# Assignment 1

Changyan Xu (xuchangy)
changyan.xu@mail.utoronto.ca

October 1, 2020

## Question 1

**(a)** See function `load_data` in `hw1_q1_code.py`.

**(b)**

1. See function `select_knn_model` in `hw1_q1_code.py`.
2. See generated plot *"training and validation accuracy vs. k"* (Fig 1).

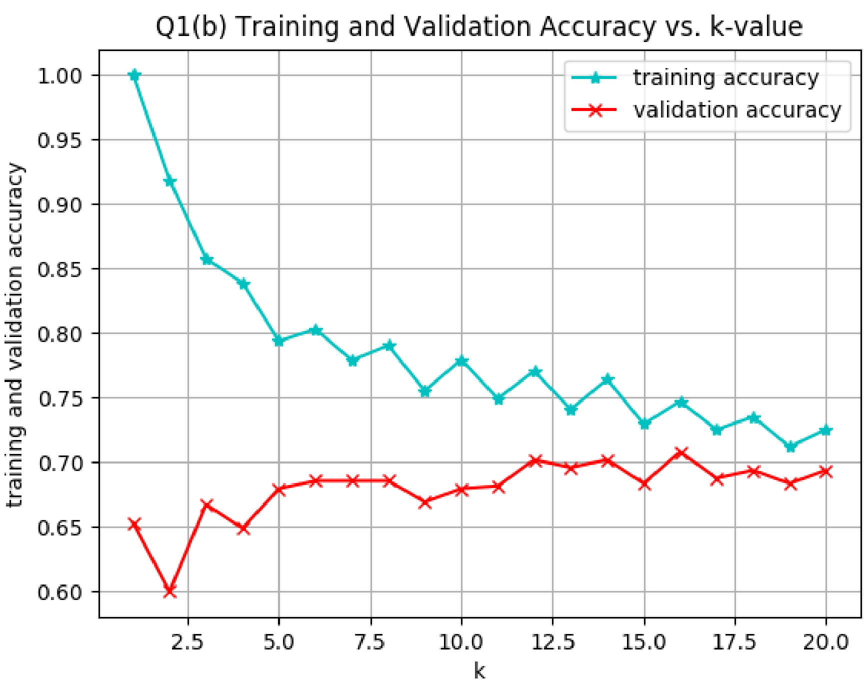



Figure 1: Training and Validation Accuracy vs. k-value.

3. Report the generated plot:
   - the training accuracy decreases from accuracy $= 1.00$ (in an almost exponential shape), as k-value increases
   - the validation accuracy gently increases as the k-value increases
   - both training and validation accuracy gradually converges to an interval of approximately $(0.70, 0.73)$

4. Choose the model with the best validation accuracy, and report its accuracy on the test data.

    - the best validation accuracy (red line with highest accuracy value) occurs at $\boxed{\text{k=16}}$. See the detailed computation in function `select_knn_model`.
    - by applying $k = 16$ on the test data, it gives an accuracy score of $\boxed{0.7}$.

**(c)**

1. See generated plot *"training and validation accuracy vs. k"* (Fig 2).

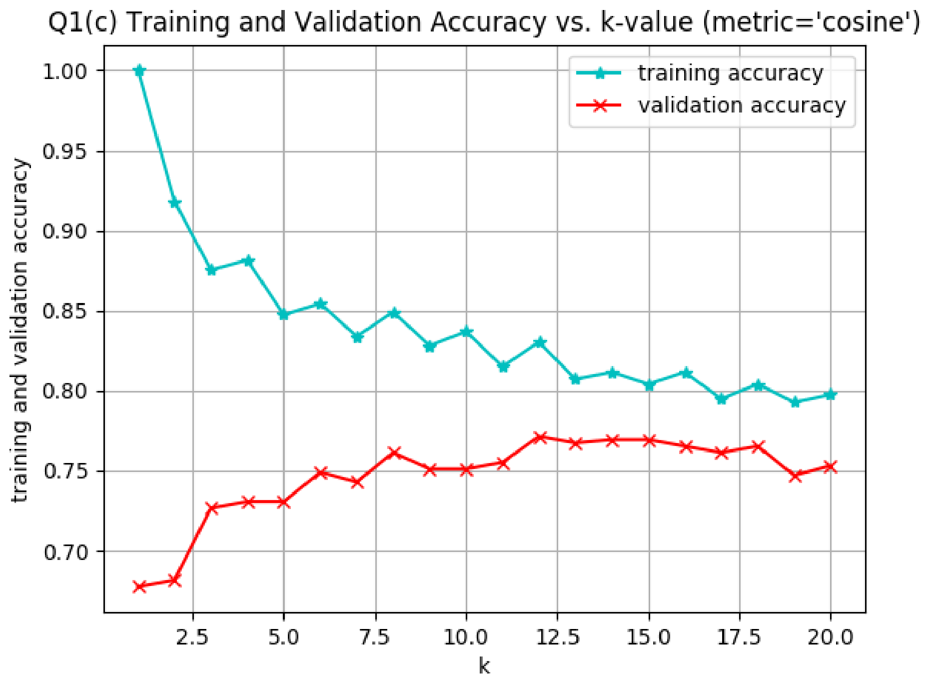



Figure 2: Training and Validation Accuracy vs. k-value. (metric='cosine')

2. Report the generated plot:

    - The accuracy has improved by passing `metric='cosine'` to `KNeighborsClassifier`.
    - Both training and validation accuracy gradually converges to an interval of approximately $(0.75, 0.80)$

3. Choose the model with the best validation accuracy, and report its accuracy on the test data.

    - the best validation accuracy (red line with highest accuracy value) occurs at $\boxed{\text{k=12}}$. See the detailed computation in function `select_knn_model`.
    - by applying $k = 12$ on the test data, it gives an accuracy score of approximately $\boxed{0.79}$.

4. How does `metric='cosine'` compute the distance between data points? And why might it perform better than the Euclidean metric (default) here?

   - The cosine distance only consider the direction (or angle) between the vectors, but does not consider the similarities on the magnitude of two vectors.

     - $\cos(0°) = 1$
     - $\cos(90°) = 0, \cos(-90°) = 0$
     - $\cos(180°) = -1$
   - Whereas the Euclidean distance calculates the magnitude of distance between to data points.
   - Instead of comparing and finding the nearest data points with certain threshold, the cosine distance checks angle between the data vectors and classify with 1, 0 or -1 cosine values.
   - In this case, the input data has significant large number of features, which led to the 'Curse of Dimensionality'. Cosine distance helps with reducing the dimensions of the data. I believe the mentioned reason led to a better performance of accuracy score on test datasets.
   - Reference:

`https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d`

```
hw1_q1_code
/usr/local/bin/python3.8 /Users/ChangyanXu/Desktop/CSC311/HW1/code/hw1_q1_code.py
(b) k_best: 16, test_score: 0.7
(c) k_best: 12, test_score: 0.7918367346938775

Process finished with exit code 0
```

Figure 3: Output by running the file `hw1_q1_code.py`)

# Question 2

$$y = \sum_{j=1}^{D} w_j x_j + b$$
$$\mathcal{R}(\mathrm{w}) = \frac{\lambda}{2}\mathrm{w}^\top \mathrm{w} = \frac{\lambda}{2}\sum_{j=1}^{D} w_j^2$$
$$\mathcal{J}_{reg}^{\beta}(\mathrm{w}) = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 + \frac{1}{2}\sum_{j=1}^{D}\beta_j w_j^2$$

**(a)** First,

$$\mathcal{J}_{reg}^{\beta}(\mathrm{w}) = \mathcal{J} + \mathcal{R}$$

the partial derivative of $\mathcal{J}_{reg}^{\beta}$ is :

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j} = \frac{\partial \mathcal{J}}{\partial w_j} + \frac{\partial \mathcal{R}}{\partial w_j} \qquad \frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial b} = \frac{\partial \mathcal{J}}{\partial b} + \frac{\partial \mathcal{R}}{\partial b} \tag{1}$$

$$\mathcal{J} = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(\sum_{j=1}^{D} w_j x_j^{(i)} + b - t^{(i)})^2$$

The derivative of $w_j$ for loss function $\mathcal{J}$based on matrix derivative:

$$\frac{\partial \mathcal{J}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(\sum_{j'=1}^{D} w_{j'} x_{j'}^{(i)} + b - t^{(i)}) = \boxed{\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)})}$$

The derivative of $b$ for loss function $\mathcal{J}$based on matrix derivative:

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(\sum_{j'=1}^{D} w_{j'} x_{j'}^{(i)} + b - t^{(i)}) = \boxed{\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})}$$

Next, we are finding the derivatives of regulation term.

$$\mathcal{R}(\mathrm{w}) = \frac{1}{2}\sum_{j=1}^{D}\beta_j w_j^2$$

The derivative of $w_j$ for loss function $\mathcal{R}$ based on matrix derivative:

$$\frac{\partial \mathcal{R}}{\partial w_j} = \frac{1}{2}\sum_{j=1}^{D} 2\beta_j w_j = \boxed{\beta_j w_j}$$

The derivative of $b$ for loss function $\mathcal{R}$based on matrix derivative:

$$\frac{\partial \mathcal{R}}{\partial b} = \boxed{0}$$

Therefore, from equation(1) and previous derivatives,

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j} = \frac{\partial \mathcal{J}}{\partial w_j} + \frac{\partial \mathcal{R}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)}) + \beta_j w_j \tag{2}$$

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial b} = \frac{\partial \mathcal{J}}{\partial b} + \frac{\partial \mathcal{R}}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} (y^{(i)} - t^{(i)}) \tag{3}$$

The update of $w_j$ is :

$$\begin{aligned}
w_j &\leftarrow w_j - \alpha \frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j} \\
&\leftarrow w_j - \alpha\left(\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)}) + \beta_j w_j\right) \\
&\leftarrow w_j - \frac{\alpha}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)}) - \alpha\beta_j w_j \\
w_j &\leftarrow \boxed{(1 - \alpha\beta_j) w_j - \frac{\alpha}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)})}
\end{aligned}$$

The update of $b$ is:

$$\begin{aligned}
b &\leftarrow b - \alpha \frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial b} \\
&\leftarrow \boxed{b - \frac{\alpha}{N}\sum_{i=1}^{N} (y^{(i)} - t^{(i)})}
\end{aligned}$$

The reason to call this regulation form "weight decay" is because every time we update the weight $w_j$ not only with the gradient $\nabla \mathcal{J}$, but also subtract it from $(\alpha\beta_j)w_j$ which leads less than the $w_j$ from the last time, therefore,$w_j$ has a tendency of decaying to zero.

**(b)** Since we already know our model is :

$$y = \sum_{j=1}^{D} w_j x_j$$

In order to derive the system of linear equation of the following form for $\mathcal{J}_{reg}^{\beta}$:

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j} = \sum_{j'=1}^{D} \mathrm{A}_{jj'} w_{j'} - \mathrm{c}_j = 0 \tag{4}$$

from equation (2) we get:

$$\begin{aligned}
\frac{\partial \mathcal{J}^{\beta}_{reg}}{\partial w_j} &= \frac{\partial \mathcal{J}}{\partial w_j} + \frac{\partial \mathcal{R}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}(y^{(i)} - t^{(i)}) + \beta_j w_j = 0 \\
&= \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)}((\sum_{j'=1}^{D} w_{j'} x_{j'}^{(i)} - t^{(i)}) + \beta_j w_j = 0 \\
&= \boxed{\frac{1}{N}\sum_{j'=1}^{D}(\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)}) w_{j'} - \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} + \beta_j w_j} = 0
\end{aligned}$$

Combined the upper solution with the equation (4) we have:

$$\mathrm{A}_{jj'} = \boxed{\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)}} \quad \& \quad \mathrm{c}_j = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} - \beta_j w_j \tag{5}$$

**(c)** From the previous question answer we have:

$$\mathbf{A} = \frac{1}{N}\mathrm{X}^{\top}\mathrm{X} \qquad \mathbf{c} = \frac{1}{N}\mathrm{X}^{\top}\mathrm{t} - \boldsymbol{\beta}\mathrm{w}$$

Also from equation(4) and above equations, we know that:

$$\begin{aligned}
\mathbf{A}\mathrm{w} &= \mathbf{c} \\
\frac{1}{N}\mathrm{X}^{\top}\mathrm{X}\mathrm{w} &= \frac{1}{N}\mathrm{X}^{\top}\mathrm{t} - \boldsymbol{\beta}\mathrm{w} \\
(\mathrm{X}^{\top}\mathrm{X} + N\boldsymbol{\beta}\mathrm{I})\mathrm{w} &= \mathrm{X}^{\top}\mathrm{t} \\
\mathrm{w} &= \boxed{(\mathrm{X}^{\top}\mathrm{X} + N\boldsymbol{\beta}\mathrm{I})^{-1}\mathrm{X}^{\top}\mathrm{t}}
\end{aligned}$$

## Question 3

From the question we knew that our linear model is:

$$y = \sum_{j=1}^{D} w_j x_j + b = \mathbf{w}^\top \mathbf{x} + b$$

Because we set

$$\mathbf{X} = \begin{bmatrix} \mathbf{1} & [\mathbf{x}^{(1)}]^\top \\ \mathbf{1} & [\mathbf{x}^{(2)}]^\top \\ \vdots & \\ \mathbf{1} & [\mathbf{x}^{(N)}]^\top \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

Therefore, our model for vector y is:

$$\mathbf{y} = \boxed{\mathbf{X}\mathbf{w} = Nb + [\mathbf{x}^{(1)}]^\top \mathbf{w} + \cdots + [\mathbf{x}^{(N)}]^\top \mathbf{w}} \tag{6}$$

since we set the cost function as the average loss over the training set:

$$\begin{aligned}
\mathcal{J} &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y^{(i)}, t^{(i)}) \\
&= \frac{1}{N}\sum_{i=1}^{N} (1 - cos(y^{(i)} - t^{(i)})) \\
&= \frac{1}{N}[N - \sum_{i=1}^{N} cos(y^{(i)} - t^{(i)})] \\
&= 1 - \frac{1}{N}\sum_{i=1}^{N} cos(y^{(i)} - t^{(i)}) \\
&= 1 - \frac{1}{N}\sum_{i=1}^{N} cos(\sum_{j=1}^{D} w_j x_j^{(i)} + b - t^{(i)})
\end{aligned}$$

Therefore, when we take the derivative of the cost function $\mathcal{J}$:

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \mathbf{y}} &= \frac{\partial}{\partial \mathbf{y}}[1 - \frac{1}{N}\sum_{i=1}^{N}(1 - cos(y^{(i)} - t^{(i)}))] \\
&= \frac{1}{N}\sum_{i=1}^{N} sin(y^{(i)} - t^{(i)}) \\
&= \boxed{\frac{1}{N}\sum_{i=1}^{N} sin(\mathbf{y} - \mathbf{t})}
\end{aligned}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \frac{\partial \mathcal{J}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{w}}, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{w}} = \mathbf{X}^\top$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = [\frac{1}{N}\sum_{i=1}^{N} sin(y^{(i)} - t^{(i)})]\mathbf{X}^\top$$

$$= \boxed{\frac{1}{N}\mathbf{X}^\top sin(\mathbf{y} - \mathbf{t}))}$$

Since $\frac{\partial \mathbf{y}}{\partial \mathrm{b}} = N$,

$$\frac{\partial \mathcal{J}}{\partial \mathrm{b}} = \frac{\partial \mathcal{J}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathrm{b}}$$

$$= \frac{1}{N} \sin(\boldsymbol{y} - \boldsymbol{t})N$$

$$= \boxed{\sin(\mathbf{y} - \mathbf{t})}$$

# Question 4

**(a)** See `hw1_q4_code.py` for importing data under *Q4(a) section*.

**(b)** See 6 functions in `hw1_q4_code.py` under *Q4(b) section*.

**(c)** See Fig 4. See the code and results in *Q4(c) section* of `hw1_q4_code.py`.

- The training error slowly increases from error approx. 0 to 0.8, as $\lambda$ increases from 0.00005 to 0.005
- The test error first sharply decreases from error approx. 2.6 to 1.1, and reach its minimum (min test error $\boxed{\approx 1.10}$) at $\boxed{\lambda \approx 0.0013}$ (denoted with the black vertical line). Then it gradually increases to $< 1.4$ as $\lambda = 0.005$

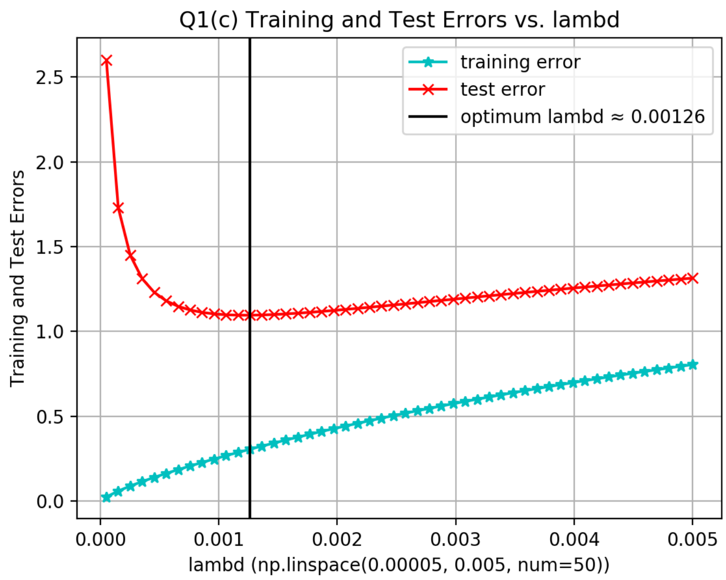



Figure 4: Training and Test Errors vs. $\lambda$.

**(d)** See Fig 5. See the code and results in *Q4(d) section* of `hw1_q4_code.py`.

1. What is the value of $\lambda$ proposed by your cross validation procedure?
   - Optimal Model: It is better to choose $\boxed{\text{5-Fold CV}}$ instead of 10-Fold CV in this case.
     - Because comparing to the 10-Fold CV, the 5-Fold CV has a lower generalisation error (, as the green curve is generally lower than the blue curve).
   - Optimal Lambda: $\boxed{\lambda \approx 0.00126}$
     - we find the lowest cv error over the 5-fold cross validation
2. Comment on the shapes of the error curves.

- The lines of test error, 5-Fold CV_Error and 10-Fold CV_Error have the very similar shape of first sharp drop and then slowly climbing up a bit.
- Only the line of training error has the much different shape of gradually growing errors as $\lambda$ increases.

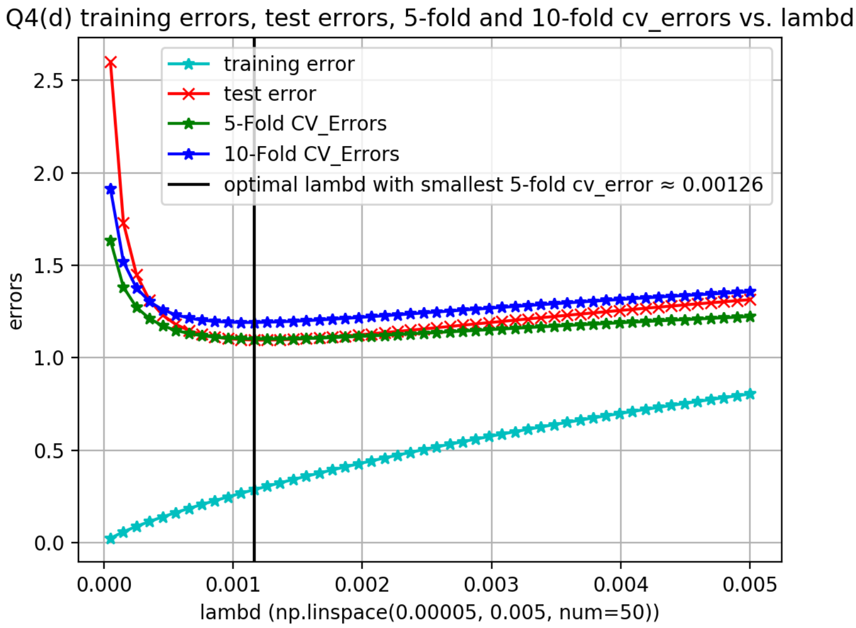
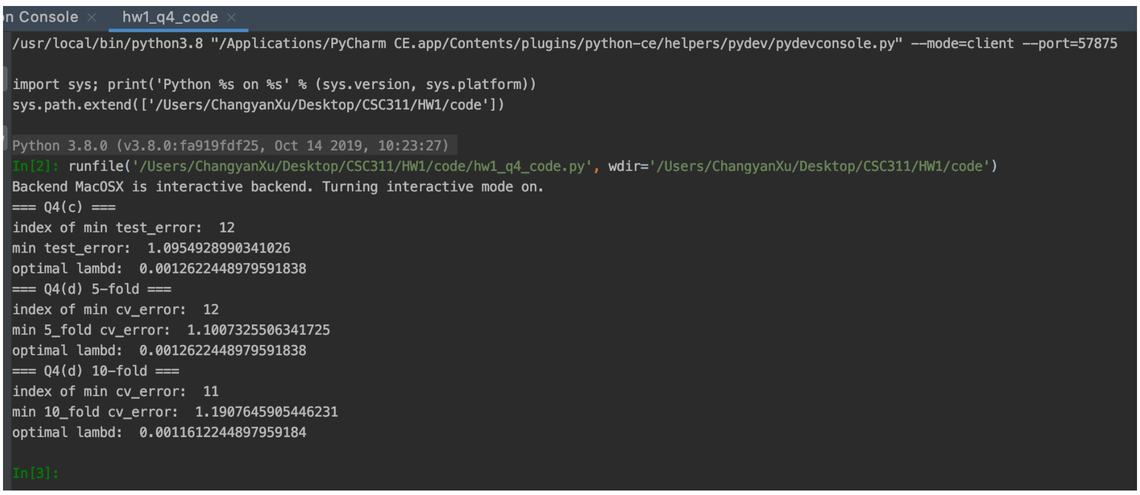



Figure 5: Training Errors, Test Errors, 5-fold Cross Validation Errors and 10-fold Cross Validation Errors vs. $\lambda$.



```
n Console    hw1_q4_code
/usr/local/bin/python3.8 "/Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/pydev/pydevconsole.py" --mode=client --port=57875

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/ChangyanXu/Desktop/CSC311/HW1/code'])

Python 3.8.0 (v3.8.0:fa919fdf25, Oct 14 2019, 10:23:27)
In[2]: runfile('/Users/ChangyanXu/Desktop/CSC311/HW1/code/hw1_q4_code.py', wdir='/Users/ChangyanXu/Desktop/CSC311/HW1/code')
Backend MacOSX is interactive backend. Turning interactive mode on.
=== Q4(c) ===
index of min test_error:  12
min test_error:  1.0954928990341026
optimal lambd:  0.0012622448979591838
=== Q4(d) 5-fold ===
index of min cv_error:  12
min 5_fold cv_error:  1.1007325506341725
optimal lambd:  0.0012622448979591838
=== Q4(d) 10-fold ===
index of min cv_error:  11
min 10_fold cv_error:  1.1907645905446231
optimal lambd:  0.0011612244897959184

In[3]:
```

Figure 6: Output by running the file `hw1_q4_code.py`)