CSE512 HW4

Tim Zhang (110746199)

1 SGD for Multiclass Classification with Kernels and Costs

1.1 Question 2: Updates

1.1.1 (a)

$$\Delta = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

1.1.2 (b)

$$\Delta = \begin{bmatrix} 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 & 0 \end{bmatrix}$$

1.1.3 (c)

Fix $\hat{y} \in \max_{y' \in \{1,\ldots,k\}} \Delta(y', y_t) + \langle \boldsymbol{w}_{t,y'}, \phi(\boldsymbol{x}_t) \rangle - \langle \boldsymbol{w}_{t,y_t}, \phi(\boldsymbol{x}_t) \rangle$.

Then:

$$\nabla_{\boldsymbol{w}_{t,i}} \ell(\boldsymbol{w}_{t,i}, \phi(\boldsymbol{x}_t), y_t) =$$

$$\nabla_{\boldsymbol{w}_{t,i}} \Delta(\hat{y}, y_t) + \langle \boldsymbol{w}_{t,\hat{y}}, \phi(\boldsymbol{x}_t) \rangle - \langle \boldsymbol{w}_{t,y_t}, \phi(\boldsymbol{x}_t) \rangle = \begin{cases} 0 & \text{if } \hat{y} = y_t = i \vee \hat{y} \neq i \text{ and } y_t \neq i \\ \phi(\boldsymbol{x}_t) & \text{if } \hat{y} = i \text{ and } y_t \neq i \\ -\phi(\boldsymbol{x}_t) & \text{if } \hat{y} \neq i \text{ and } y_t = i \end{cases}$$

Assuming that $\Delta(y, y')$ is not a function of $\boldsymbol{w}_t$ ■

1.2 Question 3: SGD with kernels

1.2.1 (a)

Let $\alpha_{j,i}$ be constructed using the fixed $\hat{y}$ discussed above.

Then:

$$\alpha_{j,i} = \begin{cases} 0 & \text{if } \hat{y} = y_j = i \vee \hat{y} \neq i \text{ and } y_j \neq i \\ -\eta_j & \text{if } \hat{y} = i \text{ and } y_j \neq i \\ \eta_j & \text{if } \hat{y} \neq i \text{ and } y_j = i \end{cases}$$

This clearly shows that $\boldsymbol{w}_{t,i} = \sum_{j=1}^{t-1} \alpha_{j,i} \phi(\boldsymbol{x}_j)$ by observing the definition of $\alpha_{j,i}$ and of $\nabla_{\boldsymbol{w}_{j,i}} \ell(\boldsymbol{w}_{j,i}, \phi(\boldsymbol{x}_j), y_j)$. Under the assumption that each weight vector is initialized to $\mathbf{0}$.

Note that the signs are flipped in the definition of $\alpha_{j,i}$ as compared to the subgradient given that the SGD update rule is $\boldsymbol{w}_{t+1,i} = \boldsymbol{w}_{t,i} - \eta_t \boldsymbol{g}_{t,i}$.

Giving a formal inductive proof (and starting from $t = 0$) the basis case is:

$$\boldsymbol{w}_{1,i} = \boldsymbol{w}_{0,i} - \eta_1 \boldsymbol{g}_{1,i} = \mathbf{0} - \eta_1 \boldsymbol{g}_{1,i} = \mathbf{0} + \alpha_{1,i} \phi(\boldsymbol{x}_1)$$

Using the inductive hypothesis that $\boldsymbol{w}_{t,i} = \sum_{j=1}^{t-1} \alpha_{j,i} \phi(\boldsymbol{x}_j)$:

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_{t,i} - \eta_t \boldsymbol{g}_{t,i} \\ &= -\eta_t \boldsymbol{g}_{t,i} + \sum_{j=1}^{t-1} \alpha_{j,i} \phi(\boldsymbol{x}_j) \\ &= \alpha_{t,i} \phi(\boldsymbol{x}_t) + \sum_{j=1}^{t-1} \alpha_{j,i} \phi(\boldsymbol{x}_j) \\ &= \sum_{j=1}^{t} \alpha_{j,i} \phi(\boldsymbol{x}_j) \blacksquare \end{aligned}$$

1.2.2 (b)

$$\begin{aligned}
\langle \boldsymbol{w}_{t,i}, \phi(\boldsymbol{x}_t) \rangle &= \langle \sum_{j=1}^{t-1} \alpha_{j,i} \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_t) \rangle \\
&= \sum_{j=1}^{t-1} \langle \alpha_{j,i} \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_t) \rangle \\
&= \sum_{j=1}^{t-1} \alpha_{j,i} \langle \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_t) \rangle \\
&= \sum_{j=1}^{t-1} \alpha_{j,i} k(\boldsymbol{x}_j, \boldsymbol{x}_t)
\end{aligned}$$

Where the second line is from distributivity of dot product, the third line is because $\alpha_{j,i}$ is a scalar, and the last line the the definition of $k(\boldsymbol{x}, \boldsymbol{x'})$ ■

1.3 Question 4: Implementation and Use

1.3.1 (a)

NOTE: It is assumed that $X \in \mathbb{R}^{m \times d}$ and $y \in \mathbb{R}^{m \times 1}$.

```
%
% Function computes kernel SGD
%
function [alpha, Xsv] = train_mhinge_krnel_sgd(Xtr, ytr, Delta, p)
    [m, d] = size(Xtr);     % Dimensions of data
    k = 10;                 % Number of classes
    alpha = zeros(1, k);    % Matrix of alphas
    Xsv = zeros(1, d);      % Saved samples

    % Bootstrap the first example
    max = 0;
    yhat = 0;

    % Find yhat
    for i = 1: k
        loss = Delta(i, ytr(1));

        if loss > max
            max = loss;
            yhat = i;
        end
    end

    Xsv(1, :) = Xtr(1, :);  % Initialize Xsv to only the first example

    % Compute alpha_{j, i} where eta_1 = 1
    for i = 1: k
        if i == ytr(1) + 1
            alpha(1, i) = 1;
        elseif i == yhat
            alpha(1, i) = -1;
        else
            alpha(1, i) = 0;
        end
    end
```

```
    % Compute SGD over remaining training examples
    for t = 2: m
        y = ytr(t) + 1;  % Label for current training sample
        x = Xtr(t, :);   % Current sample
        cur_alpha = zeros(1, k);  % Current alpha
        eta = 1/sqrt(t);  % Learning rate

        % Determine yhat
        yhat = maximize_loss(alpha, Xsv, Delta, k, x, y, p);

        % Update Alpha
        for i = 1: k
            if i ~= yhat && i == y
                cur_alpha(1, i) = eta;

            elseif i == yhat && i ~= y
                cur_alpha(1, i) = -1 * eta;

            % Otherwise either y = yhat = i or y ~= i and yhat ~= i
            else
                cur_alpha(1, i) = 0;
            end
        end

        % If an update occured
        if any(cur_alpha)
            alpha = [alpha; cur_alpha];
            Xsv = [Xsv; x];
        end
    end
end
```

```
%
% Function computes loss at timestep
%
function loss = compute_loss(alpha, y, ker)
    loss = 0;   % Initial loss
    [t, ~] = size(alpha);

    for i = 1: t
        loss = loss + (alpha(i, y) * ker(i));
    end
end

%
% Function returns class which maximizes loss
%
function yhat = maximize_loss(alpha, Xsv, Delta, k, x, y, p)
    max = intmin;  % Maximum loss value
    yhat = 0;      % Maximum loss label

    % Precompute the kernel function
    ker = kernel(x, Xsv, p);

    % Compute loss term involving y_t
    y_loss = compute_loss(alpha, y, ker);

    % Find y' which maximizes loss
    for i = 1: k
        if i == y
            loss = Delta(i, y);
        else
            yhat_loss = compute_loss(alpha, i, ker);
            loss = Delta(i, y) + yhat_loss - y_loss;
        end

        % Update maximizer
        if loss > max
            max = loss;
            yhat = i;
        end
    end
end
```

```
%
% Function computes polynomial kernel
%
function ker = kernel(x, Xsv, p)
    [m, ~] = size(Xsv);
    ker = zeros(m, 1);

    for i = 1: m
        ker(i) = (x * Xsv(i, :)')^p;
    end
end
```

1.3.2 (b)

```
%
% Function computes the predicted classes
%
function [ypred] = test_mhinge_kernel_sgd(alpha, Xsv, Xte, p)
    [m, d] = size(Xte);
    [t, k] = size(alpha);
    ypred = zeros(m, 1);
    disp(m);

    % Predict for each test example
    for i = 1: m
        max = intmin;  % Determine most likely class
        pred = 0;

        % Precompute the kernel function
        ker = kernel(Xte(i, :), Xsv, p);

        % Calculate scores for each class predictor
        for class = 1: k
            score = 0;

            % Compute <w, x>
            for j = 1: t
                score = score + (alpha(j, class) * ker(j));
            end

            % Update prediction
            if score > max
                max = score;
                pred = class;
            end
        end

        ypred(i) = pred - 1;  % Classes range from 0-9, k from 1-10
    end
end
```

1.3.3 (c)

In the following confusion matrices the rows are the correct class and the columns are the predicted labels.

Cost Matrix $\Delta_{2.a}$

Test error = `0.0345`

Confusion matrix =

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**963**	1	2	1	0	2	9	1	1	0
1	0	**1125**	2	1	0	1	1	0	5	0
2	5	2	**995**	6	4	0	5	6	9	0
3	1	0	1	**980**	0	8	1	3	11	5
4	2	1	3	0	**951**	0	5	1	3	16
5	3	0	1	10	2	**853**	12	1	8	2
6	11	3	2	0	8	5	**925**	0	4	0
7	3	6	6	4	3	1	0	**989**	1	15
8	5	1	1	10	3	1	6	7	**936**	4
9	6	7	2	9	29	4	4	6	4	**938**

Cost Matrix $\Delta_{2.b}$

Test error = `0.0345`

Confusion matrix =

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**963**	1	2	1	0	2	9	1	1	0
1	0	**1125**	2	1	0	1	1	0	5	0
2	5	2	**995**	6	4	0	5	6	9	0
3	1	0	1	**980**	0	8	1	3	11	5
4	2	1	3	0	**951**	0	5	1	3	16
5	3	0	1	10	2	**853**	12	1	8	2
6	11	3	2	0	8	5	**925**	0	4	0
7	3	6	6	4	3	1	0	**989**	1	15
8	5	1	1	10	3	1	6	7	**936**	4
9	6	7	2	9	29	4	4	6	4	**938**

```
%
% Function ouputs the accuracy of the prediction
%
function [correct, misclassified] = test_prediction(ypred, yte)
    [m, ~] = size(yte);
    correct = zeros(10, 1);         % Correct predictions per class
    misclassified = zeros(10, 10);  % (correct class, prediction)

    for i = 1: m
        % If the prediction was wrong store prediction
        if ypred(i) ~= yte(i)
            misclassified(yte(i) + 1, ypred(i) + 1) =
            misclassified(yte(i) + 1, ypred(i) + 1) + 1;
        else
            correct(ypred(i) + 1) = correct(ypred(i) + 1) + 1;
        end
    end

    disp('Accuracy:');
    disp(sum(correct)/m);
end
```

2 Question 5: Boosting

Let $Z = \sum_{i=1}^{m} D_t(\boldsymbol{x}_i, y_i) \cdot \exp(-w_t y_i f_t(\boldsymbol{x_i}))$ and $\epsilon_t = \sum_{i:y_i \neq f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)$:

$$\begin{aligned}\sum_{i=1}^{m} D_{t+1}(\boldsymbol{x}_i, y_i) \cdot \mathbb{1}[y_i \neq f_t(\boldsymbol{x_i})] &= \sum_{i:y_i \neq f_t(\boldsymbol{x}_i)} D_{t+1}(\boldsymbol{x}_i, y_i) \\ &= \frac{1}{Z} \sum_{i:y_i \neq f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) \cdot \exp(w_t) \\ &= \frac{\exp(w_t)\epsilon_t}{Z}\end{aligned}$$

Rewriting Z:

$$\begin{aligned}Z &= \sum_{i=1}^{m} D_t(\boldsymbol{x}_i, y_i) \cdot \exp(-w_t y_i f_t(\boldsymbol{x_i})) \\ &= \exp(w_t) \sum_{i:y_i \neq f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) + \exp(-w_t) \sum_{i:y_i = f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) \\ &= \exp(w_t)\epsilon_t + \exp(-w_t) \sum_{i:y_i = f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)\end{aligned}$$

Then:

$$\begin{aligned}\sum_{i=1}^{m} D_{t+1}(\boldsymbol{x}_i, y_i) \cdot \mathbb{1}[y_i \neq f_t(\boldsymbol{x_i})] &= \frac{\exp(w_t)\epsilon_t}{\exp(w_t)\epsilon_t + \exp(-w_t) \sum_{i:y_i = f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)} \\ &= \frac{1}{(\exp(w_t)\epsilon_t)^{-1} \cdot \exp(w_t)\epsilon_t + (\exp(w_t)\epsilon_t)^{-1} \cdot \exp(-w_t) \sum_{i:y_i = f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)} \\ &= \frac{1}{1 + (\exp(w_t)\epsilon_t)^{-1} \cdot \exp(-w_t) \sum_{i:y_i = f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}\end{aligned}$$

It remains to show that $(\exp(w_t)\epsilon_t)^{-1} \cdot \exp(-w_t) \sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) = 1$:

$$
\begin{aligned}
(\exp(w_t)\epsilon_t)^{-1} \cdot \exp(-w_t) \sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) &= \frac{\exp(-w_t) \sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{\exp(w_t)\epsilon_t} \\
&= \frac{(\frac{1}{\epsilon_t} - 1)^{-1/2} \sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{(\frac{1}{\epsilon_t} - 1)^{1/2}\epsilon_t} \\
&= \frac{\sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{(\frac{1}{\epsilon_t} - 1)^{1/2}(\frac{1}{\epsilon_t} - 1)^{1/2}\epsilon_t} \\
&= \frac{\sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{(\frac{1}{\epsilon_t} - 1)\epsilon_t} \\
&= \frac{\sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{1 - \epsilon_t} \\
&= \frac{\sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)}{\sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)} = 1
\end{aligned}
$$

Where the last line follows from the fact that D_t is a probability distribution and thus $1 - \epsilon_t = 1 - \sum_{i:y_i \neq f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i) = \sum_{i:y_i=f_t(\boldsymbol{x}_i)} D_t(\boldsymbol{x}_i, y_i)$ ■

3 PCA via Successive Deflation

3.1 Question 6

3.1.1 (a)

By definition $\tilde{\boldsymbol{C}} = \frac{1}{m}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top$ then solving for $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top$:

$$\begin{aligned}
\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top &= (\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{X}((\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{X})^\top \\
&= (\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{X}\boldsymbol{X}^\top(\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top) \\
&= (\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)(\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{v}_1\boldsymbol{v}_1^\top) \\
&= (\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)(\boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top) \\
&= \boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top - \boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{X}\boldsymbol{X}^\top + \boldsymbol{v}_1\boldsymbol{v}_1^\top m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top \\
&= \boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top - \boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{X}\boldsymbol{X}^\top + m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{v}_1\boldsymbol{v}_1^\top \\
&= \boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top - \boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{X}\boldsymbol{X}^\top + m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top \\
&= \boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{X}\boldsymbol{X}^\top \\
&= (\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{X}\boldsymbol{X}^\top \\
&= (((\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{X}\boldsymbol{X}^\top)^\top)^\top \\
&= ((\boldsymbol{X}\boldsymbol{X}^\top)^\top(\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top)^\top)^\top \\
&= (\boldsymbol{X}\boldsymbol{X}^\top(\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\top))^\top \\
&= (\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{v}_1\boldsymbol{v}_1^\top)^\top \\
&= (\boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top)^\top \\
&= (\boldsymbol{X}\boldsymbol{X}^\top)^\top - (m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top)^\top \\
&= \boldsymbol{X}\boldsymbol{X}^\top - m\lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top
\end{aligned}$$

Thus $\tilde{\boldsymbol{C}} = \frac{1}{m}\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\top - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top$ ■

3.1.2 (b)

First note that $\tilde{\boldsymbol{C}} = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\top - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top = \boldsymbol{C} - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top$:

$$\begin{aligned}
\tilde{\boldsymbol{C}}\boldsymbol{v}_j &= (\boldsymbol{C} - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top)\boldsymbol{v}_j \\
&= \boldsymbol{C}\boldsymbol{v}_j - \lambda_1\boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{v}_j \\
&= \lambda_j\boldsymbol{v}_j
\end{aligned}$$

Where the last equality is due to the definition of $\lambda_j\boldsymbol{v}_j$ and the fact that $\boldsymbol{v}_1^\top\boldsymbol{v}_j = 0$ for $j \neq 1$ ■

3.1.3 (c)

It suffices to show that λ_1, $\boldsymbol{v}_1$ are no longer the largest eigenvalue/eigenvector pair of $\tilde{\boldsymbol{C}}$:

$$\begin{aligned}\tilde{\boldsymbol{C}}\boldsymbol{v}_1 &= (\boldsymbol{C} - \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^\top)\boldsymbol{v}_1 \\ &= \boldsymbol{C}\boldsymbol{v}_1 - \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^\top \boldsymbol{v}_1 \\ &= \lambda_1 \boldsymbol{v}_1 - \lambda_1 \boldsymbol{v}_1 \\ &= 0 = \lambda_1' \boldsymbol{v}_1\end{aligned}$$

Thus $\lambda_1' = 0$ and is now strictly smaller than λ_2 (since $\boldsymbol{v}_i \neq 0$ by definition) thus $\boldsymbol{u} = \boldsymbol{v}_2$ since the eigenvectors are sorted by eigenvalue ■

3.1.4 (d)

```
function [lambdas, vectors] = compute_k_eigenvectors(C, k, f):
    lambdas = zeros(k, 1);  % Vector of eigenvalues
    U = zeros(k, 1);  % Vector of eigenvectors

    for i = 1: k
        [lambda, u] = f(C);  % Call oracle function
        lambdas(i) = lambda;
        U(i) = u;

        C = C - (lambda * u * u');  % Compute deflated covariance
    end
end
```

4 Clustering with k-means

4.1 Question 7

4.1.1 (a)

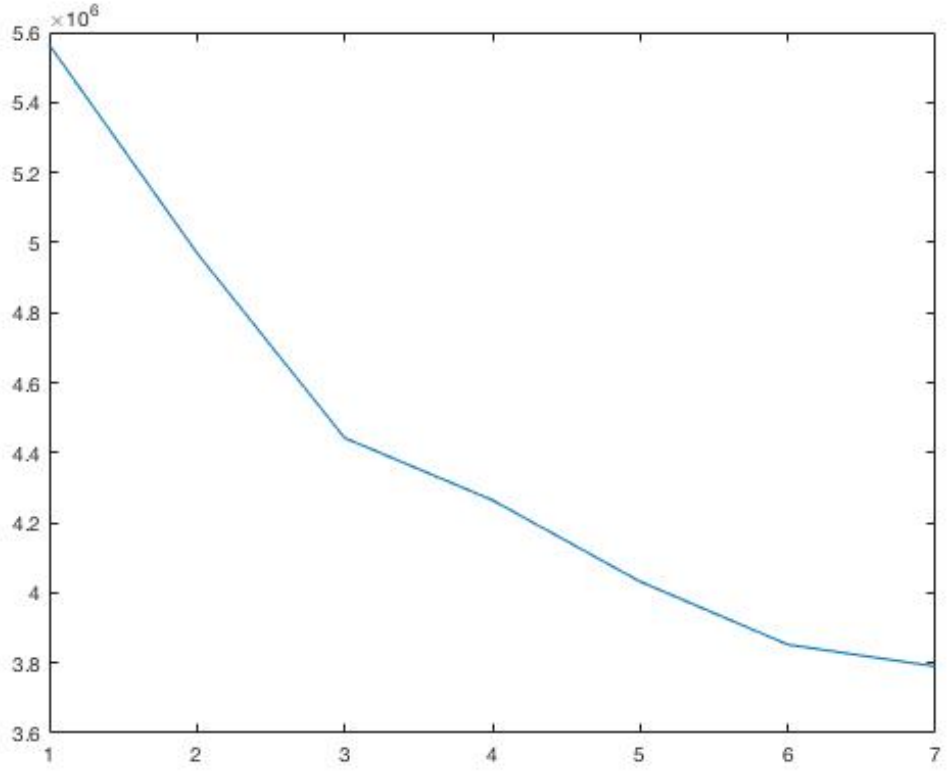
```
%
% Function computes the set of k centers via k-means clustering
%
function M = kmeans(X, k, T)
    [m, d] = size(X);   % Dimensions of data
    M = zeros(k, d);    % Matrix of k centers
    C = zeros(m, 1);    % Entries are {1, ..., k} for sample cluster

    % Sequentially assign centers as first k examples
    for i = 1: k
        M(i, :) = X(i, :);
    end

    % Iterate a max of T times
    for i = 1: T
        converged = true;  % Tracks if algorithm has converged

        % Assign each sample to nearest center
        for j = 1: m
            prev = C(j);  % Track previous center
            C(j) = compute_nearest_center(M, X(j, :), k);

            if C(j) ~= prev
                converged = false;
            end
        end

        % Terminate on convergence
        if converged
            break;
        end

        % Recompute centers
        compute_centers(M, C, X);
    end
end
```

```
%
% Function returns the index of the closest center to sample
%
function c = compute_nearest_center(M, x, k)
    c = 0;              % Index of nearest center
    closest = intmax;  % Current value closest distance

    for i = 1: k
        dist = norm(x - M(i, :));  % Compute distance via l2 norm

        if dist < closest
            c = i;
            closest = dist;
        end
    end
end

%
% Function recomputes centers as average vector of cluster
%
function compute_centers(M, C, X)
    [k, d] = size(M);
    [m, ~] = size(X);

    % Recompute center for each cluster
    for i = 1: k
        center = zeros(1, d);  % Average vector
        m_cluster = 0;         % Number of samples in cluster

        for j = 1: m
            if C(j) == i
                center = center + X(i, :);
                m_cluster = m_cluster + 1;
            end
        end

        M(i, :) = center ./ m_cluster;
    end
end
```

4.1.2 (b)

```
%
% Function plots iterations of k-means via metric
%
function plot(X, y, k_max, T, metric)
    k = 2;
    val = [];  % Holds each iterations value

    while k <= k_max
        M = kmeans(X, k, T);  % Compute centers

        if strcmp(metric, 'Sum of Squares')
            val = [val sum_of_squares(M, X, k)];
        elseif strcmp(metric, 'Purity')
            val = [val, purity(M, X, y, k)];
        else
            disp('Invalid metric');
            return
        end

        k = k + 2;  % Iterate by 2
    end

    % Plot values
    plot(val);
end

%
% Function computes sum of squares
%
function sum = sum_of_squares(M, X, k)
    [m, ~] = size(X);
    C = zeros(m, 1);  % Entries are {1, ..., k} for sample cluster
    sum = 0;

    % Assign each sample to nearest center
    for i = 1: m
        C(i) = compute_nearest_center(M, X(i, :), k);
    end
```

```
    % Calculate squared distance for each cluster
    for i = 1: m
        sum = sum + norm(X(i, :) - M(C(i), :))^2;
    end
end
```

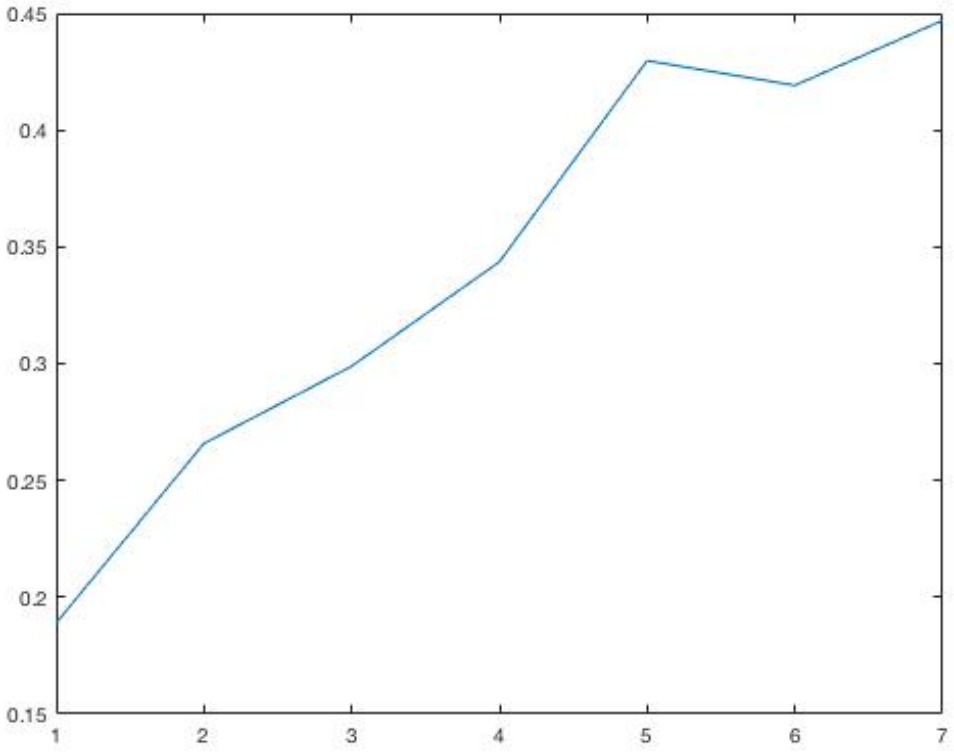


Horizontal axis represents the *ith* iteration, for example 3 is the iteration where $k = 6$.

4.1.3 (c)

```
%
% Function computes purity
%
function purity = purity(M, X, y, k)
    [m, ~] = size(X);      % Dimensions of training data
    C = zeros(m, 1);       % Entries are {1, ..., k} for sample cluster
    labels = zeros(1, k);  % Holds cluster labels by majority vote

    % Assign each sample to nearest center
    for i = 1: m
        C(i) = compute_nearest_center(M, X(i, :), k);
    end

    % Assign labels
    for i = 1: k
        dist = zeros(1, 10);  % Distribution of labels in cluster i

        for j = 1: m

            % If sample is in cluster add the label to dist
            if C(j) == i
                dist(y(j) + 1) = dist(y(j) + 1) + 1;
            end
        end

        % Store index of max class
        [~, l] = max(dist);
        labels(i) = l - 1;  % Class labels are from 0-9
    end

    error = 0;

    % Compute error
    for i = 1: m
        % If the label is different than the cluster label
        if labels(C(i)) ~= y(i)
           error = error + 1;
        end
    end
```

```
    purity = (m - error) / m;  % The proportion of correct labels
end
```



Horizontal axis represents the *ith* iteration, for example 3 is the iteration where $k = 6$.

5 Manual calculation of one round of EM for a Mixture of Gaussians

5.1 Question 8

5.1.1 (a) M Step

The likelihood is given by $\prod_{i=1}^{3} \mathbb{P}_{X\sim D_\theta}[X = x_i]$ where each $\mathbb{P}_{X\sim D_\theta}[X = x_i]$ is defined as:

$$\begin{aligned} f_\theta(x_i) &= \sum_{y=1}^{2} f_\theta(x_i, y) \\ &= \sum_{y=1}^{2} \pi_y f_\theta(x_i|y) \\ &= \sum_{y=1}^{2} \frac{\pi_y}{\sqrt{2\pi\sigma_y^2}} \exp\Big(- \frac{(x_i - \mu_y)^2}{2\sigma_y^2} \Big) \end{aligned}$$

5.1.2 (b) M Step

Computing $N_y^{(t)}$:

$$N_1^{(t)} = \sum_{i=1}^{3} Q_{i,1}^{(t)} = 1 + .4 + 0 = 1.4$$
$$N_2^{(t)} = \sum_{i=1}^{3} Q_{i,2}^{(t)} = 0 + .6 + 1 = 1.6$$

Computing $\pi_y^{(t)}$:

$$\pi_1^{(t)} = \frac{N_1}{3} = \frac{1.4}{3}$$
$$\pi_2^{(t)} = \frac{N_2}{3} = \frac{1.6}{3}$$

5.1.3 (c) M Step

Computing $\mu_y^{(t)}$:

$$\mu_1^{(t)} = \frac{1}{N_1} \sum_{i=1}^{m} Q_{i,1}^{(t)} x_i = \frac{1 + 4 + 0}{1.4} = \frac{5}{1.4}$$
$$\mu_2^{(t)} = \frac{1}{N_2} \sum_{i=1}^{m} Q_{i,2}^{(t)} x_i = \frac{0 + 6 + 20}{1.6} = \frac{26}{1.6}$$

5.1.4 (d) M Step

Computing $\sigma_y^{(t)}$:

$$\sigma_1^{(t)} = \frac{1}{N_1}\sum_{i=1}^{m} Q_{i,1}^{(t)}(x_i - \mu_1)^2 = \frac{(-\frac{3.6}{1.4})^2 + .4 \cdot (\frac{9}{1.4})^2 + 0}{1.4} = \frac{23.14285714}{1.4} = 16.53061224$$

$$\sigma_2^{(t)} = \frac{1}{N_2}\sum_{i=1}^{m} Q_{i,2}^{(t)}(x_i - \mu_2)^2 = \frac{0 + .6 \cdot (-\frac{10}{1.6})^2 + (\frac{6}{1.6})^2}{1.6} = \frac{37.5}{1.6} = 23.4375$$

5.1.5 (e) E Step

$$Q_{i,y}^{(t+1)} = \frac{\frac{\pi_y^{(t)}}{\sigma_y^{(t)}} \cdot \exp\left(- \frac{(x_i - \mu_y^{(t)})^2}{2(\sigma_y^{(t)})^2} \right)}{\sum_{j=1}^{k} \frac{\pi_j^{(t)}}{\sigma_j^{(t)}} \cdot \exp\left(- \frac{(x_i - \mu_j^{(t)})^2}{2(\sigma_j^{(t)})^2} \right)}$$

Where the value of y is the cluster c which x_i belongs to.

5.1.6 (f) E Step

Computing each term in the denominator:

$$\frac{\pi_1^{(t)}}{\sigma_1^{(t)}} \cdot \exp\left(-\frac{(x_1 - \mu_1^{(t)})^2}{2(\sigma_1^{(t)})^2} \right) = .0282304527 \cdot \exp\left(-\frac{6.612244898}{546.5222821} \right) = .027890957$$

$$\frac{\pi_2^{(t)}}{\sigma_2^{(t)}} \cdot \exp\left(-\frac{(x_1 - \mu_2^{(t)})^2}{2(\sigma_2^{(t)})^2} \right) = .0227555556 \cdot \exp\left(-\frac{232.5625}{1098.632813} \right) = .0184142673$$

$$\frac{\pi_1^{(t)}}{\sigma_1^{(t)}} \cdot \exp\left(-\frac{(x_2 - \mu_1^{(t)})^2}{2(\sigma_1^{(t)})^2} \right) = .0282304527 \cdot \exp\left(-\frac{41.32653061}{546.5222821} \right) = .0261744565$$

$$\frac{\pi_2^{(t)}}{\sigma_2^{(t)}} \cdot \exp\left(-\frac{(x_2 - \mu_2^{(t)})^2}{2(\sigma_2^{(t)})^2} \right) = .0227555556 \cdot \exp\left(-\frac{39.0625}{1098.632813} \right) = .021960684$$

$$\frac{\pi_1^{(t)}}{\sigma_1^{(t)}} \cdot \exp\left(-\frac{(x_3 - \mu_1^{(t)})^2}{2(\sigma_1^{(t)})^2} \right) = .0282304527 \cdot \exp\left(-\frac{269.8979592}{546.5222821} \right) = .017228329$$

$$\frac{\pi_2^{(t)}}{\sigma_2^{(t)}} \cdot \exp\left(-\frac{(x_3 - \mu_2^{(t)})^2}{2(\sigma_2^{(t)})^2} \right) = .0227555556 \cdot \exp\left(-\frac{14.0625}{1098.632813} \right) = .0224661407$$

$$\sum_{j=1}^{k} \frac{\pi_j^{(t)}}{\sigma_j^{(t)}} \cdot \exp\left(-\frac{(x_1 - \mu_j^{(t)})^2}{2(\sigma_j^{(t)})^2} \right) = .0463052243$$

$$\sum_{j=1}^{k} \frac{\pi_j^{(t)}}{\sigma_j^{(t)}} \cdot \exp\left(-\frac{(x_2 - \mu_j^{(t)})^2}{2(\sigma_j^{(t)})^2} \right) = .0481351405$$

$$\sum_{j=1}^{k} \frac{\pi_j^{(t)}}{\sigma_j^{(t)}} \cdot \exp\left(-\frac{(x_3 - \mu_j^{(t)})^2}{2(\sigma_j^{(t)})^2} \right) = .0396944697$$

Computing $Q_{i,j}^{(t+1)}$:

$$Q_{1,1}^{(t+1)} = .027890957/.0463052243 = .6023285152$$

$$Q_{1,2}^{(t+1)} = .0184142673/.0463052243 = .3976714848$$

$$Q_{2,1}^{(t+1)} = .0261744565/.0481351405 = .5437702316$$

$$Q_{2,2}^{(t+1)} = .021960684/.0481351405 = .4562297684$$

$$Q_{3,1}^{(t+1)} = .017228329/.0396944697 = .4340234075$$

$$Q_{3,2}^{(t+1)} = .0224661407/.0396944697 = .5659765925$$

Thus

$$Q^{(t+1)} = \begin{bmatrix} .6023285152 & .3976714848 \\ .5437702316 & .4562297684 \\ .4340234075 & .5659765925 \end{bmatrix}$$