EE2211 Tutorial 6

(Ridge Regression in Dual Form)
Question 1:

Derive the solution for linear ridge regression in dual form (see Lecture 6 notes page 16).

Answer: For $\lambda > 0$,

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}$$

$$\Rightarrow \mathbf{w} = \lambda^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w})$$

$$\Rightarrow \mathbf{w} = \lambda^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{a}$$

where

$$\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\Rightarrow \lambda\boldsymbol{a} = (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\Rightarrow \lambda\boldsymbol{a} = (\mathbf{y} - \mathbf{X}\mathbf{X}^T\boldsymbol{a})$$

$$\Rightarrow \mathbf{X}\mathbf{X}^T\boldsymbol{a} + \lambda\boldsymbol{a} = \mathbf{y}$$

$$\Rightarrow (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{a} = \mathbf{y}$$

$$\Rightarrow \boldsymbol{a} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$$

Hence,

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{a} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}.$$

(Polynomial Regression, 1D data)
Question 2:

Given the following datapairs for training:

$\{x = -10\} \rightarrow \{y = 5\}$
$\{x = -8\} \rightarrow \{y = 5\}$
$\{x = -3\} \rightarrow \{y = 4\}$
$\{x = -1\} \rightarrow \{y = 3\}$
$\{x = 2\} \rightarrow \{y = 2\}$
$\{x = 8\} \rightarrow \{y = 2\}$

EE2211 Tutorial 6

(Ridge Regression in Dual Form)

Question 1:

Derive the solution for linear ridge regression in dual form (see Lecture 6 notes page 16).

Answer: For $\lambda > 0$,

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w}$$

$$\Rightarrow \mathbf{w} = \lambda^{-1}(\mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w})$$

$$\Rightarrow \mathbf{w} = \lambda^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{a}$$

where

$$\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\Rightarrow \lambda\boldsymbol{a} = (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\Rightarrow \lambda\boldsymbol{a} = (\mathbf{y} - \mathbf{X}\mathbf{X}^T\boldsymbol{a})$$

$$\Rightarrow \mathbf{X}\mathbf{X}^T\boldsymbol{a} + \lambda\boldsymbol{a} = \mathbf{y}$$

$$\Rightarrow (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\boldsymbol{a} = \mathbf{y}$$

$$\Rightarrow \boldsymbol{a} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}$$

Hence,

$$\mathbf{w} = \mathbf{X}^T\boldsymbol{a} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{y}.$$

(Polynomial Regression, 1D data)

Question 2:

Given the following datapairs for training:

$\{x = -10\} \rightarrow \{y = 5\}$
$\{x = -8\} \rightarrow \{y = 5\}$
$\{x = -3\} \rightarrow \{y = 4\}$
$\{x = -1\} \rightarrow \{y = 3\}$
$\{x = 2\} \rightarrow \{y = 2\}$
$\{x = 8\} \rightarrow \{y = 2\}$

$$X = \begin{bmatrix} -10 \\ -8 \\ -3 \\ -1 \\ 2 \\ 8 \end{bmatrix} \quad y = \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

$$X\_b = \begin{bmatrix} 1 & -10 \\ 1 & -8 \\ 1 & -3 \\ 1 & -1 \\ 1 & 2 \\ 1 & 8 \end{bmatrix}$$

$$f(x) = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3$$

(a) Perform a 3rd-order polynomial regression and sketch the result of line fitting.
(b) Given a test point $\{\ x = 9\ \}$ predict y using the polynomial model.
(c) Compare this prediction with that of a linear regression.

Answer:

Polynomial model of 3rd order: $f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_1^2 + w_3x_1^3$.

$$\mathbf{P} = \begin{bmatrix} 1 & -10 & 100 & -1000 \\ 1 & -8 & 64 & -512 \\ 1 & -3 & 9 & -27 \\ 1 & -1 & 1 & -1 \\ 1 & 2 & 4 & 8 \\ 1 & 8 & 64 & 512 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix}.$$

Polynomial regression:

$$\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y}$$

$$= \begin{bmatrix} 6 & -12 & 242 & -1020 \\ -12 & 242 & -1020 & 18290 \\ 242 & -1020 & 18290 & -100212 \\ -1020 & 18290 & -100212 & 1525082 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & \\ -10 & -8 & \\ 100 & 64 & \\ -1000 & -512 & -2 \end{bmatrix}$$

$$= \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix}$$

Linear regression:

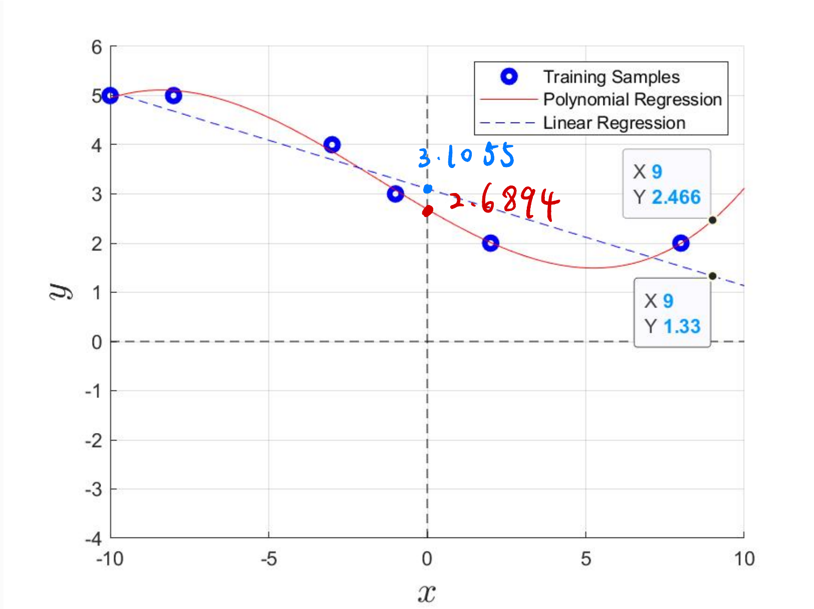
$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 6 & -12 \\ -12 & 242 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -10 & -8 & -3 & -1 & 2 & 8 \end{bmatrix} \begin{bmatrix} 5 \\ 5 \\ 4 \\ 3 \\ 2 \\ 2 \end{bmatrix} =$$

$$\begin{bmatrix} 3.1055 \\ -0.1972 \end{bmatrix}.$$

Prediction:

y_predict_Poly = 2.4661

y_predict_Linear = 1.3303



(Polynomial Regression, 3D data, Python)

Question 3:

$X = [x_1 \quad x_2 \quad x_3]$

(a) Write down the expression for a 3rd order polynomial model having a 3-dimensional input.

(b) Write down the **P** matrix for this polynomial given $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \end{bmatrix}$.

(c) Given $\mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, can a unique solution be obtained in dual form? If so, proceed to solve it.

(d) Given $\mathbf{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, can the primal ridge regression be applied to obtain a unique solution? If so, proceed to solve it.

$C(n,r) = \dfrac{(n+r-1)!}{r!\,(n-1)!}$

$= \dfrac{5!}{3! \times 2!} = \dfrac{120}{6 \times 2} = 10$

Answer:

(a) Polynomial model of 3rd order:

$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3$$

$$+ w_{12}\, x_1x_2 + w_{23}\, x_2x_3 + w_{13}\, x_1x_3 + w_{11}\, x_1^2 + w_{22}\, x_2^2 + w_{33}\, x_3^2$$
$$+ w_{211}\, x_2x_1^2 + w_{311}\, x_3x_1^2 + w_{122}\, x_1x_2^2 + w_{322}\, x_3x_2^2 + w_{133}\, x_1x_3^2 + w_{233}\, x_2x_3^2$$

$$+ \; w_{123}\, x_1x_2x_3 + w_{111}\, x_1^3 + w_{222}\, x_2^3 + w_{333}\, x_3^3 \quad \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (1)$$

(b) **P** = [

Columns 1 through 13

1	1	0	1	0	0	1	1	0	1	0	1	0
1	1	-1	1	-1	-1	1	1	1	1	-1	1	1

```
Columns 14 through 20

     0    1    0    0    1    0    1

     1    1   -1   -1    1   -1    1   ]
```

(c) Yes

$$\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{y} = \mathbf{P}^T\begin{bmatrix} 10 & 10 \\ 10 & 20 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$\hat{\mathbf{w}}^T =$[0 0 -0.1000 0 -0.1000 -0.1000 0 0 0.1000 0 -0.1000

0 0.1000 0.1000 0 -0.1000 -0.1000 0 -0.1000 0]

In python:

```
w_dual =

[ 0.  0.  -0.1  0.  0. -0.1  0.  0.1 -0.1  0.  0.  -0.1  0.  0.1  -0.1  0. -0.1  0.1 -0.1  0. ]
```

(Note: The arrangement of the polynomial terms in the columns of matrix **P** using `PolynomialFeatures` from `sklearn.preprocessing` might be different from that in equation(1).)

(d) Yes

$$\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^T\mathbf{y}$$

$\hat{\mathbf{w}}^T =$[0.0000 0.0000 -0.1000 0.0000 -0.1000 -0.1000 0.0000

0.0000 0.1000 0.0000 -0.1000 0.0000 0.1000 0.1000

0.0000 -0.1000 -0.1000 0.0000 -0.1000 0.0000]

In python:

```
w_primal = [ 9.99969302e-07  9.99972940e-07 -9.99980001e-02  9.99970098e-07

              9.99970666e-07 -9.99980000e-02  9.99967597e-07  9.99980000e-02

             -9.99980000e-02  9.99972485e-07  9.99969529e-07 -9.99980000e-02

              9.99968506e-07  9.99980000e-02 -9.99980001e-02  9.99970553e-07

             -9.99980001e-02  9.99980001e-02 -9.99980001e-02  9.99969416e-07]
```

(Note: The arrangement of the polynomial terms in the columns of matrix **P** using `PolynomialFeatures` from `sklearn.preprocessing` might be different from that in equation(1).)

Here, at $\lambda = 0.0001$, we observe a very close solution to that in (c) even though (d) constitutes an **approximation** whereas (c) is exact.

Codes:

```
import numpy as np

from numpy.linalg import inv

from sklearn.preprocessing import PolynomialFeatures

X = np.array([[1,0,1], [1,-1,1]])

y = np.array([0, 1])

## Generate polynomial features

order = 3

poly = PolynomialFeatures(order)

P = poly.fit_transform(X)

## dual solution (without ridge)

w_dual = P.T @ inv(P @ P.T) @ y

print(w_dual)

## primal ridge

reg_L = 0.0001*np.identity(P.shape[1])

w_primal_ridge = inv(P.T @ P + reg_L) @ P.T @ y

print(w_primal_ridge)
```

(Binary Classification, Python)

Question 4:

Given the training data:

$$\{x = -1\} \rightarrow \{\, y = class1 \,\}$$
$$\{x = \quad 0 \;\} \rightarrow \{\, y = class1 \,\}$$
$$\{x = 0.5\} \rightarrow \{\, y = class2 \,\}$$
$$\{x = 0.3\} \rightarrow \{\, y = class1 \,\}$$
$$\{x = 0.8\} \rightarrow \{\, y = class2 \,\}$$

Predict the class label for $\{x = -0.1\}$ and $\{x = 0.4\}$ using linear regression with signum discrimination.

Answer:

$$\mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 0.3333 \\ -1.1111 \end{bmatrix}$$

$$\mathrm{sgn}(\hat{\mathbf{y}}_t) = \mathrm{sgn}(\mathbf{X}_t\hat{\mathbf{w}}) = \mathrm{sgn}\left(\begin{bmatrix} 0.4444 \\ -0.1111 \end{bmatrix}\right) = \begin{bmatrix} class+1 \\ class-1 \end{bmatrix} \begin{matrix} \to class1 \\ \to class2 \end{matrix}$$

Codes:

```
import numpy as np
from numpy.linalg import inv
from sklearn.preprocessing import PolynomialFeatures
X = np.array([[1,-1], [1,0], [1,0.5], [1,0.3], [1,0.8]])
y = np.array([1, 1, -1, 1, -1])
## Linear regression for classification
w = inv(X.T @ X) @ X.T @ y
print(w)
Xt = np.array([[1,-0.1], [1,0.4]])
y_predict = Xt @ w
print(y_predict)
y_class_predict = np.sign(y_predict)
print(y_class_predict)
```

(Multi-Category Classification, Python)

Question 5:

Given the training data:

$$\begin{matrix} \{x=-1\} \to \{y = class1\} \\ \{x = 0\} \to \{y = class1\} \\ \{x=0.5\} \to \{y = class2\} \\ \{x=0.3\} \to \{y = class3\} \\ \{x=0.8\} \to \{y = class2\} \end{matrix}$$

(a) Predict the class label for $\{x=-0.1\}$ and $\{x=0.4\}$ based on linear regression towards a one-hot encoded target.

(b) Predict the class label for $\{x=-0.1\}$ and $\{x=0.4\}$ using a polynomial model of 5^{th} order and a one-hot encoded target.

Answer:

$$\mathbf{X} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix}, \ \mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \ \mathbf{X}_t = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.4 \end{bmatrix}.$$

(a) $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 0.4780 & 0.3333 & 0.1887 \\ -0.6499 & 0.5556 & 0.0943 \end{bmatrix}$

$$\hat{\mathbf{Y}}_t = \mathbf{X}_t\hat{\mathbf{w}} = \begin{bmatrix} 1 & -0.1 \\ 1 & 0.4 \end{bmatrix}\begin{bmatrix} 0.4780 & 0.3333 & 0.1887 \\ -0.6499 & 0.5556 & 0.0943 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5430 & 0.2778 & 0.1792 \\ 0.2180 & 0.5556 & 0.2264 \end{bmatrix} \Rightarrow \begin{bmatrix} class1 \\ class2 \end{bmatrix}$$

(b) Polynomial model of 5th order: $f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_1^2 + w_3x_1^3 + w_4x_1^4 + w_5x_1^5$

$$\mathbf{P} = \begin{bmatrix} 1.0000 & -1.0000 & 1.0000 & -1.0000 & 1.0000 & -1.0000 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.5000 & 0.2500 & 0.1250 & 0.0625 & 0.0313 \\ 1.0000 & 0.3000 & 0.0900 & 0.0270 & 0.0081 & 0.0024 \\ 1.0000 & 0.8000 & 0.6400 & 0.5120 & 0.4096 & 0.3277 \end{bmatrix}.$$

$$\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{Y} = \begin{bmatrix} 1.0000 & 0 & -0.0000 \\ -5.3031 & -3.7023 & 9.0055 \\ 5.2198 & 10.8728 & -16.0926 \\ 6.6662 & 9.4698 & -16.1360 \\ -6.4765 & -12.9099 & 19.3864 \\ -2.6199 & -7.8045 & 10.4244 \end{bmatrix}.$$

$$\mathbf{P}_t = \begin{bmatrix} 1.0000 & -0.1000 & 0.0100 & -0.0010 & 0.0001 & -0.0000 \\ 1.0000 & 0.4000 & 0.1600 & 0.0640 & 0.0256 & 0.0102 \end{bmatrix}.$$

$$\hat{\mathbf{Y}}_t = \mathbf{P}_t\hat{\mathbf{w}} = \begin{bmatrix} 1.5752 & 0.4683 & -1.0435 \\ -0.0521 & 0.4544 & 0.5977 \end{bmatrix} \Rightarrow \begin{bmatrix} class1 \\ class3 \end{bmatrix}.$$

Codes:

```
import numpy as np
from numpy.linalg import inv
from sklearn.preprocessing import PolynomialFeatures
X = np.array([[1,-1], [1,0], [1,0.5], [1,0.3], [1,0.8]])
Y = np.array([[1,0,0], [1,0,0], [0,1,0], [0,0,1], [0,1,0]])

## Linear regression for classification
W = inv(X.T @ X) @ X.T @ Y
print(W)
Xt = np.array([[1,-0.1], [1,0.4]])
y_predict = Xt @ W
print(y_predict)
y_class_predict = [[1 if y == max(x) else 0 for y in x] for x in y_predict ]
print(y_class_predict)

## Polynomial regression for
## Generate polynomial features
order = 5
poly = PolynomialFeatures(order)
## only the data column (2nd) is needed for generation of polynomial terms
reshaped = X[:,1].reshape(len(X[:,1]),1)
P = poly.fit_transform(reshaped)
reshaped = Xt[:,1].reshape(len(Xt[:,1]),1)
Pt = poly.fit_transform(reshaped)
```

```
## dual solution (without ridge)
Wp_dual = P.T @ inv(P @ P.T) @ Y
print(Wp_dual)
yp_predict = Pt @ Wp_dual
print(yp_predict)
yp_class_predict = [[1 if y == max(x) else 0 for y in x] for x in yp_predict ]
print(yp_class_predict)
```

(Multi-Category Classification, Python)

Question 6 (continued from Q3 of Tutorial 2):

Get the data set "`from sklearn.datasets import load_iris`". Use Python to perform the following tasks.

(a) Split the database into two sets: 74% of samples for training, and 26% of samples for testing. Hint: you might want to utilize `from sklearn.model_selection import train_test_split` for the splitting.
(b) Construct the target output using one-hot encoding.
(c) Perform a linear regression for classification (without inclusion of ridge, utilizing one-hot encoding for the learning target) and compute the number of test samples that are classified correctly.
(d) Using the same training and test sets as in above, perform a 2nd order polynomial regression for classification (again, without inclusion of ridge, utilizing one-hot encoding for the learning target) and compute the number of test samples that are classified correctly. Hint: you might want to use `from sklearn.preprocessing import PolynomialFeatures` for generation of the polynomial matrix.

<u>Codes:</u>

```
## (a) split data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(

iris_dataset['data'], iris_dataset['target'], test_size=0.26, random_state=0)



## (b) one-hot encoding

# Ytr_onehot = list()

# for i in y_train:

#     letter = [0, 0, 0]

#     letter[i] = 1

#     Ytr_onehot.append(letter)

# Yts_onehot = list()
```

```
# for i in y_test:

#     letter = [0, 0, 0]

#     letter[i] = 1

#     Yts_onehot.append(letter)

from sklearn.preprocessing import OneHotEncoder

onehot_encoder=OneHotEncoder(sparse=False)

reshaped = y_train.reshape(len(y_train), 1)

Ytr_onehot = onehot_encoder.fit_transform(reshaped)

reshaped = y_test.reshape(len(y_test), 1)

Yts_onehot = onehot_encoder.fit_transform(reshaped)



## (c) Linear Classification

w = inv(X_train.T @ X_train) @ X_train.T @ Ytr_onehot

print(w)

# calculate the output based on the estimated w and test input X and then assign
to one of the classes based on one hot encoding

yt_est = X_test.dot(w);

yt_cls = [[1 if y == max(x) else 0 for y in x] for x in yt_est ]

print(yt_cls)

# compare the predicted y with the ground truth

m1 = np.matrix(Yts_onehot)

m2 = np.matrix(yt_cls)

difference = np.abs(m1 - m2)

print(difference)

# calculate the error rate/accuracy

correct = np.where(~difference.any(axis=1))[0]
```

```
accuracy = len(correct)/len(difference)

print(len(correct))

print(accuracy)


## (d) Polynomial Classification

import numpy as np

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(2)

P = poly.fit_transform(X_train)

Pt = poly.fit_transform(X_test)

if P.shape[0] > P.shape[1]:

    wp = inv(P.T @ P) @ P.T @ Ytr_onehot

else:

    wp = P.T @ inv(P @ P.T) @ Ytr_onehot

print(wp)

yt_est_p = Pt.dot(wp);

yt_cls_p = [[1 if y == max(x) else 0 for y in x] for x in yt_est_p ]

print(yt_cls_p)

m1 = np.matrix(Yts_onehot)

m2 = np.matrix(yt_cls_p)

difference = np.abs(m1 - m2)

print(difference)

correct_p = np.where(~difference.any(axis=1))[0]

accuracy_p = len(correct_p)/len(difference)

print(len(correct_p))

print(accuracy_p)
```

(c) Correct prediction 28/39

(d) Correct prediction 38/39

Question 7

MCQ: there could be more than one answer. Given three samples of two-dimensional data points $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 3 & 3 \end{bmatrix}$ with corresponding target vector $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$. Suppose you want to use a full third-order polynomial model to fit these data. Which of the following is/are true?

a) The polynomials model has 10 parameters to learn
b) The polynomial learning system is an under-determined one
c) The learning of the polynomial model has infinite number of solutions
d) The input matrix $\mathbf{X}$ has linearly dependent samples
e) None of the above

Answer: a, b, c, d

Question 8

MCQ: there could be more than one answer. Which of the following is/are true?

a) The polynomial model can be used to solve problems with nonlinear decision boundary.

b) The ridge regression cannot be applied to multi-target regression.

c) The solution for learning feature $\mathbf{X}$ with target $\mathbf{y}$ based on linear ridge regression can be written as $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$ for $\lambda > 0$. As λ increases, $\hat{\mathbf{w}}^T\hat{\mathbf{w}}$ decreases.

d) If there are four data samples with two input features each, the full second-order polynomial model is an over-determined system.

Answer: a, c

(c) Correct prediction 28/39

(d) Correct prediction 38/39

Question 7

MCQ: there could be more than one answer. Given three samples of two-dimensional data points
$$\mathbf{X} = \begin{array}{cc} x_1 & x_2 \\ \end{array} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 3 & 3 \end{bmatrix} \begin{array}{c} ① \\ ② \\ ③ \end{array}$$
with corresponding target vector $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$. Suppose you want to use a full third-order polynomial model to fit these data. Which of the following is/are true?

$③ = 3 \times ①$

(a) 1
$x_1 \quad x_2$
$x_1^2 \quad x_2^2 \quad x_1 x_2$
$x_1^3 \quad x_2^3 \quad x_1^2 x_2 \quad x_2^2 x_1$

⑩

a) The polynomials model has 10 parameters to learn
b) The polynomial learning system is an under-determined one
c) The learning of the polynomial model has infinite number of solutions
d) The input matrix $\mathbf{X}$ has linearly dependent samples
e) None of the above

(b) P : 3×10 wide
under → infinite

Answer: a, b, c, d

Question 8

MCQ: there could be more than one answer. Which of the following is/are true?

a) The polynomial model can be used to solve problems with nonlinear decision boundary.

b) The ridge regression cannot be applied to multi-target regression.

c) The solution for learning feature $\mathbf{X}$ with target $\mathbf{y}$ based on linear ridge regression can be written as $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$ for $\lambda > 0$. As λ increases, $\hat{\mathbf{w}}^T\hat{\mathbf{w}}$ decreases.

d) If there are four data samples with two input features each, the full second-order polynomial model is an over-determined system.

(c) $w = (X^T X + \lambda I)^{-1} X^T y \downarrow$

$\lambda \uparrow, \quad \lambda I \uparrow$

$(X^T X + \lambda I)^{-1} \downarrow$

Answer: a, c

(d) $X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \end{bmatrix}$

⑥

P: $1, \; x_1, \; x_2$
$x_1 x_2, \; x_1^2, \; x_2^2$

P : 4×6 → wide → under