10-601 Machine Learning: Homework 5

Out: Monday, October 24, 2016
Due 5:30 p.m. Wednesday, November 2, 2016
TAs: Ben Cowley, Varshaa Naganathan, Simon Shaolei Du

Instructions

- **Homework Submission:** Submit your answers and results to Gradescope. You will need to specify which pages go with which question. Please submit Questions 1.2, 1.3, 2, 3, and 4 as separate questions. We provide a LaTeX template which you can use to type up your solutions. Please check Piazza for updates about the homework.
- **Autolab Submission:** On the Homework 5 autolab page, you can click on the "download handout" link to download the submission template, which is a tar archive containing a blank placeholder pdf for your written solutions, a LaTeX template that you can use, and Octave `.m` files (in the `/code` folder) for each programming question. Replace each of these files with your solutions for the corresponding problem, remove any `.mat` files, and make sure you do not change the structure of the folder (i.e., leave the `.m` files inside the `/code` folder). To submit, make sure the top-level directory has the name `hw5`, and create a new tar archive with the command `tar -czvf hw5.tar hw5` (on linux + mac). Submit your archived solutions to Autolab by clicking the "Submit File" button.

  DO NOT change the name of any of the files or folders in the submission template. In other words, your submitted files should have exactly the same names as those in the submission template.
- **Collaboration policy**: The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes (including code) are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved. Please refer to the course webpage.

1 *k*-means Clustering [60 pts]

In this problem you will implement Lloyd's method for the k-means clustering problem and answer several questions about the k-means objective, Lloyd's method, and k-means++.

Recall that given a set $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ of n points in d-dimensional space, the goal of k-means clustering is to find a set of centers $c_1, \dots, c_k \in \mathbb{R}^d$ that minimize the k-means objective:

$$\sum_{j=1}^{n} \min_{i \in \{1,\dots,k\}} \|x_j - c_i\|^2, \tag{1}$$

which measures the sum of squared distances from each point x_j to its nearest center.

In class we discussed that finding the optimal centers for the k-means objective is NP-hard, which means that there is likely no algorithm that can efficiently compute the optimal centers. Instead, we often use Lloyd's method, which is a heuristic algorithm for minimizing the k-means objective that is efficient in practice and often outputs reasonably good clusterings. Lloyd's method maintains a set of centers $c_1, \dots, c_k$ and a partitioning of the data X into k clusters, $C_1, \dots, C_k$. The algorithm alternates between two steps: (i) improving the partitioning $C_1, \dots, C_k$ by reassigning each point to the cluster with the nearest center, and (ii) improving the centers $c_1, \dots, c_k$ by setting c_i to be the mean of those points in the set C_i for $i = 1, \dots, k$. Typically, these two steps are repeated until the clustering converges (i.e, the partitioning $C_1, \dots, C_k$ remains unchanged after an update). Pseudocode is given below:

1. Initialize the centers $c_1, \dots, c_k$ and the partition $C_1, \dots, C_k$ arbitrarily.
2. Do the following until the partitioning $C_1, \dots, C_k$ does not change:
   i. For each cluster index i, let $C_i = \{x \in X : x \text{ is closer to } c_i \text{ than any other center}\}$, breaking ties arbitrarily but consistently.
   ii. For each cluster index i, let $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.

1.1 Implementing Lloyd's Method

In the remainder of this problem you will implement and experiment with Lloyd's method and the k-means++ algorithm on the two dimensional dataset shown in Figure 1. **Note:** Use only native Octave functions—higher-level functions, such as `pdist2`, are not supported by Autolab. Autolab should take ~1 min to grade your submission. If it is taking longer, there is an error, most likely with convergence.

(a) [8 pts] Complete the function `[a] = update_assignments(X, C, a)`. The input `X` is the $n \times d$ data matrix, `C` is the $k \times d$ matrix of current centers, and `a` is the $n \times 1$ vector of current cluster assignments. That is, `C(i,:)` is the center for cluster i, and `a(j)` denotes the cluster label $(1, \dots, k)$ for the j^{th} data point. Your function should output a new $n \times 1$ vector of cluster assignments so that each point is assigned to the cluster with the center that has the smallest Euclidean distance with the point.

Solution:

```
function a = update_assignments(X, C, a)
  % Compute the matrix of all pairwise distances between rows of X and C
  n = size(X, 1);
  k = size(C, 1);
  D = zeros(n, k);
  for i = 1:k
    D(:, i) = sum(bsxfun(@minus, X, C(i,:)) .^2, 2);
  end
  % Update the assignments vector to be the argmin of each row of D.
  [~, a] = min(D, [], 2);
end
```

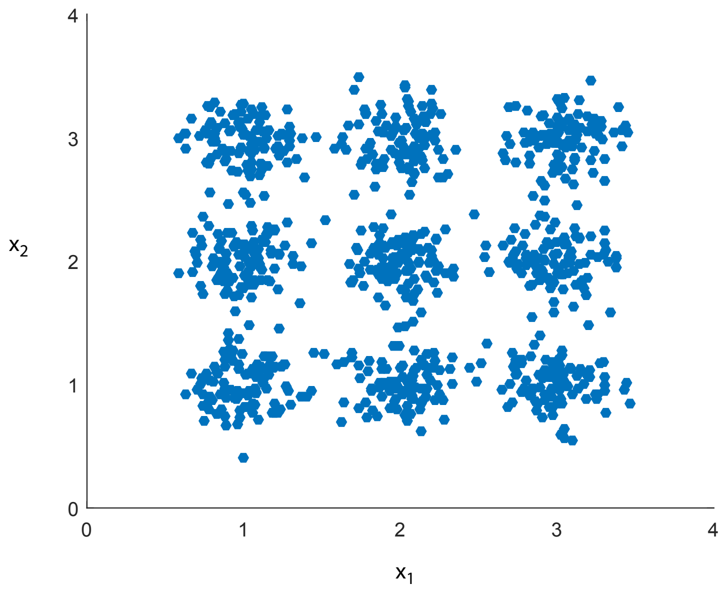


Figure 1: The 2D dataset X used for problem 1.

(b) [8 pts] Complete the function `[C] = update_centers(X, C, a)`. The input arguments are as in part (a). Your function should output a $k \times d$ matrix `C` whose i^{th} row is the optimal cluster center for those points in cluster i.

Solution:

```
function C = update_centers(X, C, a)
  k = size(C, 1);
  for i = 1:k
    % Set C(i,:) to be the mean of those rows of X for which a == i.
    C(i,:) = mean(X(a == i,:), 1);
  end
end
```

(c) [8 pts] Complete the function `[C, a] = lloyd_iteration(X, C0)`. This function takes a data matrix `X`, initial centers `C0` and runs Lloyd's method until convergence. Specifically, alternate between updating the assignments and updating the centers until the the assignments stop changing. Your function should output the final $k \times d$ matrix `C` of centers and final the $n \times 1$ vector `a` of assignments.

Solution:

```
function [C, a] = lloyd_iteration(X, C)
  a = zeros(size(X,1), 1);
  while 1
    % Compute the new assignments
    a_new = update_assignments(X, C, a);
    % If none of them have changed since the last iteration then halt
    if all(a_new == a)
      break
    end
    % Update the assignments
    a = a_new;
    % Update the centers
    C = update_centers(X, C, a);
  end
end
```

(d) [8 pts] Complete the function `[obj] = kmeans_obj(X, C, a)`. This function takes the $n \times d$ data matrix `X`, a $k \times d$ matrix `C` of centers, and a $n \times 1$ vector `a` of cluster assignments. Your function should output the value of the k-means objective of the provided clustering.

Solution:

```
function obj = kmeans_obj(X, C, a)
  k = size(C, 1);
  obj = 0.0;
  for i = 1:k
    % Compute X(j,:) - C(i,:) for each row of X with a(j) == i.
    deltas = bsxfun(@minus, X(a == i, :), C(i, :));
    % Add the sum of squared norms of the rows of delta to the objective.
    obj = obj + sum(sum(deltas.^2));
  end
end
```

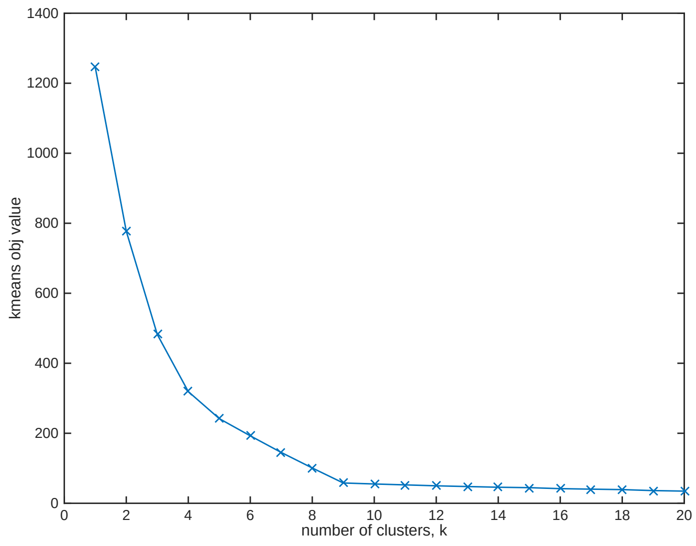
We provide you with a function `[C, a, obj] = kmeans_cluster(X, k, init, num_restarts)` that takes an $n \times d$ data matrix `X`, the number `k` of clusters, a string `init` which must be either `'random'` or `'kmeans++'`, and a number of restarts `num_restarts`. This function runs Lloyd's method `num_restarts` times and outputs the best clustering it finds. You may use this function when answering the following questions. When `init` is set to `'random'` then the centers are initialized uniformly at random, and when `init` is set to `'kmeans++'` the centers are initialized using the D^2 weighting of k-means++.

1.2 Experiment 1: The effect of *k* on Lloyd's method

In parts (e) and (f) of this problem you will investigate the effect of the number of clusters k on Lloyd's method and a simple heuristic for choosing a good value for k. You can load the data for this problem by running the command `load kmeans_data` in Matlab or Octave. This will introduce a 900×2 matrix `X`, where each row corresponds to a single point in the dataset shown in Figure 1.

(e) [6 pts] Finish coding the script `plot_kmeans_sweep_k.m`, which plots the k-means objective value obtained by running `kmeans_cluster(X, k, 'random', 10)` for each value of `k` in $\{1, \dots, 20\}$. The x-axis of your plot should be the value of `k` used, and the y-axis should be the `k`-means objective. Include only the plot in your written solutions.

Solution:



Not required in the solution:

```
% Script that plots kmean obj vs. number of clusters, k
%
```

```
% Finish the script where indicated
%
% HW5, 1.f

load kmeans_data
    % returns X (N x 2): data matrix for N samples

objs = [];  % store obj values for each k

%%%%%% COMPLETE CODE HERE %%%%%%%%
% Use kmeans_cluster.m to obtain
%   the objective function valuesfor k=1,...,20
% Save objective function values in the variable 'objs'

for k = 1:20
  [~, ~, obj] = kmeans_cluster(X, k, 'random', 10);
  objs(k) = obj;
end

%%%%%% DO NOT CHANGE FOLLOWING CODE %%%%%%%%
f = figure;
plot(objs, 'x-');
ylabel('kmeans obj value');
xlabel('number of clusters, k');
```

(f) [6 pts] One heuristic for choosing the value of k is to pick the value at the "elbow" or bend of the plot produced in part (e), since increasing k beyond this point results in relatively little gain. Based on your plot, what value of k would you choose? Does this agree with your intuitions when looking at the data?

Solution: The elbow in the plotted curve occurs at $k = 9$. Looking at the plot of the data, we see that there are 9 reasonably well separated Gaussian clusters. So the value of $k = 9$ matches the intuition.

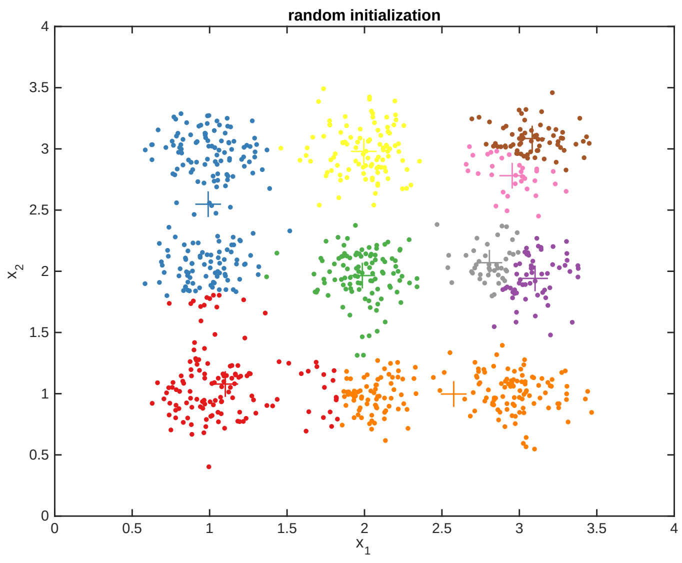
1.3 Experiment 2: The effect of initialization on Lloyd's method

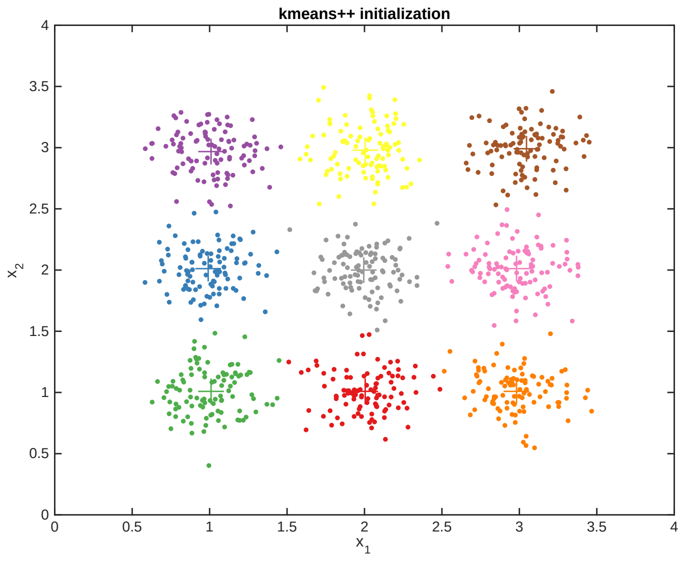
In parts (h) and (i) you will investigate the effect of the center initialization on Lloyd's method. In particular, you will compare random initialization with the k-means++ initialization. Note that the provided `kmeans_cluster` function has both kinds of initialization already implemented. In this problem, you just need to run the code and answer questions.

(g) [4 pts] For both types of intialization, use the script `plot_kmeans_init_projs.m` to plot projections similar to Figure 1 except that the points are colored based on their class label, as well as the optimized centers. Include both plots in your written solutions.

Solution:

Give full credit for two plots. Random initialization should look worse (clusters not clearly identified), but `kmeans++` may show errors as well, depending on the initialization.





(h) [6 pts] For each value of `init` in {`'random'`, `'kmeans++'`}, report the mean and standard deviation of the k-means objective value returned by `kmeans_cluster(X, 9, init, 1)` over 1000 runs. Note that `num_restarts` = 1. Since the initialization is random, your answers will vary each time you perform this experiment. However, there should be a clear gap in performance between the two initialization methods. Submit the values and the script code.

Solution:

```
load kmeans_data
rng(10601);
num_trials = 1000;
for init = {'random', 'kmeans++'}
  objs = [];
  for i = 1:num_trials
    [~, ~, obj] = kmeans_cluster(X, 9, init{1}, 1);
    objs(i) = obj;
  end
  fprintf('Average for init=%s: %f +/- %f\n', init{1}, mean(objs), std(objs))
end
```

The output of the above code gives average objective value of 87.6 ± 25.4 for random initialization and 75.9 ± 21.25 for k-means++ initialization. These values are random, so we will accept any output for which the random initialization objective is between 85 to 90, the k-means++ initialization objective is between 73 to 78, and the standard deviation is between 20 to 25.

(i) [6 pts] Intuitively, a good center initialization for the 2D dataset shown in Figure 1 is when we pick one center from each of the 9 natural Gaussian clusters. When two centers land in the same Gaussian cluster, they will split that cluster in half and leave only 7 centers to cover the remaining 8 Gaussian clusters. Explain in 1–2 sentences why using k-means++ initialization is more likely to choose one sample from each cluster than random initialization.

Solution: When using k-means++ initialization, each point is chosen to be the j^{th} center with probability proportional to its squared distance to the first $j-1$ centers. For this dataset, once a center has been chosen from a given Gaussian cluster, points belonging to any other Gaussian cluster will be further away than points in the same Gaussian cluster, so k-means++ will be more likely to sample them.

2 Z-scoring before applying PCA? [20 pts]

Consider the $3 \times N$ matrix X be a dataset with 3 variables and N samples. PCA follows these steps when applied to X:

1. recenter X such that the sample mean $\frac{1}{N}\sum_{n=1}^{N} X_n = \mathbf{0}$
2. identify ordered orthonormal principal components $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \in \mathbb{R}^3$
3. project recentered X onto the principal components: $\mathbf{u}_i^T X$
4. identify ordered variances $\sigma_1, \sigma_2, \sigma_3$ for the principal components,

   where $\sigma_i = \frac{1}{N}\sum\limits_{n=1}^{N}(\mathbf{u}_i^T X_n)^2$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3$

We can also think about PCA as performing an eigendecomposition on the 3×3 covariance matrix Σ. Given the eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ (i.e., the basis vectors), we can compute each eigenvalue (i.e., the variances) as $\sigma_i = \mathbf{u}_i^T \Sigma \mathbf{u}_i$. PCA orders the eigenvalues such that $\sigma_1 \geq \sigma_2 \geq \sigma_3$.

Now consider drawing the columns of $X = [X_1, \ldots, X_N]$ from the given multivariate Gaussian distribution:

$$X_n \sim \mathcal{N}\left(\mu = \begin{bmatrix} 5 \\ 5 \\ 10 \end{bmatrix}, \Sigma = \begin{bmatrix} 6 & -3 & 0 \\ -3 & 6 & 0 \\ 0 & 0 & 10 \end{bmatrix}\right)$$

We apply PCA to X and ask the following questions.

Note: For the following problems, feel free to use Matlab's built-in `pca`, `mvnrnd`, `cov2corr`, etc., to help guide your answers. Use $N \geq 1000$ samples when generating the data X. Do not include code in your answers. Provide analytical, exact solutions; do *not* give numerical solutions from Matlab.
Hint: The fastest way to solve these problems is with intuition and to check with Matlab.

1. [4 pts.] What are the ordered variances $\sigma_1, \sigma_2, \sigma_3$ for the ordered principal components?

   Solution:
   $\sigma_1 = 10$
   $\sigma_2 = 9$
   $\sigma_3 = 3$

2. [4 pts.] What are the ordered basis vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \in \mathbb{R}^3$ for the ordered principal components?

   Solution:

$$[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \end{bmatrix}$$

One limitation of PCA is that it is scale variant. For example, consider two variables, where one represents age (in years) and one represents height (in feet). Because the range of ages (0 to 100 years) is much larger than the range of height (4 to 7 feet), PCA will trivially identify the age variable for the first principal component. However, the first principal component should be able to reconstruct both age and height, which are likely correlated.

To overcome this, a typical procedure before applying PCA is to z-score each variable (i.e., for each x_i, subtract its mean and divide by its standard deviation). This transforms the covariance matrix Σ into a correlation matrix $\tilde{\Sigma}$. Note that $\hat{\Sigma}_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}}\sqrt{\Sigma_{jj}}}$.

3. [3 pts.] Let's say we z-score X to obtain $\tilde{X}$. Then, $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \tilde{x}_3]^T \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$. What are the elements of $\tilde{\mu}$ and $\tilde{\Sigma}$?

   Solution: Note that

$$\tilde{X} = \begin{bmatrix} (x_1 - 5)/\sqrt{6} \\ (x_2 - 5)/\sqrt{6} \\ (x_3 - 10)/\sqrt{10} \end{bmatrix}.$$

   Now using the definition of mean and covariance matrix, we obtain

$$\tilde{X} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix} = \mathcal{N}\left(\tilde{\mu} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tilde{\Sigma} = \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

We now apply PCA to $\tilde{X}$.

4. [3 pts.] What are the ordered variances $\tilde{\sigma}_1, \tilde{\sigma}_2, \tilde{\sigma}_3$ for the ordered principal components?

   Solution:
   $\sigma_1 = 1.5$
   $\sigma_2 = 1$
   $\sigma_3 = 0.5$

5. [3 pts.] What are the ordered basis vectors $\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \tilde{\mathbf{u}}_3 \in \mathbb{R}^3$ for the ordered principal components?

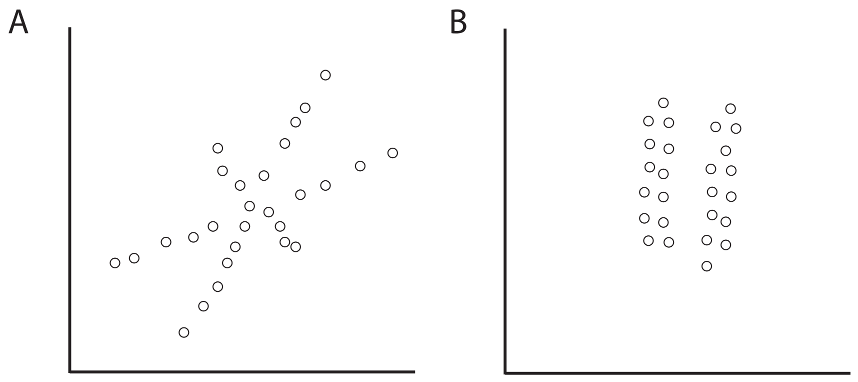
   Solution: $[\tilde{u}_1, \tilde{u}_2, \tilde{u}_3] = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix}$

6. [3 pts.] Do your answers for 4 and 5 differ from your answers to 1 and 2? Why or why not?

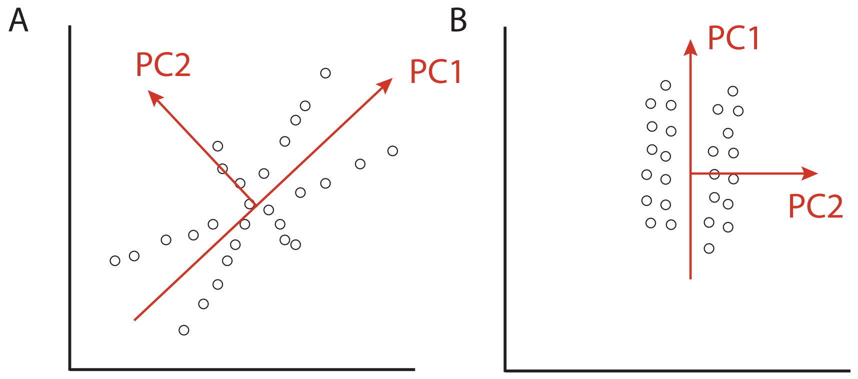
   Solution: Yes. Because we z-scored the data, x_3 decreased its relative variance to x_1 and x_2. The correlation between x_1 and x_2 then dominate the first principal component.

3 Lots of dots [20 pts]

1. [10 pts] For each of the two data sets shown below, draw vectors that correspond to the first and second principal components that would be produced by Principal Components Analysis (PCA). For each diagram, label which vector is the first component, and which is the second.



Solution:



2. [5 pts] Is it possible for you to draw a third principal component for the first of these datasets (in A)? Please do, or explain in one sentence why it is impossible.

   Solution: This is not possible because PCA cannot have more principal components than the number of features (which is 2 for this dataset).

3. [5 pts] PCA is typically performed before clustering high-dimensional data. For this procedure, what assumption do we make about the data's covariance structure in relation to the distances between clusters? Explain why this assumption holds or does not hold for the second dataset (in B). Be brief (2-3 sentences).

   Solution: We make the assumption that the directions of largest covariance are dominated by the distances between clusters rather than the inter-cluster distances. The assumption does not hold for the second dataset because the clusters largely overlap in the first principal component but do not overlap in the second principal component.

4 PCA two ways [Extra Credit: 10 pts]

Like the primal and dual problems for SVM, we can have two perspectives of PCA: maximizing the variance or minimizing the reconstruction error. We will find that both these views lead to the same optimization problem. This question uses a lot of linear algebra—make sure the sizes of each matrix and vector correctly match up!

4.1 [Extra credit: 5 pts] PCA: Maximizing the variance

Consider N data points $X_1, \dots, X_N \in \mathbb{R}^p$, such that the sample mean of X is zero: $\frac{1}{N}\sum_{i=1}^{N} X_i = \mathbf{0}$. Also consider projection vector $\mathbf{u} \in \mathbb{R}^p$, where $\|\mathbf{u}\|_2 = 1$. We would like to maximize the sample variance $\tilde{V}[\mathbf{u}^T X]$, which is the sample variance of X projected onto $\mathbf{u}$. The sample variance of N samples of a variable z is $\tilde{V}[z] = \frac{1}{N}\sum_{i=1}^{N} z_i^2$, where $\frac{1}{N}\sum_{i=1}^{N} z_i = 0$. We can write an optimization problem to maximize the sample variance:

$$\max_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \quad \tilde{V}[\mathbf{u}^T X] \tag{2}$$

Reformulate Eqn. 2 to the following optimization problem, and define the $p \times p$ matrix Σ:

$$\max_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \quad \mathbf{u}^T \Sigma \mathbf{u} \tag{3}$$

Show all work:

Solution: $\tilde{V}[\mathbf{u}^T X] = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}^T X_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}^T X_i)(\mathbf{u}^T X_i)^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}^T X_i X_i^T \mathbf{u}$
$= \mathbf{u}^T[\frac{1}{N}\sum_{i=1}^{N} X_i X_i^T]\mathbf{u} = \mathbf{u}^T\Sigma\mathbf{u}$, where $\Sigma = \frac{1}{N}\sum_{i=1}^{N} X_i X_i^T$.

Thus, the two objective functions are the same.

4.2 [Extra credit: 5 pts] PCA: Minimizing the reconstruction error

Consider the same variables as those defined in Question 4.1. Instead of maximizing the projected variance, we now seek to minimize the reconstruction error. In other words, we would like to minimize the difference between X and the reconstructed $\mathbf{u}\mathbf{u}^T X$. Note that $\mathbf{u}\mathbf{u}^T X$ first projects X onto $\mathbf{u}$, and then projects this scalar back into the p-dimensional space along the $\mathbf{u}$ axis. Minimzing the reconstruction error can be written as the following optimization problem:

$$\min_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \frac{1}{N}\sum_{i=1}^{N} \|X_i - \mathbf{u}\mathbf{u}^T X_i\|_2^2 \tag{4}$$

Reformulate Eqn. 4 to the following optimization problem, with the same Σ as defined in Section 4.1.

$$\max_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \mathbf{u}^T \Sigma \mathbf{u} \tag{5}$$

Show all work:

Solution: Note $\frac{1}{N}\sum_{i=1}^{N} \|X_i - \mathbf{u}\mathbf{u}^T X_i\|_2^2 = \frac{1}{N}\sum_{i=1}^{N} (X_i - \mathbf{u}\mathbf{u}^T X_i)^T (X_i - \mathbf{u}\mathbf{u}^T X_i)$
$= \frac{1}{N}\sum_{i=1}^{N} X_i^T X_i - 2(\mathbf{u}^T X_i)^2 + \mathbf{u}^T\mathbf{u}(\mathbf{u}^T X_i)$.

We can ignore the $X_i^T X_i$ term because it does not depend on $\mathbf{u}$. Also, we can make use of the constraint that $\|\mathbf{u}\|_2 = 1$ to let $\mathbf{u}^T\mathbf{u} = 1$. This leads to the optimization problem:

$$\min_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} -\frac{1}{N}\sum_{i=1}^{N} (\mathbf{u}^T X_i)^2 = \max_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \frac{1}{N}\sum_{i=1}^{N} (\mathbf{u}^T X_i)^2$$

This optimization problem has the same objective function as in the solution for Question 4.1, which leads to the optimization problem:

$$\max_{\substack{\mathbf{u} \\ \text{s.t. } \|\mathbf{u}\|_2=1}} \mathbf{u}^T \Sigma \mathbf{u}$$