Homework #2

ECE 461/661: Introduction to Machine Learning for Engineers
Prof. Carlee Joe-Wong and Prof. Gauri Joshi

Due: Friday, Feb 21, 2025
8:59 pm PT/11:59 pm ET/Saturday, Feb 22, 2025 at 6:59 am CAT

Please remember to show your work for all problems and to write down the names of any students that you collaborate with. The full collaboration and grading policies are available on the course website: https://www.andrew.cmu.edu/course/18-661/. You are strongly encouraged (but not required) to use Latex to typeset your solutions.

Your solutions should be uploaded to Gradescope (https://www.gradescope.com/) in PDF format by the deadline. We will not accept hardcopies. If you choose to hand-write your solutions, please make sure the uploaded copies are legible. Gradescope will ask you to identify which page(s) contain your solutions to which problems, so make sure you leave enough time to finish this before the deadline. We will give you a 30-minute grace period to upload your solutions in case of technical problems.

1 Naïve Bayes Parameters *[10 points]*

The naïve Bayes approach assumes that the feature vectors are independent given the label, that is, for any given data point $\mathbf{x_i} = [x_{i1}, x_{i2}, \cdots, x_{id}]^\top$ and its label y_i, we have

$$\mathbb{P}(x_{i1}, x_{i2}, \cdots, x_{id}, y_i) = \mathbb{P}(y_i) \prod_{j=1}^{d} \mathbb{P}(x_{ij} \mid y_i) \tag{1}$$

Suppose that we are given the data set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]^\top$, where $\mathbf{X} \in \mathbb{R}^{N \times 4}$, consisting of N data points and 4 features and, its label $\mathbf{y} = [y_1, y_2, \cdots, y_N]^\top$. Assume that the first feature has 2, the second has 3, the third has 4 and the fourth has 5 possible values: Let $X_1 \in \{1, 2\}$, $X_2 \in \{1, 2, 3\}$, $X_3 \in \{1, 2, 3, 4\}$, and $X_4 \in \{1, 2, 3, 4, 5\}$. Also, there are 4 possible labels, i.e., $Y \in \{1, 2, 3, 4\}$ $\forall i$.

a. *[5 points]* Determine the number of **free parameters** θ and π that one has to estimate using the naïve Bayes framework. (We use θ for the probabilities of a feature given a label (e.g., $\theta_{3,2,4} = P(X_3 = 2 \mid Y = 4)$) and π for the class priors (e.g., $\pi_3 = P(Y = 3)$) similar to the lecture slides.)

b. *[4 points]* If the features are not independent conditioned on the label, one has to estimate the entire joint distribution $\mathbb{P}(X_1, X_2, X_3, X_4 \mid Y = i)$, for all $i \in \{1, 2, 3, 4\}$. Determine the number of **free parameters** that one has to estimate in such a scenario.

c. *[1 points]* Based on the numbers of parameters you found in parts (a) and (b), explain one advantage of assuming conditional independence.

Solution:

a. As there are 4 labels, one needs to estimate 3 **free parameters for π values**. (i.e., π_1, π_2, π_3) for prior probabilities (Note that the last one can be found using normalization, $\pi_4 = 1 - \sum_{i=1}^{3} \pi_i$).

Also, for each label, for each possible value of each feature, we need to estimate the probability. For example, for feature X_2, we need to find $P(X_2 = 1|Y = i)$, $P(X_2 = 2|Y = i)$, $P(X_2 = 3|Y = i)$ for all $i \in \{1, 2, 3, 4\}$. Observe that for any i, using normalization, we just need the number of possible feature values minus 1 **free parameters**, (i.e., $P(X_2 = 3|Y = i) = 1 - P(X_2 = 1|Y = i) - P(X_1 = 2|Y = i)$). Therefore, for the second feature, we need $(3 - 1) \times 5 = 10$ parameter θ estimates.

Generalizing this, we need (#of possible values for feature j -1)$\times$(#of labels) parameter estimate for feature j. It makes;

(a) $(2 - 1) \times 4 = 4$ parameters is needed for the first

(b) $(3 - 1) \times 4 = 8$ parameters is needed for the second

(c) $(4 - 1) \times 4 = 12$ parameters is needed for the third feature.

(d) $(5 - 1) \times 4 = 16$ parameters is needed for the third feature.

In total, **we need** 40 **free parameters for θ estimates**.

Therefore, we need a total of 43 **free parameters π and θ estimates.**

b. Using the same reasoning, we would still **need to estimate** 3 **free parameters for** π values.

However, this time, we need to estimate the joint distribution of features given labels. For $Y = i$ for all $i \in \{1, 2, 3, 4\}$, we need to find $P(X_1 = x_1, X_1 = x_2, X_3 = x_3, X_4 = x_4 \mid Y = i)$ for all possible values of x_1, x_2, x_3, and x_4. We are given that x_1 can take 2, x_2 can take 3, x_3 can take 4 and x_4 can take 5 different values. Therefore, we need to estimate $2 \times 3 \times 4 \times 5 - 1 = 120 - 1 = 119$ parameters for each $Y = i$. -1 is again to account for the observation that we will not need to estimate the parameter for one of the tuples (x_1, x_2, x_3, x_4) because we know that,

$$\sum_{x_1, x_2, x_3, x_4} P(X_1 = x_1, X_1 = x_2, X_3 = x_3, X_4 = x_4 \mid Y = i) = 1, \text{ for } i \in \{1, 2, 3, 4\}.$$

Therefore, for 4 possible labels, **we need** $119 \times 4 = 476$ θ **values.**

Then, we **need** $476 + 3 = 479$ **free parameters in total.**

c. With naïve assumption of Naive Bayes, we need a much smaller number of parameter estimates. The number of parameters grows exponentially with # of possible values of features if conditional independence is not assumed.

2 Naïve Bayes in Practice *[12 points]*

In this problem we will use the naïve Bayes algorithm to classify movie reviews as positive, neutral or negative. A simple approach involves maintaining a vocabulary of words that commonly occur in movie reviews and using the frequency of their occurrence in the three classes to classify movie reviews.

We are given the vocabulary $\mathcal{V} = \{$ 1:"incredible", 2:"plot", 3:"great", 4:"amazing", 5:"okay", 6:"decent", 7:"movie", 8:"no", 9:"acting", 10:"waste" $\}$. We will use V_i to represent the i^{th} word in $\mathcal{V}$. The training set provided includes four positive reviews:

- "great movie amazing"
- "incredible movie"

- "great acting amazing plot"
- "amazing acting amazing plot"

two neutral reviews:

- "okay movie"
- "decent no amazing acting"

and two negative reviews:

- "amazing waste"
- "no movie plot"

Recall that the naïve Bayes classifier is a generative classifier, where the probability of an input $\mathbf{x} = [x_1, x_2, \cdots, x_n]^\top$ depends on its class y. In our case the input vector $\mathbf{x}$ corresponding to each movie review has $n = 10$ equal to the number of words in the vocabulary $\mathcal{V}$, where each entry x_i is equal to the number of times word V_i occurs in $\mathbf{x}$.

a. *[2 points]* Calculate the naïve Bayes estimates of $\Pr(y = 1)$, $\Pr(y = 2)$ and $\Pr(y = 3)$ from the training data, where $y = 1$ corresponds to positive reviews, $y = 2$ to neutral reviews and $y = 3$ to negative reviews.

b. *[1 points]* List the feature vector $\mathbf{x}$ for each positive review in the training set

c. *[2 points]* In the naïve Bayes model, the likelihood of a sentence with feature vector $\mathbf{x}$ given a class c is

$$\Pr(\mathbf{x} \mid y = c) = \prod_{k=1}^{n} (\theta_{c,k})^{x_k}$$

where $\theta_{c,k}$ is the weight of word k among all words of class c. Calculate the maximum likelihood estimate of $\theta_{1,4}, \theta_{1,7}, \theta_{2,4}, \theta_{2,7}, \theta_{3,4}$ and $\theta_{3,7}$

d. *[3 points]* Given a new review "amazing movie", decide whether it is positive, neutral or negative, based on the naïve Bayes classifier, learned from the above data.

e. *[4 points]* Use Laplacian smoothing with $\alpha = 1$ to decide whether the review "decent movie" is positive, neutral or negative. In one sentence, describe the problem we would encounter if we had not used Laplacian smoothing.

Solution:

a. The naïve Bayes estimates of $\Pr(y = c)$ is the fraction of samples in the training set labelled c. Therefore $\Pr(y = 1) = 4/8 = 1/2$, $\Pr(y = 2) = 1/4$ and $\Pr(y = 3) = 1/4$

b.
- $x_3 = x_4 = x_7 = 1$, $x_i = 0$ for all other i
- $x_1 = x_7 = 1$, $x_i = 0$ for all other i
- $x_2 = x_3 = x_4 = x_9 = 1$, $x_i = 0$ for all other i
- $x_4 = 2$, $x_2 = x_9 = 1$, $x_i = 0$ for all other i

c. The Maximum Likelihood Estimate of $\theta_{c,k}$ is given by

$$\hat{\theta}_{c,k} = \frac{\text{\# of times word } k \text{ shows up in data points labeled as } c}{\text{\#total trials for data points labeled as } c}$$

Therefore, $\hat{\theta}_{1,4} = 4/13$, $\hat{\theta}_{1,7} = 2/13$, $\hat{\theta}_{2,4} = 1/6$, $\hat{\theta}_{2,7} = 1/6$, $\hat{\theta}_{3,4} = 1/4$, and $\hat{\theta}_{3,7} = 1/4$

d. Naive Bayes classification chooses the class $\hat{y}$ that maximizes the MAP of an observed data point $\mathbf{x}$

$$\hat{y} = \text{argmax}_c \Pr(y = c) \Pr(\mathbf{x}|y = c)$$

which, for our case, is equivalent to

$$\hat{y} = \text{argmax}_{c \in \{0,1\}} \left(\log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} \right)$$

For the movie review "amazing movie", with $c = 1$,

$$\begin{aligned} \log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log(1/2) + \log \hat{\theta}_{1,4} + \log \hat{\theta}_{1,7} \\ &= \log(1/2) + \log 4/13 + \log 2/13 \\ &= -1.6258 \end{aligned}$$

with $c = 2$

$$\begin{aligned} \log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log(1/4) + \log \hat{\theta}_{2,4} + \log \hat{\theta}_{2,7} \\ &= \log(1/4) + \log 1/6 + \log 1/6 \\ &= -2.1583 \end{aligned}$$

with $c = 3$

$$\begin{aligned} \log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log(1/4) + \log \hat{\theta}_{3,4} + \log \hat{\theta}_{3,7} \\ &= \log(1/4) + \log 1/5 + \log 1/5 \\ &= -2 \end{aligned}$$

Therefore $\hat{y} = 1$ and the review is classified as positive.

e. Using Laplacian smoothing with $\alpha = 1$, $\hat{\theta}_{c,k}$ becomes

$$\hat{\theta}_{c,k} = \frac{(\text{\# of times word } k \text{ shows up in data points labeled as } c) + 1}{(\text{\#total trials for data points labeled as } c) + |\mathcal{V}|}$$

Therefore, for the movie review "decent movie", with $c = 1$

$$\begin{aligned} \log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log \left(\frac{1}{2} \right) + \log \hat{\theta}_{1,6} + \log \hat{\theta}_{1,7} \\ &= \log \left(\frac{1}{2} \right) + \log \left(\frac{0+1}{13+10} \right) + \log \left(\frac{2+1}{13+10} \right) \\ &= -2.547 \end{aligned}$$

with $c = 2$

$$\begin{aligned}\log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log\left(\frac{1}{4}\right) + \log \hat{\theta}_{2,6} + \log \hat{\theta}_{2,7} \\ &= \log\left(\frac{1}{4}\right) + \log\left(\frac{1+1}{6+10}\right) + \log\left(\frac{1+1}{6+10}\right) \\ &= -2.4082\end{aligned}$$

and with $c = 3$

$$\begin{aligned}\log \Pr(y = c) + \sum_k x_k \log \hat{\theta}_{c,k} &= \log\left(\frac{1}{4}\right) + \log \hat{\theta}_{3,6} + \log \hat{\theta}_{3,7} \\ &= \log\left(\frac{1}{4}\right) + \log\left(\frac{0+1}{5+10}\right) + \log\left(\frac{1+1}{5+10}\right) \\ &= -2.6532\end{aligned}$$

Therefore $\hat{y} = 2$ and the review "decent movie" is classified as neutral.

Had Laplacian smoothing not been used, the likelihood of $y = 1$ and $y = 3$ would always be $-\infty$ or undefined (or zero if not using log-likelihood).

3 Logistic Regression in Practice *[9 points]*

Given a training set $\mathcal{D} = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}, \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right)\right\}$ where ($\mathbf{x}^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \{0, 1\}$) is the feature vector and the binary label for data point i, we want to find the parameters $\hat{\mathbf{w}}$ that maximize the likelihood for the training set, assuming a parametric model of the form

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp\left(-\mathbf{w}^T \mathbf{x}\right)}.$$

The conditional log likelihood of the training set is

$$L(\mathbf{w}) = \sum_{i=1}^{n} y^{(i)} \log p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right) + \left(1 - y^{(i)}\right) \log\left(1 - p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right)$$

and the gradient is

$$\nabla L(\mathbf{w}) = \sum_{i=1}^{n} \left(y^{(i)} - p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) \mathbf{x}^{(i)} \tag{2}$$

a. *[2 points]* Is it possible to get a closed form for the parameters $\hat{\mathbf{w}}$ that maximize the conditional log likelihood? If it is possible, give the closed-form solution. If not, how would you compute $\hat{\mathbf{w}}$ in practice? Explain your method to find $\hat{\mathbf{w}}$ in a few sentences or provide a short pseudo-code.

For a binary logistic regression model, we predict $y = 1$ when $p(y = 1 \mid \mathbf{x}) \geq 0.5$. Assume that the decision boundary occurs when $P(y = 1 \mid \mathbf{x}, \mathbf{w}) = P(y = 0 \mid \mathbf{x}, \mathbf{w})$.

b. *[4 points]* Find the decision boundary, which is the set of $\mathbf{x}$ satisfying $P(y = 1 \mid \mathbf{x}, \mathbf{w}) = P(y = 0 \mid \mathbf{x}, \mathbf{w})$.

c. *[1 points]* Is this model a linear classifier?

d. *[2 points]* Now, let us assume that in our application of this model, our tolerance of mis-classifying samples with the true label $y = 0$ is much lower than our tolerance to mis-classifying samples with true label $y = 1$ (i.e., we do not want to estimate a data point as belonging to class 0 if its true class is 1. However, we can tolerate the error of estimating a data point belong to class 1 if its true class is 0.) For example, in predicting the presence of a disease, we may be much more tolerant of mis-classifying a healthy person (true label $y = 0$) as having the disease ($y = 1$) compared to mis-classifying a person with the disease (true label $y = 1$) as not having it ($y = 0$).

Now suppose you have trained a logistic regression model to obtain the model weights $\mathbf{w}$. How would you adjust the prediction with weights $\mathbf{w}$ to decrease the mis-classification of samples with true label $y = 1$?

Solution:

a. There is no closed form expression for maximizing the conditional log likelihood. One has to consider iterative optimization methods, such as gradient descent (GD), to compute $\hat{\mathbf{w}}$. In GD, we first initialize the weight vector and set a learning rate. Then we iteratively calculate the gradient using the expression in Eq. 2 and update our weight vector in the negative gradient direction. See the pseudo-code below:

1 Initialize the weight vector $\mathbf{w}$, and set the learning rate α.

2 Repeat until convergence:

i Compute gradient: $\mathbf{g} \leftarrow \sum_{i=1}^{n} \left(y^{(i)} - p\left(y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w}\right)\right) \mathbf{x}^{(i)}$.

ii Update weights: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{g}$.

iii Check for convergence criteria.

b. Using the parametric form for the decision boundary where $p(y = 1 \mid \mathbf{x}) = p(y = 0 \mid \mathbf{x}) = 0.5$:

$$\begin{aligned} p(y = 1 \mid \mathbf{x}) = \frac{1}{2} &\Longrightarrow \frac{1}{1 + \exp\left(-\mathbf{w}^T \mathbf{x}\right)} = \frac{1}{2} \\ &\Longrightarrow 1 + \exp\left(-\mathbf{w}^T \mathbf{x}\right) = 2 \\ &\Longrightarrow \exp\left(-\mathbf{w}^T \mathbf{x}\right) = 1 \\ &\Longrightarrow -\mathbf{w}^T \mathbf{x} = 0 \\ &\Longrightarrow \mathbf{w}^T \mathbf{x} = 0 \end{aligned}$$

so the decision boundary is $\{\mathbf{x} : \mathbf{w}^T \mathbf{x} = 0\}$.

c. It is linear.

d. Decrese the threshold for $\hat{y} = 1$ estimates. For example, use the classification rule $p(\hat{y} = 1 \mid \mathbf{x}) \geq 0.3$ instead of threshold 0.5.

4 Solving Logistic Regression *[12 points]*

The cross-entropy loss function for a logistic regression task on a dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ can be written as

$$\mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{n} \left(y_i \mathbf{w}^\top \mathbf{x}_i - \log\left(1 + \exp\left(\mathbf{w}^\top \mathbf{x}_i\right)\right)\right). \tag{3}$$

In this question, we will show that finding the MLE (maximum likelihood estimator) is equivalent to minimizing the cross-entropy loss in Eq. (3). However, unlike linear regression, it is difficult to derive a closed-form solution for logistic regression.

a. *[3 points]* Starting from the definition of negative log-likelihood (Eq. (4) below), show that it is equal to the cross-entropy loss (Eq. (3)).

$$-\ell(\mathbf{w}) = -\log\left(\prod_{i=1}^{n}\left(\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{y_i}\left(1-\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{1-y_i}\right) \tag{4}$$

b. *[4 points]* Since it is difficult to directly optimize Eq. (3), gradient descent algorithms are usually used to find the optimum, as is discussed in the lecture. Show that the negative log-likelihood is a convex function. You may use the facts that the sum of convex functions is also convex, and that if f and g are both convex, twice differentiable and g is non-decreasing, then $g(f)$ is convex.

[Hint]: A function f is convex if for any $x_1, x_2 \in \text{Domain}(f)$,

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2), \forall t \in [0,1] \tag{5}$$

Also, a twice-differentiable function is convex if and only if the Hessian is positive semi-definite.

We now turn our attention to another algorithm known as *iterative weighted least squares*, which is based on the Newton-Raphson algorithm. Let $\mathbf{w}_k$ denote the parameter vector $\mathbf{w}$ at the k^{th} iteration. Then, the update rule of iterative weighted least-squares is as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \left(\nabla^2\mathcal{L}(\mathbf{w}_k)\right)^{-1}\nabla\mathcal{L}(\mathbf{w}_k), \tag{6}$$

where $\nabla\mathcal{L}(\mathbf{w}), \nabla^2\mathcal{L}(\mathbf{w})$ are the gradient and Hessian of the negative log-likelihood in Eq. (3) respectively. Consider the following notations:

- $\mathbf{y}$ is an $N \times 1$ column vector with i^{th} element being the label y_i,
- $\mathbf{X}$ is an $N \times d$ matrix with $\mathbf{x}_i^\top$ being the i^{th} row,
- $\mathbf{W}_k$ is an $N \times N$ diagonal matrix with i^{th} diagonal element being $\frac{\exp\left(-\mathbf{w}_k^\top\mathbf{x}_i\right)}{\left(1+\exp\left(-\mathbf{w}_k^\top\mathbf{x}_i\right)\right)^2}$,
- $\mathbf{p}_k$ is an $N \times 1$ column vector with i^{th} element being $\frac{1}{1+\exp\left(-\mathbf{w}_k^\top\mathbf{x}_i\right)}$
- $\mathbf{z}_k = \mathbf{X}\mathbf{w}_k + \mathbf{W}_k^{-1}(\mathbf{y} - \mathbf{p}_k)$

It can be shown that:

- $\nabla\mathcal{L}(\mathbf{w}_k) = -\mathbf{X}^\top(\mathbf{y} - \mathbf{p}_k)$
- $\nabla^2\mathcal{L}(\mathbf{w}_k) = \mathbf{X}^\top\mathbf{W}_k\mathbf{X}$

c. *[5 points]* Using the above, prove that

$$\mathbf{w}_{k+1} = \left(\mathbf{X}^\top\mathbf{W}_k\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{W}_k\mathbf{z}_k, \tag{7}$$

Note: The name of the algorithm comes from the observation that the algorithm at each step is solving the weighted least squares problem,

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}_k}\left(\mathbf{z}_k - \mathbf{X}\mathbf{w}_k\right)^\top\mathbf{W}_k\left(\mathbf{z}_k - \mathbf{X}\mathbf{w}_k\right). \tag{8}$$

Solution:

a.

$$
\begin{aligned}
-\ell(\mathbf{w}) &= -\log\left(\prod_{i=1}^{n}\left(\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{y_i}\left(1-\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{1-y_i}\right)\\
&= -\sum_{i=1}^{n}\left(\log\left(\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{y_i}+\log\left(1-\frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x}_i)}\right)^{1-y_i}\right)\\
&= -\sum_{i=1}^{n}\left(y_i\log\left(\frac{\exp(\mathbf{w}^\top\mathbf{x}_i)}{1+\exp(\mathbf{w}^\top\mathbf{x}_i)}\right)+(1-y_i)\log\left(\frac{1}{1+\exp(\mathbf{w}^\top\mathbf{x}_i)}\right)\right)\\
&= -\sum_{i=1}^{n}\left(y_i\mathbf{w}^\top\mathbf{x}_i-y_i\log(1+\exp(\mathbf{w}^\top\mathbf{x}_i))-(1-y_i)\log(1+\exp(\mathbf{w}^\top\mathbf{x}_i))\right)\\
&= -\sum_{i=1}^{n}\left(y_i\mathbf{w}^\top\mathbf{x}_i-\log(1+\exp(\mathbf{w}^\top\mathbf{x}_i))\right)=\mathcal{L}(\mathbf{w})
\end{aligned}
$$

b. We will first show that $\log(1+e^x)$ is a convex function.

$$
\begin{aligned}
\frac{d(\log(1+e^x))}{dx} &= \frac{e^x}{1+e^x}=\frac{1}{1+e^{-x}}\\
\frac{d^2(\log(1+e^x))}{dx^2} &= \frac{e^{-x}}{(1+e^{-x})^2}\geq 0.
\end{aligned}
$$

The negative log-likelihood is given by

$$
\mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{n}\left[y_i\mathbf{w}^\top\mathbf{x}_i-\log\left(1+\exp\left(\mathbf{w}^\top\mathbf{x}_i\right)\right)\right]
$$

Observe that linear functions of the form $\mathbf{w}^\top\mathbf{x}_i$ are both convex and concave. Assume, $g(z)=\log(1+e^z)$ and $f(\mathbf{x})=\mathbf{w}^\top\mathbf{x}_i$. Since, g is convex and non-decreasing and f is convex, the function $\log\left(1+\exp\left(\mathbf{w}^\top\mathbf{x}_i\right)\right)$ is convex. Hence, the $\mathcal{L}(w)$ is of the form

$$
\begin{aligned}
\mathcal{L}(w) &= -\sum_{i=1}^{n}[linear-convex]\\
&= \sum_{i=1}^{n}[linear+convex]\\
&= \sum_{i=1}^{n}[convex+convex]\\
&= \sum_{i=1}^{n}convex\\
&= convex
\end{aligned}
$$

c.

$$\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k - \left(\nabla^2 \mathcal{L}(\mathbf{w}_k)\right)^{-1} \nabla \mathcal{L}(\mathbf{w}_k) \\
&= \mathbf{w}_k + \left(\mathbf{X}^\top \mathbf{W}_k \mathbf{X}\right)^{-1} \mathbf{X}^\top (\mathbf{y} - \mathbf{p}_k) \\
&= \left(\mathbf{X}^\top \mathbf{W}_k \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{W}_k \left(\mathbf{X}\mathbf{w}_k + \mathbf{W}_k^{-1}(\mathbf{y} - \mathbf{p}_k)\right) \\
&= \left(\mathbf{X}^\top \mathbf{W}_k \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{W}_k \mathbf{z}_k
\end{aligned}$$

5 SVMs: Hinge loss and mistake bounds *[12 points]*

Suppose we build a predictive model $\hat{y} = \text{sgn}(\mathbf{w}^\top \mathbf{x})$, where $\text{sgn}(z) = 1$ if z is positive and -1 otherwise. Here we are dealing with binary predictions where each label is in $\{-1, 1\}$. The classification loss counts the number of mistakes (i.e., points where $y \neq \text{sgn}(\mathbf{w}^\top \mathbf{x})$) that we make with $\mathbf{w}$ on a dataset. Then we train the SVM model on N data points $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$. The SVM loss function can be viewed as a relaxation to the classification loss. The *hinge loss* on a data point $(\mathbf{x}, y)$ is defined as:

$$\ell((\mathbf{x}, y), \mathbf{w}) = \max\left[0, 1 - y\mathbf{w}^\top \mathbf{x}\right], \tag{9}$$

where $\mathbf{x} \in \mathbb{R}^D$ and $y \in \{-1, 1\}$. The SVM attempts to minimize:

$$L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \ell((\mathbf{x}_i, y_i), \mathbf{w}) + \lambda \|\mathbf{w}\|_2^2, \tag{10}$$

where $\lambda > 0$ is the regularization parameter.

a. *[4 points]* Suppose that for some $\mathbf{w}$ we have a correct prediction of y_i with $\mathbf{x}_i$, i.e. $y_i = \text{sgn}(\mathbf{w}^\top \mathbf{x}_i)$. What range of values can the hinge loss, $\ell((\mathbf{x}, y), \mathbf{w})$, take on this correctly classified example? Points that are classified correctly and which have non-zero hinge loss are referred to as *margin mistakes*. Also, what is the possible range of the hinge loss for an incorrectly classified example?

b. *[5 points]* Let $M(\mathbf{w})$ be the number of mistakes made when we use the weight vector $\mathbf{w}$ to classify our dataset (i.e., the number of training data points for which $y_i \neq \text{sgn}(\mathbf{w}^\top \mathbf{x})$). Note that $\mathbf{w}$ can be an arbitrary weight vector (not necessarily the one that solves the SVM optimization problem). Show that:

$$\frac{1}{N} M(\mathbf{w}) \leq \frac{1}{N} \sum_{i=1}^{N} \max\left[0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i\right]. \tag{11}$$

In other words, the average hinge loss on our dataset is an upper bound on the average number of mistakes we make on our dataset. By minimizing the average hinge loss with $\mathbf{w}$, we can ensure that the SVM model makes few mistakes.

c. *[3 points]* Assume that the training data is separable. However, it turns out that the minimizer $\mathbf{w}^\star$ of the objective in Eq. (10) mis-classifies some training samples. How should we adjust λ (i.e., make it larger or smaller) to make the SVM work properly on training data? Why? Explain in 1-2 sentences.

Solution:

a. For the prediction to be correct, either $y_i = 1$ and $\text{sgn}(\mathbf{w}^T\mathbf{x}) = 1$ or $y_i = -1$ and $\text{sgn}(\mathbf{w}^T\mathbf{x}) = -1$. In either case, the loss function evaluates to $\ell((\mathbf{x}, y), \mathbf{w}) = \max\{0, 1 - |\mathbf{w}^T\mathbf{x}|\}$, which only be in the range $[0, 1]$.

The answer $[0, 1)$ is also acceptable as the hinge loss equals 1 when $\mathbf{w}^T x = 0$, i.e., it is on the classification boundary, where the classification is ambiguous.

Also, for an incorrectly classified example, either $y_i = 1$ and $\text{sgn}(\mathbf{w}^T\mathbf{x}) = -1$ or $y_i = -1$ and $\text{sgn}(\mathbf{w}^T\mathbf{x}) = 1$. In either case, the loss function evaluates to $\ell((\mathbf{x}, y), \mathbf{w}) = \max\{0, 1 - |\mathbf{w}^T\mathbf{x}|\}$, which only be in the range $[1, +\infty)$. Again the answer $(1, +\infty)$ is also acceptable.

b. The range of hinge loss that takes on the misclassfied example is $[1, +\infty]$,

Let M denote the set of misclassified examples and C denote the set of correctly classified examples. Then we have

$$\max[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i] \in [1, +\infty], i \in M,$$
$$\max[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i] \in [0, 1), i \in C.$$

Then

$$\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}\max\left[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right] \\
&= \frac{1}{N}(\sum_{i\in M}\max\left[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right] + \sum_{i\in C}\max\left[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right]) \\
&\geq \frac{1}{N}(\sum_{i\in M}\max\left[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right]) \\
&\geq \frac{1}{N}(\sum_{i\in M} 1) \\
&= \frac{1}{N}M(\mathbf{w}).
\end{aligned}$$

c. Make λ smaller. As we know that the dataset is linearly separable, a low-bias solution (with lower λ) should classify all N samples correctly.

6 Support Vector Machine - Slack Variables & Duality Intuition

[10 points]

In hard SVM, our aim is to maximize the margin when samples are linearly separable. The margin is defined as the distance of the closest point to the classification boundary. In lecture, we showed that this is equivalent to solving the following hard-SVM optimization problem:

$$\min_{\boldsymbol{w}, b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1, \quad \forall \quad i \in [n] \tag{12}$$

where $\boldsymbol{w}, \mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ and n denotes the number of data points. Here each $(\mathbf{x}_i, y_i)$ pair denotes a training data point.

a. *[1 points]* Explain in one or two sentences why the hard-SVM formulation given above may fail when applied to real-world datasets.

To address this issue, we use the soft-SVM formulation by introducing slack variables ξ_i for each data point $i = 1, 2, \ldots, n$:

$$\min_{\boldsymbol{w}, b, \xi_1, \ldots, \xi_n} \left\{ \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{i=1}^{n} \xi_i \right\}$$
$$\text{s.t.} \quad \forall i, \; y_i(\boldsymbol{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

b. *[4 points]* Discuss how the introduction of slack variables, ξ_i, resolves the impracticalities of the hard-SVM formulation that you identified in part (a). Why do we use "$\geq 1 - \xi_i$" instead of "≥ 1" in the constraints and why is it necessary to include "$C \sum_i \xi_i$" in the minimization objective? Your answer should be no more than three or four sentences long.

c. *[4 points]* Further, as we see in the class, the dual formulation of the soft-SVM problem can be preferred.

Primal	Dual
$\min_{\boldsymbol{w}, b, \xi_1, \ldots, \xi_n} \left\{ \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{i=1}^{n} \xi_i \right\}$ s.t. $\forall i, \; y_i(\boldsymbol{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$	$\max_{\boldsymbol{\alpha}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right\}$ s.t. $\forall i, \; 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$

where the dual variables are represented by α_i and $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$.

Find the number of variables to be optimized in the primal and dual forms of SVM. Then, suppose that we decide to use some other kernels (e.g., radial basis kernel, polynomial kernel) instead of the linear kernel ($k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$). Does this change the number of dual variables we need to use?

d. *[1 points]* Why might you prefer to solve the SVM in the dual formulation as opposed to the primal formulation?

Solution:

a. The real world data is generally not linearly separable. There is no solution to the minimization problem of hard-SVM satisfying all constraints.

b. Slack variables relax the constraint in case our training data is not separable. Using $\geq 1 - \xi_i$ instead of ≥ 1 in the constraints lets us have wrong classified data samples. We also need a penalty depending on how much incorrectly classified those samples, therefore, $C \sum_i \xi_i$ term appears in the objectives.

c. Primal: $\mathbf{w} \to d$, $b \to 1$, $\xi \to n$. Dual: $\boldsymbol{\alpha} \to n$. The kernel function does not affect the number of variables to optimize in the dual form.

d. The dual has fewer variables to optimize.

7 Implementing SVMs *[15 points]*

In this problem, you will experiment with SVMs on a real-world dataset. You will implement a linear SVM (i.e., an SVM using the original features). You should implement it from scratch without using any libraries

like scikit-learn, although you can use existing packages to solve quadratic programs. If you are not sure if the library you want to use is allowed, please ask the TAs or post a question on Piazza. **We provide you with a guided Python Notebook (hw2_svm_imp.ipynb) on which you will find parts you need to fill in. You can use this notebook on JupyterLab / JupyterNotebook / GoogleColab.**

Please append your notebook file as a PDF at the end of your submission so that your code and cell outputs can be seen on the same submission PDF. If you use JupyterLab / JupyterNotebook / GoogleColab, you can save the notebook as PDF from File→Print. You can also use online '*ipynb to pdf file converter*s. Please do not forget to append your notebook as a PDF to the end of your submission PDF for us to grade this question.

Dataset: We have provided the *Heart Disease* dataset from UCI's machine learning data repository.[1] The provided binary classification dataset has 13 input features, and 303 samples.

- By running the first cell of the given notebook, you download the `ucimlrepo` package. It will help us download the dataset. For this step, you should have an internet connection.
- In the second cell, we provide all required libraries to complete the homework.
- In the third cell, we download the dataset (again, you need an internet connection), one-hot encode some categorical features, and split it into train/test parts using a set seed.

You should run those three cells without any modification. After running the first three cells, you should have *X_train* (shape of $(216, 22)$), *y_train* (shape of $(216,)$), *X_test* (shape of $(83, 22)$), and *y_test* (shape of $(83,)$) variables.

a. *[3 points]* Let $x_k^{(1)}, \cdots, x_k^{(N)}$ be the values of feature k for all training set (*X_train*) points where N is the number of samples in the training set. Preprocess the training (*X_train*) and test data (*X_test*) by

- Computing the mean $\overline{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_k^{(i)}$ of each feature and subtracting it from all values of this feature.
- Dividing each feature by its standard deviation, defined as

$$s_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_k^{(i)} - \overline{x}_k)^2}$$

for feature k. Here, $\overline{x}_k$ is the sample mean of this feature.

Save the normalized data samples as variables named *X_train_normalized* for the training set and *X_test_normalized* for the test set. Note that the shape of these variables should be the same as the shapes of *X_train* and *X_test*, respectively.

This type of preprocessing is useful for SVMs, as SVMs attempt to maximize the distance between the separating hyperplane and the support vectors. If one feature (i.e., one dimension in this space) has very large values, it will dominate the other features when calculating this distance. Rescaling the features will ensure that they all have the same influence on the distance metric.

Note that the mean and standard deviation should be estimated from the *training data (X_train)* and then applied to both datasets. Explain why this is the case. Also, report the mean and the standard deviation of the first and the last features computed on the training data (*X_train*).

[1] https://archive.ics.uci.edu/dataset/45/heart+disease.

b. *[7 points]* Now, implement the `trainSVM` function. Follow the description given in the cell with title *Train SVM*. The input of `trainSVM` contains normalized training data (*X_train_normalized*) and training labels *y_train*, as well as the parameter C. The output of `trainSVM` contains the optimal weight vector ($\boldsymbol{w}$), bias term (b) and slack variables ($\boldsymbol{\xi}$). In your implementation, solve the SVM in its primal form:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \forall\, 1 \leq i \leq N$$
$$\xi_i \geq 0, \ \forall\, 1 \leq i \leq N$$

To solve the above quadratic problem, we encourage you to use the CVXPY [2] Python package. You can also use any package of your choice. If you are not familiar with CVXPY, they have their own very extensive tutorial/user guide at: https://www.cvxpy.org/tutorial/index.html. After implementing the desired function,

- Solve the problem using `trainSVM` with normalized training set (*X_train_normalized*), training labels (*y_train*) and $C = 1$. What values do you find for the first 3 weights $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$, b, and first 3 slack variables $\{\xi_1, \xi_2, \xi_3\}$?
- Solve the problem using `trainSVM` with normalized training set (*X_train_normalized*), training labels (*y_train*) and $C = 0$. What values do you find for the first 3 weights $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$, b, and first 3 slack variables $\{\xi_1, \xi_2, \xi_3\}$?
- Do you see any difference between the slack variables you found with $C = 0$ and $C = 1$? Does that explain why we are introducing $C\sum_i \xi_i$ term into the objective for Soft-SVM? Explain your answer in 1-2 sentences.

c. *[5 points]* Now, we will write a function for evaluation in the cell titled *Eval SVM*. Write a function `evalSVM` that takes four inputs: *X_eval* -features for evaluation- (we will use either normalized train set features (*X_train_normalized*) or normalized test set features (*X_test_normalized*) to calculate train/test set accuracy, respectively), *y_eval* -true labels of the evaluation data- (this will be again used to input train set labels (*y_train*) or test set labels (*y_test*)), and the trained SVM parameters $\mathbf{w}$ and b.

This function should estimate the classes of all samples in the given *X_eval* for evaluation using an SVM model with parameters $\mathbf{w}$ and b. Then, the function should compare the estimated labels with *y_eval* (the true labels), and return the accuracy of the SVM model on this data, which is the fraction of data points that are correctly classified.

Now, for all possible $C = a \times 10^q$ values where $a \in \{1, 3, 6\}$ and $q \in \{-4, -3, -2, -1, 0, 1\}$, find the optimal values of $\mathbf{w}$ and b using `trainSVM` with training labels (*y_train*) and normalized training set (*X_train_normalized*) and Then, using your `evalSVM` function, calculate the accuracy over the normalized training set (*X_train_normalized*) and normalized test set (*X_test_normalized*).

Make two plots of your results, one showing the training accuracy vs. the value of C and one showing the test accuracy vs. the value of C. Use a log-scale[3] on the x-axis for both plots. Also, report the values of C that maximise the training and test accuracies, along with the accuracies that are reached with these values of C.

7.1 Deliverables

- Normalisation of `X_train` and `X_test` *[3 points]*

[2] https://www.cvxpy.org/index.html
[3] You can use *log* option of https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xscale.html

- Correct implementation of data normalisation on `X_train` and `X_test`
- Explanation of why the mean and standard deviation used for normalising both training and test datasets are estimated from the training data.
- Report of the correct mean and standard deviation of the first and last features of the training data.

- SVM training *[7 points]*
  - Correct implementation of the `trainSVM` function.
  - Report of the first three weights, bias, and first three slack variables for both $C = 1$ and $C = 0$.
  - Identification of the difference between the slack variables when $C = 1$ and $C = 0$. And, an explanation of the connection between this difference and the $C \sum_i \xi_i$ term in the Soft-SVM objective.

- SVM evaluation *[5 points]*
  - Correct implementation of the `evalSVM` function.
  - Plot of training accuracy against the instructed values of C.
  - Plot of test accuracy against the instructed values of C.
  - Report of the value of C that maximises training accuracy and the corresponding training accuracy
  - Report of the value of C that maximises test accuracy and the corresponding test accuracy

Solution:

a. The first feature mean: 54.99074, standard deviation: 9.07784

The last feature mean: 0.50462, standard deviation: 0.49997

We apply the same transformation for a matter of consistency of the distribution/model being learned. More precisely, by applying a transformation on the training data—subtracting the mean of the training data and scaling by its standard deviation—we are working with a different data distribution. Therefore, since we originally assume the train and test data have the same distribution, we must apply the same transformation to the test data.

b. With C = 1, first three weights: [-0.01280 0.51706 0.27813], b: [0.08109], first three slack variables: [-1.70119e-10 -1.64395e-10 -1.69587e-10] ([0 0 0] is also accepted since negative slack variables is just small numerical error of the package. Also, these first three slack variables may differ depending on the solver used, however, they are always small numbers close to zero.)

With C = 0, first three weights: [3.09523e-06 -8.18802e-06 -9.46615e-06], b: [-10.44762], first three slack variables: [429.58840 434.02071 414.34670]. Note, that these values differ depending on the solver used. The values given above are for the recommended and default solver, `cp.CLARABEL`.

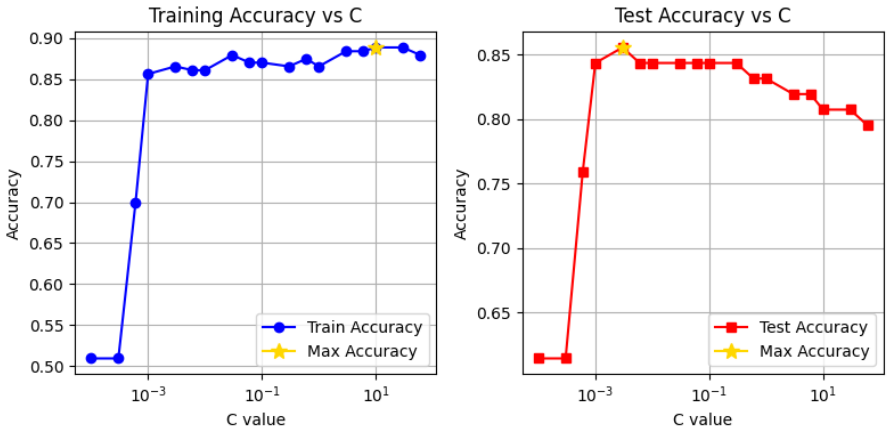
When we use $C = 0$, the slack variables are very high because we are not penalizing them for having a high value. Therefore, the objective can be zero (which is the minimal possible objective) using large slack variables such that the constraints are satisfied.

c. The maximum training accuracy is 0.8889 when C is 10.000 for training accuracy.

The maximum test accuracy is 0.8554 when C is 0.003 for test accuracy.

Plots can be found below.

Solution:



8 Non-linear Basis Functions *[20 points]*

8.1 Derivation *[6 points]*

In class, we have learned that by using non-linear basis functions, we can fit a non-linear model (w.r.t. to the data) while preserving linearity w.r.t. the model parameters. In this problem, we are given a training data set $\mathcal{D}$ that consists of N points, (x_i, y_i) for $i = 1, \cdots, N$, and we consider a regression problem as follows:

$$y_i = w_0 + w_1 \sin(x_i) + w_2 \cos(x_i) + w_3 \sin(2x_i) + w_4 \cos(2x_i) + \cdots + w_{2k-1} \sin(kx_i) + w_{2k} \cos(kx_i) \quad (13)$$

Here x_i and y_i are scalar real numbers.

1. *[2 points]* What is the basis function $\boldsymbol{\phi}(x)$ for this problem?

Solution: $\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ \sin(x) \\ \vdots \\ \cos(kx) \end{bmatrix}$

2. *[2 points]* Express the residual sum of squares error in terms of $\{\boldsymbol{\phi}(x_i); i = 1, 2, \ldots, N\}$.

Solution: $RSS = \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i))^2$

3. *[2 points]* What parameter values $\boldsymbol{w} = \begin{bmatrix} w_0, & w_1, & \cdots, & w_{2k} \end{bmatrix}^\top$ minimize the residual sum of squares error?

Solution: $\boldsymbol{w} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$, where $\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}(x_1) & \boldsymbol{\phi}(x_2) & \cdots & \boldsymbol{\phi}(x_N) \end{bmatrix}^\top$

8.2 Implementation

Now we want to implement sinusoidal regression on the $N = 61$ data points given to you in `nonlinear-regression-data.csv`. In `nonlinear-regression.py`, implement the following functions:

- `phi(x)`: which returns the basis function you got in the derivation part.
- `fit(X_train, Y_train, k)`: which fits the nonlinear model given the the degree `k`.
- `predict(X)`: which predicts the values of the dependent variable for a new set of covariates using the learned model.
- `rmse(X_val, Y_val)`: which computes the RMSE error on the validation set using the learned model.

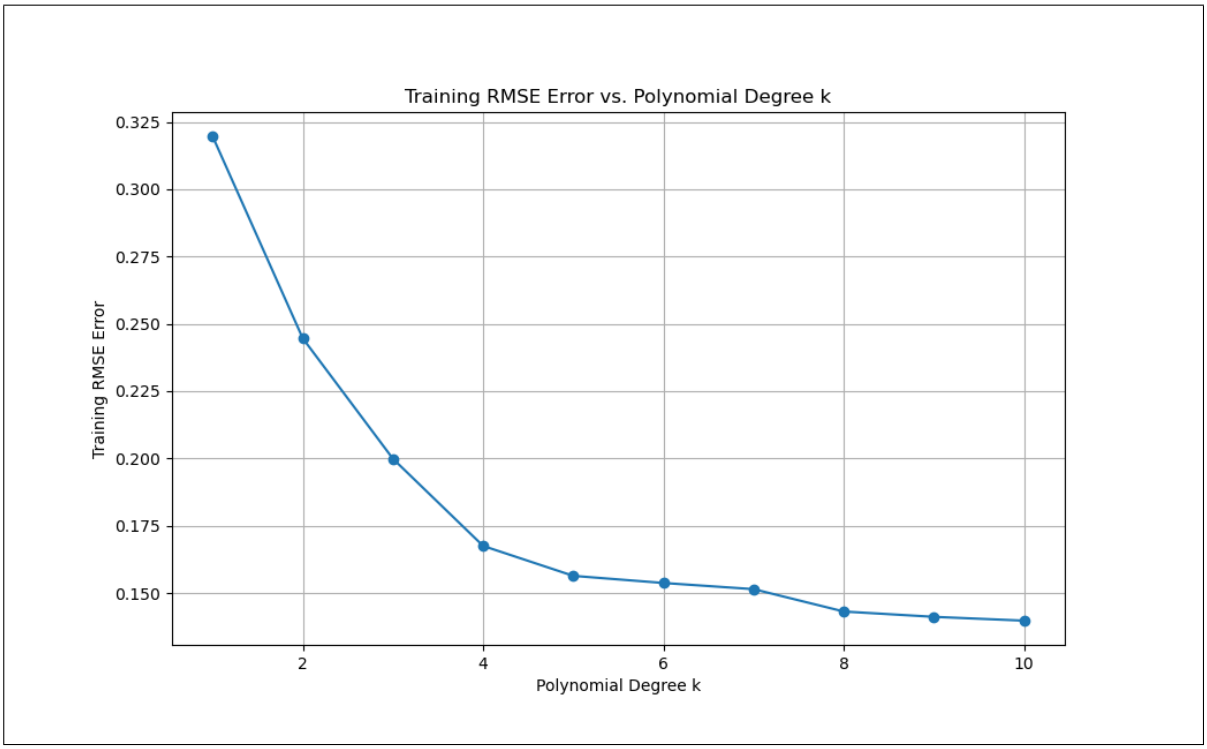
Solution: See nonlinear-regression-solution.py

8.3 Evaluation *[14 points]*

Now, randomly divide the dataset into two parts: training set (45 points) and validation set (16 points). By varying k from 1 to 10, obtain the RMSE error on the validation set and the training set.

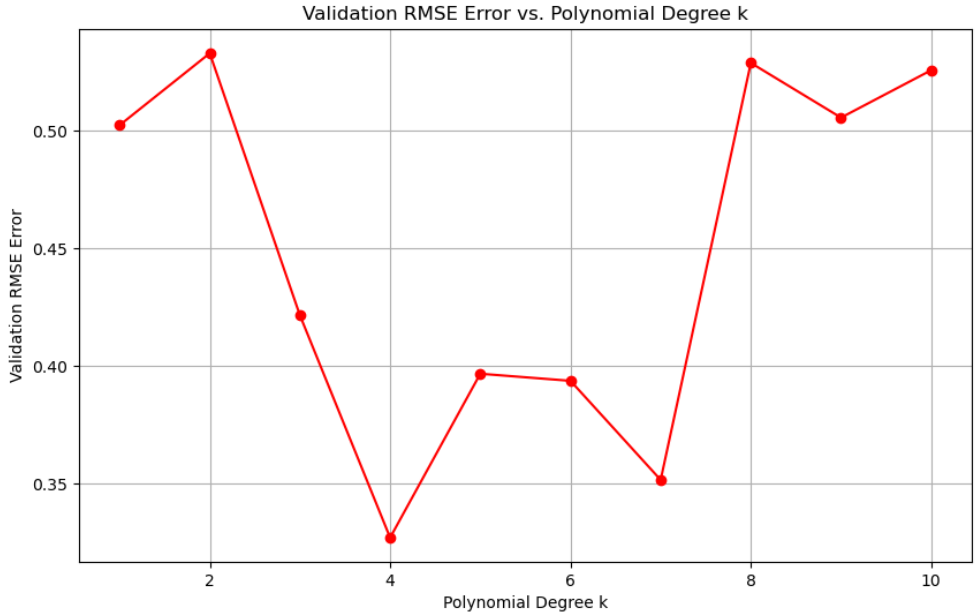
1. *[4 points]* Plot the training error (RMSE error on the training set) versus k. From the plot, which value of k gives you the minimum training error?

Solution: $k = 10$.



2. *[4 points]* Plot the validation error versus k. Which value of k gives you the minimum validation error?

Solution: $k = 4$.

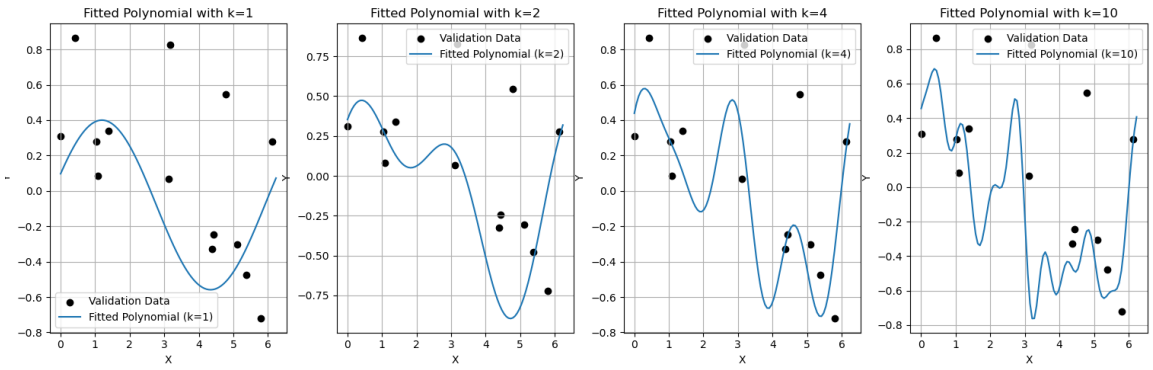


3. *[2 points]* Compare the optimal k values that you found for the training error and validation error. Explain why they are different or the same.

Solution: It is because to minimize the training error, you can always choose a more complex model, *i.e.,* higher values of k. On the other hand, to minimize validation error, we should choose a model that generalizes to unseen data.

4. *[4 points]* For each $k = 1, 3, 5, 10$, create a scatter plot of the validation data points. On the same plot, for each k, draw a line for the fitted polynomial.

Solution:



Please do not forget to append your Python code and notebook to the end of your submission for us to grade this question.