CS189

1. (a) By myself. consulting piazza.

(b) I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

Yao Cao

HW3

2. (a) $Z \sim N(0,1)$

$Y|X \sim N(Xw_1 + w_0, 1)$

(b) $L = P(Y_1, Y_2 \cdots Y_n | X_1, X_2, \cdots X_n)$

$$\overset{iid}{=} P(Y_1|X_1)P(Y_2|X_2)\cdots P(Y_n|X_n)$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y_i - x_i w_1 - w_0)^2}{2}\right)$$

$$l = \log L = \sum_{i=1}^{n}\left(\log\frac{1}{\sqrt{2\pi}} - \frac{(y_i - x_i w_1 - w_0)^2}{2}\right)$$

$$= -\sum_{i=1}^{n}\frac{(y_i - x_i w_1 - w_0)^2}{2} + \text{const}.$$

$$\frac{\partial l}{\partial w_1} = -\sum_{i=1}^{n}(y_i - x_i w_1 - w_0)\cdot(-x_i)$$

$$= \sum_{i=1}^{n}(x_i y_i - x_i^2 w_1 - w_0 x_i) = 0$$

$$\Rightarrow \left(\sum_{i=1}^{n} x_i^2\right) w_1 = \sum_{i=1}^{n}(x_i y_i - w_0 x_i)$$

$$\hat{w}_1 = \frac{\sum_{i=1}^{n}(x_i y_i - w_0 x_i)}{\sum_{i=1}^{n} x_i^2} \quad ①$$

$$\frac{\partial^2 l}{\partial w_1^2} = \sum_{i=1}^{n}(-x_i^2) < 0$$

so this is maximum

$$\frac{\partial l}{\partial w_0} = -\sum_{i=1}^{n}(y_i - x_i w_1 - w_0)\cdot(-1) = 0$$

$$= \sum_{i=1}^{n}(y_i - x_i w_1 - w_0) = 0$$

$$\Rightarrow n w_0 = \sum_{i=1}^{n} y_i - \left(\sum_{i=1}^{n} x_i\right) w_1$$

$$\hat{w}_0 = \frac{1}{n}\sum_{i=1}^{n} y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) w_1 \quad ②$$

$$\frac{\partial^2 l}{\partial w_0^2} = \sum_{i=1}^{n}(-1) < 0$$

so this is maximum.

combine ① & ②, we have

$$\hat{w}_1 = \frac{\overline{X_n Y_n} - \overline{X_n}\,\overline{Y_n}}{\overline{X_n^2} - \overline{X_n}^2}$$

where $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\overline{Y_n} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ and so on ....

$$\hat{w}_0 = \overline{Y_n} - \overline{X_n}\hat{w}_1 = \overline{Y_n} - \overline{X_n}\cdot\frac{\overline{X_n Y_n} - \overline{X_n}\,\overline{Y_n}}{\overline{X_n^2} - \overline{X_n}^2}$$

(c) $Y|X \sim U[-0.5 + Xw, 0.5 + Xw]$

(d) $P(Y|X) = 1$ when $-0.5 \leq Y - Xw \leq 0.5$

$= 0$ elsewhere

$L = P(Y_1, \cdots Y_n | X_1 \cdots X_n)$

$= \prod_{i=1}^{n} P(Y_i|X_i) = 1$ when $-0.5 \leq Y_i - X_i w \leq 0.5$ for every i

$= 0$ elsewhere

for every i,

$-0.5 + X_i w \leq Y_i \leq 0.5 + X_i w$ satisfy if

$\max(Y_i) \leq 0.5 + \min(X_i) w$ &

$\min(Y_i) \geq -0.5 + \max(X_i) w$.

$$\Rightarrow \frac{\max(Y_i) - 0.5}{\min(X_i)} \leq w \leq \frac{\min(Y_i) + 0.5}{\max(X_i)}$$

so $\hat{w}$ is not unique

(e) as n gets large, we have more accurate estimate of w.

If we have more data, we can test each w value more times and can more easily tell if it is likely to be the true value or not

(4) $P(W|Data) = P(W|y_1, y_2, \cdots y_n, x_1, x_2, \cdots x_n)$

$$\propto P(y_1, \cdots y_n | W, x_1, \cdots x_n)\, P(W)$$

$$\propto \prod_{i=1}^{n} P(Y_i | W, X_i) \cdot P(W)$$

$$\propto \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(Y_i - X_i W)^2}{2}\right] \cdot \frac{1}{\sqrt{2\pi}\cdot\sigma} \exp\left[-\frac{W^2}{2\sigma^2}\right]$$

$$\propto \exp\left[-\frac{\sum_{i=1}^{n}(Y_i - X_i W)^2}{2} - \frac{W^2}{2\sigma^2}\right]$$

$$\propto \exp\left[-\frac{W^2(n\overline{X_n^2}\sigma^2 + 1) - 2nW\overline{X_nY_n}\cdot\sigma^2 + n\overline{Y_n^2}\cdot\sigma^2}{2\sigma^2}\right]$$

$$\propto \exp\left[-\frac{W^2 - \frac{2n\overline{X_nY_n}\sigma^2}{n\overline{X_n^2}\cdot\sigma^2+1}\cdot W + Const}{\frac{2\sigma^2}{n\overline{X_n^2}\sigma^2+1}}\right]$$

$\boxed{\text{mean}}$ $$E[W|Data] = \frac{n\cdot\overline{X_nY_n}\,\sigma^2}{n\cdot\overline{X_n^2}\,\sigma^2+1} \quad ①$$

$\boxed{\text{variance}}$ $$Var[W|Data] = \frac{\sigma^2}{n\overline{X_n^2}\,\sigma^2+1} \quad ②$$

so $P(W|Data) \sim N\Big(E[W|Data], Var[W|Data]\Big)$

(g) $L = P(Y_1, \cdots Y_n \mid x_1 \cdots x_n, w)$

$$= \prod_{i=1}^{n} P(Y_i \mid \vec{x}_i, \vec{w})$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(Y_i - \vec{W}^T \vec{x}_i)^2}{2}\right]$$

$$\ell = \log L \propto \sum_{i=1}^{n} \left(-\frac{(Y_i - \vec{w}^T \vec{x}_i)^2}{2}\right)$$

$$\underset{\vec{w}}{\arg\max}\, \ell = \underset{\vec{w}}{\arg\min} \sum_{i=1}^{n} (Y_i - \vec{x}_i^T \vec{w})^2$$

$$= \underset{\vec{w}}{\arg\min} \|\vec{Y} - X\vec{w}\|_2^2$$

where $\vec{Y} = (Y_1, \cdots Y_n)^T$

$$X = \begin{pmatrix} - x_1^T - \\ - x_2^T - \\ \vdots \\ - x_n^T - \end{pmatrix}$$

which is the same problem of OLS

(h) $P(W \mid \text{Data}) = P(\vec{W} \mid Y_1, \cdots Y_n, \vec{x}_1, \cdots \vec{x}_n)$

$$\propto P(Y_1, \cdots Y_n \mid \vec{W}, \vec{x}_1, \cdots \vec{x}_n)\, P(\vec{w})$$

$$= \prod_{i=1}^{n} P(Y_i \mid \vec{w}, \vec{x}_i) \cdot P(\vec{w})$$

$$\propto \prod_{i=1}^{n} \exp\left[-\frac{(Y_i - \vec{x}_i^T \cdot \vec{w})^2}{2}\right] \cdot \exp\left[-\frac{1}{2}\vec{W}^T \Sigma^{-1} \vec{W}\right]$$

$$\propto \exp\left[-\frac{\sum_{i=1}^{n}(Y_i - \vec{x}_i^T \vec{w})^2 + \frac{1}{\sigma^2}\vec{W}^T \vec{W}}{2}\right]$$

$$\Sigma = \begin{pmatrix} \sigma^2 & & & \\ & \sigma^2 & & \\ & & \ddots & \\ & & & \sigma^2 \end{pmatrix} \in \mathbb{R}^{d \times d}$$

$$\propto \exp\left[-\frac{\|\vec{Y}\|_2^2 - 2\sum_{i=1}^{n} Y_i \vec{X}_i^T \vec{w} + \sum_{i=1}^{n} \vec{w}^T \vec{X}_i \vec{X}_i^T w + \frac{1}{\sigma^2}\vec{w}^T \vec{w}}{2}\right]$$

$$\Sigma^{-1} = \frac{1}{\sigma^2} I$$

$$\propto \exp\left[-\frac{\vec{Y}^T \vec{Y} - 2\vec{w}^T X^T \vec{Y} + w^T X^T X w + \frac{1}{\sigma^2}\vec{w}^T \vec{w}}{2}\right]$$

$$\propto \exp\left[-\frac{w^T(\frac{1}{\sigma^2} I + X^T X) w - 2\vec{w}^T X^T \vec{Y} + \text{const.}}{2}\right]$$

mean $\quad E[\vec{w}] = (\frac{1}{\sigma^2} I + X^T X)^{-1} X^T \vec{Y}$

variance $\quad Var[\vec{w}] = (\frac{1}{\sigma^2} I + X^T X)^{-1} = \Sigma'$

$$\vec{W} \mid \text{Data} \sim N\left((\tfrac{1}{\sigma^2} I + X^T X)^{-1} X^T \vec{Y},\ (\tfrac{1}{\sigma^2} I + X^T X)^{-1}\right)$$

(i) more data, more accurate the estimation

at small n, σ^2 influences the distribution more

at large n, σ^2 has smaller influence on the distribution

3\. (a) ① $E[\hat{X} - u]$

$= E\left[\frac{1}{n}(X_1 + \cdots + X_n) - u\right]$

$= \frac{1}{n} \cdot nE[X] - u$

$= 0$

② $E[\hat{X} - u]$

$= E\left[\frac{1}{n+1}(X_1 + \cdots + X_n) - u\right]$

$= \frac{1}{n+1} \cdot nE[X] - u$

$= \left(\frac{n}{n+1} - 1\right) u = \frac{-1}{n+1} u.$

③ $E\left[\frac{1}{n+n_0}(X_1 + \cdots + X_n) - u\right]$

$= \left(\frac{n}{n+n_0} - 1\right) u = \frac{-n_0}{n+n_0} u.$

④ $E[0 - u] = -u.$

(b) $Var\left[\frac{1}{n}(X_1 + \cdots + X_n)\right]$

$= \frac{1}{n^2} Var[X_1 + \cdots + X_n]$

$= \frac{1}{n^2} \cdot n\, Var[X] = \frac{1}{n} \sigma^2$

$Var\left[\frac{1}{n+1}(X_1 + \cdots + X_n)\right]$

$= \frac{1}{(n+1)^2} \cdot n Var[X] = \frac{n}{(n+1)^2} \sigma^2$

$Var\left[\frac{1}{n+n_0}(X_1 + \cdots + X_0)\right] = \frac{n}{(n+n_0)^2} \sigma^2$

$Var[0] = 0$

$$\text{bias}(\hat{x}) = E[\hat{x} - u]$$
$$= E[\hat{x}] - u$$
$$\Rightarrow E[\hat{x}] = \text{bias}(\hat{x}) + u$$

$$Var(\hat{x}) = E((\hat{x} - E(\hat{x}))^2)$$
$$= E(\hat{x}^2) - E(\hat{x})^2$$

(c) $E[(\hat{x} - x')^2]$

$$= E[\hat{x}^2 - 2\hat{x}x' + x'^2]$$
$$= E[\hat{x}^2] - 2E[\hat{x}x'] + E[x'^2]$$
$$= E[\hat{x}^2] - 2E[\hat{x}]E[x] + E[x^2]$$

$\hat{x}$, x' independent.

$$= Var(\hat{x}) + (\text{bias}(\hat{x}) + u)^2 - 2(\text{bias}(\hat{x}) + u)u + \sigma^2 + u^2$$

$$= Var(\hat{x}) + \text{bias}^2 + u^2 + 2\text{bias} \cdot u - 2\text{bias} \cdot u - 2u^2 + \sigma^2 + u^2$$
$$= Var(\hat{x}) + \text{bias}^2(\hat{x}) + \sigma^2 \quad ①$$

$E[(\hat{x} - u)^2]$

$$= E[(\hat{x}^2 - 2u\hat{x} + u^2]$$
$$= E[\hat{x}^2] - 2uE[\hat{x}] + u^2$$
$$= Var(\hat{x}) + (\text{bias} + u)^2 - 2u(\text{bias} + u) + u^2$$
$$= Var(\hat{x}) + \text{bias}^2 + 2 \cdot \text{bias} \cdot u + u^2 - 2u(\text{bias}) - 2u^2 + u^2$$
$$= Var(\hat{x}) + \text{bias}^2(\hat{x}) \quad ②$$

the test error ① includes Variance and bias term as well as an irreducible error σ^2

the true error ② includes only variance and bias

but they are only off by a constant σ^2, so the $\hat{x}$ makes ① smallest should also makes ② smallest

(d) ① $Var(\hat{x}) + bias^2(\hat{x}) = \frac{1}{n}\sigma^2 + 0 = \frac{1}{n}\sigma^2$

② $\frac{n}{(n+1)^2}\sigma^2 + \frac{1}{(n+1)^2}u^2 = \frac{n\sigma^2 + u^2}{(n+1)^2}$

③ $\frac{n}{(n+n_0)^2}\sigma^2 + \frac{n_0^2}{(n+n_0)^2}u^2 = \frac{n\sigma^2 + n_0^2 u^2}{(n+n_0)^2}$

④ $0 + u^2 = u^2$

(e) ① when $n_0 = 0$ $\quad \frac{n\sigma^2 + n_0^2 u^2}{(n+n_0)^2} = \frac{\sigma^2}{n}$

② when $n_0 = 1$ $\quad \frac{n\sigma^2 + n_0^2 u^2}{(n+n_0)^2} = \frac{n\sigma^2 + u^2}{(n+1)^2}$

③ ……

④ when $n_0 \to \infty$ $\quad \frac{n\sigma^2 + n_0^2 u^2}{(n+n_0)^2} = u^2$

(f) as n_0 increases, bias increases, variance decreases

(g) error $= \frac{n\sigma^2 + (\alpha n)^2 u^2}{(n+\alpha n)^2} = \frac{\sigma^2 + \alpha^2 n u^2}{n(1+\alpha)^2}$

$$\frac{\partial(error)}{\partial\alpha} = \frac{2\alpha n u^2}{n(1+\alpha)^2} - \frac{2(\sigma^2 + \alpha^2 n u^2)}{n(1+\alpha)^3}$$

$$= \frac{2\alpha n u^2 + 2\alpha^2 n u^2 - 2\sigma^2 - 2\alpha^2 n u^2}{n(1+\alpha)^3}$$

$$= \frac{2\alpha n u^2 - 2\sigma^2}{n(1+\alpha)^3} = 0$$

$$\alpha = \frac{\sigma^2}{n u^2}$$

(h) $\alpha \to \infty$

(i) $x' = X - u_0$

$$E[X'] = E[X - u_0] = E[X] - u_0 = u - u_0$$

$$\begin{aligned} Var[X'] &= Var[X - u_0] \\ &= Var[X] + Var[u_0] \\ &= Var[X] = \sigma^2 \end{aligned}$$

(j) We should pick a λ that minimizes ($bias^2$ + Var)
(using validation)

$\lambda \to \infty$ corresponds to a very small $|w|$

$\alpha \to \infty$ corresponds to very small mean u

4. (a) $E[\hat{w}] = E[(X^TX)^{-1}X^T\vec{y}]$

$$= (X^TX)^{-1}X^T E[\vec{y}^* + z]$$
$$= (X^TX)^{-1}X^T(\vec{y}^* + 0)$$
$$= (X^TX)^{-1}X^T\vec{y}^* = w^*$$

$$E[\|y^* - X\hat{w}\|_2^2]$$
$$= E[(y^* - X\hat{w})^T(y^* - X\hat{w})]$$
$$= E[\vec{y}^{*T}\vec{y}^* + \hat{w}^TX^TX\hat{w} - 2\vec{y}^{*T}X\hat{w}]$$
$$= \vec{y}^{*T}\vec{y}^* + E(\hat{w}^TX^TX\hat{w}) - 2\vec{y}^{*T}Xw^* \qquad ①$$

$$\|y^* - E[X\hat{w}]\|_2^2 + E[\|X\hat{w} - E[X\hat{w}]\|_2^2]$$
$$= (y^* - Xw^*)^T(y^* - Xw^*) + E[(X\hat{w} - Xw^*)^T(X\hat{w} - Xw^*)]$$
$$= y^{*T}y^* + w^{*T}X^TXw^* - 2y^{*T}Xw^* + E[\hat{w}^TX^TX\hat{w} + w^{*T}X^TXw^* - 2w^{*T}X^TX\hat{w}]$$
$$= \vec{y}^{*T}\vec{y}^* + w^{*T}X^TXw^* - 2y^{*T}Xw^* + E[\hat{w}^TX^TX\hat{w}] + w^{*T}X^TXw^* - 2w^{*T}X^TXw^*$$
$$= \vec{y}^*\vec{y}^* + E[\hat{w}^TX^TX\hat{w}] - 2y^{*T}Xw^* \qquad ②$$

① = ②

(b) $\text{Var}[\hat{w}] = \text{Var}[(X^TX)^{-1}X^T\vec{y}]$

$= \text{Var}[(X^TX)^{-1}X^T(y^* + z)]$

$= \text{Var}[(X^TX)^{-1}X^T z]$

$= (X^TX)^{-1}X^T \Sigma_z X(X^TX)^{-1}$

$= \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1}$

$= \sigma^2(X^TX)^{-1}$

$\hat{w} \sim N(w^*, \sigma^2(X^TX)^{-1})$

$$\Sigma_z = \begin{pmatrix} \sigma^2 & & & \\ & \sigma^2 & & \\ & & \ddots & \\ & & & \sigma^2 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

$\vec{z} \sim N(0, \Sigma_z)$

(c) $\frac{1}{n}E[\|X\hat{w} - Xw^*\|_2^2]$

$= \frac{1}{n}E[(X\hat{w} - Xw^*)^T(X\hat{w} - Xw^*)]$

$= \frac{1}{n}E[\text{trace}((X\hat{w} - Xw^*)(X\hat{w} - Xw^*)^T)]$

$= \frac{1}{n}\text{trace}(\text{Var}(X\hat{w}))$

$= \frac{1}{n}\text{trace}(X\,\text{Var}(\hat{w})\,X^T)$

$= \frac{1}{n}\text{trace}(X\sigma^2(X^TX)^{-1}X^T)$

$= \frac{1}{n}\text{trace}(\sigma^2(X^TX)^{-1}X^TX)$

$= \frac{\sigma^2}{n}\cdot d$

(d) $X = \begin{pmatrix} 1 & x_1 & \cdots & x_1^D \\ 1 & x_2 & & x_2^D \\ \vdots & \vdots & & \vdots \\ 1 & x_n & & x_n^D \end{pmatrix}$ $\qquad \vec{y}^* = \begin{pmatrix} w_1 x_1 + w_0 \\ \vdots \\ w_1 x_n + w_0 \end{pmatrix}$

$$w^* = (X^T X)^{-1} X^T \vec{y}^*$$

$$\text{bias}^2 = \|\vec{y}^* - Xw^*\|_2^2 = (\vec{y}^* - X(X^TX)^{-1}X^T\vec{y}^*)^T(\vec{y}^* - X(X^TX)^{-1}X^T\vec{y}^*)$$
$$= \vec{y}^{*T}\vec{y}^* + \vec{y}^{*T}X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T\vec{y}^* - 2\vec{y}^{*T}X(X^TX)^{-1}X^T\vec{y}^*$$
$$= \vec{y}^{*T}\vec{y}^* - \vec{y}^{*T}X(X^TX)^{-1}X^T\vec{y}^*$$

$$\vec{y}^* \in \mathrm{col}(X)$$

$$w^* = (w_0, w_1, 0, 0, \cdots 0)^T \in \mathbb{R}^{D+1}$$

$$\text{bias}: \|y^* - Xw^*\|_2 = 0$$

$$\frac{1}{n}E[\|y^* - X\hat{w}\|_2^2]$$
$$= \text{bias}^2 + \text{variance} = 0 + \frac{\sigma^2}{n}(D+1) \le \varepsilon$$

$$n \geq \frac{\sigma^2(D+1)}{\varepsilon}$$

$$n \propto D$$

- larger D, need proportionally larger number of numbers

(f) $\frac{1}{n}\cdot$ Variance $= \frac{\sigma^2 d}{n} = \frac{\sigma^2(D+1)}{n}$ increases as D increases

bias $= \|y^* - Xw^*\|_2$

$\frac{1}{n}\cdot$ bias $= |e^{\alpha} - \phi_D(\alpha)|$

$\leq \frac{1}{(D+1)!}\cdot 4^{D+1}$ decreases as D increases

(h) In plot (e), $\varepsilon \propto D$, $\varepsilon \propto \frac{1}{n}$

In plot (f), $\varepsilon \propto \frac{1}{n}$. (at a certain D)

when D is small, $\frac{4^{D+1}}{(D+1)!}$ is dominant, and

ε decreases as D increases

when D is large, $\frac{\sigma^2(D+1)}{n}$ is dominant, and

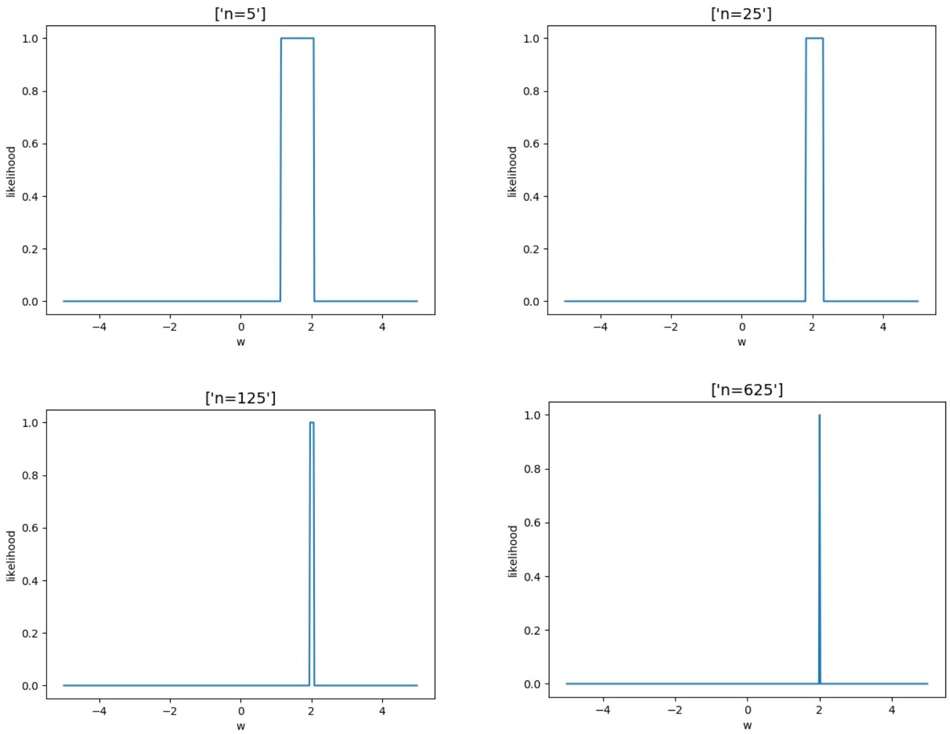
ε increases as D increases

In (e), error only includes variance, so as D gets large, error gets large

In (f), error = bias^2 + variance, and as D gets large, bias decreases, variance increases, so we have a optimal D for the smallest error

2e

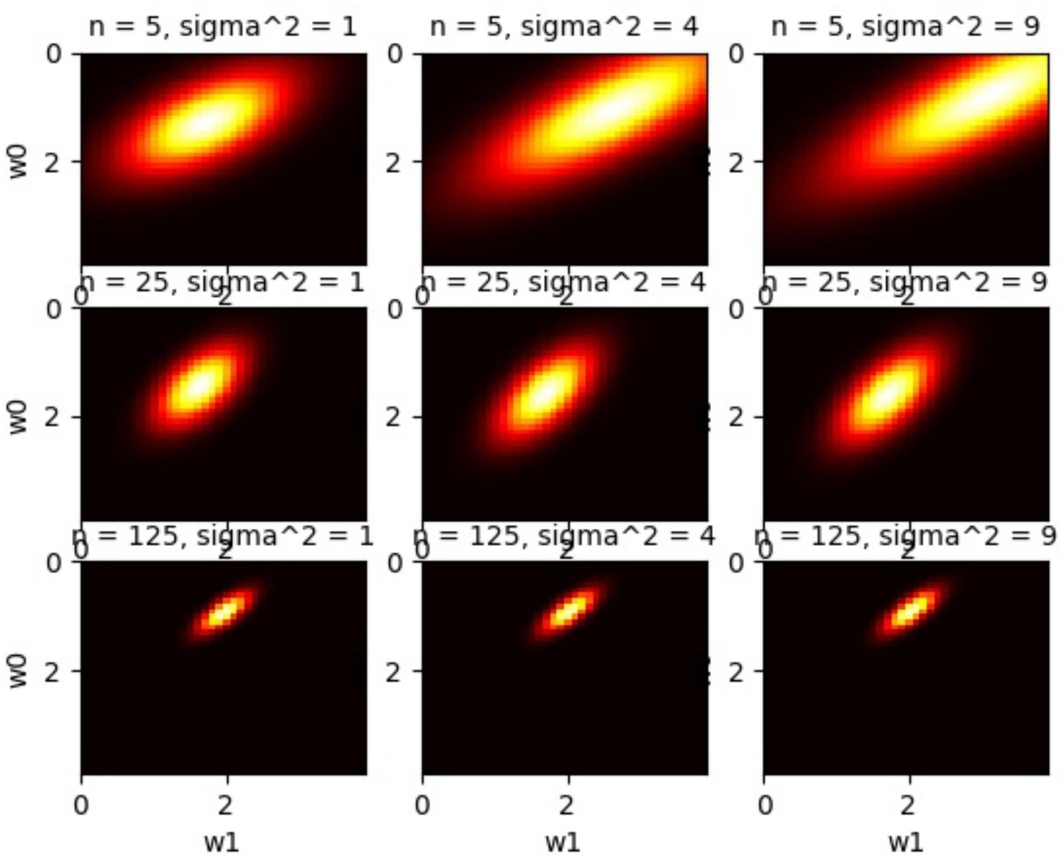
```
import numpy as np
from numpy.random import normal, uniform
import matplotlib.pyplot as plt
from numpy import max, min


def likelihood_function(Y, X, w):
	print('\n')
	print('min_w: {}'.format((max(Y)-0.5)/min(X)))
	print('max_w: {}'.format((min(Y)+0.5)/max(X)))
	print('\n')

	if (Y-X*w < 0.5).all() and (Y-X*w >= -0.5).all():
		return 1
	else:
		return 0

for n in [5,25,125,625]:
	##generate n data points
	true_w = 2
	X = uniform(0.2, 0.3, size = n)
	Z = uniform(-0.5, 0.5, size = n)
	Y = X * true_w + Z

	####calculate likelihood as function of w
	W = np.arange(-5, 5, 0.02)
	N = W.shape[0]
	likelihood = np.zeros(N)
	for j in range(N):
		likelihood[j] = likelihood_function(Y, X, W[j])

	plt.plot(W, likelihood)
	plt.xlabel('w', fontsize=10)
	plt.ylabel('likelihood', fontsize=10)
	plt.title(['n=' + str(n)], fontsize=14)
	plt.savefig('{}.jpg'.format(n))
	plt.show()
```


['n=5']
['n=25']
['n=125']
['n=625']
likelihood
w
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

2i

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import normal, uniform
from numpy.linalg import inv
from scipy.stats import multivariate_normal
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

def likelihood_function(X, Y, var, w):
  d = X.shape[1]
  mean = (inv(1.0/var * np.eye(d) + X.T @ X) @ X.T @ Y).ravel()
  covar = inv(1.0/var * np.eye(d) + X.T @ X)
  return multivariate_normal.pdf(w, mean=mean, cov=covar)

def mean_function(X, Y, var):
  d = X.shape[1]
  return (inv(1.0/var * np.eye(d) + X.T @ X) @ X.T @ Y).ravel()

fig , ax = plt.subplots(3, 3)

for axi, n in enumerate([5, 25, 125]):
  # generate data
  w_true = np.array([[1], [2]])
  X = uniform(size = 2*n).reshape((n, 2))
  Z = normal(size = n).reshape((n, 1))
  Y = X @ w_true + Z

  for axj, var in enumerate([1, 4, 9]):
    print(mean_function(X, Y, var))
    # compute likelihood
    W0 = np.arange(0, 4, 0.1)
    W1 = np.arange(0, 4, 0.1)
    N = W0.shape[0]
    likelihood = np.ones([N,N]) # likelihood as a function of w_1 and w_0
    for i in range(N):
      for j in range(N):
        w = np.array([W0[i], W1[j]])
        likelihood[i, j] = likelihood_function(X, Y, var, w)
```

```
    # plotting the likelihood

    # for 2D likelihood using imshow
    ax[axi, axj].imshow(likelihood, cmap='hot', aspect='auto',extent=[W1.min(), W1.max(),
W0.max(), W0.min()])
    ax[axi, axj].set_xlabel('w1')
    ax[axi, axj].set_ylabel('w0')
    ax[axi, axj].set_title('n = {}, sigma^2 = {}'.format(n, var), fontsize = 10)

plt.savefig('2i.jpg')
plt.tight_layout()
plt.show()
```

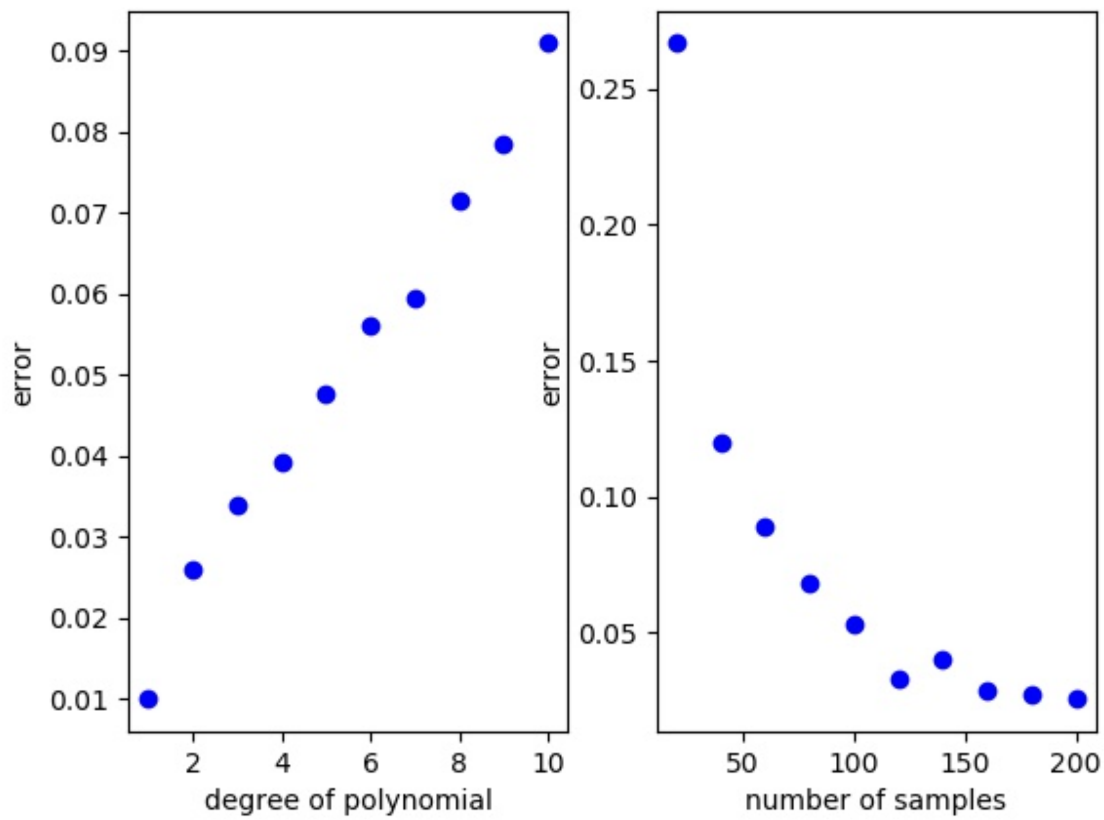


4ef-code

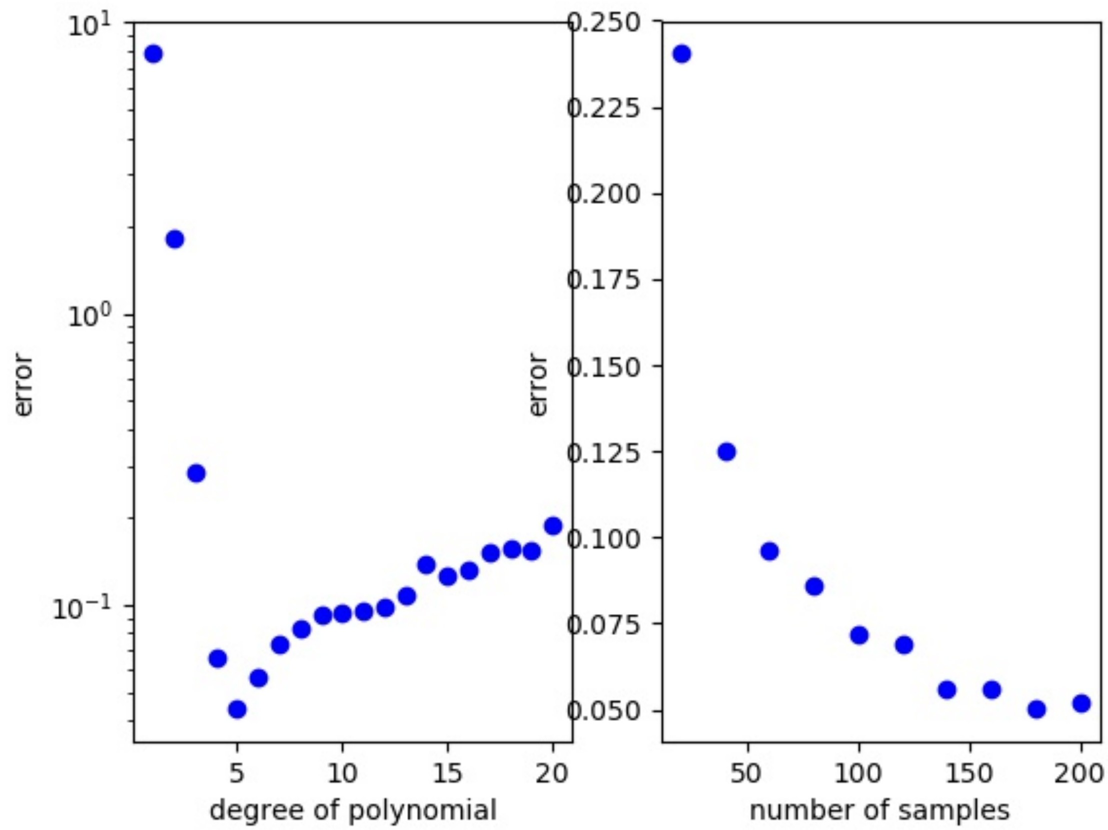
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import uniform, normal
from numpy.linalg import inv, norm

# assign problem parameters
w1 = 1
w0 = 1
#interval = [-1, 1]
interval = [-4, 3]

# generate data
# np.random might be useful
def error_function(D, n, func = 'p', repeat = 40):
    error = np.zeros(repeat)
    for j in range(repeat):
        alpha = uniform(interval[0], interval[1], n).reshape((n, 1))
        noise = normal(size = n).reshape((n, 1))
        print(alpha)
        print(noise)
        X = np.ones((n, 1))
        for i in range(1, D+1):
            X = np.hstack((X, alpha**i))
        print(X)

        if func == 'p':
            y_true = w1 * alpha + w0
        if func == 'exp':
            y_true = np.exp(alpha)

        print(y_true)

        y_noise = y_true + noise

        w_hat = inv(X.T @ X) @ X.T @ y_noise

        error[j] = norm(X @ w_hat - y_true)**2/n
    return np.mean(error)


# fit data with different models
# np.polyfit and np.polyval might be useful
```

```
# plotting figures
# sample code
plt.figure()
plt.subplot(121)

deg = 20
error = np.zeros(deg)
n = 120
for i in range(deg):
    error[i] = error_function(i+1, n, 'exp')

plt.semilogy(np.arange(1, deg+1), error, 'o', color = 'b')
plt.xlabel('degree of polynomial')
plt.ylabel('error')

plt.subplot(122)
error = np.zeros(10)

D = 4
for i in range(10):
    error[i] = error_function(D, (i+1)*20, 'exp')
plt.plot(np.arange(20, 220, 20), error, 'o', color = 'b')

plt.xlabel('number of samples')
plt.ylabel('error')
plt.savefig('4f.jpg')
plt.show()
```

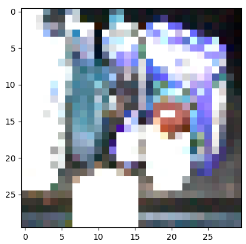
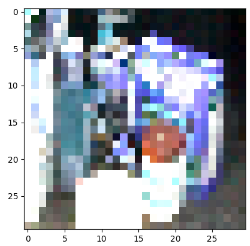
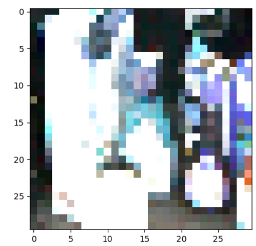
4e-figure


0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
error
2
4
6
8
10
degree of polynomial
0.25
0.20
0.15
0.10
0.05
error
50
100
150
200
number of samples

4f-figure



5a

the control vectors of image 0 is [0. -1. 0.]
the control vectors of image 10 is [-1.   -0.45111084 -1.   ]
the control vectors of image 20 is [0.   0.   0.37368774]

5b

we cannot do inversion with a 2700*2700 singular matrix since the rank of X.T @ X is at most n, which is 91 in this case

5c

below is the output **without standardization:**
the **training errors** for lambda = [0.1, 1, 10, 100, 1000] are [8.79622646e+10
4.04115399e+11   7.31977324e+09   9.62356666e+09
  2.15645320e+10]

5d

below is the output **with standardization**:
the training errors for lambda = [0.1, 1, 10, 100, 1000] are [3.25574737e-07
2.91051229e-05   1.59038146e-03   3.47731220e-02
  2.54402961e-01]

5e

below is the output **without standardization:**
the **validation errors** for lambda = [0.1, 1, 10, 100, 1000] are [9.12815657e+10
2.78337869e+11   6.67654211e+09   5.21116488e+09
  3.70661677e+10]

below is the output **with standardization**:
the **validation errors** for lambda = [0.1, 1, 10, 100, 1000] are [0.86807707
0.86210293 0.82750762 0.72465309 0.7250142]

as lambda increase, bias increases(training error reflects bias), and variance decreases(validation error reflects bias+variance)

5f

the condition number **without standardization** is 281997.87107584067
the condition number **with standardization** is 444.7259317111113

5-code

```
import pickle
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv, norm, svd

class HW3_Sol(object):

    def __init__(self):
        pass

    def load_data(self):
        self.x_train = pickle.load(open('x_train.p','rb'), encoding='latin1')
        self.y_train = pickle.load(open('y_train.p','rb'), encoding='latin1')
        self.x_test = pickle.load(open('x_test.p','rb'), encoding='latin1')
        self.y_test = pickle.load(open('y_test.p','rb'), encoding='latin1')

    def compose_x(self, x_raw):
        n = x_raw.shape[0]
        d = x_raw.shape[1]*x_raw.shape[2]*x_raw.shape[3]
        return x_raw.reshape((n, d))

    def OLS(self, X, U):
        return inv(X.T @ X) @ X.T @ U

    def ridge(self, X, U, lambd):
        d = X.shape[1]
        return inv(X.T @ X + lambd * np.eye(d)) @ X.T @ U

    def error(self, X, U, Pi):
        n = U.shape[0]
        f_norm = norm(X @ Pi - U, ord = 'fro')
        return 1.0/n * f_norm**2
```

```
    def standardize(self, X):
        return X/255.0 * 2 - 1

    def kappa(self, X, lambd):
        d = X.shape[1]
        A = X.T @ X + lambd * np.eye(d)
        s = svd(A, compute_uv = False)
        return s[0]/s[-1]

    def visualize(self, i):
        plt.imshow(self.x_train[i])
        plt.savefig('training_image_{}'.format(i))
        plt.show()

if __name__ == '__main__':

    hw3_sol = HW3_Sol()

    hw3_sol.load_data()

    ##############(a)###############
    for i in [0, 10, 20]:
        hw3_sol.visualize(i)
        print('the control vectors of image {} is {}'.format(i, hw3_sol.y_train[i]))


    ###############(b)##############
    X = hw3_sol.compose_x(hw3_sol.x_train)
    U = hw3_sol.y_train
    X_val = hw3_sol.compose_x(hw3_sol.x_test)
    U_val = hw3_sol.y_test
    #Pi = hw3_sol.OLS(X, U)

    print('we cannot do inversion with a 2700*2700 singular matrix since the rank
of X.T @ X is at most n, which is 91 in this case')
    ###############(c+e)##############
    print('below is the output without standardization:')
```

```
    error = np.zeros(5)
    error_validation = np.zeros(5)
    for i, lambd in enumerate([0.1, 1, 10, 100, 1000]):
        Pi = hw3_sol.ridge(X, U, lambd)
        error[i] = hw3_sol.error(X, U, Pi)
        error_validation[i] = hw3_sol.error(X_val, U_val, Pi)


    print('the training errors for lambda = {} are {}'.format([0.1, 1, 10, 100, 1000],
error))
    print('the validation errors for lambda = {} are {}'.format([0.1, 1, 10, 100, 1000],
error_validation))

    ###############(d+e)###############
    print('below is the output with standardization:')
    X = hw3_sol.standardize(X)
    X_val = hw3_sol.standardize(X_val)

    error = np.zeros(5)
    error_validation = np.zeros(5)
    for i, lambd in enumerate([0.1, 1, 10, 100, 1000]):
        Pi = hw3_sol.ridge(X, U, lambd)
        error[i] = hw3_sol.error(X, U, Pi)
        error_validation[i] = hw3_sol.error(X_val, U_val, Pi)


    print('the training errors for lambda = {} are {}'.format([0.1, 1, 10, 100, 1000],
error))
    print('the validation errors for lambda = {} are {}'.format([0.1, 1, 10, 100, 1000],
error_validation))

    print('as lambda increase, bias increases(training error reflects bias), and
variance decreases(validation error reflects bias+variance)')

    ###############(f)###############
    X = hw3_sol.compose_x(hw3_sol.x_train)
```

```
    print('the condition number without standardization is
{}'.format(hw3_sol.kappa(X, 100)))

    X = hw3_sol.standardize(X)
    print('the condition number with standardization is {}'.format(hw3_sol.kappa(X,
100)))
```