Homework 1

A1.

(a)

Bias is the difference between the expected value of our model and the true value we are trying to predict. When our model oversimplifies the "true distribution", we tend to have high bias.

Variance describes the sensitivity of our model to data changes (i.e. how much our estimation will change as the data changes). When our model is overly complex (i.e. overfitting the data), we tend to have high variance.

Bias-variance tradeoff is about finding the balance between bias and variance when modeling the distribution. We want to find the model that minimizes both bias and variance (i.e. the model that is not too simple or too complex), generalizing the trend, while keeping the necessary details to closely fit the data.

(b)

When model complexity increases, its bias usually decreases while its variance tend to increase.

When model complexity decreases, its bias usually increases while its variance tend to decrease.

e.g. In the case of linear regression, when we increase the degree of the polynomial, we are increasing the complexity of the model. As a result, the model will be able to fit the data more closely, but it will also be more sensitive to changes in the data (more oscillation in the graph), which is equivalent of saying that the bias will decrease, while variance will increase. The opposite is true when we decrease the degree (complexity) of the polynomial.

(c)

False. When we collect more data from the same distribution, the variance typically decreases.

Justification: More data helps the model better understand the underlying "true distribution", reducing its sensitivity to small fluctuations in the training set (the noises get "averaged out"). Therefore, the prediction is more stable and the variance decreases.

(d)

The validation set should be used for hyperparameter tuning.

If we use the training or the test set for hyperparameter tuning, the resulting model will be biased towards the specific training or test set. That's why validation exist in the first place (to minimizes the bias in hyperparameter tuning).

Let's take LOO as an example:

- Assume we have a dataset D with a set of hyperparameters to tune.
- Let $D \setminus j$ be training data with jth data point (x_j, y_j) moved to validation set.
- We train the model $f_{D \setminus j}$ with $D \setminus j$ dataset.
- We estimate the true error of $f_{D \setminus j}$ on the predicting y_j ($\{x_j, y_j\}$ is the validation set).
- We repeat the process for all jth data point in D (for each data point we leave out we learn a $f_{D \setminus j}$).
- Average over all data points j to get the LOO error.
- Repeat the process for all hyperparameters to get the LOO error for each hyperparameter.
- Pick the hyperparameter with the lowest LOO error.
- Train the model with the chosen hyperparameter on the entire dataset D.
- We test the model on the test set to get its performance on unseen data.

(e)

False. The training error of a function on the training set typically gives an underestimation of the true error.

Justification: When we train a model on a training set, the model will learn the pattern from the specific dataset, which means that it should predict the training set pretty well (i.e. the training error will be low). However, the pattern in the specific training set, especially for small datasets, might not generalize well for unseen data. So, the model might not predict unseen data as well (i.e. the true error might be higher). Therefore, the training error should be an underestimation of the true error.

A2.

(a)

$$
\begin{aligned}
\widehat{\lambda}_{\mathsf{MLE}} &= \arg\max_{\lambda} \mathbb{P}\{x \mid \lambda\} \\
&= \arg\max_{\lambda} \ \log \mathbb{P}\{x \mid \lambda\} \\
&= \arg\max_{\lambda} \ \log \prod_{i=1}^{n} \mathbb{P}\{x_i \mid \lambda\} \\
&= \arg\max_{\lambda} \ \sum_{i=1}^{n} \log \mathbb{P}\{x_i \mid \lambda\} \\
&= \arg\max_{\lambda} \ \sum_{i=1}^{n} \log e^{-\lambda} \frac{\lambda_i^x}{x_i!} \\
&= \arg\max_{\lambda} \ \sum_{i=1}^{n} \log e^{-\lambda} + \log \lambda_i^x - \log x_i! \\
&= \arg\max_{\lambda} \ \sum_{i=1}^{n} -\lambda + x_i \log \lambda - \log x_i! \\
&\Rightarrow \frac{\mathrm{d}}{\mathrm{d}\lambda} \sum_{i=1}^{n} -\lambda + x_i \log \lambda - \log x_i! = 0 && \text{(set derivative to 0)} \\
&\Rightarrow \sum_{i=1}^{n} -1 + \frac{x_i}{\lambda} = 0 \\
&\Rightarrow \sum_{i=1}^{n} \frac{x_i}{\lambda} = \sum_{i=1}^{n} 1 \\
&\Rightarrow \boxed{\widehat{\lambda}_{\mathsf{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i}
\end{aligned}
$$

(b)

$$
\begin{aligned}
\widehat{\lambda}_{\mathsf{MLE}} &= \frac{1}{n} \sum_{i=1}^{n} x_i && \text{(from part (a))} \\
&= \frac{1}{5}(3 + 7 + 5 + 0 + 2) && \text{(plug in the first 5 data points)} \\
&= \boxed{3.4}
\end{aligned}
$$

$$
\mathsf{Poi}(x = 4 \mid \lambda = 3.4) = \boxed{e^{-3.4} \cdot \frac{3.4^4}{4!} \approx 0.1858}
$$

(c)

$$\widehat{\lambda}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{(from part (a))}$$

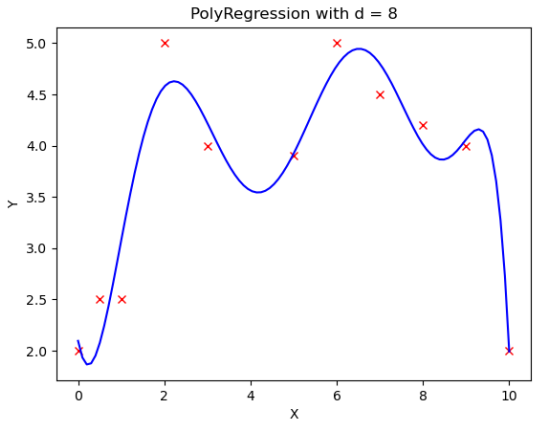
$$= \frac{1}{6}(3+7+5+0+2+8) \qquad \text{(plug in the first 6 data points)}$$

$$= \boxed{\frac{25}{6}}$$

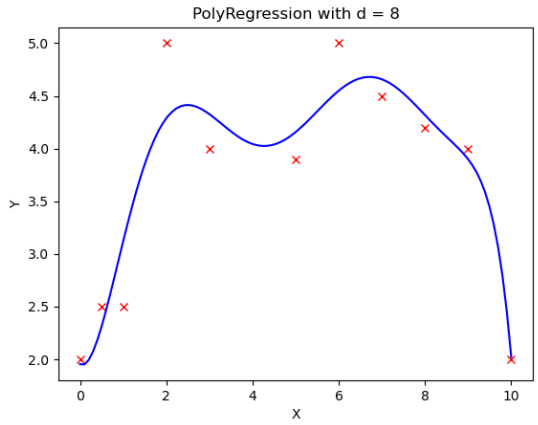
$$\text{Poi}(x = 5 \mid \lambda = \frac{25}{6}) = \boxed{e^{-\frac{25}{6}} \cdot \frac{(\frac{25}{6})^5}{5!} \approx 0.1623}$$

A3.

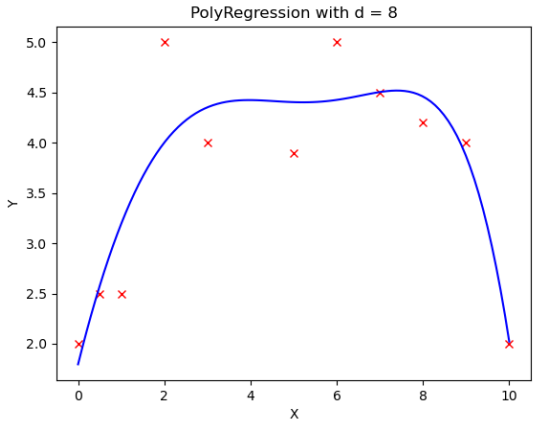
(b)



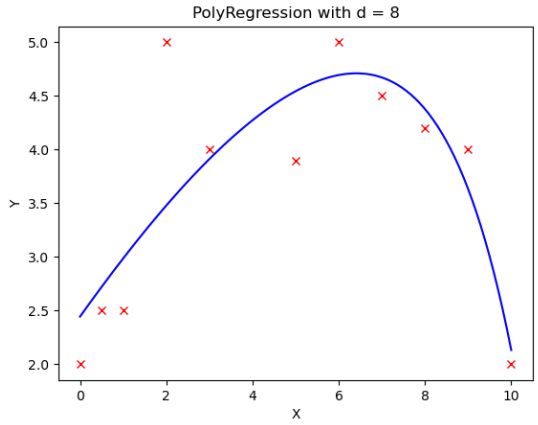
(a) $\lambda = 0$



(b) $\lambda = 1e-7$



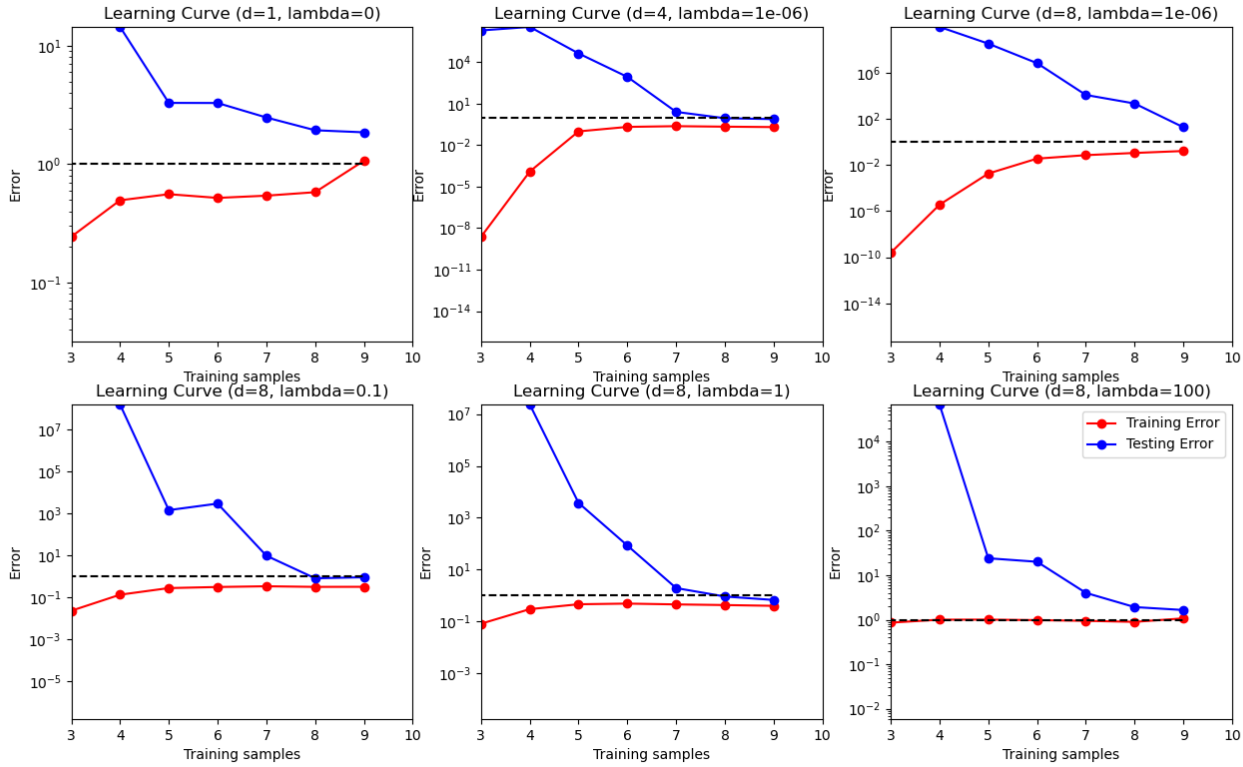
(c) $\lambda = 1e-4$



(d) $\lambda = 1e-1$

As shown above, as regularization parameter λ increases (from 0 to 0.1), the model puts more penalty on large weights which decreases variance. It resulted in smoother and less complex curves, which indicates that the model generalize better to new data points but has higher bias at the same time.

A4.


Learning Curve (d=1, lambda=0)
Learning Curve (d=4, lambda=1e-06)
Learning Curve (d=8, lambda=1e-06)
Learning Curve (d=8, lambda=0.1)
Learning Curve (d=8, lambda=1)
Learning Curve (d=8, lambda=100)
Error
Training samples
Training Error
Testing Error

A5.

(a)

$$
\begin{aligned}
\widehat{W} &= \underset{W}{\text{argmin}} \sum_{i=1}^{n} \|W^\top x_i - y_i\|_2^2 + \lambda \|W\|_F^2 \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[\sum_{i=1}^{n} (e_j^\top W^\top x_i - e_j^\top y_i)^2 + \lambda \|W e_j\|_2^2 \right] \\
&\qquad\qquad (\arg\max_{j=0,\ldots,9} e_{j+1}^\top \widehat{W}^\top x_i \text{ and } W = \begin{bmatrix} w_1 & \ldots & w_k \end{bmatrix}) \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[\sum_{i=1}^{n} (w_j^\top x_i - e_j^\top y_i)^2 + \lambda \|w_j\|_2^2 \right] \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[\|X w_j - Y e_j\|_2^2 + \lambda \|w_j\|_2^2 \right] \\
&\qquad\qquad (\text{ Where } X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d} \text{ and } Y = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix}^\top \in \mathbb{R}^{n \times k}) \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[(X w_j - Y e_j)^\top (X w_j - Y e_j) + \lambda w_j^\top w_j \right] \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[(X w_j)^\top X w_j - (X w_j)^\top Y e_j - (Y e_j)^\top X w_j + (Y e_j)^\top Y e_j + \lambda w_j^\top w_j \right] \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[w_j^\top X^\top X w_j - 2 w_j^\top X^\top Y e_j + e_j^\top Y^\top Y e_j + \lambda w_j^\top w_j \right] \\
&= \underset{W}{\text{argmin}} \sum_{j=1}^{k} \left[w_j^\top (X^\top X + \lambda I) w_j - 2 w_j^\top X^\top Y e_j + e_j^\top Y^\top Y e_j \right]
\end{aligned}
$$

We take gradient w.r.t w_j:

$$
\begin{aligned}
&\nabla_{w_j} \sum_{j=1}^{k} \left[w_j^\top (X^\top X + \lambda I) w_j - 2 w_j^\top X^\top Y e_j + e_j^\top Y^\top Y e_j \right] \\
=&\sum_{j=1}^{k} \nabla_{w_j} (w_j^\top (X^\top X + \lambda I) w_j) - 2 \nabla_{w_j} (w_j^\top X^\top Y e_j) + \nabla_{w_j} (e_j^\top Y^\top Y e_j) \\
=&\sum_{j=1}^{k} 2 (X^\top X + \lambda I) w_j - 2 X^\top Y e_j \qquad\qquad (\text{Since } X^\top X + \lambda I \text{ is symmetric}) \\
=&2 (X^\top X + \lambda I) \sum_{j=1}^{k} w_j - 2 X^\top Y \sum_{j=1}^{k} e_j \\
=&2 (X^\top X + \lambda I) W - 2 X^\top Y I \\
=&2 (X^\top X + \lambda I) W - 2 X^\top Y
\end{aligned}
$$

(a contd.)

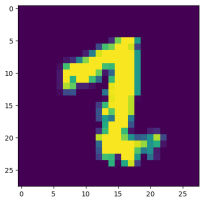
Setting the gradient to zero, we get:

$$2(X^\top X + \lambda I)\widehat{W} - 2X^\top Y = 0$$
$$\Rightarrow (X^\top X + \lambda I)\widehat{W} = X^\top Y$$
$$\Rightarrow \boxed{\widehat{W} = (X^\top X + \lambda I)^{-1} X^\top Y}$$

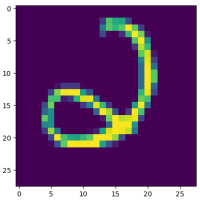
(c)

Training Error: 14.805%
Test Error: 14.66%

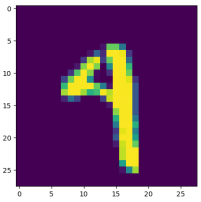
(d)



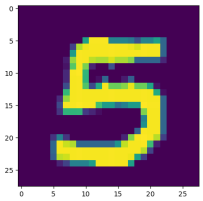
(a) $y = 2, \widehat{y} = 1$



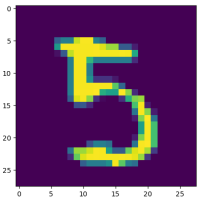
(b) $y = 2, \widehat{y} = 6$



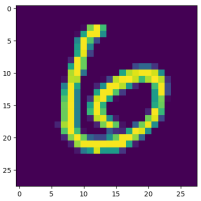
(c) $y = 4, \widehat{y} = 9$



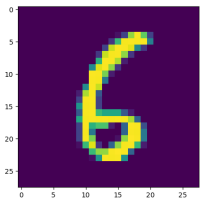
(d) $y = 5, \widehat{y} = 3$



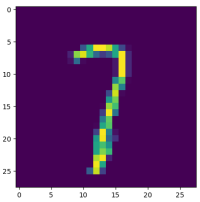
(e) $y = 5, \widehat{y} = 3$



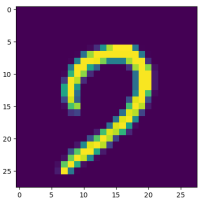
(f) $y = 6, \widehat{y} = 4$



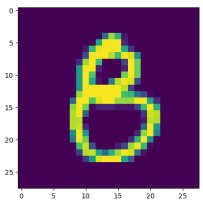
(g) $y = 6, \widehat{y} = 8$



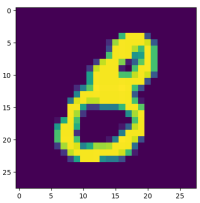
(h) $y = 7, \widehat{y} = 1$



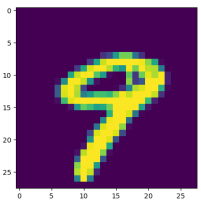
(i) $y = 7, \widehat{y} = 9$



(j) $y = 8, \widehat{y} = 3$



(k) $y = 8, \widehat{y} = 3$



(l) $y = 9, \widehat{y} = 7$

It seems that the classifier tends to confuse digits with similar structural features (similar shapes and strokes) or those that are poorly written.
Digits like 1 and 7, 1 and 2, 4 and 9 can be easily mixed up if the handwriting is ambiguous (true even for human).
However, for digits that are structurally similar like 5 and 3, 8 and 3, the handwriting are pretty clear for human to distinguish, but the classifier still having a hard time telling them apart.

A6.

(a)

15 hours