1 consider dataset $\{x_1, \ldots, x_N\}$. $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|_2^2$.

a) what is minimum value of J when $k = n$?

↳ suppose each datapoint aligns with the prototype of some cluster. Then each prototype has 1 datapoint in it such that

- For each k, $r_{nk} = 1$ for only one datapoint x_n.
- For that $r_{nk} = 1$, $\|x_n - \mu_k\|_2^2 = 0$.

Then, $J = 0$.

Since $J \geq 0$, it follows that the minimum of J is $\boxed{J = 0}$.

(sum of squared terms)

b) consider the regularized objective function $J = \sum_{k=1}^{K} \left[\lambda \|\mu_k\|_2^2 + \sum_{n=1}^{N} r_{nk} \|x_n - \mu_k\|_2^2 \right]$.

Suppose all r_{nk} are known. Find $\underset{\mu_k}{\operatorname{argmin}} \left(\lambda \|\mu_k\|_2^2 + \sum_{n=1}^{N} r_{nk} \|x_n - \mu_k\|_2^2 \right)$.

$$\frac{\partial J}{\partial \mu_k} = \frac{\partial}{\partial \mu_k} \left(\lambda \|\mu_k\|_2^2 + \sum_{n=1}^{N} r_{nk} \|x_n - \mu_k\|_2^2 \right)$$

$$= 2\lambda \mu_k + \sum_{n=1}^{N} -2 r_{nk} (x_n - \mu_k)$$

$$= 0$$

$$\Rightarrow 2 \left(\lambda \mu_k - \sum_{n=1}^{N} r_{nk} (x_n - \mu_k) \right) = 0$$

$$\Rightarrow \lambda \mu_k = \sum_{n=1}^{N} r_{nk} x_n - \sum_{n=1}^{N} r_{nk} \mu_k$$

$$\Rightarrow \mu_k \left(\lambda + \sum_{n=1}^{N} r_{nk} \right) = \sum_{n=1}^{N} r_{nk} x_n$$

$$\Rightarrow \mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\lambda + \sum_{n=1}^{N} r_{nk}}$$

→

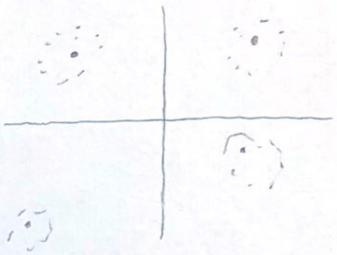
-Zack Bergin

The optimal μ_k is $\mu_k = \dfrac{\sum_{n=1}^{N} r_{nk} x_n}{\lambda + \sum_{n=1}^{N} r_{nk}}$.

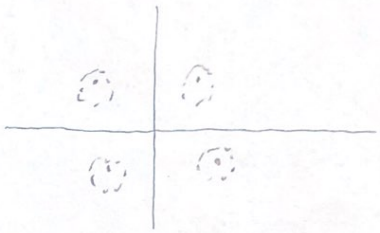
This regularization would affecting the refitting step of K-means because it directly affects how μ_k is calculated.

Namely, as λ gets large, μ_k approaches 0.

Consequently, regularization causes all of the cluster centers to concentrate toward each other at the origin.



unregularized



regularized

2] Unlabeled data $x_n \in \mathbb{R}^m$, $n = 1, \dots, N$. Use L_1 distance; $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|_1$.

Let $C_k = \{ n \mid r_{nk} = 1 \}$. Goal: minimize $f(\mu_k) = \sum_{n \in C_k} \| x_n - \mu_k \|_1$ wrt μ_k.

Let x_{ni} be i^{th} element in x_n, μ_{ki} be i^{th} element in μ_k.

Then, $f(\mu_k) = \sum_{n \in C_k} \| x_n - \mu_k \|_1 = \sum_{n \in C_k} \sum_{i=1}^{m} | x_{ni} - \mu_{ki} | = \sum_{i=1}^{m} \sum_{n \in C_k} | x_{ni} - \mu_{ki} |$.

Hence, we can optimize for each element of μ_k separately.

↳ Note: $\sum_{n \in C_k}$ iterates over all data points in k^{th} cluster.

- - -

a) Find $\partial f(x)$ of $f(x) = |x|$ for $x < 0$, $x > 0$ and $x = 0$.

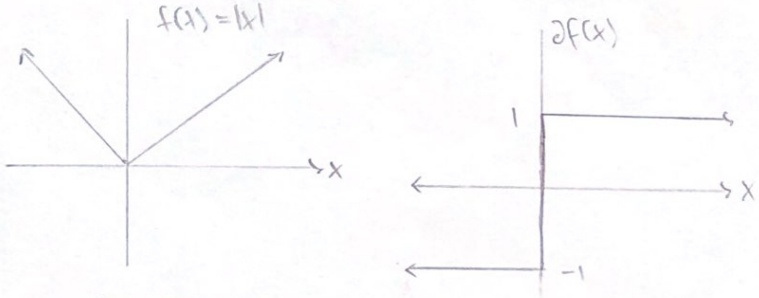
When $x < 0$, $f(x)$ is differentiable with $f'(x) = -1$.
Hence the subgradient, which equals the derivative of f at points where f is differentiable, is given by $\partial f(x) = \{-1\}$.

Similarly when $x > 0$, $\partial f(x) = \{+1\}$.

At $x = 0$, use the definition of subgradient to write that $\forall z \in \text{dom} f$, g satisfies $f(z) \geq f(0) + g(z - 0) \iff |z| \geq gz \ \forall z$.

$|z| \geq gz$ is satisfied iff subgradient $g \in [-1, 1]$.
Hence for $x = 0$, $\partial f(x) = [-1, 1]$.

Graphical representation:



$$\partial f(x) = \begin{cases} \{-1\}, & x < 0 \\ [-1, 1], & x = 0 \\ \{+1\}, & x > 0 \end{cases}$$

b) Show that $\bar{y}^* = y_{\frac{N_k+1}{2}}$ minimizes $\sum_{j=1}^{N_k} |y_j - \bar{y}| = f(\bar{y})$ given the assumptions.

Suppose $f(\bar{y})$ is convex and subdifferentiable everywhere.

Then, f is subdifferentiable at $\bar{y}^* = y_{\frac{N_k+1}{2}}$.

Since N_k is odd, $\bar{y}^*$ exists and is the median of $\{y_1, \dots, y_{N_k}\}$.

Since $y_1 < y_2 < \cdots < y_{N_k}$, the following is true:

- $(y_j - \bar{y}^*) < 0$ for $j = 1, \dots, \frac{N_k+1}{2} - 1$
- $(y_j - \bar{y}^*) = 0$ for $j = \frac{N_k+1}{2}$
- $(y_j - \bar{y}^*) > 0$ for $j = \frac{N_k+1}{2} + 1, \dots, N_k$

Then by part a, the subgradient of $|y_j - \bar{y}^*|$ is the following:

- $\{-1\}$ for $j = 1, \dots, \frac{N_k+1}{2} - 1$
- $[-1, 1]$ for $j = \frac{N_k+1}{2}$
- $\{+1\}$ for $j = \frac{N_k+1}{2} + 1, \dots, N_k$

Hence, $\partial f(\bar{y}^*) = \sum_{j=1}^{\frac{N_k+1}{2}-1} (-1) + \sum_{j=\frac{N_k+1}{2}+1}^{N_k} (+1) + [\text{subgradient of } |y_{\frac{N_k+1}{2}} - \bar{y}^*|]$.

Yet, there is an equal number of terms on each side of the median because N_k is odd, so the first two sums add to equal 0.

Also, $0 \in [\text{subgradient of } |y_{\frac{N_k+1}{2}} - \bar{y}^*|]$.

Hence, $0 \in \partial f(\bar{y}^*)$.

Recalling above that f is convex and subdifferentiable at $\bar{y}^*$, by the provided theorem, it follows that $\bar{y}^*$ minimizes $\sum_{j=1}^{N_k} |y_j - \bar{y}|$.

□

c) 2 Step algorithm that minimizes J...

0) Initialization: Start with some random initialization of the prototypes μ_k for $1 \leq k \leq K$.

1) Assignment: For each datapoint, find the prototype that is closest to it. Mathematically, set $r_{nk} = \begin{cases} 1 & \text{if } k = \underset{1 \leq j \leq K}{\operatorname{argmin}} \|x_n - \mu_j\|^2 \\ 0 & \text{else} \end{cases}$

2) Refitting: Suppose all indicators r_{nk} are known.
Must find prototypes that minimize $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|_1$.

Let μ_k = [median of all datapoints in cluster k].

↳ consequence of pt. b.

One iteration includes step 1 and step 2.
Keep going until a stopping criterion.

This algorithm is advantageous compared to k-means because it is computationally more simple in the step of refitting.

Whereas k-means sets μ_k equal to the mean of datapoints in cluster k, a $O(n)$ operation, this algorithm sets μ_k to the median, which if implemented well can be done in $O(1)$ time.

Also note that the median does not consider drastic outliers, whereas the mean does, showing another reason why this algorithm is advantageous.

3] Consider $A = \begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix}$

a) Find eigenvalues and eigenvectors of A, then normalize eigenvectors.

$$p(\lambda) = \det(A - \lambda I) = \det \begin{bmatrix} 3-\lambda & 2 \\ 2 & -\lambda \end{bmatrix} = (3-\lambda)(-\lambda) - (2)(2) = -3\lambda + \lambda^2 - 4$$

$$\Rightarrow p(\lambda) = (\lambda - 4)(\lambda + 1)$$

$$\Rightarrow \lambda_1 = 4, \quad \lambda_2 = -1 \text{ are eigenvalues}$$

Look at respective eigenspaces.

↳ $\lambda_1 = 4$: $\ker(A - 4I) = \ker \begin{bmatrix} -1 & 2 \\ 2 & -4 \end{bmatrix} = \text{span}\begin{pmatrix} 2 \\ 1 \end{pmatrix}$

$\Rightarrow \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ is an eigenvector for $\lambda_1 = 4$.

Normalized, obtain $u_1 = \dfrac{\begin{pmatrix} 2 \\ 1 \end{pmatrix}}{\sqrt{2^2+1^2}} = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$\lambda_2 = -1$: $\ker(A + I) = \ker \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix} = \text{span}\begin{pmatrix} 1 \\ -2 \end{pmatrix}$

$\Rightarrow \begin{pmatrix} 1 \\ -2 \end{pmatrix}$ is an eigenvector for $\lambda_1 = -1$.

Normalized, obtain $u_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

$\lambda_1 = 4$ is eigenvalue w/ eigenvector $\frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

$\lambda_2 = -1$ is eigenvalue w/ eigenvector $\frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

b) Eigenvalue decomposition of A given by $A = Q \Lambda Q^{-1}$ where Q's i^{th} column has i^{th} eigenvector and Λ is diagonal with corresponding eigenvalues $\Lambda_{ii} = \lambda_i$.

Hence, $A = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ -2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ -2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix}^{-1}$

$\therefore$ $A = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ -2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ -2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix}^{-1}$

4] A symmetric real matrix M is positive definite if $\forall z \neq 0$, $z^T M z > 0$.

a) Let $A = \begin{bmatrix} 9 & 6 \\ 6 & a \end{bmatrix}$. what should a satisfy so A is positive definite?

Let $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ be some arbitrary vector.

Then, $z^T A z = \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 9 & 6 \\ 6 & a \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 9z_1 + 6z_2 \\ 6z_1 + az_2 \end{bmatrix}$

$= z_1(9z_1 + 6z_2) + z_2(6z_1 + az_2)$

$= 9z_1^2 + 12z_1z_2 + az_2^2$

$= 9z_1^2 + 12z_1z_2 + 4z_2^2 + az_2^2 - 4z_2^2$

$= (3z_1 + 2z_2)^2 + az_2^2 - 4z_2^2$

Now, if $z_2 = 0$ then $z_1 \neq 0$ and $(3z_1 + 2z_2)^2 + az_2^2 - 4z_2^2 = 9z_1^2 > 0$.

If $z_2 \neq 0$, then $(3z_1 + 2z_2)^2 + az_2^2 - 4z_2^2 > 0$ provided that

$az_2^2 - 4z_2^2 > 0 \iff \underline{a > 4}$.

$\therefore$ A is positive definite if $a > 4$

b) Suppose B is positive definite. Show B^{-1} is positive definite.

Pick some column vector $z \neq 0$.

B^{-1} exists. So, $\exists$ vector w of same dimension as z such that $w = B^{-1}z$.

$w = B^{-1}z \Rightarrow z = Bw$ and $z^T = w^T B^T$.

Then, $z^T B^{-1} z = (w^T B^T) B^{-1} (Bw)$

$= w^T B^T w$

$= (w^T Bw)^T$

$= w^T Bw$ b/c $w^T Bw$ is a scalar.

> 0 by positive definiteness of B

$\Longrightarrow B^{-1}$ is positive definite by definition.

□

c) Show that data covariance matrix S in PCA is positive semi-definite.

S is defined as $S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$ of dimension $D \times D$.

↳ N is number of data points, $N > 0$.

x_n is a data point, $x_n \in \mathbb{R}^D$.

$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$ is the mean of all datapoints.

Let v be some arbitrary column vector dimension $D \times 1$ such that $v \neq 0$.

Then, $v^T S v = v^T \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T v$

$$= \frac{1}{N}\sum_{n=1}^{N}(v^T x_n - v^T\bar{x})(x_n^T - \bar{x}^T)v$$

$$= \frac{1}{N}\sum_{n=1}^{N}(v^T x_n - v^T\bar{x})(x_n^T v - \bar{x}^T v)$$

$$= \frac{1}{N}\sum_{n=1}^{N}(v^T x_n - v^T\bar{x})^2 \quad \text{b/c } v^T x_n = x_n^T v,\ v^T\bar{x} = \bar{x}^T v$$

$$\geq 0$$

Hence, S is positive semidefinite.

□

Zack Berger

Question 5

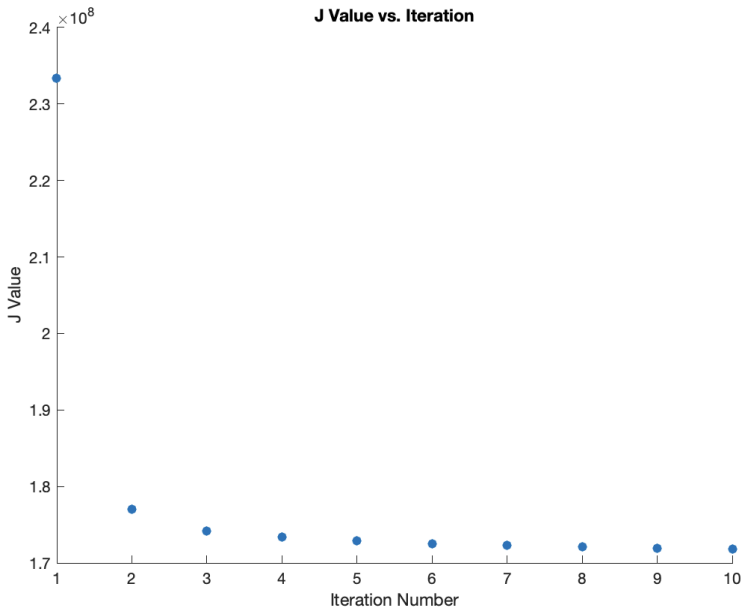
Part A - Visualization

Using imread and imshow, obtained the following graphic of the Bruin Bear:

Part B - Algorithm

Using the specifications for the K-means algorithm and running 10 iterations yielded the following graph of J vs. iterations.

The graph's asymptotic behavior as the number of iterations gets large indicates that the K-means algorithm will converge after many iterations. The insignificant changes in J as the number of iterations could be used to initiate a stopping condition for the algorithm.



Part C - Compression

As K increases, the quality of the compressed image improves. More of the original colors are retained, and thus the overall resolution of the image is greater…

K = 4

- After last iteration, J = $1.718261597502673 \times 10^{8}$

K = 8

- After last iteration, J = $8.043382692074148 \times 10^{7}$

K = 8

- After last iteration, J = $3.742910684965859 \times 10^{7}$

Part D - Compression Ratio

To store the original image, you would need 2,880,000 bits.

> 300x400 = 120,000 pixels
> Each pixel stores 3 elements vector taking 8x3 = 24 bits
> 120,000 (pixels) x 24 (bits/pixel) = 2,880,000 bits

To calculate how many bits are required to store the image using K centers, do the following:

> 300x400 = 120,000 pixels
> Each pixel stores 1 index = 8 bits
> 120,000 (pixels) x 8 (bits/pixel) = 960,000 bits
>
> K values stored, each with 24 bit precision
> K x 24 bits = 24K bits
>
> Total storage = (24K + 960,000) bits

This computation lead to the following:

- K = 4: 960,096 bits
- K = 8: 960,192 bits
- K = 16: 960,384 bits

Compute compression ratios as (Uncompressed Size) / (Compressed Size):

- K = 4: 2.9997
- K = 8: 2.9994
- K = 16: 2.9988

Code Attachment

```
% Import 300x400 picture of Bruin Bear, then display it
pic = imread('/Users/zackberger/Desktop/ML/HW/HW_7/UCLA_Bruin.jpg');

% Initialize K-means algorithm; K > 0
K = 16;

prototypes = zeros(3,K);            % Initialize prototypes to zero
prototypes(:,1) = [229, 249, 250];  % Set the first prototype to first pixel


% Set the rest of the prototypes using furthest-first heuristic
for k = 2 : K

   % Start with the first pixel as the furthest pixel
   % fpd = furthest_pixel_distance
   furthest_pixel = [229;249;250];
   fpd = 0;

   % Iterate over all pixels
   for r = 1 : 300
      for c = 1 : 400

         current_pixel = double([pic(r,c,1);pic(r,c,2);pic(r,c,3)]);

         % Find the min distance of current_pixel to previously computed prototypes
         % Denote it cpd. Start with distance to first prototype
         cpd = norm(current_pixel - prototypes(:,1))^2;

         % Iterate over all previously computed prototypes
         for p = 2 : (k - 1)
            % Obtain distance to next prototype (npd)
            npd = norm(current_pixel - prototypes(:,p))^2;
            if npd < cpd
               cpd = npd;
            end
         end

         % If the closest prototype distance is the furthest pixel
         % distance, we have a new furthest pixel
         if cpd > fpd
            fpd = cpd;
            furthest_pixel = current_pixel;
         end

      end
   end

   % Set the furthest pixel as the next prototype
   prototypes(:,k) = furthest_pixel;

end


% Run K-means for 10 iterations
J = zeros(1,10);                % Track objective for each iteration
pixel_indices = zeros(300,400);     % Each pixel refers to its closest prototype
Num_Iterations = 10;
```

```
for i = 1 : Num_Iterations

    % Assign each pixel to closest prototype
    for r = 1 : 300
        for c = 1 : 400

            current_pixel = double([pic(r,c,1);pic(r,c,2);pic(r,c,3)]);

            % Calculate closest prototype.
            % Start with distance to first.
            % prototype
            cpd = norm(current_pixel - prototypes(:,1))^2;
            closest_prototype = 1;

            % Iterate over all other prototypes
            for p = 2 : K
                npd = norm(current_pixel - prototypes(:,p))^2;
                if npd < cpd
                    cpd = npd;
                    closest_prototype = p;
                end
            end

            % Set closest prototype
            pixel_indices(r,c) = closest_prototype;

        end
    end

    % Now, pixel_indicies contains the number prototype that each pixel is
    % a part of. Re-estimate center of each cluster using this info
    prototypes = zeros(3,K);    % Refresh the prototype values
    num_samples = zeros(1,K);   % Number of datapoints in each cluster

    for r = 1 : 300
        for c = 1 : 400

            current_pixel = double([pic(r,c,1);pic(r,c,2);pic(r,c,3)]);

            num_samples(1,pixel_indices(r,c)) = num_samples(1,pixel_indices(r,c)) + 1;
            prototypes(:, pixel_indices(r,c)) = prototypes(:, pixel_indices(r,c)) + current_pixel;

        end
    end

    prototypes = prototypes ./ num_samples;



    % Calculate objective function
    total_dist = 0;
    for r = 1 : 300
        for c = 1 : 400

            for k = 1 : K

                current_pixel = double([pic(r,c,1);pic(r,c,2);pic(r,c,3)]);

                if pixel_indices(r,c) == k
                    total_dist = total_dist + norm(current_pixel - prototypes(:,k))^2;
                end
```

```
            end
        end
    end

    J(i) = total_dist;
end

% Set the picture to correct pixels
for r = 1 : 300
    for c = 1 : 400
        pic(r,c,:) = prototypes(:, pixel_indices(r,c));
    end
end

imshow(pic);

% Plot objective function vs. iterations
% iterations = [1,2,3,4,5,6,7,8,9,10];
% scatter(iterations, J, 'filled');

% title("J Value vs. Iteration");
% xlabel("Iteration Number");
% ylabel("J Value");
```