Student : Ya Zou

CS434 Machine Learning and Data Mining – Homework 3

Naïve Bayes and Neural Networks

▶ Q1 Prove Bernoulli Naïve Bayes has a linear decision boundary [4pts]. Prove the Bernoulli Naïve Bayes model described above has a linear decision boundary. Specifically, you'll need to show that class 1 will be predicted only if $b + \mathbf{w}^T\mathbf{x} > 0$ for some parameters b and $\mathbf{w}$. To do so, show that:

$$\frac{P(y=1|x_1,...,x_d)}{P(y=0|x_1,...,x_d)} > 1 \implies b + \sum_{i=1}^{d} w_i x_i > 0 \tag{4}$$

As part of your report, explicitly write expressions for the bias b and each weight w_i.

Hints: Expand Eq. (3) by substituting the posterior expressions from Eq. (1) & (2) into Eq. (3). Take the log of that expression and combine like-terms.

Eq(1): $$P(y=1|x_1,...,x_d) = \frac{P(y=1)\prod_{i=1}^{d} P(x_i|y=1)}{P(x_1,...,x_d)} \propto \theta_1 \prod_{i=1}^{d} \theta_{i1}^{x_i}(1-\theta_{i1})^{1-x_i}$$

Eq(2) $$P(y=0|x_1,...,x_d) = \frac{P(y=0)\prod_{i=1}^{d} P(x_i|y=0)}{P(x_1,...,x_d)} \propto \theta_0 \prod_{i=1}^{d} \theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}$$

Lets substitut eq(1) & eq(2) into eq(3)

$$\frac{\theta_1 \prod_{i=1}^{d} \theta_{i1}^{x_i}(1-\theta_{i1})^{1-x_i}}{\theta_0 \prod_{i=1}^{d} \theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}} > 1$$

lets take log on both sides like we did in previous assignment :

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0} * \prod_{i=1}^{d} \frac{\theta_{i1}^{x_i}}{\theta_{i0}^{x_i}} * \frac{(1-\theta_{i1})^{1-x_i}}{(1-\theta_{i0})^{1-x_i}}\right) > \log(1)$$

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \left(\sum_{i=1}^{d} \log\left(\frac{\theta_{i1}^{x_i}}{\theta_{i0}^{x_i}} * \frac{(1-\theta_{i1})^{1-x_i}}{(1-\theta_{i0})^{1-x_i}}\right)\right) > 0$$

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \left(\sum_{i=1}^{d} \log\left(\frac{\theta_{i1}}{\theta_{i0}}\right)^{x_i} + \log\left(\frac{(1-\theta_{i1})}{(1-\theta_{i0})}\right)^{1-x_i}\right) > 0$$

$$= \log\left(\frac{\theta_1}{\theta_0}\right) + \left(\sum_{i=1}^{d} x_i \log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) + (1-x_i)\log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right) > 0$$

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \left(\sum_{i=1}^{d} x_i \log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) + \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right) - x_i \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right) > 0$$

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{d}\left(x_i\left(\log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) - \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right) + \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right) > 0$$

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{d}\left[x_i\left(\log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) - \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right)\right] + \sum_{i=1}^{d} \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right) > 0$$

From here we can tell the term $\sum_{i=1}^{d} \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)$ is independent on x, thus we can rerange the equation to the following:

$$\Rightarrow \log\left(\frac{\theta_1}{\theta_0}\right) + \sum_{i=1}^{d}\left(\log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right) + \sum_{i=1}^{d}\left[x_i\left(\log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) - \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right)\right] > 0$$

$\Rightarrow$ The given equation can be written as $\frac{P(y=1 \mid x_1 \cdots x_d)}{P(y=0 \mid x_1 \cdots x_d)} > 1 \Rightarrow b + \sum_{i=1}^{d} w_i x_i > 0$, thus we know: $b = \log\left(\frac{\theta_1}{\theta_0}\right) + \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)$

$$w_i = \left(\log\left(\frac{\theta_{i1}}{\theta_{i0}}\right) - \log\left(\frac{1-\theta_{i1}}{1-\theta_{i0}}\right)\right)$$

$w_i =$ (10

▶ [2] Duplicate Features in Naive Bayes [1pts]. Consider a Naive Bayes model with a single feature X_1 for a binary classification problem. Assume a uniform prior such that $P(y=0) = P(y=1)$. Suppose the model predicts class 1 for an example x then we know:

$$P(y=1 | X_1 = x_1) > P(y=0 | X_1 = x_1) \quad (5)$$

Now suppose we make a mistake and duplicate the X_1 feature in our data – now we have two identical inputs X_1 and X_2. Show that the predicted probability for class 1 is higher than it was before; that is, prove:

$$P(y=1 | X_1 = x_1, X_2 = x_1) > P(y=1 | X_1 = x_1) \quad (6)$$

Hints: Use the assumption that $P(y=0) = P(y=1)$ to transform the posterior expressions in Eq. (6) and Eq. (5). As X_2 is an exact copy of X_1, $P(X_2 = x_1 | y)$ is the same as $P(X_1 = x_1 | y)$ for any y.

Let's apply Bayes theorem on Eq.(5):

$\Rightarrow P(y=1 | X_1 = x_1) > P(y=0 | X_1 = x_1)$ ——— (5')

$\Rightarrow \frac{P(X_1 = x_1 | y=1) P(y=1)}{P(x_1 = x_1 | y=1) P(y=1) + P(X_1 = x_1 | y=0) P(y=0)} > \frac{P(x_1 = x_1 | y=0) P(y=0)}{P(x_1 = x_1 | y=1) P(y=1) + P(x_1 = x_1 | y=0) P(y=0)}$

from equation we know the denominators are the same, so we can get $P(X_1 = x_1 | y=1) > P(X_1 = x_1 | y=0)$

$$\frac{P(x_1 | y=1) P(y=1)}{P} > \frac{P(x_1 | y=0) P(y=0)}{P}$$

D) $P(x_1 | y=1) > P(x_1 | y=0)$ (C+A)

E) $P(x_2 | y=1) > P(x_2 | y=0)$ (D+B)

▶ Q2 Duplicate Features in Naïve Bayes [1pts]. Consider a Naïve Bayes model with a single feature X_1 for a binary classification problem. Assume a uniform prior such that $P(y = 0) = P(y = 1)$. Suppose the model predicts class 1 for an example $\mathbf{x}$ then we know:

$$P(y = 1|X_1 = x_1) > P(y = 0|X_1 = x_1) \tag{5}$$

Now suppose we make a mistake and duplicate the X_1 feature in our data – now we have two identical inputs X_1 and X_2. Show that the predicted probability for class 1 is higher than it was before; that is, prove:

$$P(y = 1|X_1 = x_1, X_2 = x_2) > P(y = 1|X_1 = x_1) \tag{6}$$

Hints: Use the assumption that $P(y = 0) = P(y = 1)$ to simplify the posterior expressions in Eq. (6) and Eq. (5). As X_2 is an exact copy of X_1, $P(X_2 = x_2|y)$ is the same as $P(X_1 = x_1|y)$ for any example.

Given:

① $P(y=0) = P(y=1)$

② $X_1 \equiv X_2$

③ $P(y_1=1 | X_1=x_1) > P(y=0 | X_1=x_1)$ (5)

$$\Rightarrow \frac{P(X_1=x_1|y=1)\,P(y_1=1)}{P(X_1=x_1|y=1)P(y=1)+P(X_1=x_1|y=0)P(y=0)} > \frac{P(X_1=x_1|y=0)\,P(y=0)}{P(X_1=x_1|y=1)P(y=1)+P(X_1=x_1|y=0)P(y=0)}$$

We know $P(y=0) = P(y=1)$ and the denominator of the eq are the same, so we can get: $P(X_1=x_1|y=1) > P(X_1=x_1|y=0)$.

Since we know As X_2 is an exact copy of X_1, $P(X_2=x_2|y)$ is the same as $P(X_1=x_1|y)$ and $P(X_1=x_1|y=1) > P(X_1=x_1|y=0)$

thus we can know:

$$P(X_2=x_2|y=1) > P(X_2=x_2|y=0)$$

Since we need to prove

$$\underbrace{P(y=1|X_1=x_1, X_2=x_2)}_{①} > \underbrace{P(y=1|X_1=x_1)}_{②} \quad \text{Eq(6)}$$

$$P(y=1 \mid X_1=x_1, X_2=x_2) = \frac{P(X_1=x_1 \mid y=1)\,P(X_2=x_2 \mid y=1)\,P(y=1)}{P(x_1 \mid y=1)\,P(x_2 \mid y=1)\,P(y=1) + P(x_1 \mid y=0)\,P(x_2 \mid y=0)\,P(y=0)}$$

① ②

we know that $P(y=1) = P(y=0)$, thus we can get:

$$P(y=1 \mid X_1=x_1, X_2=x_2) = \frac{P(X_1=x_1 \mid y=1)^2}{P(X_1=x_1 \mid y=1)^2 + P(X_1=x_1 \mid y=0)^2}$$

$$= \frac{1}{1+\left(\frac{P(X_1=x_1 \mid y=0)}{P(X_1=x_1 \mid y=1)}\right)^2}$$

according the result we got, we can know that

$$\underbrace{P(y=1 \mid X_1=x_1)}_{②} = \frac{}{1+\left(\frac{P(X_1=x_1 \mid y=0)}{P(X_1=x_1 \mid y=1}\right)}$$

We've already know $P(X_1=x_1 \mid y=1) > P(X_1=x_1 \mid y=0)$

so we know $\frac{P(X_1=x_1 \mid y=0)}{P(X_1=x_1 \mid y=1)} < 1$. then we can tell $\frac{1}{1+\left(\frac{P(X_1=x_1 \mid y=0)}{P(X_1=x_1 \mid y=1)}\right)^2} > \frac{1}{1+\left(\frac{P(X_1=x_1 \mid y=0)}{P(X_1=x_1 \mid y=1}\right)}$

Therefore we proved $P(y=1 \mid X_1=x_1, X_2=x_2) > P(y=1 \mid X_1=x_1)$

► Q4 Learning Rate [2pts]. The learning rate (or step size) in stochastic gradient descent controls how large of a step in the direction of the loss gradient we take our parameters at each iteration. The batch size determines how many data points we use to estimate the gradient. Modify the hyperparameters to run the following experiments:

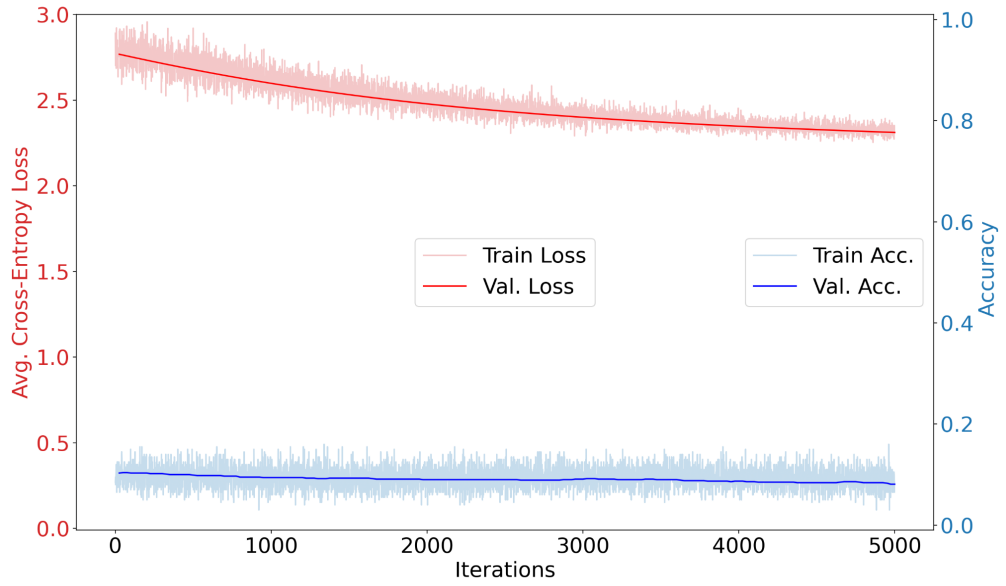
1. Step size of 0.0001 (leave default values for other hyperparameters)
2. Step size of 5 (leave default values for other hyperparameters)
3. Step size of 10 (leave default values for other hyperparameters)

Include these plot in your report and answer the following questions:

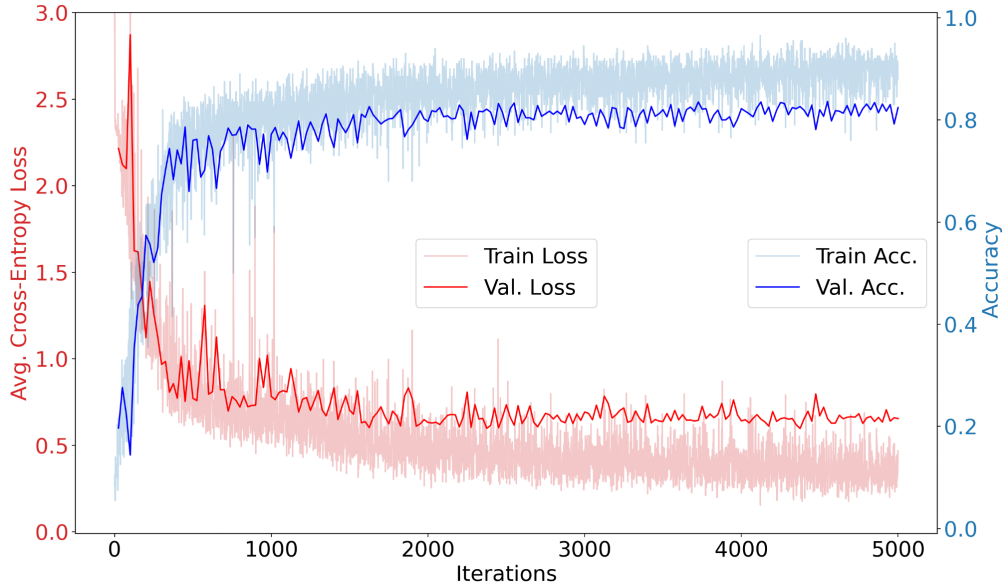
a) Compare and contrast the learning curves with your curve using the default parameters. What do you observe in terms of smoothness, shape, and what performance they reach?

b) For (a), what would you expect to happen if the max epochs were increased?

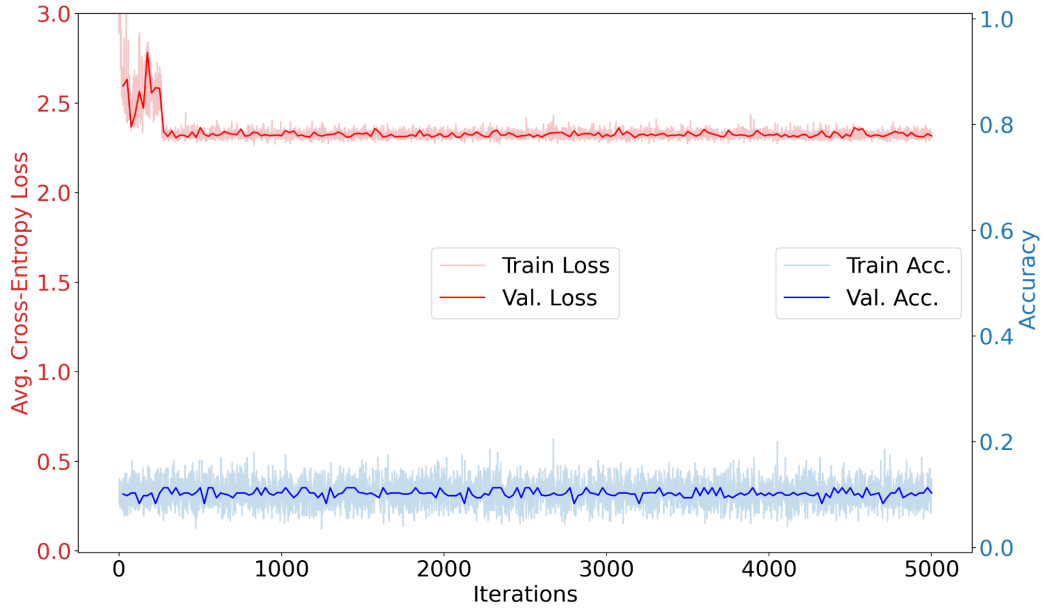
4.1. The step size of 0.0001:



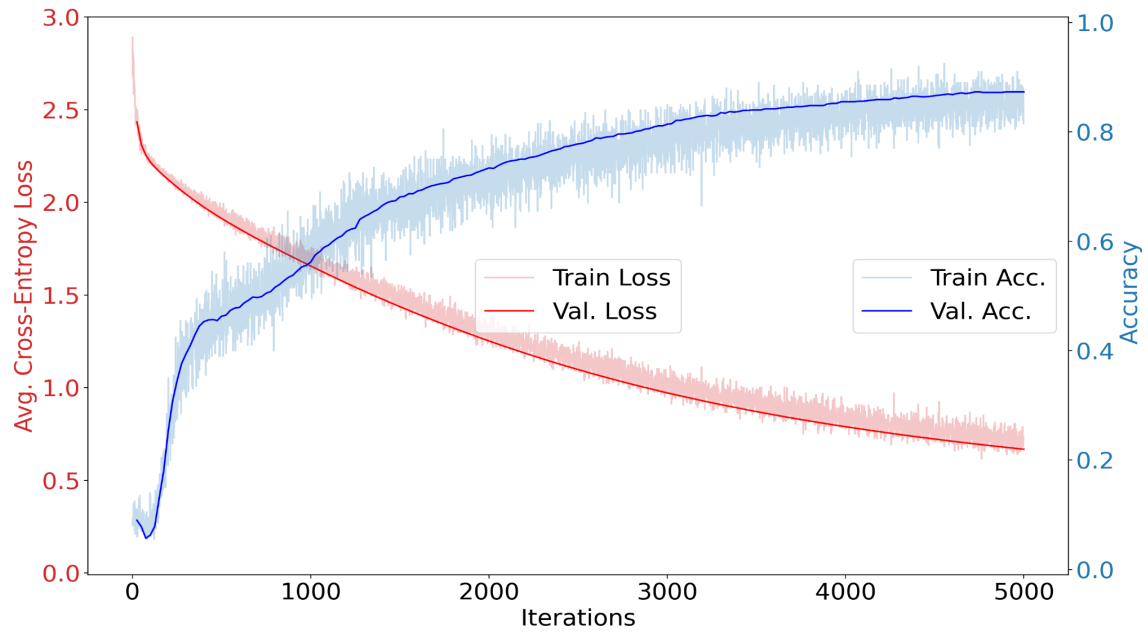
4.2. The step size of 5:



4.3. The step size of 10:



The Default step size = 0.01:



4a:

Description of the graph of default step size = 0.01: The two lines(blue and read) in the graph intersect at 1000 iterations, the red line represents the loss that has a gradually decreasing slope which means that with increasing iterations the average Cross-Entropy Loss is decreasing

towards 0, and the blue line represents the accuracy that has an increasing slope which means that with increasing of iterations the accuracy is increasing towards 1.

Compare the graph of default step size = 0.01 and the graph of step size = 0.0001:The graph for the step size of 0.0001 shows that the two lines (blue and red) do not intersect, unlike the graph for the default step size of 0.01, where they intersect at 1000 interactions. All the lines in the two graphs look very smooth. The blue line represents accuracy and appears to have a “straight-line shape, indicating that, with increasing iterations, the accuracy neither increases nor decreases, remaining at an accuracy value of about 10 %. On the other hand, the red line represents the loss, which starts decreasing from 3.0 to 2.5, and at 2.5 the Avg, Cross-Entropy Loss stops decreasing.

Compare the graph for default step size = 0.01 and the graph of step size = 5: The two lines in the graph for the default step size of 0.01 show a smooth appearance, whereas the two lines in the graph for the step size of 5 appear erratic. The graph with a step size of 0.01 intersects at 4000 iterations, while the lines in the graph with a step size of 5 intersect around 200 iterations. The line representing losses has a negative slope, guiding it toward an average cross-entropy loss of 0 as the number of iterations increases. Meanwhile, the line representing accuracies has an increasing trend, signifying that with more iterations, the accuracy rises towards 100%. The overall shape of the graph closely resembles the default graph with a step size of 0.01; however, the only distinction is that the graph with a step size of 0.01 is notably more stable and less variable compared to the graph with a step size of 5.

Compare the graph for default step size = 0.01 and the graph of step size = 10: The two lines in the graph with a step size of 10 show some jitter, unlike the very smooth lines in the graph with the default step size of 0.01. The graph with the default size of 0.01 intersects at 1000 iterations, while the two lines in the graph with a step size of 10 run parallel to each other, similar to the graph with a step size of 5. The loss line begins with fluctuations in the first 300 iterations and then levels off, forming a nearly flat line at approximately 2.3 average cross-entropy loss. The accuracy line remains constant at 0.1 throughout 5000 iterations.

4b, if the max epoch were increased, I would expect the frequency of weight adjustments to rise. This progression signifies a transformation of the curve from initially underfitting the data, gradually achieving the best fit, and eventually inclining towards overfitting the data. Excessive epochs can lead to overfitting of the training data, implying insufficient learning and poor performance on the test data. Conversely, too few epochs result in the model inadequately fitting the data, leading to underfitting and suboptimal performance on training and test datasets. Identifying the optimal number of epochs is crucial for achieving the most favorable fit for both training and test data, ensuring effective learning without overfitting.

▶ Q5 ReLU's and Vanishing Gradients [3pts]. Modify the hyperparameters to run the following experiments:

1. 5-layer with Sigmoid Activation (leave default values for other hyperparameters)
2. 5-layer with Sigmoid Activation with 0.1 step size (leave default values for other hyperparameters)
3. 5-layer with ReLU Activation (leave default values for other hyperparameters)

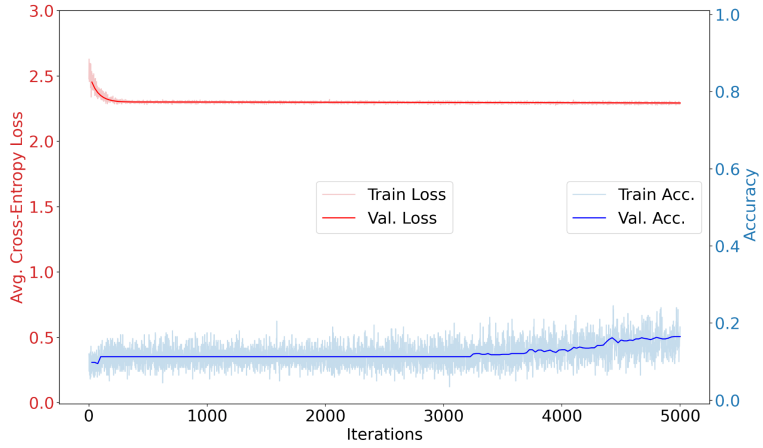
Include these plot in your report and answer the following questions:

a) Compare and contrast the learning curves you observe and the curve for the default parameters in terms of smoothness, shape, and what performance they reach. Do you notice any differences in the relationship between the train and validation curves in each plot?

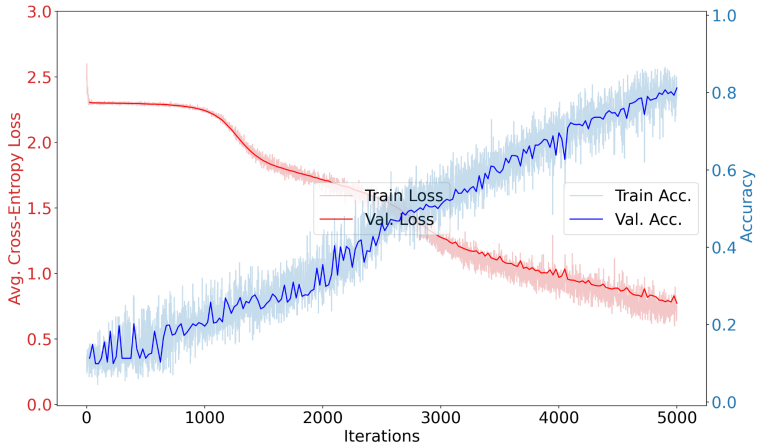
b) If you observed increasing the learning rate in (2) improves over (1), why might that be?

c) If (3) outperformed (1), why might that be? Consider the derivative of the sigmoid and ReLU functions.

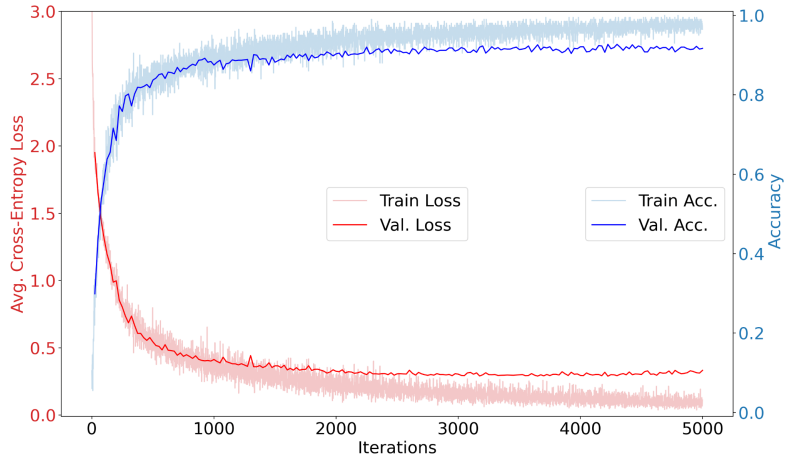
5.1 5-layer with Sigmoid Activation(default value for other hyperparameters)



5.2 5-layer with Sigmoid Activation with 0.1 step size(default value for other hyperparameters)



5.3 5-layer with ReLU activation(default value for other hyperparameters)



5a.

Compare the graph of a 5-layer model with Sigmoid Activation (utilizing default values for other hyperparameters) to the default graph. The accuracy lines, depicted in red and blue (representing losses), show remarkable smoothness, resembling closely the default graph. However, the structural differences between the two graphs are pronounced. In the 5-layer model with Sigmoid Activation, the two lines run parallel, while in the default graph, they intersect at 1000 iterations. The loss line steadily ascends towards a 2.3 Cross-Entropy Loss with increasing iterations. Conversely, the accuracy line shows an increase from a 10% to a 20% accuracy rate. Furthermore, I noticed the training loss disappeared after approximately 100 iterations in the graph of a 5-layer model with Sigmoid Activation.

Compare the graph of a 5-layer model with Sigmoid Activation, utilizing a 0.1 step size and default values for other hyperparameters, to the default graph: The red line (representing loss) shows smoothness from 0 to 3000 iterations but becomes somewhat erratic thereafter. On the other hand, the blue line demonstrates considerable volatility throughout the entire 5000 iterations. In contrast, both lines in the default graph maintain a high degree of smoothness. The shape of the two lines in the graph of the 5-layer model with Sigmoid Activation and a 0.1 step size intersect at approximately 2500 iterations, whereas the two lines in the default graph also intersect but at a different iteration value. The red line in the former decreases with increasing iterations from 0 to 5000, while the blue line increases, reaching an accuracy rate of 80%. Also, I noticed that we barely can see any train loss in the graph with the 5-layer model with Sigmoid Activation with 0.1 step size.

Compare the 5-layer with ReLU activation(default value for other hyperparameters) and the default graph: In the 5-layer with the ReLU activation graph, both red and blue lines are not smooth like the two lines in the default graph. However, two lines intersect in both graphs but at a different iteration value, In the 5-layer with ReLU activation graph, two lines intersect at iterations of approximately 80 iterations. The accuracy of the blue line increases at the

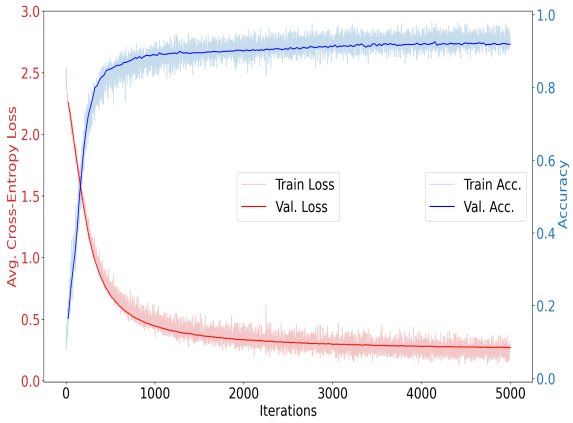
beginning from approximately 500 iterations then the accuracy rate keeps steady at 80% accuracy, but for the red line, it dramatically decreases from 3.0 to 0.4 Avg. Cross-Entropy-Loss with increasing 0 to 1000 iterations, but after 1000 iterations the loss keeps at 0.4 from 1000 iterations to 5000 iterations. Additionally, I observed that in the graph depicting a 5-layer configuration with ReLU activation, at approximately 2000 iterations, both the training loss and training accuracy deviate slightly from the corresponding lines of loss and accuracy.

5b. When we make the step size bigger from 0.01 to 0.1, it helps the learning rate get better. The step size controls how fast our model learns from the data. If we use a smaller step size, it means the model changes very little each time, so we need more rounds of learning. But with a bigger step size, the changes are larger, so we need fewer rounds of learning. We did two experiments, keeping the number of rounds the same. The one with the bigger step size of 0.1 worked better than the one with the smaller step size of 0.01. Think of it like this: the run with the bigger step size is better because, during each round, the weights change more in the right direction. This helps us get closer to the best weights for our model.

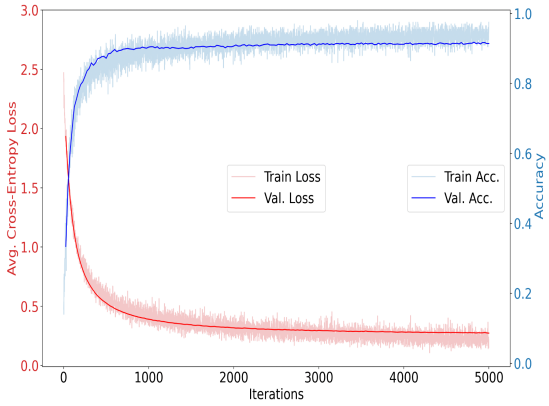
5c.It might be that ReLU is widely considered superior to Sigmoid as an activation function in neural networks, and this preference is largely attributed to the distinctive characteristics of their derivatives. The derivative of the ReLU function is both straightforward and computationally efficient. When dealing with positive input values, the derivative remains constant, ensuring a consistent and substantial gradient during the backpropagation process. In contrast, the Sigmoid function, commonly used in the past, exhibits a derivative that diminishes significantly for extreme input values, giving rise to the vanishing gradient problem. This phenomenon hampers the learning process as the gradients approach zero, resulting in sluggish convergence or even stagnation during training. ReLU's simplicity and its ability to circumvent the vanishing gradient problem make it a preferred choice, particularly in the realm of deep learning networks, where efficient training is paramount for optimal model performance.

> ▶ Q6 Measuring Randomness [1pt]. Using the default hyperparameters, set the random seed to 5 different values and report the validation accuracies you observe after training. What impact does this randomness have on the certainty of your conclusions in the previous questions?

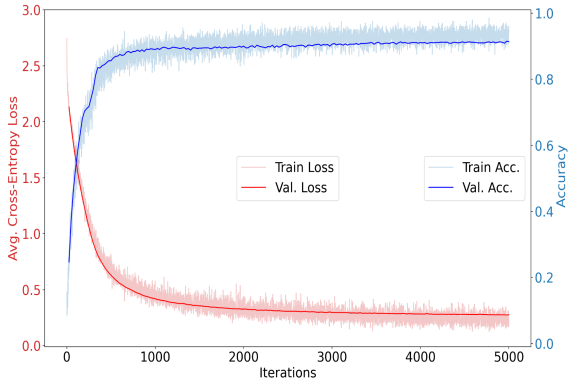
I changed the seed value to five distinct values (1, 35, 75, 205, 500). Based on my observations, the final accuracy rates varied each time(according to the printed validation accuracy rate on the terminal). However, an analysis of the corresponding graphs revealed that the overall accuracy range remained relatively stable, predominantly within the approximate range of 86% to 92%.



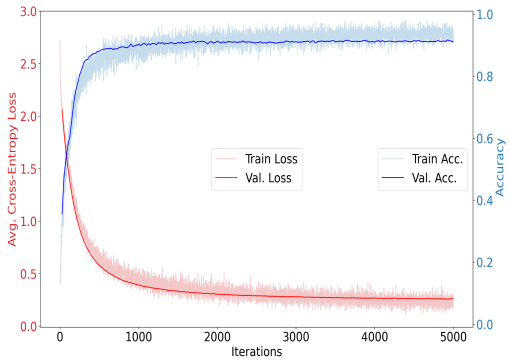
Seed = 1



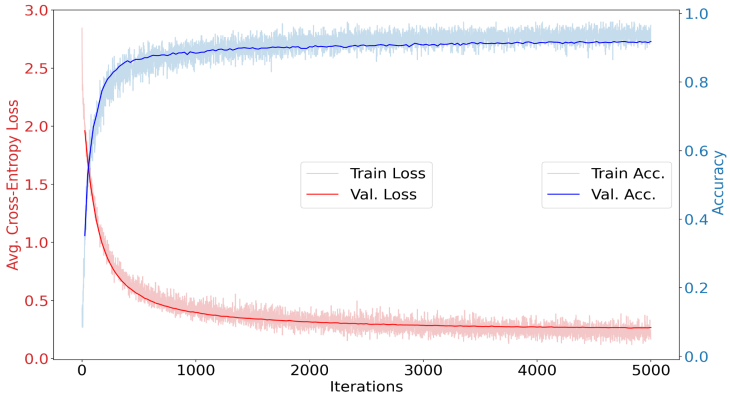
Seed = 35



Seed = 75



Seed = 205



Seed = 500

3 Debriefing (required in your report)

1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did you work on it mostly alone or did you discuss the problems with others?
4. How deeply do you feel you understand the material it covers (0%–100%)?
5. Any other comments?

1. 9 hours
2. Difficult
3. Mostly with TA
4. 80%
5. No