Hi everyone! Hope everyone had a good recess week!

EE2211 Introduction to Machine Learning

T14 & T22, Chua Dingjuan elechuad@nus.edu.sg

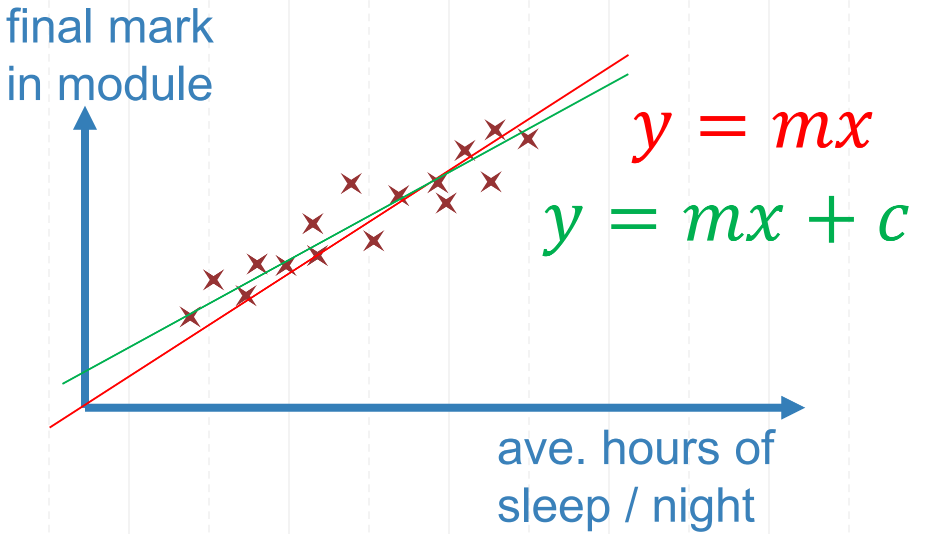
Materials @ tiny.cc/ee2211tut

Tutorial 6

Linear Regression + + +

What is Linear Regression?

	$1 \quad \ldots \quad d$	y	
	ave. hrs of sleep / night	final mark in module	
x_1	5	23	y_1
x_2	4	45	y_2
x_3	3	89	y_3
x_4	8	46	y_4
x_5	9	90	y_5
…	8.5	80	…
x_m	…	…	y_m

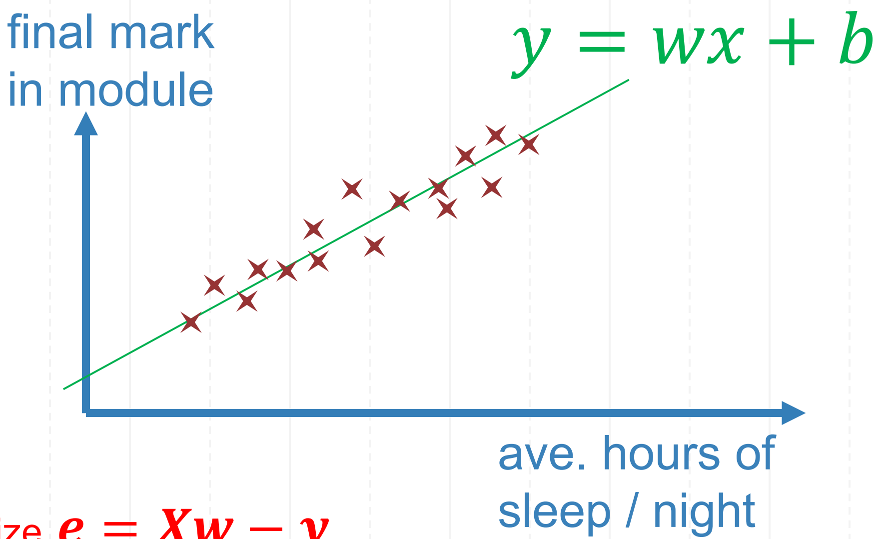


What is Linear Regression?

	1 … d	*y*	
	ave. hrs of sleep / night	final mark in module	
x_1	5	23	y_1
x_2	4	45	y_2
x_3	3	89	y_3
x_4	8	46	y_4
x_5	9	90	y_5
…	8.5	80	…
x_m	…	…	y_m

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} 23 \\ 45 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 1 & 4 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix}$$



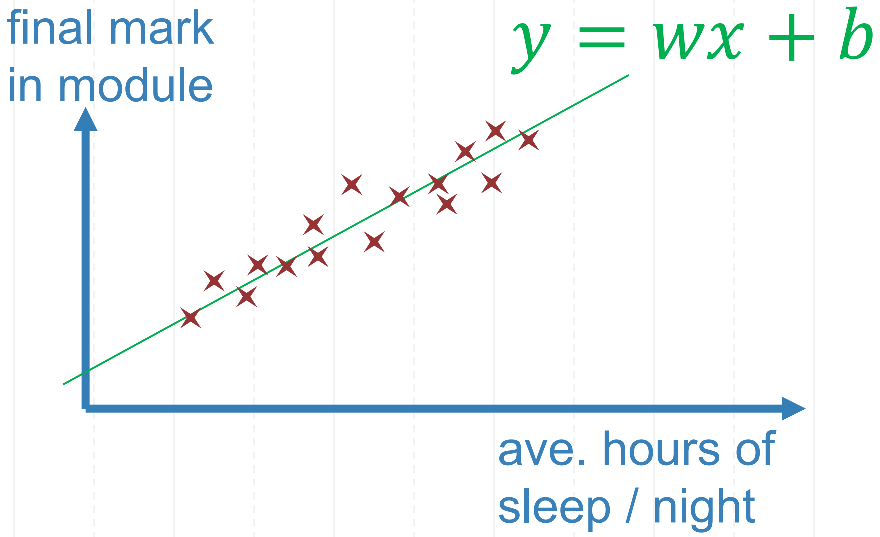
Minimize $\boldsymbol{e} = \boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$

Solution for overdet : $\hat{\boldsymbol{w}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$

Over Determined m > d	Under Determined m < d
No exact solution = approximate solution	infinite number of solutions = constrained solution $\mathbf{w} = \mathbf{X}^T\mathbf{a}$
Left inv $\mathbf{X}^{\dagger} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ $\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$ Least squares	Right inv $\mathbf{X}^{\dagger} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ $\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$ Least norms

Extension into....

	1 … d	*y*	
	ave. hrs of sleep / night	final mark in module	
x_1	5	23	y_1
x_2	4	45	y_2
x_3	3	89	y_3
x_4	8	46	y_4
x_5	9	90	y_5
…	8.5	80	…
x_m	…	…	y_m



Extension of concepts into :

- Binary Classification
- Multi-Category Classification
- Multiple Outputs Classification
- Polynomial Regression
- Ridge Regression

What if there are Multiple Outputs...?

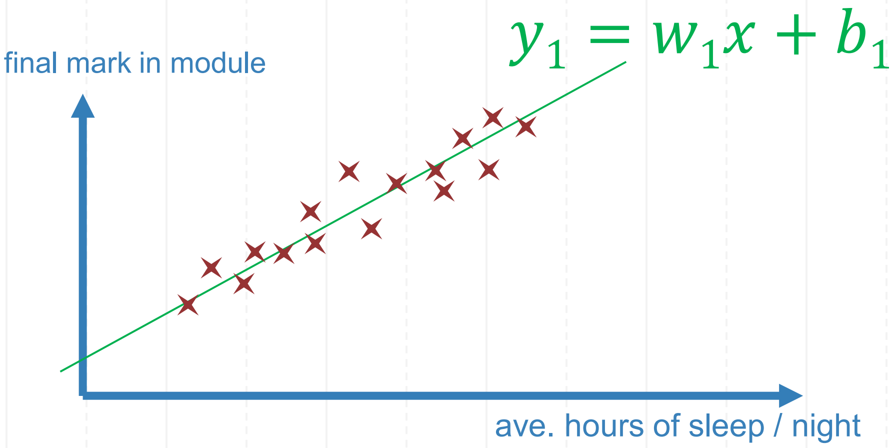
	1 … d	y_1	y_2
	ave. hrs of sleep / night	final mark in module	assignment mark
x_1	5	23 $y_{1,1}$	50 $y_{1,2}$
x_2	4	45 $y_{2,1}$	48 $y_{2,2}$
x_3	3	89 $y_{3,1}$	45 $y_{3,2}$
x_4	8	46 $y_{4,1}$	62 $y_{4,2}$
x_5	9	90 $y_{5,1}$	58 $y_{5,2}$
…	8.5	80 …	60 …
x_m	…	… $y_{m,1}$	… $y_{m,2}$

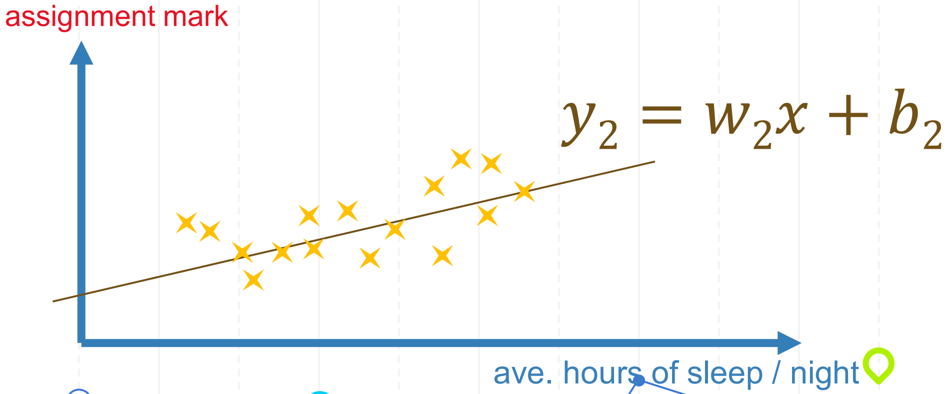
h

$$\boldsymbol{Y} = \boldsymbol{XW}$$

$$m \times h = (m \times d) \cdot (d \times h)$$

$$\begin{bmatrix} y_{1,1} & y_{2,1} \\ y_{2,1} & y_{2,2} \\ \ldots & \ldots \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \ldots \end{bmatrix} \begin{bmatrix} b_1 & b_2 \\ w_1 & w_2 \end{bmatrix}$$





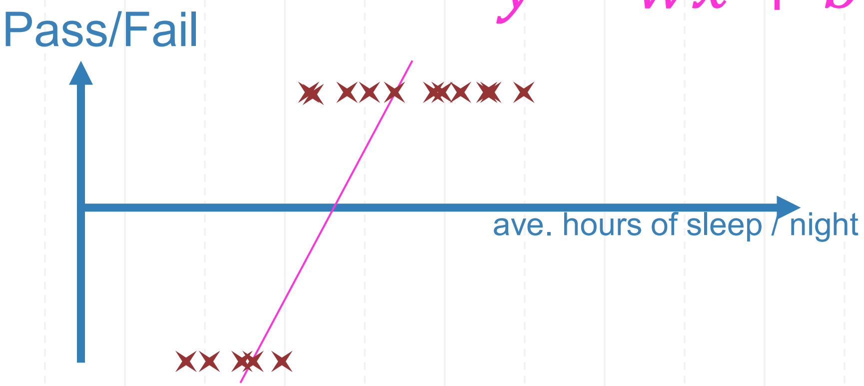
What if the question changes to binary classification?

	1 … d	y	
	ave. hrs of sleep / night	final pass/fail in module	-1 = fail 1 = pass
x_1	5	-1	y_1
x_2	4	-1	y_2
x_3	3	1	y_3
x_4	8	-1	y_4
x_5	9	1	y_5
…	8.5	1	…
x_m	…	…	y_m

$$\boldsymbol{y} = \boldsymbol{Xw}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} -1 \\ 1 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 1 & 4 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b \\ w \end{bmatrix}$$

$$y = wx + b$$



Steps :
- Assign target output/s into {-1,1} (or {0,1})
- Solve for linear regression as usual.
- When using model to predict output, use sgn() or threshold to identify class of output.

If the question changes to multi-category classification?

	1	…	d	$[y_1, y_2, y_3]$
	ave. hrs of sleep / night		final grade in module	Class1 = A, Class2 = B, Class3 = C
x_1	5		C	[0,0,1]
x_2	4		B	[0,1,0]
x_3	3		A	[1,0,0]
x_4	8		B	[0,1,0]
x_5	9		A	[1,0,0]
…	8.5		A	[1,0,0]
x_m	…		…	…
x_{test}	**6**		**???**	$[y_1, y_2, y_3]$
				$[0, 0.5, 0.9]$

$$Y = XW$$

$$\begin{bmatrix} y_{1,1} & y_{2,1} & y_{3,1} \\ y_{2,1} & y_{2,2} & y_{3,2} \\ \dots & \dots & \dots \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ \dots & \dots & \dots \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 1 & 4 \\ 1 & \dots \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$$

Steps :

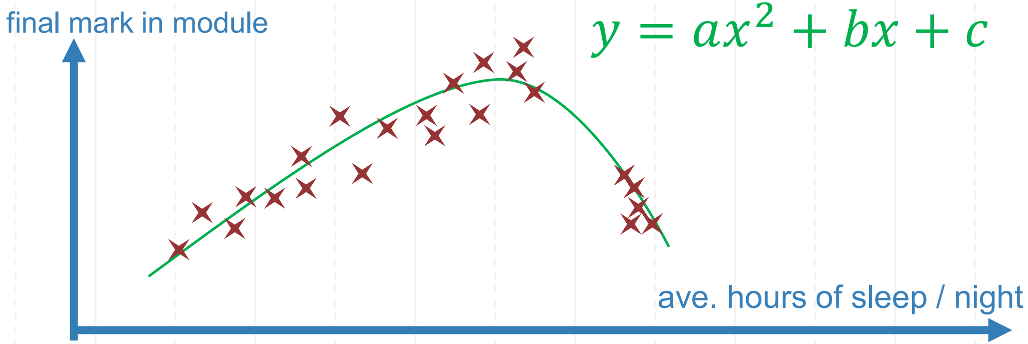
- Assign multiple outputs using one-hot encoding.
- Solve for linear regression as usual.
- When using model to predict output, position of largest output determines class label. ➔ $\arg\max_{i=1,\dots,C}$

What if Regression is NOT Linear?

	1 … d	y	
	ave. hrs of sleep / night	final mark in module	
x_1	5	23	y_1
x_2	4	45	y_2
x_3	3	89	y_3
x_4	8	46	y_4
x_5	9	90	y_5
…	8.5	80	…
x_m	13	30	y_m

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} \Rightarrow \begin{bmatrix} 23 \\ 45 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 4 & 16 \\ 1 & \dots & \dots \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix}$$



$$y = ax^2 + bx + c$$

Steps :

- Form full polynomial expression :

$$w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j +$$

- Solve for polynomial regression as usual.
- When using output to predict output, as usual.

Over Determined m > d	Under Determined m < d
No exact solution = approximate solution	infinite number of solutions = constrained solution $\mathbf{w} = \mathbf{X}^T\mathbf{a}$
Left inv $\mathbf{X}^{\dagger} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ $\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$ Least squares	Right inv $\mathbf{X}^{\dagger} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ $\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$ Least norms

Ridge Regression / Regularization

Primal	Dual Form
$\hat{w} = \left(X^T X + \lambda I\right)^{-1} X^T y$	$\hat{w} = X^T\left(X X^T + \lambda I\right)^{-1} y$

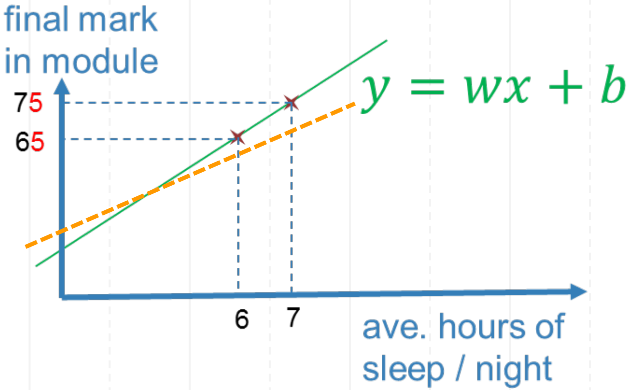
- In ridge regression, an additional term is added during minimization:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 + \lambda \mathbf{w}^T \mathbf{w}$$

- In applications where number of samples m, much smaller than d (polynomial regression is one example), ridge regression can be useful

- Effect of λ reduces w ➔ same Δx → smaller Δy
➔ predictions are less sensitive to training data ➔ training error ↑

- If XX^T / X^TX are non-invertible, the additional term λI makes it invertible



Over Determined m > d	Under Determined m < d
No exact solution = approximate solution	infinite number of solutions = constrained solution $\mathbf{w} = \mathbf{X}^T\mathbf{a}$
Left inv $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ $\hat{\mathbf{w}} = \mathbf{X}^\dagger\mathbf{y}$ Least squares	Right inv $\mathbf{X}^\dagger = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ $\hat{\mathbf{w}} = \mathbf{X}^\dagger\mathbf{y}$ Least norms

https://www.youtube.com/watch?v=Q81RR3yKn30&ab_channel=StatQuestwithJoshStarmer

Discussion of Solutions 6

Q1,2,3,4,5,6,7,8

(self-study)

Question2

Given the data pairs for training:
(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.

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

(a) SOLUTION

Full polynomial of third order with only one x input feature :

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

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

$$\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y} = \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix}$$

(b) {x = 9}

$$y = X.\hat{w} = [1 \quad 9 \quad 81 \quad 729] \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix} = 2.466$$

Question2

(a) SOLUTIONS – PYTHON CODES…

$$\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y} = \begin{bmatrix} 2.6894 \\ -0.3772 \\ 0.0134 \\ 0.0029 \end{bmatrix}$$

```
>>> z
array([ 0.00285772,  0.01343815, -0.37722517,  2.68935636])
```

```
#SOLUTION v1 - polyfit function within numpy#

x = np.array([-10, -8, -3, -1,  2,  8])
y = np.array([5, 5, 4, 3, 2, 2])

z = np.polyfit(x, y, 3)

model = np.poly1d(z)

ypredicted = model(9)
```

```
>>> ypredicted
2.4660977113619755
```

```
#---- SOLUTION v2 - Solving Step by Step ------#

x = np.array([[-10, -8, -3, -1,  2,  8]]).T
y = np.array([[5, 5, 4, 3, 2, 2]]).T

P = np.column_stack((np.ones(len(x)),x,x*x,x**3))

w = np.linalg.inv(P.T @ P) @ P.T @ y

xt = np.array([[9]]).T

testx = np.column_stack((np.ones(len(xt)),xt,xt*xt,xt**3))

ypredicted = testx @ w
```

same

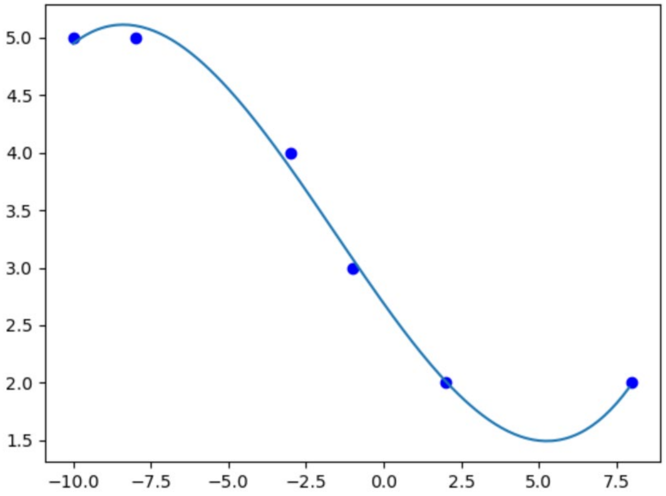
```
>>> P
array([[    1.,   -10.,   100., -1000.],
       [    1.,    -8.,    64.,  -512.],
       [    1.,    -3.,     9.,   -27.],
       [    1.,    -1.,     1.,    -1.],
       [    1.,     2.,     4.,     8.],
       [    1.,     8.,    64.,   512.]])
```

```
#-- SOLUTION v3 - Step by Step Using scikit --#

x = np.array([[-10, -8, -3, -1,  2,  8]]).T
y = np.array([[5, 5, 4, 3, 2, 2]]).T

from sklearn.preprocessing import PolynomialFeatures as skpf
polyfn = skpf(3)
P=polyfn.fit_transform(x)

w = np.linalg.inv(P.T @ P) @ P.T @ y

xt = np.array([[9]]).T

testx = polyfn.fit_transform(xt)

ypredicted = testx @ w
```

```
>>> w
array([[ 2.68935636],
       [-0.37722517],
       [ 0.01343815],
       [ 0.00285772]])
```

Question2

(a) PYTHON PLOTTING




```
#--- Plotting---#
plt.plot(x,y,'bo')
xline= np.arange(min(x),max(x),0.1)
yline=model(xline)

plt.plot(xline,yline)
plt.show()
```

Question2

Given the data pairs for training:
(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.

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

(c) SOLUTION

Linear Regression

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

$$\hat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 3.105 \\ -0.197 \end{bmatrix}$$

$$y = X.\hat{w} = [1 \quad 9] \begin{bmatrix} 3.105 \\ -0.197 \end{bmatrix} = 1.330$$

```
#### Q2C ####
x = np.array([[-10, -8, -3, -1,  2,  8]]).T
y = np.array([[5, 5, 4, 3, 2, 2]]).T

X = np.column_stack((np.ones((len(x),1)), x))

w = np.linalg.inv(X.T @ X) @ X.T @ y

ylinear = np.array([1 , 9]) @ w
```

Question3

(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 $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \end{bmatrix}$.

$\begin{array}{ccc} x_1 & x_2 & x_3 \end{array}$

(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.

(a) SOLUTION

(a) Polynomial model of 3rd order:

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$+ w_{12}\, x_1 x_2 + w_{23}\, x_2 x_3 + w_{13}\, x_1 x_3 + w_{11}\, x_1^2 + w_{22}\, x_2^2 + w_{33}\, x_3^2$$

$$+ w_{211}\, x_2 x_1^2 + w_{311}\, x_3 x_1^2 + w_{122}\, x_1 x_2^2 + w_{322}\, x_3 x_2^2 + w_{133}\, x_1 x_3^2 + w_{233}\, x_2 x_3^2$$

$$+ w_{123}\, x_1 x_2 x_3 + w_{111}\, x_1^3 + w_{222}\, x_2^3 + w_{333}\, x_3^3 \qquad (1)$$

Question3

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

(b) SOLUTION BY HAND

- By including additional terms involving the products of pairs of components of $\mathbf{x}$, we obtain a quadratic model: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j$.

By including even more terms, we can derive our own cubic model :

$$f_w(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{k=1}^{d} w_{ijk} x_i x_j x_k$$

For $X = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \end{bmatrix}$ ➔ There are three input features ➔ d = 3

$$f_w(x) = w_0 + \underline{\sum_{i=1}^{3} w_i x_i} + \sum_{i=1}^{3} \sum_{j=1}^{3} w_{ij} x_i x_j + \sum_{i=1}^{3} \sum_{j=1}^{3} \sum_{k=1}^{3} w_{ijk} x_i x_j x_k$$

Step by Step :

1) $\sum_{i=1}^{3} w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3$

Question3

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

(b) SOLUTION BY HAND

$$f_w(x) = w_0 + \sum_{i=1}^{3} w_i x_i + \underline{\sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij} x_i x_j} + \underline{\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} w_{ijk} x_i x_j x_k}$$

Step by Step :

2) $\sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij} x_i x_j$

$= w_{11}x_1x_1 + w_{12}x_1x_2 + w_{13}x_1x_3 + w_{22}x_2x_2 + w_{23}x_2x_3 + w_{33}x_3x_3$

$= w_{11}x_1^2 + w_{12}x_1x_2 + w_{13}x_1x_3 + w_{22}x_2^2 + w_{23}x_2x_3 + w_{33}x_3^2$

3) $\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} w_{ijk} x_i x_j x_k = w_{111}x_1x_1x_1 + w_{112}x_1x_1x_2 + w_{113}x_1x_1x_3 + w_{122}x_1x_2x_2 +$
$w_{123}x_1x_2x_3 + w_{133}x_1x_3x_3 + w_{222}x_2x_2x_2 + w_{223}x_2x_2x_3 + w_{232}x_2x_3x_2 + w_{233}x_2x_3x_3 + w_{333}x_3x_3x_3$
$= w_{111}x_1^3 + w_{112}x_1^2x_2 + w_{113}x_1^2x_3 + w_{122}x_1x_2^2 + w_{123}x_1x_2x_3 + w_{133}x_1x_3^2 + w_{222}x_2^3 + w_{223}x_2^2x_3 +$
$w_{233}x_2x_3^2 + w_{333}x_3^3$

Question3

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

(b) SOLUTION BY HAND

$$f_w(x) = w_0 + \sum_{i=1}^{3} w_i x_i + \sum_{i=1}^{3}\sum_{j=1}^{3} w_{ij} x_i x_j + \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} w_{ijk} x_i x_j x_k$$

Formula for total number of terms, contributed from student! →

$C = \frac{(n+r-1)!}{r!(n-1)!}$, where n = number of input features +1, and r = degree of polynomial

For 3 input features, degree of polynomial 3, n = 3 + 1, r = 3 → $C = \frac{(n+r-1)!}{r!(n-1)!} = \frac{6!}{3!3!} = 20$ → 20 terms

For 1 input feature1, degree of polynomial 4, n = 1 + 1, r = 4→ $C = \frac{(n+r-1)!}{r!(n-1)!} = \frac{5!}{4!1!} = 5$ → 5 terms

Question3

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

(b) SOLUTION BY CODE

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2], a^3, a^2b, ab^2, b^3...

Lecture slide 25 : **2nd order** polynomial model

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}\,x_1x_2 + w_{11}\,x_1^2 + w_{22}\,x_2^2$$

Note : polynomial model generated by scikit is NOT IN THE SAME sequence as your notes…

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures as skpf

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

polyfn = skpf(3)

P=polyfn.fit_transform(x)
```

```
>>> P
array([[ 1.,  1.,  0.,  1.,  1.,  0.,  1.,  0.,  0.,  1.,  1.,  0.,  1.,
         0.,  0.,  1.,  0.,  0.,  0.,  1.],
       [ 1.,  1., -1.,  1.,  1., -1.,  1.,  1., -1.,  1.,  1., -1.,  1.,
         1., -1.,  1., -1.,  1., -1.,  1.]])
```

Question3

(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) SOLUTION

Using dual form, a unique solution (invertible) can be solved without involving ridge regression (λ).
Same as solving like a typical under-determined system.

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

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

w = P.T @ inv(P @ P.T) @ y
```

```
>>> w
array([[ 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. ]])
```

```
>>> w_ridge
array([[ 9.99959639e-07],
       [ 9.99966005e-07],
       [-9.99980001e-02],
       [ 9.99970894e-07],
       [ 9.99969188e-07],
       [-9.99980001e-02],
       [ 9.99969075e-07],
       [ 9.99980000e-02],
       [-9.99980001e-02],
       [ 9.99971348e-07],
       [ 9.99969302e-07],
       [-9.99980000e-02],
       [ 9.99968165e-07],
       [ 9.99980001e-02],
       [-9.99980000e-02],
       [ 9.99969075e-07],
       [-9.99980001e-02],
       [ 9.99980000e-02],
       [-9.99980000e-02],
       [ 9.99967597e-07]])
```

(d) SOLUTION

If we were to now involve ridge regression (λ) :

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

```
ridge = 0.0001

w_ridge = inv(P.T @ P + ridge*np.identity(len(P.T))) @ P.T @ y
```

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Question4

Given the training data:

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

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

SOLUTION

Applying binary classification :

$$y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 0.5 \\ 1 & 0.3 \\ 1 & 0.8 \end{bmatrix} \Rightarrow w = \begin{bmatrix} 0.33 \\ -1.11 \end{bmatrix}$$

$$\hat{y}_t = \boldsymbol{X_t \hat{w}} = \boldsymbol{sgn}\left(\begin{bmatrix} 1 & -0.1 \\ 1 & 0.4 \end{bmatrix} \begin{bmatrix} 0.33 \\ -1.11 \end{bmatrix}\right)$$
$$= \text{sgn}\left(\begin{bmatrix} 0.44 \\ -0.11 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} class1 \\ class2 \end{bmatrix}$$

```python
import numpy as np
from numpy.linalg import inv

x = np.array([[-1, 0, 0.5, 0.3, 0.8]]).T
y = np.array([[1, 1, -1, 1, -1]]).T

X = np.column_stack((np.ones(len(x)),x))

w = inv(X.T @ X) @ X.T @ y

xtest = np.array([[-0.1, 0.4]]).T
Xtest = np.column_stack((np.ones(len(xtest)),xtest))

y_predict = Xtest @ w

y_predict_class = np.sign(y_predict)
```

```
>>> y_predict
array([ 0.44444444, -0.11111111])
>>> y_predict_class
array([ 1., -1.])
```

Question5

Given the training data:

$$\{x=-1\} \rightarrow \{y=class1\}$$
$$\{x=\ \ 0\} \rightarrow \{y=class1\}$$
$$\{x=0.5\} \rightarrow \{y=class2\}$$
$$\{x=0.3\} \rightarrow \{y=class3\}$$
$$\{x=0.8\} \rightarrow \{y=class2\}$$

(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 5th order and a one-hot encoded target.

(a) SOLUTION

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

$$\hat{\mathbf{Y}}_t=\mathbf{X}_t\hat{\mathbf{w}}=\begin{bmatrix}1 & -0.1\\ 1 & 0.4\end{bmatrix}\begin{bmatrix}0.4780 & 0.3333 & 0.1887\\ -0.6499 & 0.5556 & 0.0943\end{bmatrix}$$
$$=\begin{bmatrix}0.5430 & 0.2778 & 0.1792\\ 0.2180 & 0.5556 & 0.2264\end{bmatrix} \Rightarrow \begin{bmatrix}class1\\ class2\end{bmatrix}$$

```
import numpy as np
from numpy.linalg import inv

x = np.array([[-1, 0, 0.5, 0.3, 0.8]]).T
Y = np.array([[1,0,0], [1,0,0], [0,1,0], [0,0,1], [0,1,0]])

X = np.column_stack((np.ones(len(x)),x))

w = inv(X.T @ X) @ X.T @ Y

xtest = np.array([[-0.1, 0.4]]).T
Xtest = np.column_stack((np.ones(len(xtest)),xtest))

y_predict= Xtest @ w

#Creates an array which returns argmax
y_class_predict = [[1 if y == max(x) else 0 for y in x] for x in y_predict ]

#Alternative code below returns the argmax values per row
y_class_predict1 = np.argmax(y_predict,axis=1)
```

element is one if it's equal to max. of row

for each element in this row

row

for each row

matrix

```
>>> y_class_predict
[[1, 0, 0], [0, 1, 0]]
>>> y_class_predict1
array([0, 1], dtype=int64)
```

Question5

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

(b) SOLUTION

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

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

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

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

```python
#-- Q5B -- #
## Polynomial regression
from sklearn.preprocessing import PolynomialFeatures as skpf

x = np.array([[-1, 0, 0.5, 0.3, 0.8]]).T
xtest = np.array([[-0.1, 0.4]]).T

polyfn = skpf(5)
P=polyfn.fit_transform(x)

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

Ptest = polyfn.fit_transform(xtest)

y_predict = Ptest @ wp

y_class_predict = np.argmax(y_predict,axis=1)
```

```
>>> y_predict
array([[ 1.57522369,  0.46828063, -1.04350432],
       [-0.05207932,  0.45436978,  0.59770954]])
>>> y_class_predict
array([0, 2], dtype=int64)
```

```python
from sklearn.preprocessing import OneHotEncoder
enc=OneHotEncoder(categories=[[0,1,2]],sparse=False)
tmp = y_class_predict.reshape((len(y_class_predict),1))
enc.fit_transform(tmp)
```

Question6

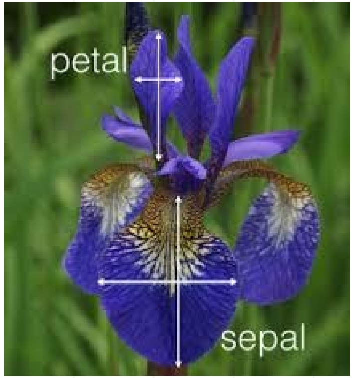
Get the data set “from sklearn.datasets import load_iris”. Use Python to perform the following tasks.

(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.




```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

Question6

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.

```
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)
```

```
>>> y_train.shape
(111,)
>>> y_test.shape
(39,)
```

(b) Construct the target output using one-hot encoding.

One-Hot Syntax Example:

```
>>> enc = OneHotEncoder(sparse=False)
>>> enc
OneHotEncoder(sparse=False)
>>> test=np.array([[0,1,2,3,4,5]]).T
>>> test
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
>>> onehot = enc.fit_transform(test)
>>> onehot
array([[1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1.]])
```

```
## (b) one-hot encoding
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)
```

Question6

(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.

m1
-m2

```
matrix([[ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0., -1.,  1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0., -1.,  1.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  1., -1.]])
```

```
>>> ~difference.any(axis=1)
matrix([[ True],
        [ True],
        [ True],
        [False],
        [ True],
        [ True],
        [ True],
        [False],
        [ True],
        [ True],
        [False],
        [False],
        [ True],
        [ True],
        [False],
        [ True],
        [False],
        [ True],
        [ True],
        [ True],
        [ True],
        [False],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [ True],
        [False],
        [ True],
        [ True],
        [ True],
        [False],
        [ True],
        [False],
        [False]])
```

```
## (c) Linear Classification
w = inv(X_train.T @ X_train) @ X_train.T @ Ytr_onehot
print(w)

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)

m1 = np.matrix(Yts_onehot)        #np.matrix here is optional
m2 = np.matrix(yt_cls)
difference = np.abs(m1 - m2)
print(difference)
correct = np.where(~difference.any(axis=1))[0]
accuracy = len(correct)/len(difference)

print(len(correct))
print(accuracy)
```

Y_test in one-hot format

predicted Y value in one-hot format

any() function returns True if **any** item is true, otherwise it returns False.

```
>>> np.where(~difference.any(axis=1))
(array([ 0,  1,  2,  4,  5,  6,  8,  9, 12, 13, 15, 17, 18, 19, 20, 22, 23,
        24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36], dtype=int64), array([0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0], dtype=int64))
```

Question6

(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.

```
## (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)
```

NUS National University of Singapore

Q7 More than one correct answer

A) $w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2 + w_{112} x_1^2 x_2 + w_{122} x_1 x_2^2 + w_{111} x_1^3 + w_{222} x_2^3$

$\Rightarrow$ 10 terms $\Rightarrow$ A is true

2 input features

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

$\Rightarrow$ 10 unknowns, 3 data pts.

$\therefore$ underdet $\Rightarrow$ B is true

C The learning of the polynomial model has infinite number of solutions

$\Rightarrow$ underdet $\Rightarrow \infty$ solns

$\therefore$ C is true

D The input matrix X has linearly dependent samples

$\Rightarrow$ Yes D is true

None of the above

Q8 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 $\mathbf{w}\hat{} = (\mathbf{X}T\mathbf{X}+\lambda\mathbf{I})-1\ \mathbf{X}T\mathbf{y}$ for $\lambda > 0$. As λ increases, $\mathbf{w}\hat{}T\mathbf{w}\hat{}$ decreases. ✓

D If there are four data samples with two input features each, the full second-order polynomial model is an overdetermined system. ✗

Question1

Derive the solution for linear ridge regression in dual form (see Lecture 6 notes page 20).

SOLUTION

(Derivation as homework, hint: start off with $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$ and make use of $\mathbf{w} = \mathbf{X}^T\mathbf{a}$ with $\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$)

Expand, make w the subject

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

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

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

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

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

Similar to tutorial 4 q6,

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

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

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

we need to still get rid of this!

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

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

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

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

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

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