CSE 417T Machine Learning

Hw5, Due: Nov.8, 2016

Chih Yun Pai

1. (a) *Show that the Gram matrix G with* $G_{i,j} = \vec{x}_i^T \vec{z}_j$ *can be expressed in terms of a pure matrix multiplication.*

Define $\vec{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}_{d\times 1}$, $\vec{z}_j = \begin{bmatrix} z_{j1} \\ z_{j2} \\ \vdots \\ z_{jd} \end{bmatrix}_{d\times 1}$,

where $x_{ik}, \mathrm{z}_{jk}, n, m, d \in \mathbb{R}$ *and* $i = \overline{1,n}$, $j = \overline{1,m}$, $k = \overline{1,d}$ $\in \mathbb{R}$

$$\mathrm{X} = [\vec{x}_1, \quad \vec{x}_2, \quad \dots \quad \vec{x}_n]_{d\times n} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1d} & x_{2d} & \cdots & x_{nd} \end{bmatrix}_{d\times n} ,$$

$$\mathrm{Z} = [\vec{z}_1, \quad \vec{z}_2, \quad \dots \quad \vec{z}_m]_{d\times m} = \begin{bmatrix} z_{11} & z_{21} & \cdots & z_{m1} \\ z_{12} & z_{22} & \cdots & z_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ z_{1d} & z_{2d} & \cdots & z_{md} \end{bmatrix}_{d\times m} ,$$

**Be caution, the notation* x_{ik} *and* z_{jk} *is different from the matrix notation.*

$Given \;\; G_{i,j} = [\vec{x}_i^T]_{1\times d}[\vec{z}_j]_{d\times 1} \in \mathbb{R}$

$$\Rightarrow \mathrm{G}_{n\times m} = \begin{bmatrix} \vec{x}_1^T\vec{z}_1 & \vec{x}_1^T\vec{z}_2 & \cdots & \vec{x}_1^T\vec{z}_m \\ \vec{x}_2^T\vec{z}_1 & \vec{x}_2^T\vec{z}_2 & \cdots & \vec{x}_2^T\vec{z}_m \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x}_n^T\vec{z}_1 & \vec{x}_n^T\vec{z}_2 & \cdots & \vec{x}_n^T\vec{z}_m \end{bmatrix}_{n\times m}$$

$$= \begin{bmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \vdots \\ \vec{x}_n^T \end{bmatrix}_{n\times d} [\vec{z}_1, \quad \vec{z}_2, \quad \dots \quad \vec{z}_m]_{d\times m}$$

$$= [\vec{x}_1, \quad \vec{x}_2, \quad \dots \quad \vec{x}_n]_{n\times d}^T [\vec{z}_1, \quad \vec{z}_2, \quad \dots \quad \vec{z}_m]_{d\times m}$$

$$= \mathrm{X}_{n\times d}^T \mathrm{Z}_{d\times m} = [\mathrm{X}^T \mathrm{Z}\;\;]_{n\times m}$$

1.(b) *Define two matrices* $S, R \;\in\; \mathbb{R}^{n\times m}, S_{i,j} = \vec{x}_i^T\vec{x}_i$, $R_{i,j} = \vec{z}_j^T z_j$.
Show that the Squared-Euclidean distance matrix $D^2 \;\in \mathbb{R}^{n\times m}$*, defined as* $D_{i,j}^2 = (\vec{x}_i - \vec{z}_j)^2$

$$S_{i,j} = \vec{x}_i^T\vec{x}_i \;\; \Rightarrow S = \begin{bmatrix} \vec{x}_1^T\vec{x}_1 & \vec{x}_1^T\vec{x}_1 & \cdots & \vec{x}_1^T\vec{x}_1 \\ \vec{x}_2^T\vec{x}_2 & \vec{x}_2^T\vec{x}_2 & \cdots & \vec{x}_2^T\vec{x}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x}_n^T\vec{x}_n & \vec{x}_n^T\vec{x}_n & \cdots & \vec{x}_n^T\vec{x}_n \end{bmatrix}_{n\times m} ,$$

$R_{i,j} = \vec{z}_j^T z_j \;\; \Rightarrow R = \begin{bmatrix} \vec{z}_1^T \vec{z}_1 & \vec{z}_2^T \vec{z}_2 & \cdots & \vec{z}_m^T \vec{z}_m \\ \vec{z}_1^T \vec{z}_1 & \vec{x}_2^T \vec{x}_2 & \cdots & \vec{z}_m^T \vec{z}_m \\ \vdots & \vdots & \ddots & \vdots \\ \vec{z}_1^T \vec{z}_1 & \vec{z}_2^T \vec{z}_2 & \cdots & \vec{z}_m^T \vec{z}_m \end{bmatrix}_{n\times m}$, where $i = \overline{1,n}$, $j = \overline{1,m} \in$

$\mathbb{R}$

And from part (a),

$$G_{i,j} = \vec{x}_i^T \vec{z}_j \Rightarrow G = \begin{bmatrix} \vec{x}_1^T \vec{z}_1 & \vec{x}_1^T \vec{z}_2 & \cdots & \vec{x}_1^T \vec{z}_m \\ \vec{x}_2^T \vec{z}_1 & \vec{x}_2^T \vec{z}_2 & \cdots & \vec{x}_2^T \vec{z}_m \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x}_n^T \vec{z}_1 & \vec{x}_n^T \vec{z}_2 & \cdots & \vec{x}_n^T \vec{z}_m \end{bmatrix}_{n\times m}$$

Given $D_{i,j}^2 = (\vec{x}_i - \vec{z}_j)^2$

$$\Rightarrow D^2 = \begin{bmatrix} (\vec{x}_1 - \vec{z}_1)^2 & (\vec{x}_1 - \vec{z}_2)^2 & \cdots & (\vec{x}_1 - \vec{z}_m)^2 \\ (\vec{x}_2 - \vec{z}_1)^2 & (\vec{x}_2 - \vec{z}_2)^2 & \cdots & (\vec{x}_2 - \vec{z}_m)^2 \\ \vdots & \vdots & \ddots & \vdots \\ (\vec{x}_n - \vec{z}_1)^2 & (\vec{x}_n - \vec{z}_2)^2 & \cdots & (\vec{x}_n - \vec{z}_m)^2 \end{bmatrix}_{n\times m}$$

$D_{i,j}^2 = (\vec{x}_i - \vec{z}_j)^2$

$= \;\; \vec{x}_i \cdot \vec{x}_i - 2\vec{x}_i \cdot \vec{z}_j + \vec{z}_j \cdot \vec{z}_j$

$= \vec{x}_i^T \vec{x}_i - 2\vec{x}_i^T \vec{z}_j + \vec{z}_j^T \vec{z}_j$

$= S_{i,j} - 2G_{i,j} + R_{i,j}$

$\Rightarrow D^2 = S - 2G + R$

What mathematical property do all entries of D satisfy?

$\Rightarrow$ Each element $D_{i,j} \geq 0$, and represents the distance between $\vec{x}_i$ and $\vec{z}_j$.

What are the diagonal entries of D?

$\Rightarrow D_{i,i} = (S_{i,i} - 2G_{i,i} + R_{i,i})^{\frac{1}{2}}$, or $D_{i,i} = (\vec{x}_i \cdot \vec{x}_i - 2\vec{x}_i \cdot \vec{z}_i + \vec{z}_i \cdot \vec{z}_i)^{\frac{1}{2}}$

where $i = \overline{1, \min(n,m)}$

What are the diagonal entries of D assuming that X = Z (we want the distance among all data points in X)?

$\Rightarrow$ When X=Z, D is a n by n matrix with diagonal elements all zeros, because the distance to each point itself equals to zero.

What do you need to obtain the true Euclidean distance matrix D?

$\Rightarrow D = (S - 2G + R)^{\frac{1}{2}}$ *or* $D = (\vec{x}_i \cdot \vec{x}_i - 2\vec{x}_i \cdot \vec{z}_j + \vec{z}_j \cdot \vec{z}_j)^{\frac{1}{2}}$, where $i = \overline{1,n}$, $j = \overline{1,m}$. In implementation, $D =$ sqrt$(S - 2G + R)$ is much faster than the other expression.

---

1. (c)

X=rand(100,10000);

Z=rand(100,2000);

tic;D=l2distanceSlow(X,Z);toc

tic;D=l2distance(X,Z);toc

For l2distanceSlow, Elapsed time is 61.7 seconds.

Changing the 3rd loop(line 22~line 26) in the l2distanceSlow.m to only one line (`D(i,j)=sqrt(sum((X(:,i)-Z(:,j)).^2));`) and running the code above, Elapsed time is 31.5 .

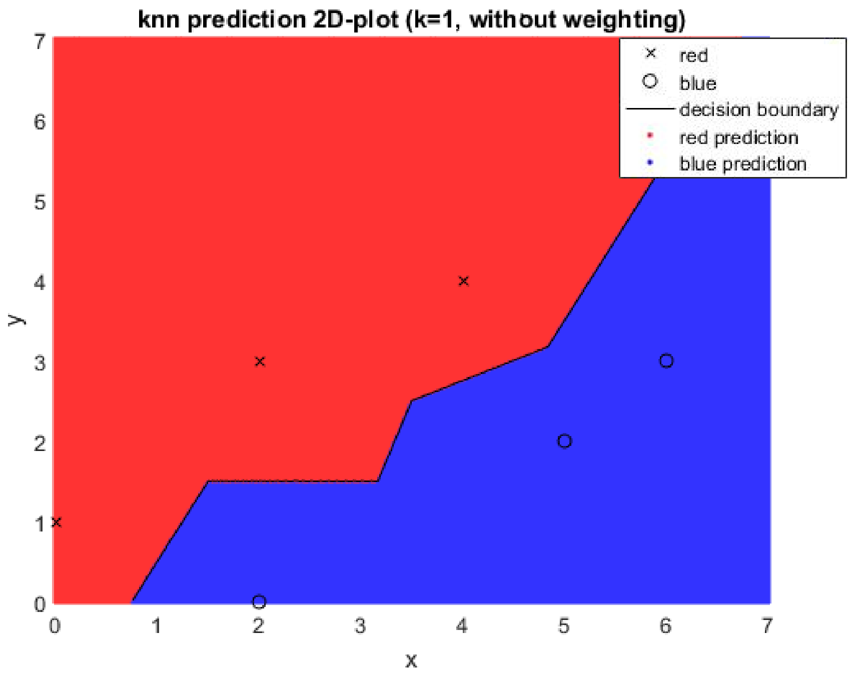
Running the new method D=l2distance (X,Z), the function implements part (a) in one line

```
D = sqrt(repmat(sum(X'.*X',2),1,m)-2*(X'*Z)+repmat(sum(Z.*Z,1),n,1));
```

, the execution time is 0.518 seconds.

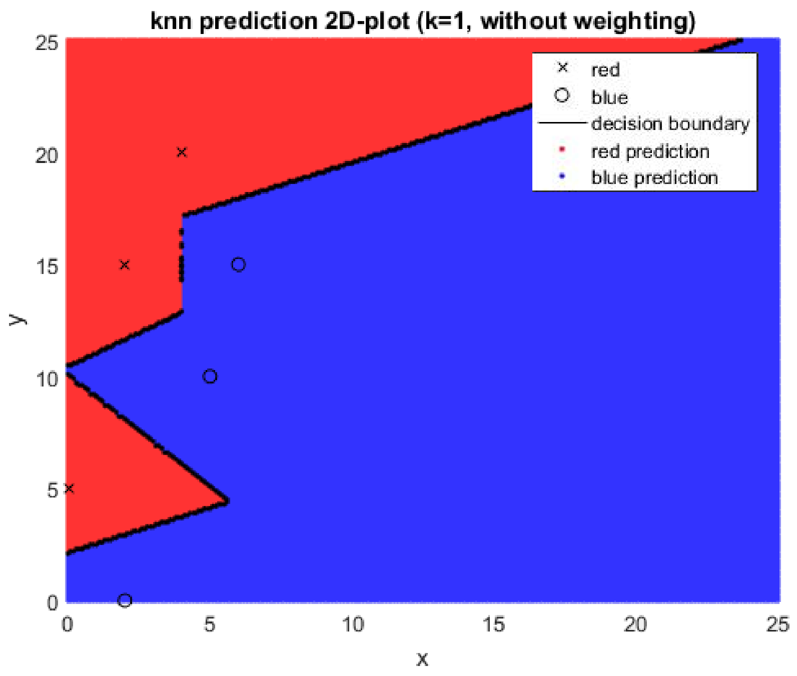
2. (a)

knn boundary for k = 1 and without weighting plot:



2. (b)

After multiplying y-coordinate by 5, knn boundary for k = 1 and without weighting plot:

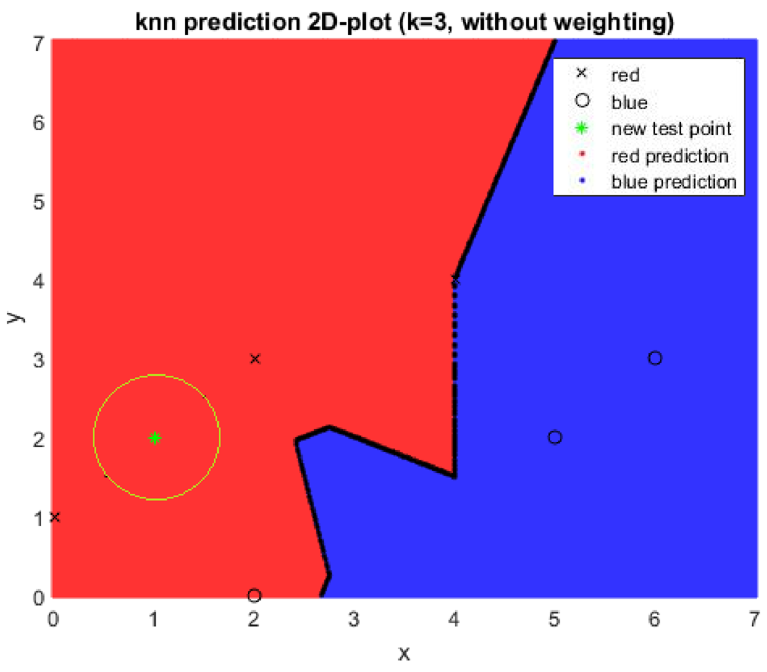


After multiplying y-coordinates by 5, the distribution of points distorted. When this happen in real case, it might cause over-fitting for some of test points. (e.g. Point (1,10) might be red, yet in the distorted version, it is blue.)

2. (c)

3nn without weighting, the new point is classified to red.

It's classification can be changed by adding at least two "o: blue" points inside the green circle (center: (1,2), radius: $\sqrt{2}$), because for now the nearest two points are "x: red".



2. (d)

Without any modification, Knn's complexity is O(dn).

Other strategies to improve efficiency, such as, KD-tree which's complexity is O(d*log(n)) and CNN which's complexity is uncertain yet at most O(dn).

3. (a) Implementation of finknn.m

3. (b) Implementation of analyze.m

What classification accuracy would you expect from a random classifier?

-> For 40-classes faces data set and for 10-classes digit data set, the random classifier would make prediction of around 1/40 and 1/10 accuracy respectively. Because it classifies randomly, its prediction approximate to expectation of the classes.

3. (c)

Max accuracy, Improved compare to 1st method, decreased compare to 1st method

Implementation of knnclassifier.m

- 1st method: Picking k-nn with plainly compare the occurrence frequency of the nearest k-nn. And the prediction accuracy in each of 10 times shuffling training set and test set is consistent with the result.

	1-nn	2-nn	3-nn	4-nn	5-nn	6-nn	7-nn	8-nn	9-nn
Face Recognition (≅0.02 seconds)	95.83%	95.83%	91.67%	90.83%	90.00%	87.50%	85.00%	85.00%	81.67%
Handwritten digits Recognition (≅2.2 seconds)	95.02%	93.92%	94.97%	94.62%	94.97%	94.62%	94.67%	94.22%	94.32%

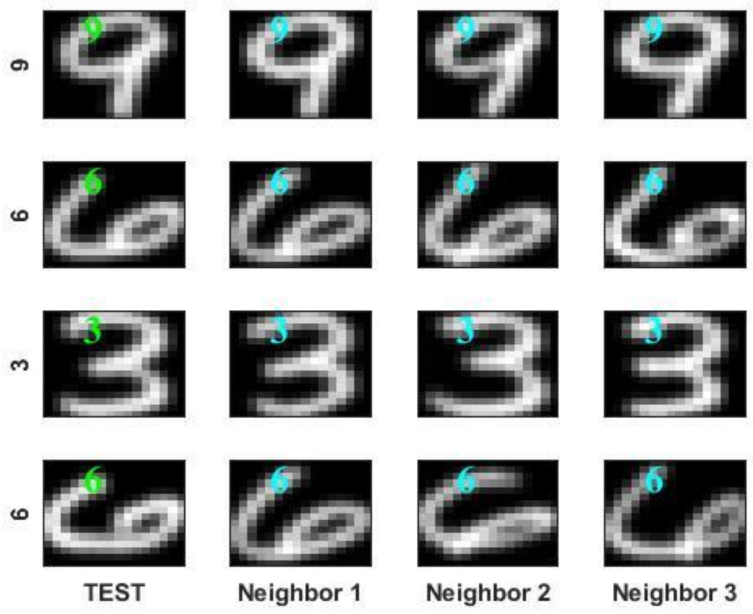
- 2nd method: Giving each k-nn a weight of $\frac{1}{Distance}$, summing distance for unique classification respectively, and pick the largest sum of weighting for classification.

	1-nn	2-nn	3-nn	4-nn	5-nn	6-nn	7-nn	8-nn	9-nn
Face Recognition (≅0.02 seconds)	95.83%	95.83%	91.67%	93.33%	89.17%	90.00%	87.50%	87.50%	85.83%

Handwritten digits Recognition (≅2.2 seconds)	95.02%	95.02%	94.97%	95.22%	95.27%	95.07%	94.97%	94.97%	94.62%

- 3nd method: Giving each k-nn a weight of $\frac{1}{k}$, summing distance for unique classification respectively, and pick the largest sum of weighting for classification.

→ The result is totally the same as the 1st method.


9
6
3
6
TEST
Neighbor 1
Neighbor 2
Neighbor 3