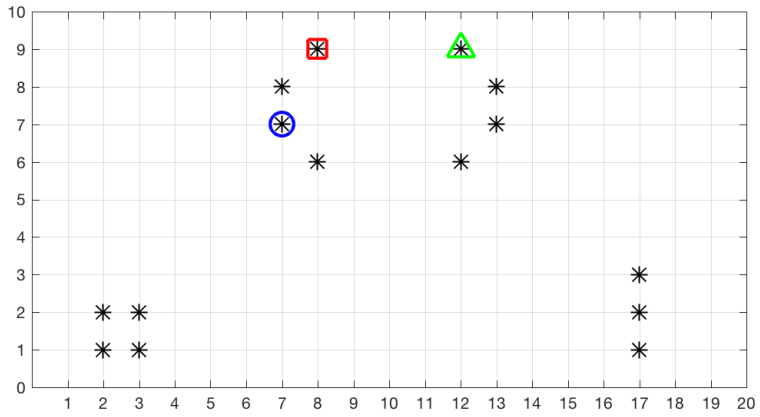


Figure 1: Part (a): Dataset with first initialization for K-means.

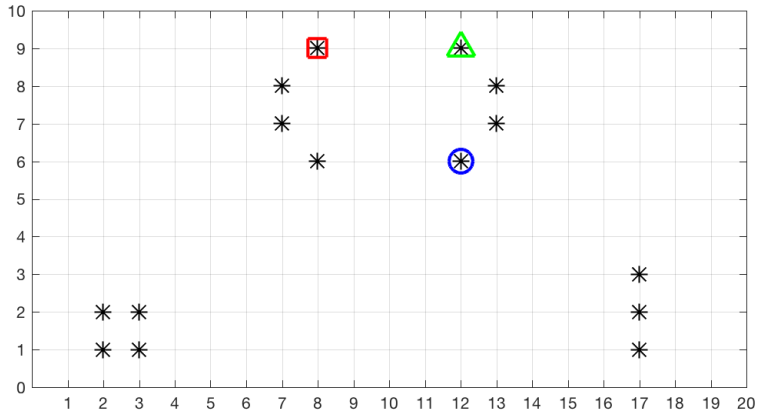


Figure 2: Part (b): Dataset with second initialization for K-means.

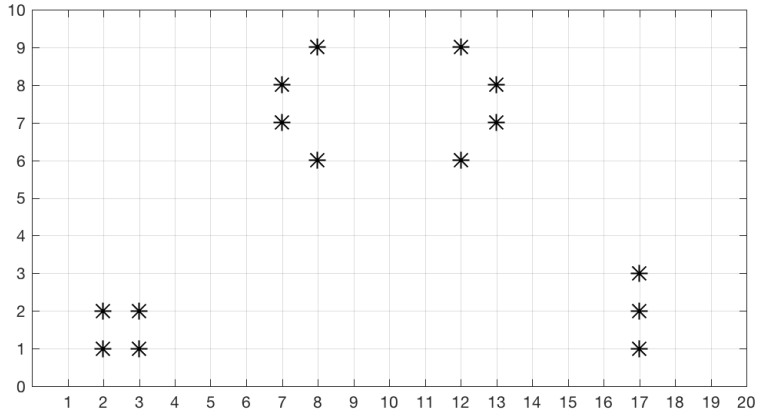


Figure 3: Blank dataset for you to implement K-means.

5 Performance Measures for Face Detection in Images [15 points]

Consider a face detection problem, where the goal is to build a system that can automatically detect faces in images. Two research groups develop systems for this problem using slightly different approaches. In both cases, the central component of the system is a binary classifier which, when applied to a 24×24 image, decides whether or not it is a face. The two groups train their classifiers using different learning algorithms. Moreover, when given a new image, they also apply their classifiers in slightly different ways: group A tests 24×24 regions of the image taking strides of size 2 (so, for example, for a 100×100 image, $(39)^2$ regions would be tested); group B tests 24×24 regions of the image taking strides of size 5 (so here, for a 100×100 image, only $(16)^2$ regions would be tested).[1] On a standard benchmark suite of test images that contains 300 faces altogether, the two groups have the following performances (assume the regions tested by both systems include all the 300 true face regions):

[1] In practice, the 24×24 classifier would also be applied to multiple scaled versions of the input image; we ignore this issue here for simplicity.

Research group	Number of regions tested	Number of faces detected correctly	Number of non-face regions detected as faces
A	20,000	280	100
B	12,500	270	60

1. **[10 points]** Based on the above numbers, calculate the TPR (recall), TNR, and precision of each group's system as tested. Also calculate the resulting geometric mean (GM) and F_1 measures for each system. If you were to select a system based on the GM measure, which system would you choose? Would your choice change if you were to select a system based on the F_1 measure? (Note, the geometrict mean (GM) is defined as $\sqrt{TPR \times TNR}$.)

2. **[5 points]** Which performance measure would be more suitable for this problem – the GM measure or the F_1 measure? Why?