CS 189 Introduction to Machine Learning
Spring 2018

HW3

Your self-grade URL is http://eecs189.org/self_grade?question_ids=1_1,1_2,2_1,2_2,2_3,2_4,2_5,2_6,2_7,2_8,2_9,3_1,3_2,3_3,3_4,3_5,3_6,3_7,3_8,3_9,3_10,4_1,4_2,4_3,4_4,4_5,4_6,4_7,4_8,5_1,5_2,5_3,5_4,5_5,5_6,6.

This homework is due **Friday, Feb 10th at 10pm.**

2 Probabilistic Model of Linear Regression

Both ordinary least squares and ridge regression have interpretations from a probabilistic standpoint. In particular, assuming a generative model for our data and a particular noise distribution, we will derive least squares and ridge regression as the maximum likelihood and maximum *a-posteriori* parameter estimates, respectively. This problem will walk you through a few steps to do that. (Along with some side digressions to make sure you get a better intuition for ML and MAP estimation.)

(a) Assume that X and Y are both one-dimensional random variables, i.e. $X, Y \in \mathbb{R}$. Assume an affine model between X and Y: $Y = Xw_1 + w_0 + Z$, where $w_1, w_0 \in \mathbb{R}$, and $Z \sim N(0,1)$ is a standard normal (Gaussian) random variable. Assume w_1, w_0 are fixed parameters (i.e., they are not random). **What is the conditional distribution of Y given X?**

Solution: When we condition on the event $X = x$, the expression $Xw_1 + w_0$ becomes $xw_1 + w_0$, so

$$Y|(X = x) \sim xw_1 + w_0 + Z.$$

Now $xw_1 + w_0 + Z$ is a constant plus a standard normal, so

$$xw_1 + w_0 + Z \sim N(xw_1 + w_0, 1).$$

The conditional density becomes:

$$p(Y|X = x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2}(Y - (xw_1 + w_0))^2\}. \tag{1}$$

(b) Given n points of training data $\{(X_1,Y_1),(X_2,Y_2),\dots,(X_n,Y_n)\}$ generated in an iid fashion by the probabilistic setting in the previous part, **derive the maximum likelihood estimator for w_1, w_0 from this training data.**

Solution: The log likelihood function is given by

$$\begin{aligned}\sum_{i=1}^{n} \log p(Y_i|X=X_i) &= \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2}(Y-(X_i w_1 + w_0))^2) \\ &= -\frac{1}{2}\sum_{i=1}^{n}(Y_i - (X_i w_1 + w_0))^2 + \text{Constant}\end{aligned} \tag{2}$$

Differentiating the log likelihood with respect to w_1 and w_0, we obtain:

$$\frac{\partial}{\partial w_1}\sum_i \log p(Y_i|X=X_i) = \sum_{i=1}^{n} X_i(X_i w_1 + w_0 - Y_i)$$
$$\frac{\partial}{\partial w_0}\sum_i \log p(Y_i|X=X_i) = \sum_{i=1}^{n} (X_i w_1 + w_0 - Y_i)$$

Setting both of the partial derivatives to zero, we immediately get two equations with two unknowns. The terms that only have Y_i and $Y_i X_i$ are pulled to the other side of the equality:

$$w_1 \sum_{i=1}^{n} X_i^2 + w_0 \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} X_i Y_i$$
$$w_1 \sum_{i=1}^{n} X_i + w_0 n = \sum_{i=1}^{n} Y_i$$

Then, these can be solved directly:

$$w_1 = \frac{n\sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2}$$
$$w_0 = \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} X_i Y_i}{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2}.$$

To observe the significance of these terms, we now introduce a few notations. Define the sample mean, sample variance (unadjusted) and sample cross-covariance as follows:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i,$$
$$\hat{s}_X^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - \bar{X}^2$$
$$\hat{s}_{XY} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) = \frac{1}{n}\sum_{i=1}^{n} X_i Y_i - \bar{X}\bar{Y},$$

Then, we have

$$w_0 = \bar{Y} - \bar{X} w_1, \text{and}$$
$$w_1 = \frac{\hat{s}_{XY}}{\hat{s}_X^2}.$$

(c) Now, consider a different generative model. Let $Y = Xw + Z$, where $Z \sim U[-0.5, 0.5]$ is a continuous random variable uniformly distributed between -0.5 and 0.5. Again assume that w is a fixed parameter. **What is the conditional distribution of Y given X?**

Solution: As before, w is fixed, so we have

$$\begin{aligned}
P(Y = y|X = x) &= P(z = y - Xw|X = x) \\
&= P(z = y - xw) \\
&= \begin{cases} 1 \text{ if } -0.5 < y - xw < 0.5 \\ 0 \text{ otherwise} \end{cases} \\
&= \begin{cases} 1 \text{ if } -0.5 + xw < y < 0.5 + xw \\ 0 \text{ otherwise} \end{cases}
\end{aligned} \tag{3}$$

(d) Given n points of training data $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ generated in an i.i.d. fashion in the setting of the part (c) **derive a maximum likelihood estimator of w.** Assume that $X_i > 0$ for all $i = 1, \ldots, n$. (Note that MLE for this case need not be unique; but you are required to report only one particular estimate.)

Solution: Noting that $X_i > 0$, we find that the likelihood function is given by

$$\begin{aligned}
\Pi_i p(Y_i|X_i) &= \Pi_{i=1}^n \mathbf{1}\{-0.5 + X_i w < Y_i < 0.5 + X_i w\} \\
&= \Pi_{i=1}^n \mathbf{1}\left\{\frac{Y_i - 0.5}{X_i} < w < \frac{Y_i + 0.5}{X_i}\right\}.
\end{aligned}$$

For this case, maximizing likelihood is equivalent to ensuring that all the indicators have value 1, which in turn requires that all those regions intersect:

$$\max_i \left\{\frac{Y_i - 0.5}{X_i}\right\} < w < \left\{\min_i \frac{Y_i + 0.5}{X_i}\right\}.$$

So any value of w satisfying these constraints is a valid MLE. E.g., $w = \max_i \left\{\frac{Y_i - 0.4}{X_i}\right\}$ is a valid MLE.

(e) Take the model $Y = Xw + Z$, where $Z \sim U[-0.5, 0.5]$. **Use a computer to simulate n training samples $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ and illustrate what the likelihood of the data looks like as a function of w after $n = 5, 25, 125, 625$ training samples. Qualitatively describe what is happening as n gets large.**

(You may use the starter code. Note that you have considerable design freedom in this problem part. You get to choose how you draw the X_i as well as what true value w you want to illustrate. You have total freedom in using additional python libraries for this problem part. No restrictions.)

Solution: We generated samples assuming $Y = 3X + Z$. We assumed that the model parameter is between 0 and 4 (you can assume any range that includes the true parameters.)

and we calculated the likelihood by using the formula we obtained above for different values of $w$in this interval. To do so, for a specific w we compute the likelihood $\Pi_{i=1}^{n} Pr(Y_i|X_i;w)$ which is either zero or 1 because the probability distribution is uniform. In the figure below, likelihood of the data can be seen as a function of w for different sizes of the data. Notice the y-axis values in the different plots. We can observe from the plots that increasing the number of samples leads the posterior to concentrate around the true value of $w = 3$ in this case.

The code here uses a simple approach of just giving each training sample a veto into whether or not a particular w is feasible or not.

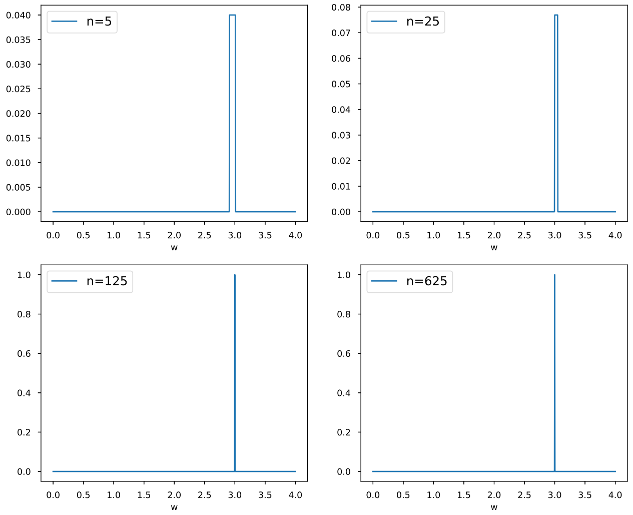


Figure 1: Plot of the likelihood of data as a function of w for different sample size

```
import numpy as np
import matplotlib.pyplot as plt

sample_size = [5,25,125,625]
plt.figure(figsize=[12, 10])
for k in range(4):
    n = sample_size[k]
    X = 2+2*np.random.random(n)  # generating n X with U[2, 4]
    Z = np.random.uniform(-0.5,0.5,n) # generating i.i.d random uniform noise
    Y = 3*X+Z # computing y from x and z assuming w=3
    N = 1001
    W = np.linspace(0,4,N)
    likelihood = np.ones(N) # likelihood as a function of w
    for i1 in range(N):
        w = W[i1]
        for i in range(n):
            # Yi has uniform distribution in [wXi-0.5,wXi+0.5]
            if abs(Y[i]-w*X[i]) > 0.5:
                likelihood[i1]=0
    likelihood /= sum(likelihood)
    plt.subplot(2, 2, k+1)
    plt.plot(W, likelihood)
    plt.xlabel('w', fontsize=10)
    plt.legend(['n=' + str(n)], fontsize=14)
# plt.savefig('uni_noise_post.pdf') # command to save figure
plt.show()
```

(f) (One-dimensional Ridge Regression) Now, let us return to the case of Gaussian noise. Given n points of training data $\{(X_1,Y_1),(X_2,Y_2),\cdots,(X_n,Y_n)\}$ generated according to $Y_i = X_iW + Z_i$, where $Z_i \sim N(0,1)$ are iid standard normal random variables. Assume $W \sim N(0,\sigma^2)$ is also a normal random variable and is independent of both the Z_i's and the X_i's. **Use Bayes' Theorem to derive the posterior distribution of W given the training data. What is the mean of the posterior distribution of W given the data?**

Hint: Compute the posterior up-to proportionality and try to identify the distribution by completing the square.

Solution: By Bayes' Theorem, we have

$$\begin{aligned}
P(w|X_1,Y_1,\cdots,X_n,Y_n) &\propto \Pi_{i=1}^n P(Y_i|X_i,w)\cdot P(w) \\
&\propto \Pi_{i=1}^n \exp\left\{-\frac{1}{2}(Y_i - wX_i)^2\right\}\exp\left\{-0.5\frac{w^2}{\sigma^2}\right\} \\
&\propto \Pi_{i=1}^n \exp\left\{-\frac{1}{2}(Y_i^2 - 2Y_iwX_i + w^2X_i^2)\right\}\exp\left\{-0.5\frac{w^2}{\sigma^2}\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(\sum_{i=1}^n Y_i^2 - 2\sum_{i=1}^n Y_iwX_i + w^2\sum_{i=1}^n X_i^2\right)\right\}\exp\left\{-0.5\frac{w^2}{\sigma^2}\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\left(\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}\right)w^2 - (2\sum_{i=1}^n X_iY_i)w\right] + \text{Constant}\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}\right)\left(w^2 - \frac{2\sum_{i=1}^n X_iY_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}w\right) + \text{Constant}\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}\right)\left(w - \frac{\sum_{i=1}^n X_iY_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}\right)^2 + \text{Constant}\right\}
\end{aligned}$$

To get to the last line, we used a trick of completing the square. Consider the term we would need to add to

$$\left(w^2 - \frac{2\sum_{i=1}^n X_iY_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}w\right)$$

to get

$$\left(w - \frac{\sum_{i=1}^n X_iY_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}\right)^2.$$

The term we would add does not depend on w, only the sample points. From the perspective of the distribution on w, this is just a constant, so we folded it into the constant term.

The final step is to notice that this posterior of w given $X_1, Y_1, \cdots, X_n, Y_n$ is a normal distribution. To see the connection, ignore the constant term in the posterior

$$P(w|X_1, Y_1, \cdots, X_n, Y_n) \propto \exp\left\{-0.5\left(\sum_{i=1}^{n} X_i^2 + \frac{1}{\sigma^2}\right)\left(w - \frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}\right)^2\right\}.$$

Recall the density for the normal distribution $\mathcal{N}(\mu, \sigma^2)$ is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \propto \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}.$$

Comparing the two we conclude that

$$W|\{X_i, Y_i, i = 1, \ldots, n\} \sim \mathcal{N}\left(\frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}, \frac{1}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}\right).$$

Note that the mean of the posterior is

$$\frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n X_i^2 + \frac{1}{\sigma^2}}. \tag{4}$$

Since the posterior is Gaussian, we also conclude that the mode of the distribution is the mean and hence the mean is also the maximum *a posteriori* estimate (MAP) of W.

(g) Consider n training data points $\{(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2), \cdots, (\mathbf{x}_n, Y_n)\}$ generated according to $Y_i = \mathbf{w}^\top \mathbf{x}_i + Z_i$ where $Y_i \in \mathbb{R}, \mathbf{w}, \mathbf{x}_i \in \mathbb{R}^d$ with $\mathbf{w}$ fixed, and $Z_i \sim N(0,1)$ iid standard normal random variables. **Argue why the maximum likelihood estimator for w is the solution to a least squares problem.**

Solution: Since the logarithm is a monotonically increasing function, we can just look at the log likelihood function.

$$\begin{aligned} \sum_i \log p(Y_i|\mathbf{x}_i) &= -\frac{1}{2}\sum_{i=1}^{n}(Y_i - (\mathbf{x}_i^\top \mathbf{w}))^2 + \text{Constant} \\ &= -\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{Y}\|_2^2 + \text{Constant}, \end{aligned}$$

with $\mathbf{X} \in \mathbb{R}^{n \times d}$ whose i^{th} row is $\mathbf{x}_i^\top$ and $\mathbf{Y} \in \mathbb{R}^n$ whose i^{th} entry is Y_i.

At this point, we are done. We have shown that maximizing the maximum-likelihood objective is the same as maximizing a negative squared error objective, which is the same as minimizing the squared error objective, the same as ordinary least squares.

However, if we wanted to, we could just keep solving this.

Differentiate the log likelihood with respect to $\mathbf{w}$ and setting the gradient to zero, we get

$$\nabla_{\mathbf{w}} \sum_i \log p(Y_i|\mathbf{x}_i) = \mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{Y}).$$

Setting the gradient to zero, we get

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}.$$

which is the same solution as the least squares problem.

(h) (Multi-dimensional ridge regression) Consider the setup of the previous part: $Y_i = \mathbf{W}^\top\mathbf{x}_i + Z_i$, where $Y_i \in \mathbb{R}, \mathbf{W}, \mathbf{x}_i \in \mathbb{R}^d$, and $Z_i \sim N(0,1)$ iid standard normal random variables. Now we treat $\mathbf{W}$ as a random vector and assume a prior knowledge about its distribution. In particular, we use the prior information that the random variables W_j are i.i.d. $\sim N(0,\sigma^2)$ for $j = 1,2,\ldots,d$. **Derive the posterior distribution of W given all the $\mathbf{x}_i, Y_i$ pairs. What is the mean of the posterior distribution of the random vector W?**

Hint: Use hints from part (f) and the following identities: For $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ and $\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}$ we have $\mathbf{X}^\top\mathbf{X} = \sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top$ and $\mathbf{X}^T\mathbf{Y} = \sum_{i=1}^n \mathbf{x}_i Y_i$.

Solution: Note that, we have the following prior distribution on $\mathbf{W}$:

$$\begin{aligned} P(\mathbf{w}) = \Pi_{j=1}^d P(w_j) \\ &= \Pi_{j=1}^d \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{w_j^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\mathbf{w}^2}{2\sigma^2}\right) \\ &\propto \exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\}. \end{aligned}$$

Furthermore, to compute the likelihood of the data, we observe that $\mathbf{Y}|\mathbf{X},\mathbf{w} \sim N(\mathbf{X}^\top\mathbf{w}, 1)$. Concretely, we have

$$p(\mathbf{Y}|\mathbf{X},\mathbf{w}) \propto \exp\left\{-\frac{1}{2}\|\mathbf{Y} - (\mathbf{X}^\top\mathbf{w})\|^2\right\}. \tag{5}$$

Now by Bayes' Theorem, we have

$$\begin{aligned}
P(\mathbf{w}|\mathbf{x}_1,Y_1,\cdots,\mathbf{x}_n,Y_n) &\propto \Pi_{i=1}^{n} P(Y_i|\mathbf{x}_i,\mathbf{w})\ \Pi_{j=1}^{d} P(w_j) \\
&\propto \Pi_{i=1}^{n} \exp\left\{-0.5\left(Y_i-\mathbf{x}_i^\top\mathbf{w}\right)^2\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \Pi_{i=1}^{n} \exp\left\{-0.5\left(Y_i^2-2Y_i\mathbf{x}_i^\top\mathbf{w}+(\mathbf{x}_i^\top\mathbf{w})^2\right)\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \exp\left\{-0.5\left(\sum_{i=1}^{n}Y_i^2-2\sum_{i=1}^{n}Y_i\mathbf{x}_i^\top\mathbf{w}+\sum_{i=1}^{n}(\mathbf{x}_i^\top\mathbf{w})^2\right)\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \exp\left\{-0.5\left(\sum_{i=1}^{n}Y_i^2-2\sum_{i=1}^{n}Y_i\mathbf{x}_i^\top\mathbf{w}+\sum_{i=1}^{n}w^\top\mathbf{x}_i\mathbf{x}_i^\top\mathbf{w}\right)\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \exp\left\{-0.5\left(\sum_{i=1}^{n}Y_i^2-2\sum_{i=1}^{n}Y_i\mathbf{x}_i^\top\mathbf{w}+\mathbf{w}^\top\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top\mathbf{w}\right)\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \exp\left\{-0.5\left(\sum_{i=1}^{n}Y_i^2-2\sum_{i=1}^{n}(\mathbf{x}_iY_i)^\top\mathbf{w}+\mathbf{w}^\top\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top\mathbf{w}\right)\right\}\exp\left\{-\frac{\|\mathbf{w}\|_2^2}{2\sigma^2}\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\mathbf{w}^\top(\mathbf{X}^\top\mathbf{X}+\frac{1}{\sigma^2}I_d)\mathbf{w}-2(\mathbf{X}^\top\mathbf{Y})^\top\mathbf{w}\right]\right\},
\end{aligned} \tag{6}$$

with $\mathbf{X}\in\mathbb{R}^{n\times d}$ whose i^{th} row is $\mathbf{x}_i^\top$ and $\mathbf{Y}\in\mathbb{R}^n$ whose i^{th} entry is Y_i. The last line uses the fact that

$$\mathbf{X}^\top\mathbf{X}=\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top$$

and

$$\mathbf{X}^\top\mathbf{Y}=\sum_{i=1}^{n}\mathbf{x}_iY_i.$$

Now, as we did earlier, we complete the square, except in the matrix case. To make the notation easier, let's define the matrix $\mathbf{M}$ as

$$\mathbf{M}=(\mathbf{X}^\top\mathbf{X}+\frac{1}{\sigma^2}I_d).$$

Then, noticing that $\mathbf{M}$ is symmetric, we see that:

$$\begin{aligned}
P(\mathbf{w}|\mathbf{x}_1,Y_1,\cdots,\mathbf{x}_n,Y_n) &\propto \exp\{-\frac{1}{2}[\mathbf{w}^\top\mathbf{M}\mathbf{w}-2(\mathbf{X}^\top\mathbf{Y})^\top\mathbf{w}]\} \\
&\propto \exp\{-\frac{1}{2}[(\mathbf{w}-\mathbf{M}^{-1}(\mathbf{X}^\top\mathbf{Y}))^\top\mathbf{M}(\mathbf{w}-\mathbf{M}^{-1}(\mathbf{X}^\top\mathbf{Y}))]+\text{Constant}\}
\end{aligned} \tag{7}$$

How did we know this was how we should complete the square? The $\mathbf{w}$ part on the sides is clear as is the central $\mathbf{M}$ since we knew we wanted the $\mathbf{w}^\top\mathbf{M}\mathbf{w}$ term. Now, because we knew that we wanted to get an $(\mathbf{X}^\top \mathbf{Y})^\top \mathbf{w}$ term as the linear term in the square, we had to cancel the $\mathbf{M}$ in the middle with an $\mathbf{M}^{-1}$. Of course, as argued earlier, the constant here absorbs the compensation for the term that we added while completing the square.

Therefore, the posterior of w given $\mathbf{x}_1, Y_1, \cdots, \mathbf{x}_n, Y_n$ is a multivariate Gaussian. The mean of this multivariate Gaussian is

$$(\mathbf{X}^\top \mathbf{X} + \frac{1}{\sigma^2} I_d)^{-1} \mathbf{X}^\top \mathbf{Y}. \tag{8}$$

(i) Consider $d = 2$ and the setting of the previous part. **Use a computer to simulate and illustrate what the *a-posteriori* probability looks like for the W model parameter space after $n = 5, 25, 125$ training samples for different values of σ^2.**

(Again, you may use the starter code. And like problem (e), there are no restrictions for using additional python libraries for this part as well.)

Solution: First, we assumed a variance for the random variable W and we generate d numbers from $N(0, \sigma^2)$. Then, we generated the samples using these parameters and computed a value proportional to the probability using Equation 6 from above. We could have normalized to get the exact posterior probability; our results would have been equivalent as far the plots would go.

In the figures below, the *a-posteriori* probability of the model parameters given data can be seen in the model space $\mathbf{W} = (W_1, W_2)$. As we can see in the plots, increasing the number of samples results in shrinking the zone of likely parameters.

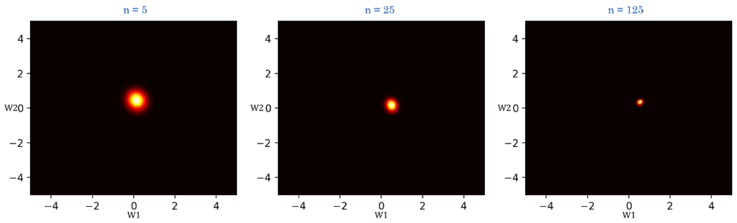


Figure 2: Heatplots of the *a-posteriori* probability of the model given the training data, plotted in the model space (W_1, W_2) for different sample sizes with $\sigma = 0.5$

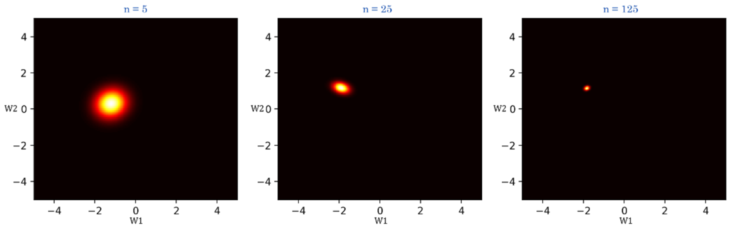


Figure 3: Heatplots of the *a-posteriori* probability of the model given the training data, plotted in the model space (W_1, W_2) for different sample sizes with $\sigma = 1.5$

```
import numpy as np
import matplotlib.pyplot as plt
Samples=[5,25,125]
```

```
Sigma = [0.5**2,1.5**2] #plotting for two different variance
w1_s = [0.4,1.3] # true W1 drawn from the gaussian with s.d 0.5 and 1.5
   ↪ respectively
w2_s = [0.5,-1.7] # true W2 drawn from the gaussian with s.d 0.5 and 1.5
   ↪ respectively
for s in range(2):
    sigma=Sigma[s]
    A=w1_s[s]
    B=w2_s[s]
    plt.figure()
    for count in range(3):
        n = Samples[count]
        # X = (X1,X2) Y = AX1+AX2+Z
        X1 = np.random.normal(0,1, n) # generating random samples for X1
   ↪ from normal dist
        X2 = np.random.normal(0,1, n)
        Z = np.random.normal(0,1, n)
        Y = A*X1 +B*X2 +Z
        N = 201
        W = np.linspace(-5,5,N)
        prob = np.ones([N,N])
        for i1 in range(N):
            w1 = W[i1] #taking different values for w1 from -5 to 5 to
   ↪ compute something proportional to the posteriori probability
            for i2 in range(N):
                w2 = W[i2]
                L=1
                for i in range(n): # this part can vectorized as well
                    L = L*np.exp(-0.5*(Y[i]-X1[i]*w1-X2[i]*w2)**2)
                L = L*np.exp(-0.5*(w1**2+w2**2)/sigma)
                prob[i1][i2]=L
        plt.subplot(1,3,count+1)
        plt.imshow(np.flipud(prob), cmap='hot', aspect='auto',extent
   ↪ =[-5,5,-5,5])
        plt.show()
```

3 Simple Bias-Variance Tradeoff

Consider a random variable X, which has unknown mean μ and unknown variance σ^2. Given n iid realizations of training samples $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ from the random variable, we wish to estimate the mean of X. We will call our estimate of X the random variable $\hat{X}$, which has mean $\hat{\mu}$. There are a few ways we can estimate μ given the realizations of the n samples:

1. Average the n samples: $\frac{x_1+x_2+\ldots+x_n}{n}$.

2. Average the n samples and one sample of 0: $\frac{x_1+x_2+\ldots+x_n}{n+1}$.

3. Average the n samples and n_0 samples of 0: $\frac{x_1+x_2+\ldots+x_n}{n+n_0}$.

4. Ignore the samples: just return 0.

In the parts of this question, we will measure the *bias* and *variance* of each of our estimators. The *bias* is defined as

$$E[\hat{X}-\mu]$$

and the *variance* is defined as

$$\mathrm{Var}[\hat{X}].$$

(a) **What is the bias of each of the four estimators above?**

Solution: Using the linearity of expectation, we write $E[\hat{X}-X]$ as $E[\hat{X}]-E[X]=E[\hat{X}]-\mu$, so we have the following biases:

(a) $E[\hat{X}]=E[\frac{X_1+X_2+...+X_n}{n}]=\frac{n\mu}{n} \implies \text{bias}=0$

(b) $E[\hat{X}]=E[\frac{X_1+X_2+...+X_n}{n+1}]=\frac{n\mu}{n+1} \implies \text{bias}=-\frac{1}{n+1}\mu$

(c) $E[\hat{X}]=E[\frac{X_1+X_2+...+X_n}{n+n_0}]=\frac{n\mu}{n+n_0} \implies \text{bias}=-\frac{n_0}{n+n_0}\mu$

(d) $E[\hat{X}]=0 \implies \text{bias}=-\mu$

(b) **What is the variance of each of the four estimators above?**

Solution: The two key identities to remember are $\mathrm{Var}[A+B]=\mathrm{Var}[A]+\mathrm{Var}[B]$ (when A and B are independent) and $\mathrm{Var}[kA]=k^2\mathrm{Var}[A]$, where A and B are random variables and k is a constant.

(a) $\mathrm{Var}[\hat{X}]=\mathrm{Var}[\frac{X_1+X_2+...+X_n}{n}]=\frac{1}{n^2}\mathrm{Var}[X_1+X_2+\ldots+X_n]=\frac{1}{n^2}(n\sigma^2)=\frac{\sigma^2}{n}$

(b) $\mathrm{Var}[\hat{X}]=\mathrm{Var}[\frac{X_1+X_2+...+X_n}{n+1}]=\frac{1}{(n+1)^2}\mathrm{Var}[X_1+X_2+\ldots+X_n]=\frac{1}{(n+1)^2}(n\sigma^2)=\frac{n}{(n+1)^2}\sigma^2$

(c) $\mathrm{Var}[\hat{X}]=\mathrm{Var}[\frac{X_1+X_2+...+X_n}{n+n_0}]=\frac{1}{(n+n_0)^2}\mathrm{Var}[X_1+X_2+\ldots+X_n]=\frac{1}{(n+n_0)^2}(n\sigma^2)=\frac{n}{(n+n_0)^2}\sigma^2$

(d) $\mathrm{Var}[\hat{X}]=0$

(c) Suppose we have constructed an estimator $\hat{X}$ from some samples of X. We now want to know how well $\hat{X}$ estimates a fresh (new) sample of X. Denote this fresh sample by X'. Note that X' is an i.i.d. copy of the random variable X. **Derive a general expression for the expected squared error $E[(\hat{X}-X')^2]$ in terms of σ^2 and the bias and variance of the estimator $\hat{X}$. Similarly, derive an expression for the expected squared error $E[(\hat{X}-\mu)^2]$. Compare the two expressions and comment on the differences between them, if any.**

Solution: Since $\hat{X}$ is a function of X, we conclude that the random variables $\hat{X}$ and X' are independent of each other. Now we provide two ways to solve the first problem.

Method 1: In this method, we use the trick of adding and subtracting a term to derive the desired expression:

$$\begin{aligned}
E[(\hat{X}-X')^2] &= E[(\hat{X}-\mu+\mu-X')^2] \\
&= E[(\hat{X}-\mu)^2 + \underbrace{E[(\mu-X')^2]}_{=\mathrm{Var}(X')=\sigma^2} \\
&= E[(\hat{X}-\mu)^2 + \sigma^2 \\
&= E[(\hat{X}-E[\hat{X}]+E[\hat{X}]-\mu)^2 + \sigma^2 \\
&= \underbrace{E[(\hat{X}-E[\hat{X}])^2}_{=\mathrm{Var}(\hat{X})} + \underbrace{(E[\hat{X}]-\mu)^2}_{=\mathrm{bias}^2} + 2\underbrace{E[(\hat{X}-E[\hat{X}])\cdot(E[\hat{X}]-\mu)]}_{=0} + \sigma^2
\end{aligned}$$

Method 2: In this method, we make use of the definition of variance. We have

$$\begin{aligned}
E[(\hat{X}-X')^2] &= E[\hat{X}^2]+E[X'^2]-2E[\hat{X}X'] \\
&= (\mathrm{Var}(\hat{X})+(E[\hat{X}])^2)+(\mathrm{Var}(X')+(E[X'])^2)-2E[\hat{X}X'] \\
&= ((E[\hat{X}])^2-2E[\hat{X}X']+(E[X'])^2)+\mathrm{Var}(\hat{X})+\underbrace{\mathrm{Var}(X')}_{=\mathrm{Var}(X)} \\
&= (E[\hat{X}]-\underbrace{E[X']}_{=E[X]=\mu})^2+\mathrm{Var}(\hat{X})+\mathrm{Var}(X) \\
&= \underbrace{(E[\hat{X}]-\mu)^2}_{=\mathrm{bias}^2}+\mathrm{Var}(\hat{X})+\sigma^2
\end{aligned}$$

The first term is equivalent to the bias of our estimator squared, the second term is the variance of the estimator, and the last term is the irreducible error.

Now let's do $E[(\hat{X}-\mu)^2]$.

$$\begin{aligned}
E[(\hat{X}-\mu)^2] &= E[\hat{X}^2]+E[\mu^2]-2E[\hat{X}\mu] \tag{9} \\
&= (Var(\hat{X})+E[\hat{X}]^2)+(Var(\mu)+E[\mu]^2)-2E[\hat{X}\mu] \tag{10} \\
&= (E[\hat{X}]^2-2E[\hat{X}\mu]+E[\mu]^2)+Var(\hat{X})+Var(\mu) \tag{11} \\
&= (E[\hat{X}]-E[\mu])^2+Var(\hat{X})+Var(\mu) \tag{12} \\
&= (E[\hat{X}]-\mu)^2+Var(\hat{X}). \tag{13}
\end{aligned}$$

Notice that these two expected squared errors resulted in the same expressions except for the σ^2 in $E[(\hat{X}-X')^2]$. The error σ^2 is considered "irreducible error" because it is associated with the noise that comes from sampling from the distribution of X. This term is not present in the second derivation because μ is a fixed value that we are trying to estimate.

(d) For the following parts, we will refer to expected total error as $E[(\hat{X}-\mu)^2]$. It is a common mistake to assume that an unbiased estimator is always "best." Let's explore this a bit further. **Compute the expected squared error for each of the estimators above.** **Solution:** Adding the previous two answers:

(a) $\frac{\sigma^2}{n}$

(b) $\frac{1}{(n+1)^2}(\mu^2+n\sigma^2)$

(c) $\frac{1}{(n+n_0)^2}(n_0^2\mu^2+n\sigma^2)$

(d) μ^2

(e) **Demonstrate that the four estimators are each just special cases of the third estimator, but with different instantiations of the hyperparameter n_0.**

Solution: The derivation for the third estimator works for *any* value of n_0. The first estimator is just the third estimator with n_0 set to 0:

$$\frac{x_1+x_2+\ldots+x_n}{n+n_0} = \frac{x_1+x_2+\ldots+x_n}{n+0} + \frac{x_1+x_2+\ldots+x_n}{n}.$$

The second estimator is just the third estimator with n_0 set to 1:

$$\frac{x_1+x_2+\ldots+x_n}{n+n_0} = \frac{x_1+x_2+\ldots+x_n}{n+1}.$$

The last estimator is the limiting behavior as n_0 goes to ∞. In other words, we can get arbitrarily close to the fourth estimator by setting n_0 very large:

$$\lim_{n_0\to\infty}\frac{x_1+x_2+\ldots+x_n}{n+n_0} = 0.$$

(f) **What happens to bias as n_0 increases? What happens to variance as n_0 increases?**

Solution:

One reason for increasing the samples of n_0 is if you have reason to believe that X is centered around 0. In increasing the number of zeros we are injecting more confidence in our belief that the distribution is centered around zero. Consequently, in increasing the number of "fake" data, the variance decreases because your distriubtion becomes more peaked. Examining the expressions for bias and variance for the third estimator, we can see that larger values of n_0 result in decreasing variance $(\frac{n}{(n+n_0)^2}\sigma^2)$ but potentially increasing bias $(\frac{n_0\mu}{n+n_0})$. Hopefully you can see that there is a trade-off between bias and variance. Using an unbiased estimator is not always optimal nor is using an estimator with small variance always optimal. One has to carefully trade-off the two terms in order to obtain minimum squared error.

(g) Say that $n_0 = \alpha n$. **Find the setting for α that would minimize the expected total error, assuming you secretly knew μ and σ.** Your answer will depend on σ, μ, and n.

Solution: First, we write our expression for the total error in terms of α:

$$\frac{1}{(n+n_0)^2}(n_0^2\mu^2+n\sigma^2)$$

$$\frac{1}{(n+\alpha n)^2}((\alpha n)^2\mu^2+n\sigma^2)$$

$$\frac{1}{(1+\alpha)^2}\frac{1}{n^2}(\alpha^2n^2\mu^2+n\sigma^2)$$

$$\frac{1}{(1+\alpha)^2}(\alpha^2\mu^2+\frac{\sigma^2}{n})$$

$$\frac{\alpha^2}{(1+\alpha)^2}\mu^2+\frac{1}{(1+\alpha)^2}\frac{\sigma^2}{n}$$

Now take the derivative with respect to α and set it equal to 0:

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

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

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

$$\alpha=\frac{\sigma^2}{n\mu^2}.$$

(h) For this part, let's assume that we had some reason to believe that μ *should be small* (close to 0) and σ *should be large*. In this case, **what happens to the expression in the previous part? Solution:** The value of α can be quite large, since the solution has a small value of μ in the denominator. In mathematical terms, we could write the limit as μ goes to 0:

$$\lim_{\mu\to 0}\frac{\sigma^2}{n\mu^2}=\infty.$$

(i) In the previous part, we assumed there was reason to believe that μ *should be small*. Now let's assume that we have reason to believe that μ is not necessarily small, but *should be close to some fixed value* μ_0. **In terms of X and μ_0, how can we define a new random variable X' such that X' is expected to have a small mean? Compute the mean and variance of this new random variable.**

Solution: Shift the random variable X by the constant guess μ_0 to get the random variable $X'=X-\mu_0$. Let's calculate the mean and variance of this new random variable:

$$E[X']=E[X-\mu_0]=E[X]-\mu_0=\mu-\mu_0\approx 0.$$

The last line ($\mu-\mu_0\approx 0$) comes from the assumption that μ is close to μ_0.

We can also calculate the variance of X':

$$\mathrm{Var}[X'] = \mathrm{Var}[X - \mu_0] = \mathrm{Var}[X] - \mathrm{Var}\mu_0 = \mathrm{Var}[X] - 0 = \mathrm{Var}[X].$$

This is a useful step to understand the relation between X and X', but not necessary for a full solution to the question asked.

(j) Draw a connection between α in this problem and the regularization parameter λ in the ridge-regression version of least-squares. **What does this problem suggest about choosing a regularization coefficient and handling our data-sets so that regularization is most effective**? This is an open-ended question, so do not get too hung up on it.

Solution: The key lesson is another reminder that regularization reduces variance, at the cost of increasing bias, by forcing solutions towards zero. But the bias-variance trade-off is not always the same. If we first center our data around some prior, so that the model is supposed to be close to zero anyways, then we can use larger values of α or λ and reduce variance considerably for a small cost in bias. It may also be instructive to realize that regularization can be thought of as adding "fake" training data which is uniformly zero.

4 Estimation and approximation in linear regression

In typical applications, we are dealing with data generated by an *unknown* function (with some noise), and our goal is to estimate this function. So far we used linear and polynomial regressions. In this problem we will explore the quality of polynomial regressions when the true function is not polynomial.

Suppose we are given a full column rank feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and an observation vector $\mathbf{y} \in \mathbb{R}^n$. Let the vector $\mathbf{y}$ represent the noisy measurement of a true signal $\mathbf{y}^*$:

$$\mathbf{y} = \mathbf{y}^* + \mathbf{z}, \tag{14}$$

with $\mathbf{z} \in \mathbb{R}^n$ representing the random noise in the observation $\mathbf{y}$, where $z_j \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. We define the vectors $\mathbf{w}^*$ and $\hat{\mathbf{w}}$ as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{y}^* - \mathbf{X}\mathbf{w}\|_2^2 \qquad \text{and} \qquad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

Observe that for a given true signal $\mathbf{y}^*$ the vector $\mathbf{w}^*$ is fixed, but the vector $\hat{\mathbf{w}}$ is a random variable since it is a function of the random noise $\mathbf{z}$. Note that the vector $\mathbf{X}\mathbf{w}^*$ is the best linear fit of the true signal $\mathbf{y}^*$ in the column space of $\mathbf{X}$. Similarly, the vector $\mathbf{X}\hat{\mathbf{w}}$ is the best linear fit of the observed noisy signal $\mathbf{y}$ in the column space of $\mathbf{X}$.

After obtaining $\hat{\mathbf{w}}$, we would like to bound the error $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$, which is the *prediction error* incurred based on the specific n training samples. In this problem we will see how to get a good estimate of this prediction error. When using polynomial features, we will also learn how to decide the degree of the polynomial when trying to fit a noisy set of observations from a smooth function.

Remark: You can use the closed form solution for OLS and results from Discussion 3 for all parts of this problem. For parts (a)-(c), assume that the feature matrix $\mathbf{X}$ and the true signal vector $\mathbf{y}^*$

are fixed (and not random). Furthermore, in all parts the expectation is taken over the randomness in the noise vector $\mathbf{z}$.

(a) **Show that** $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$ and use this fact to **show that**

$$\mathbb{E}\left[\|\mathbf{y}^* - \mathbf{X}\hat{\mathbf{w}}\|_2^2\right] = \|\mathbf{y}^* - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}]\|_2^2 + \mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}\|_2^2\right].$$

Note that the above decomposition of the squared error corresponds to the sum of bias-squared and the variance of our estimator $\mathbf{X}\hat{\mathbf{w}}$.

Solution: First we will show that $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$. We use the following closed form expressions for the vectors $\hat{\mathbf{w}}$ and $\mathbf{w}^*$:

$$\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}^* \quad \text{and} \quad \hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}.$$

We have

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{y}^* + \mathbf{z})\\ &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}^* + (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{z}\\ &= \mathbf{w}^* + (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{z}\end{aligned}$$

Since $\mathbf{w}^*$ is fixed we obtain that $\mathbb{E}[\mathbf{w}^*] = \mathbf{w}^*$. Also notice that $\mathbb{E}[\mathbf{z}] = 0$. Using these two facts, we conclude

$$\mathbb{E}[\hat{\mathbf{w}}] = \mathbb{E}[\mathbf{w}^*] + \mathbb{E}[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{z}] = \mathbf{w}^* + (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\mathbf{z}] = \mathbf{w}^*.$$

Now we will show the second part of the question. In discussion 3, we defined projection of a vector $\mathbf{v}$ onto column space of a matrix $\mathbf{X}$ as, $P_{\mathbf{X}}(\mathbf{v}) = \mathbf{X}(\arg\min_{\mathbf{w}} \|\mathbf{v} - \mathbf{X}\mathbf{w}\|_2^2)$. As a shorthand we will donte $P_{\mathbf{X}}(\mathbf{v})$ as $\mathbf{v}_{\mathbf{P}}$.
Notice that

$$\begin{aligned}\|\mathbf{y}^* - \mathbf{y}_{\mathbf{P}}\|_2^2 &= \|\mathbf{y}^* - \mathbf{y}_P^* + \mathbf{y}_P^* - \mathbf{y}_P\|_2^2\\ &= \|\mathbf{y}^* - \mathbf{y}_P^*\|_2^2 + \|\mathbf{y}_P^* - \mathbf{y}_P\|_2^2\end{aligned}$$

Since the vector $\mathbf{y}^* - \mathbf{y}_P^*$ is orthogonal to the vector $\mathbf{y}_P^* - \mathbf{y}_P$ (go over discussion 3 to convince yourself), we can use the Pythagorean theorem for the last step of the equality. And since, $\mathbf{y}_P^* = \mathbf{X}\mathbf{w}^*$ and $\mathbf{y}_P = \mathbf{X}\hat{\mathbf{w}}$. Taking expectations on both sides in the above equality, we find that

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{y}^* - \mathbf{X}\hat{\mathbf{w}}\|_2^2\right] &= \mathbb{E}\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2^2 + \mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right]\\ &= \|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2^2 + \mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right]\end{aligned}$$

The last step makes use of the fact that $\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|$ is a fixed scalar. Using the previous result $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$, we obtain that $\mathbb{E}[\mathbf{X}\hat{\mathbf{w}}] = \mathbf{X}\mathbf{w}^*$ which yields the desired result.

(b) Recall that if $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where $\Sigma \in \mathbb{R}^{(d\times d)}$ is the covariance matrix, then for any matrix $A \in \mathbb{R}^{k\times d}$, we have $\mathbf{Av} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\Sigma\mathbf{A}^T)$. Use this fact to **show that** the distribution of the vector $\hat{\mathbf{w}}$ is given by

$$\hat{\mathbf{w}} \sim \mathcal{N}(\mathbf{w}^*, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}).$$

Solution: In the previous part, we saw that

$$\hat{\mathbf{w}} = \mathbf{w}^* + \underbrace{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top}_{=:\mathbf{A}}\mathbf{z}. \qquad (15)$$

Since $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, we get that $\mathbf{Az} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{A}\mathbf{A}^\top)$. Notice that

$$\mathbf{A}\mathbf{A}^\top = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top = (\mathbf{X}^\top\mathbf{X})^{-1}.$$

And thus we have $\mathbf{Az} \sim \mathcal{N}(\mathbf{0}, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1})$. Finally, recall that if $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and $\mathbf{c}$ is any fixed vector, then $\mathbf{v} + \mathbf{c} \sim \mathcal{N}(\mathbf{c}, \Sigma)$. Consequently, the result follows using equation (15) and the fact that $\mathbf{w}^*$ is fixed.

(c) Use part (b) to **show that**

$$\frac{1}{n}\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right] = \sigma^2\frac{d}{n}.$$

Hint: The trace trick: $\text{trace}(AB) = \text{trace}(BA)$, might be useful.

Solution: Lets first expand $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2$. We have

$$\begin{aligned}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2 &= \|\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\|_2^2 \\ &= \left(\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\right)^\top\left((\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*)\right) \\ &= (\hat{\mathbf{w}} - \mathbf{w}^*)^\top\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*).\end{aligned}$$

And thus we have

$$\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right] = \mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\right].$$

For any scalar α, we can write $\alpha = \text{trace}(\alpha)$. Using the identity $\text{trace}(AB) = \text{trace}(BA)$, we find

$$\mathbb{E}\left[\text{trace}\left((\hat{\mathbf{w}} - \mathbf{w}^*)^\top\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\right)\right] = \mathbb{E}\left[\text{trace}\left(\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right)\right].$$

Note that for a random matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we have

$$\mathbb{E}[\text{trace}(\mathbf{A})] = \mathbb{E}[\sum_{i=1}^{d} A_{ii}] = \sum_{i=1}^{d}\mathbb{E}[A_{ii}] = \text{trace}(\mathbb{E}[\mathbf{A}]).$$

This identity is a powerful one and worth remembering. Using this identity, we have

$$\mathbb{E}\left[\text{trace}\left(\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right)\right] = \text{trace}\left(\mathbf{X}^\top\mathbf{X}\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right]\right).$$

Recall that $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$. Hence, the matrix $\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right]$ is precisely the covariance matrix of the random vector $\hat{\mathbf{w}}$ and is equal to $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$. Using this fact, we obtain that

$$\begin{aligned}
\text{trace}\left(\mathbf{X}^\top\mathbf{X}\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right]\right) &= \text{trace}\left(\mathbf{X}^\top\mathbf{X}\sigma^2\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\right) \\
&= \sigma^2\text{trace}(\mathbb{I}_d) \\
&= \sigma^2 d.
\end{aligned}$$

Dividing by n proves the desired result.

To summarize, we have used the following sequence of equalities:

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right] &= \mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\right] \\
&= \mathbb{E}\left[\text{trace}\left((\hat{\mathbf{w}} - \mathbf{w}^*)^\top\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)\right)\right] \\
&= \mathbb{E}\left[\text{trace}\left(\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right)\right] \\
&= \text{trace}\left(\mathbb{E}\left[\mathbf{X}^\top\mathbf{X}(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right]\right) \\
&= \text{trace}\left(\mathbf{X}^\top\mathbf{X}\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}^*)(\hat{\mathbf{w}} - \mathbf{w}^*)^\top\right]\right) \\
&= \text{trace}\left(\mathbf{X}^\top\mathbf{X}\sigma^2\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\right) \\
&= \sigma^2\text{trace}(\mathbb{I}_d) \\
&= \sigma^2 d.
\end{aligned}$$

(d) Assume the underlying model is a noisy linear model with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$, i.e. $y_i = w_1\alpha_i + w_0 + z_i$. We construct matrix $\mathbf{X}$ by using $D+1$ polynomial features $\mathbf{P}_D(\alpha_i) = [1, \alpha_i, \ldots, \alpha_i^D]^\top$ of the *distinct* sampling points $\{\alpha_i\}_{i=1}^n$. For any $D \geq 1$, compare with model (14) and **compute $\mathbf{w}^*$ for this case. Also compute the bias ($\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2$) for this case.** Using the previous parts of this problem, **compute the number of samples n required to ensure that the average expected prediction squared error is bounded by ε?** Your answer should be expressed as a function of D, σ^2, and ε.

Conclude that as we increase model complexity, we require a proportionally larger number of samples for accurate prediction.

Solution: Notice that we can re-write the underlying model as follows:

$$\begin{aligned}
&\mathbf{y} = \mathbf{y}^* + \mathbf{z} = \mathbf{X}[w_0, w_1]^\top + \mathbf{z} \quad \text{where} \\
&\mathbf{y} = [y_1 \ldots y_n]^\top, \quad \mathbf{z} = [z_1 \ldots z_n]^\top \quad \text{and} \quad \mathbf{X} = [\mathbf{P}_1(\alpha_1) \ldots \mathbf{P}_1(\alpha_n)]^\top
\end{aligned}$$

Now observe that the true signal is indeed linear and hence we only need linear features. Consequently, we have that for $D = 1$: $\mathbf{w}^* = [w_0, w_1]^\top$. Extending for $D \geq 1$, we conclude that $\mathbf{w}^* = [w_0, w_1, \mathbf{0}]^\top$, where $\mathbf{0} \in \mathbb{R}^{D-1}$.

From the above, it is clear that $\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2 = 0$, i.e. the bias is 0.

Now, lets compute the number of samples n required to ensure that prediction error is bounded by ε. For a polynomial regression model, the matrix $\mathbf{X}$ has full column rank when the points are distinct (refer to HW2), hence $d = D+1$. From part (a) we know that the error is: $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2 = \|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2^2 + \|\mathbf{X}\mathbf{w}^* - \mathbf{X}\hat{\mathbf{w}}\|_2^2$, and using part (d) and the fact that the bias is 0, we get that:

$$\frac{1}{n}\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2\right] = \sigma^2\frac{D+1}{n}$$

and in order to upper bound the error with ε it suffices to have $n > \frac{\sigma^2(D+1)}{\varepsilon}$.

Clearly, the number of samples required for error ε increases linearly with the degree D and with the variance σ^2.

(e) Simulate the problem from part (d) for yourself. Set $w_1 = 1$, $w_0 = 1$, and sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-1,1]$. Generate $y_i = w_1\alpha_i + w_0 + z_i$ with z_i representing standard Gaussian noise.

Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.

You may show separate plots for the two scalings. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves.

(For this part, the libraries numpy.random and numpy.polyfit might be useful. You are free to use any and all python libraries.)

Solution: In the Figure 4 below, we plotted the log error vs degree of the polynomial and log of error vs the sample size for better understanding. As we can see the error increases with D and decreases with n.

```
import numpy as np
import matplotlib.pyplot as plt

deg = 19 #degree of the polynomial
step = 80
n_s = 40 #sample size
w_1 = 1
w_0 = 1
error = np.zeros((deg, step)) # defining error as a function of d and n
alpha = np.random.uniform(-1, 1, n_s) #generating alpha from unifrom dist
    ↪  with sample size n_s
z = np.random.normal(0, 1, n_s) # generating i.i.d noise with sample size
    ↪  n_s
y_s = w_1*alpha+w_0 # this is Y*
y = y_s+z
for N in range(step): # number of steps
    n = N+n_s # sample size at each step
    for d in range(1, deg+1): #degree of polynomial from 1 to deg
        w_hat = np.polyfit(alpha, y, d) #fitting degree d polynomial
        E = 0 #initializing error for this fit with degree=d and sample
    ↪ size=n
```

```
        for i in range(n):
            E = E + (np.polyval(w_hat, alpha[i])-y_s[i])**2 # adding
   ↪ error of each sample to total error
        error[d-1][N]=E/n # normalize the error
    alpha_new = np.random.uniform(-1, 1, 1) # generating a new sample for
   ↪  the next step to increase the sample size by 1
    z_new = np.random.normal(0, 1, 1) # generating noise for the new
   ↪ sample for the next step to increase the sample size by 1
    alpha = np.append(alpha, alpha_new) # adding the new sample to the
   ↪ sample array
    z = np.append(z, z_new) # adding the new noise value to the noise
   ↪ array
    y_s = w_1*alpha+w_0 # computing y*
    y = y_s+z # computing y

plt.figure()
plt.subplot(121)
plt.semilogy(np.arange(1, deg+1),error[:,-1])
plt.xlabel('degree of polynomial')
plt.ylabel('log of error')
plt.subplot(122)
plt.semilogy(np.arange(n_s, n_s+step), error[-1,:])
plt.xlabel('number of samples')
plt.ylabel('log of error')
plt.show()
```

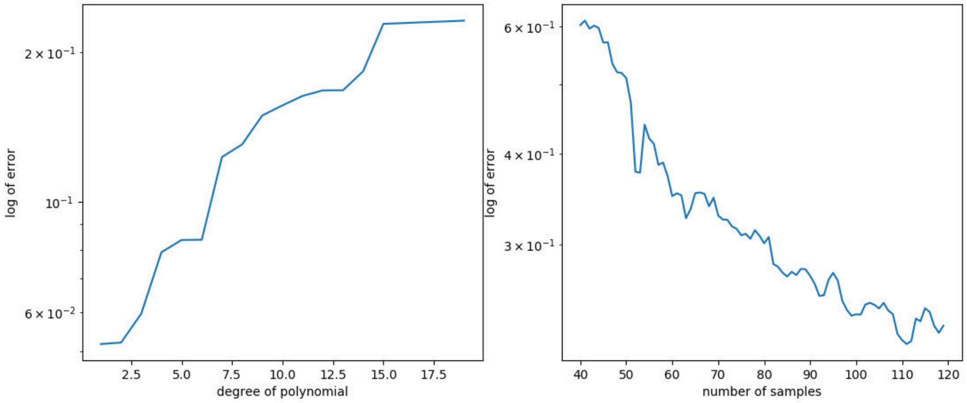


Figure 4: Average error vs D and n

(f) Assume that the underlying model is the noisy exponential function with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$ where $y_i = e^{\alpha_i} + z_i$ with *distinct* sampling points $\{\alpha_i\}_{i=1}^n$, in the interval $[-4, 3]$ and i.i.d. Gaussian noise $z_i \sim \mathcal{N}(0,1)$. We again construct matrix $\mathbf{X}$ by using $D+1$ polynomial features $[1, \alpha_i, \ldots, \alpha_i^D]^\top$ and use linear regression to fit the observations. Recall, the definitions of the bias and variance of the OLS estimator from part (a) and **show that for a fixed n, as the degree D of the polynomial increases: (1) the bias decreases and (2) the variance increases**. Use the values you derived for bias and variance and **show that for the prediction error, the optimal choice of D is given by $\mathcal{O}(\log n / \log\log n)$**.

Hint: You can directly use previous parts of this problem, Discussion 3 and Problem 4 of HW 2. You may assume that n and D are large for approximation purposes.

Solution: Using part (a), we split the prediction error in two terms: bias and variance. We first compute the bias (approximation error).

We use HW 2 Problem 4 now. For the approximation error, we use the Dth order Taylor series approximation of the function $f : \alpha \mapsto e^{\alpha}$ about the point 0, which is given (for some $\alpha' \in [0, \alpha]$) by

$$e^{\alpha} = \underbrace{\sum_{i=1,3,..}^{D} (-1)^{(i-1)/2}\frac{\alpha^i}{i!}}_{\phi_D(\alpha)} + f^{D+1}(\zeta)\frac{\alpha'^{D+1}}{(D+1)!}.$$

We derived the following bound in the last homework:

$$||f - \phi_D||_{\infty} = \sup_{\alpha\in[-3,4]} |e^{\alpha} - \phi_D(\alpha)| \leq \frac{e^3 4^{D+1}}{(D+1)!}.$$

Consequently, the approximation error for the sample $\{\alpha_i, y_i^*\}$ is bounded by $\frac{e^3 4^{D+1}}{(D+1)!}$. Summing and dividing through by n yields that

$$\text{Bias}^2 = \text{Approximation error} = \frac{1}{n}\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|^2 \leq \left(\frac{e^3 4^{D+1}}{(D+1)!}.\right)^2$$

Now using part (c) and the facts that the matrix $\mathbf{X}$ is of size $n \times (D+1)$ and $\sigma = 1$, we obtain that

$$\text{Variance} = \text{Estimation error} := \frac{1}{n}\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right] = \frac{D+1}{n}.$$

To optimally trade-off bias and variance, we set them to be equal. This provides the right scaling (upto constants). Think about why? Hint: Approximation $x + y \approx \max\{x, y\}$ for $x, y > 0$ will be useful. Using Stirling's approximation, ignoring constants and making a few approximations we find that

$$\begin{aligned}
\frac{(4e)^{D+1}}{(D+1)^{D+1}} &\approx \sqrt{\frac{D}{n}} \\
\frac{(4e)^{D+1}}{D^D} &\approx \sqrt{\frac{D}{n}} \\
\sqrt{n} &\approx \left(\frac{D}{4e}\right)^{D+1} \\
\sqrt{n} &\approx \left(\frac{D}{4e}\right)^{D} \\
\log n &\approx D\log\left(\frac{D}{4e}\right).
\end{aligned}$$

Now we verify if the approximation is valid for $D = \log n / \log\log n$:

$$\begin{aligned} D \log\left(\frac{D}{4e}\right) &\approx D \log D \\ &= \frac{\log n}{\log\log n} \cdot \log\left(\frac{\log n}{\log\log n}\right) \\ &= \frac{\log n}{\log\log n} \cdot (\log\log n - \log\log\log n) \\ &\approx \log n, \end{aligned}$$

and hence we are done.

(g) Simulate the problem in part (f) yourself. Sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-4, 3]$. Generate $y_i = e^{\alpha_i} + z_i$ with z_i representing standard Gaussian noise. **Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.** You may show separate plots for the two scalings. For scaling with D, choose $n = 120$. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves. (For this part, the libraries numpy.random and numpy.polyfit might be useful and you are free to use any and all python libraries.)

Solution: In the Figure 5 below, we plotted the log of error vs degree of the polynomial and log of error vs the sample size for better understanding.

```
import numpy as np
import matplotlib.pyplot as plt

deg = 12 #degree of the polynomial
step = 100
n_s = 20 #intial sample size
error = np.zeros((deg, step)) # defining error as a function of d and n
alpha = np.random.uniform(-4, 3, n_s) #generating alpha from unifrom dist
    ↪  with sample size n_s
z = np.random.normal(0, 1, n_s) # generating i.i.d noise with sample size
    ↪  n_s
y_s = np.exp(alpha) # this is y*
y = y_s+z
for N in range(step): # number of steps
    n = N+n_s # sample size at each step
    for d in range(1, deg+1): #degree of polynomial from 1 to deg
        w_hat = np.polyfit(alpha, y, d) #fitting degree d polynomial
        E = 0 #initializing error for this fit with degree=d and sample
    ↪ size=n
        for i in range(n):
            E = E + (np.polyval(w_hat, alpha[i])-y_s[i])**2 # adding
    ↪ error of each sample to total error
        error[d-1][N]=E/n # normalize the error
    alpha_new = np.random.uniform(-1, 1, 1) # generating a new sample for
    ↪  the next step to increase the sample size by 1
    z_new = np.random.normal(0, 1, 1) # generating noise for the new
    ↪ sample for the next step to increase the sample size by 1
    alpha = np.append(alpha, alpha_new) # adding the new sample to the
    ↪ sample array
    z = np.append(z, z_new) # adding the new noise value to the noise
    ↪ array
    y_s = np.exp(alpha) # computing y*
    y = y_s+z # computing y

plt.figure()
plt.subplot(121)
plt.semilogy(np.arange(1, deg+1),error[:,-1])
plt.xlabel('degree of polynomial')
plt.ylabel('log of error')
plt.subplot(122)
plt.semilogy(np.arange(n_s, n_s+step), error[5,:])
plt.xlabel('number of samples')
plt.ylabel('log of error')
plt.show()
```

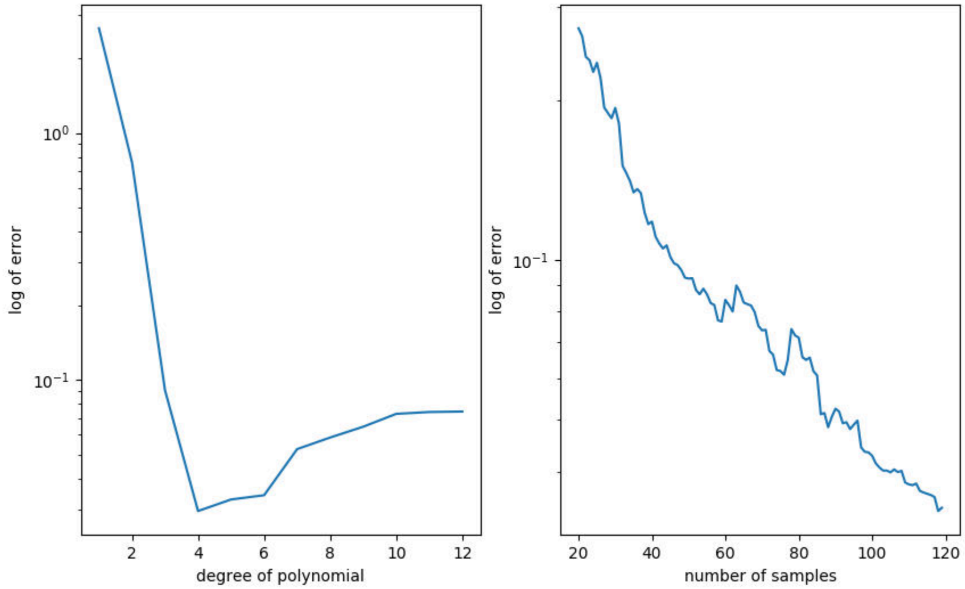


Figure 5: Average error vs D and n

(h) Comment on the differences in plots obtained in part (e) and part (g).

Solution:

In part (e), the true model was a linear model and therefore polynomial feature based OLS estimate was unbiased, i.e., we had no bias or approximation error. The variance or estimation error scales linearly with the degree of the polynomial and hence we saw in the graph that the prediction error increases monotonically with D.

For part (g), the true signal was an exponential function which can not be represented exactly by polynomials. As a result, we have an approximation error which however decreases with increase in the degree of the polynomial (as we had explored in the previous homework). However, as the model complexity (which in our case is equal to the degree of the polynomial) increases, the variance of the estimator also increases. As a result we see a trade-off between the bias and the variance. For small D, the bias is large and for large D, the variance is too big. Consequently, we see a sweet spot D^* at which the two terms are of the same order.

More generally, the approximation error measures how well our regression model fits the true data. Since we can better approximate the unknown function with higher order polynomials, this term decreases with D. However, it becomes harder to estimate the correct polynomial, since we now give ourselves extra freedom. This phenomenon is the bias variance trade-off in action!

As far as the scaling with respect to the number of samples n goes, we see that bias is unaffected by the number of samples. The variance of the estimator should only decrease with the number of samples and hence we see a decrease in prediction error with increase in number of samples.

Note that in the limit of infinite samples, the variance vanishes away and the prediction error will be equal to the bias term!

5 Robotic Learning of Controls from Demonstrations and Images

Huey, a home robot, is learning to retrieve objects from a cupboard, as shown in Fig. 6. The goal is to push obstacle objects out of the way to expose a goal object. Huey's robot trainer, Anne, provides demonstrations via tele-operation. When tele-operating the robot, Anne can look at the images captured by the robot and provide controls to Huey remotely.

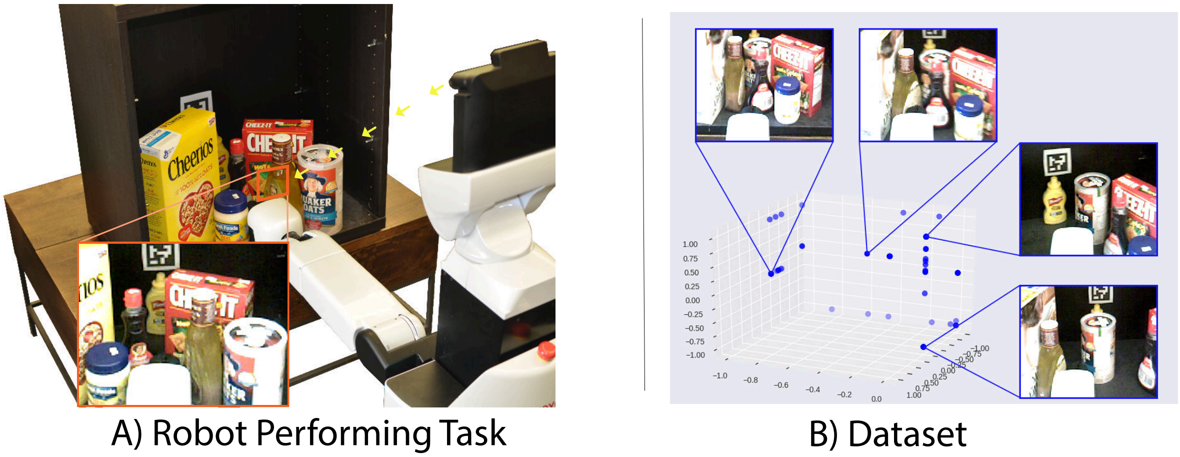
During a demonstration, Huey records the RGB images of the scene for each timestep, $x_0, x_1, ..., x_n$, where $x_i \in \mathbb{R}^{30\times30\times3}$ and the controls for his body, $u_0, u_1, \ldots, u_n$, where $u_i \in \mathbb{R}^3$. The controls correspond to making small changes in the 3D pose (i.e. translation and rotation) of his body. Examples of the data are shown in the figure.

Under an assumption (sometimes called the Markovian assumption) that all that matters for the current control is the current image, Huey can try to learn a linear *policy* π (where $\pi \in \mathbb{R}^{2700\times3}$) which linearly maps image states to controls (i.e. $\pi^\top x = u$). We will now explore how Huey can recover this policy using linear regression. Note please use **numpy** and **numpy.linalg** to complete this assignment.

(a) To get familiar with the structure of the data, **please visualize the 0th, 10th and 20th images in the training dataset. Also find out what's their corresponding control vectors.**

Solution: The corresponding controls are $[0,-1,0]$, $[-1,-0.45111084,-1]$, $[0,0,0.37368774]$

(b) Load the n training examples from x_train.p and compose the matrix X, where $X \in \mathbb{R}^{n\times2700}$. Note, you will need to flatten the images to reduce them to a single vector. The flattened



A) Robot Performing Task

B) Dataset

Figure 6: A) Huey trying to retrieve a mustard bottle. An example RGB image of the workspace taken from his head mounted camera is shown in the orange box. The angle of the view gives Huey and eye-in-hand perspective of the cupboard he is reaching into. B) A scatter plot of the 3D control vectors, or u labels. Notice that each coordinate of the label lies within the range of $[-1,1]$ for the change in position. Example images, or states x, are shown for some of the corresponding control points. The correspondence is indicated by the blue lines.

Figure 7: The 0th, 10th and 20th images in the training set.

image vector will be denoted by $\bar{x}$ (where $\bar{x} \in \mathbb{R}^{2700\times 1}$). Next, load the n examples from y_train.p and compose the matrix U, where $U \in \mathbb{R}^{n\times 3}$. Try to perform ordinary least squares to solve:

$$\min_{\pi} \|X\pi - U\|_F$$

to learn the *policy* $\pi \in \mathbb{R}^{2700\times 3}$. **Report what happens as you attempt to do this and explain why.**

Solution: The matrix is singular and not invertible. The reason is there isn't enough data to cover the high dimensional image space, so many solutions exist to solve the problem. Specifically, we only train the policy on 91 images, however a single RGB image is 30x30x3 in dimensionaility. So, with that many parameters, we can basically fit any arbitrary set of controls. (This is called the temptation to memorize in machine learning.) We can use ridge regression though to help condition the optimization to favor solutions with a small ℓ_2 norm.

(c) Now try to perform ridge regression:

$$\min_{\pi} ||X\pi - U||_2^2 + \lambda ||\pi||_2^2$$

on the dataset for regularization values $\lambda = \{0.1, 1.0, 10, 100, 1000\}$. Measure the average squared Euclidean distance for the accuracy of the policy on the training data:

$$\frac{1}{n}\sum_{i=0}^{n-1} ||\bar{x}_i^T \pi - u_i||_2^2$$

Report the training error results for each value of λ.

Solution:

The learned policy should match the training data with very low error. For all values of λ, in the specified range, you should see the error is below 10^{-8}, which is a very good fit for the dataset. It should be noted that the ability to fit training data perfectly does not necessarily mean the robot will perform well on unseen data. Even with the ridge penalties set to these values, the policy has apparently just memorized the controls.

Why was it able to do this? Remember, all the controls are small numbers between -1 and 1 while the training image data has large numbers. Consequently, small numbers for the parameters will allow these controls to be recovered. Furthermore, there are so many parameters that the work of memorizing the controls can be distributed across them. This further shrinks the parameters required to do this memorization. The ridge penalties are simply incapable of discouraging this memorization.

(d) Next, we are going to try standardizing the states. For each pixel value in each data point, x, perform the following operation:

$$x \mapsto \frac{x}{255} \times 2 - 1.$$

Since we know the maximum pixel value is 255, this rescales the data to be between $[-1, 1]$. **Repeat the previous part and report the average squared training error for each value of λ.**

Solution:

The answers for the fitting error for $\lambda = \{0.1, 1.0, 10, 100, 1000\}$ are $\{0.000, 0.000, 0.0016, 0.035, 0.25\}$. With standardization applied, we see as the regularization term is decreased the training loss is lowered. This can be interpreted as the variance of the model is increased as the possible function class is expanded.

We can also see that indeed having the smaller values in the images is forcing the system to use bigger values for the parameters if it wants to memorize the controls. The regularization penalty is now able to begin to discourage this.

(e) Evaluate both *policies* (i.e. with and without standardization on the new validation data x_test.p and y_test.p for the different values of λ. **Report the average squared Euclidean loss** and **qualitatively explain how changing the values of λ affects the performance in terms of bias and variance**.

Solution:

The answer is for $\lambda = \{0.1, 1.0, 10, 100, 1000\}$ is $\{0.87, 0.86, 0.83, 0.72, 0.73\}$ for with standardization and $\{0.77, 0.77, 0.77, 0.77, 0.77\}$ for with out.

The results with standardization illustrate that because the state space is so high dimensional the policy has trouble generalizing, thus adding bias (i.e. increasing λ can help generalization). However, increasing it too much can lead to worst performance.

We emperically see that without standardization the test error is higher, in the next section we will examine how we can characterize this.

It should be noted that the ability of Huey to generalize is still quite inadequate for a real robot policy. Later, in the course we will explore how to get significantly lower error on a larger dataset using convolutional neural networks.

(f) To better understand how standardizing improved the loss function, we are going to evaluate the *condition number* κ of the optimization, which is defined as

$$\kappa = \frac{\sigma_{\max}(X^T X + \lambda I)}{\sigma_{\min}(X^T X + \lambda I)}$$

or the ratio of the maximum singular value to the minimum singular value of the relevant matrix. Roughly speaking, the condition number of the optimization process measures how stable the solution will be when some error exists in the observations. More precisely, given a linear system $Ax = b$, the condition number of the matrix A is the maximum ratio of the relative error in the solution x to the relative error of b.

For the regularization value of $\lambda = 100$, **report the condition number with the standardization technique applied and without**.

Solution: The condition number without standardization is $\kappa = 52711697.6679$ and with standardization is $\kappa = 444.725931711$. By standardizing our data, we are able to significantly reduce the ratio of the eigenvalues, which makes our optimization less sensitive to noise in the data when performing matrix inversion.

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn the material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.