SVM

David Rajchman, Matthew-Blake Foust

December 12, 2025

1 Why SVMs

When given a dataset of two classes that can be binarily classified, we can most easily separate the classes by a hyperplane using a Perceptron and assuming the classes are linearly separable. But we must ask ourselves, what if our data isn't linearly separable? If it is, will we have one exact solution found in a reasonable time? What if we have more than two classes?

This is where the Soft-Margin Support Vector Machine(SVM) answers our machine learning prayers. The problem with the Perceptron algorithm is it will not converge if our data is not linearly separable. If our data is linearly separable we will end up with inconsistent solutions based on the hyperparameter of the learning rate and how our data is sorted. SVM's are able to combat these problems by using support vectors to find an optimal boundary, margin and unique solution that finds the smallest classification error. If the data is mostly linearly separable we can use a Soft-Margin to generalize our hyperplane.

2 SVM vs SVC

The term SVM is a general term for support vector machines, SVC states for support vector classifier, which is used for classification tasks, while SVR are used for regression tasks. Most of the terms discussed in this report are used in all SVM types, However, the hinge loss is used for SVCs only.

3 The concept of the Margin

When we draw our decision boundary, we want the boundary to be drawn as far away from each class. We call this, maximizing our margin, where the distance from the first class is equally as far from the boundary as the second class. Our total margin distance is defined as,

$$r = \frac{2}{\|w\|},$$

Normalizing $\|w\|$ helps us derive this equation, $y_n(\langle w, x_n\rangle + b) \geq r$, where $y_n \in \{-1, 1\}$ determines if x_n is on the negative or positive side of the linear boundary. The closest point(s) that can be defined as part of a class will be the "Support Vector" that defines our margin.

However when we have outliers it becomes problematic because our decision boundary and margin will be redefined by this outlier from one class and the a point closest in the other class. If we can see how the graph is drawn with an outlier we will see that our margin is drawn incorrectly and that maybe the point that is an outlier should be removed. This is where the concepts of Hard vs Soft margins come into play.

4 Hard VS Soft

Sometimes data is not linearly separable, but we would still like to classify it with an SVM. We know that hard margin SVMs enforce a decision boundary that will not accept any margin errors when classifying data, where everything on one side of the boundary gets classified as $+1$ and everything on the other side gets classified as -1. We use the following optimization problem to define our hard margin:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_n(\langle w, x_n\rangle + b) \geq 1, \quad n = 1, \ldots, N$$

On the other hand, soft margin SVMs are able to draw a decision boundary that separates two classes while maximizing the margin, even with some misclassified data points. To calculate our margin, we add a "slack" variable ξ_n, and introduce a regularization parameter $C > 0$, which trades off the size of the margin and the total amount of slack:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n \quad \text{Where } y_n(\langle w, x_n\rangle + b) \geq 1 - \xi_n, \quad \xi_n \geq 0$$

If the regularization parameter is large, then it is implied that there is low regularization as we give the slack variables a larger weight. A large C heavily penalizes any $\xi_n > 0$. Thus, when there is any heavy penalty our model will try to correctly classify outliers by shrinking the margin. From this equation, we can also see that w is regularized to prevent overfitting, but b is not so the boundary can shift without worrying about the penalties.

Here a large value of C implies low regularization, as we give the slack variables larger weight,hence giving more priority to examples that do not lie on the correct side of the margin. For every outlier, we add a penalty, and depending on how far the outlier is from the boundary, we increase the size of the penalty.

5 Hinge Loss

For a binary loss function we must count the number of mismatches between our predictions and the actual labels. So we must determine whether our x_n when used in our model, $f(x_n)$, gives us the incorrect label y_n.

We see that when using the hinge loss,

$$\ell(t) = \max\{0, 1 - t\}, \quad \text{where } t = yf(x) = y(\langle w, x\rangle + b)$$

we will return a zero when our $f(x_n)$ is correctly labeled and outside of the margin. When it is correctly labeled but inside of the margin area, It returns a t, $0 < t < 1$. If it is completely mislabeled then it returns a high loss value.

Using the hinge loss, our goal is to minimize the total loss while applying regularization to the margin, as described in Section 3.

$$\min_{w,b} \underbrace{\frac{1}{2}\|w\|^2}_{\text{regularizer}} + C\underbrace{\sum_{n=1}^{N} \max\{0, 1 - y_n(\langle w, x_n\rangle + b)\}}_{\text{error term}}$$

From this equation, we have our margin maximization and we have the error term, which is the sum of our hinge loss. C is our penalty term. If we increase C then our margin will become smaller if there are points that are misclassified and are correctly classified but within the margin. If C approaches infinity, the behavior will be identical to hard margin SVM

6 Geometric View

The hand-drawn figures below show the key concepts needed to understand SVM. The first figure shows the difference between a hard and soft margin boundary in case the dataset is linearly separable and contains an outlier measure. Its clearly visible that for the cost of 1 misclassified value the soft margin approach greatly improved the boundary position in terms of bias and effectively removes the outlier measure. The second figure shows how slack is measured and the misclasified point is highlighted

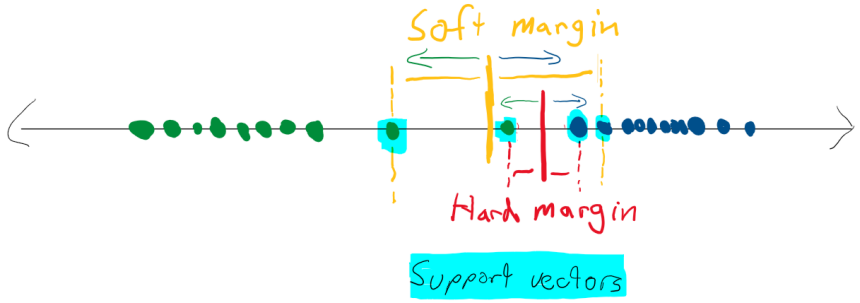

Figure 1: Soft vs Hard margin and support vectors

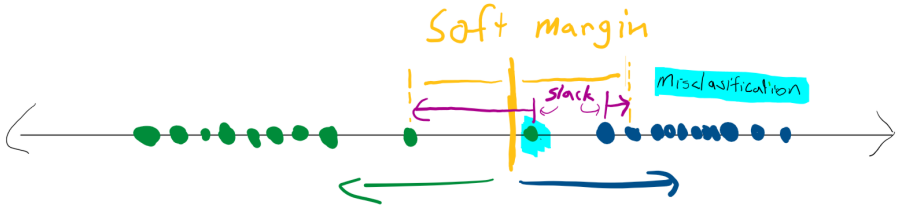

Figure 2: Slack and misclassification in soft margin SVM

7 Experimental Setup

7.1 Iris Dataset

For our experiments, we first plotted the data from the Iris dataset to see the initial separability of the two classes and used the default hyperparameters that were given:

Hyperparameter	values
C	1
Learning Rate	0.001
n epoch	1000
batch size	64

The first graph plot shows that the boundary line is not between our two classes. That the margin is probably set too small making the model too soft because there is not enough penalty for the points inside of the margin, which is most of them. We can also tell that our model works as soft margin SVM because we have blue points classified as two different classes.

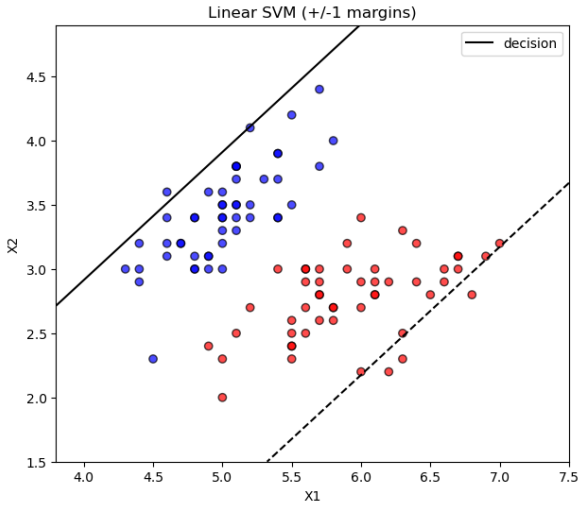

Figure 3: Initial Plot with initial Hyperparameters

In Figure 3, we assumed that it was not trained well enough and that we should increase the n-epoch, which is only a hyperparameter for training. Now that we see the transformation from Figure 3 to Figure 4, we can see that our hyperplane boundary is set correctly separating the two classes but now we might run into some overfitting problems with a high n-epoch value. However our model is still too soft. We then increased the size of C to add more penalties to make our margin size smaller.

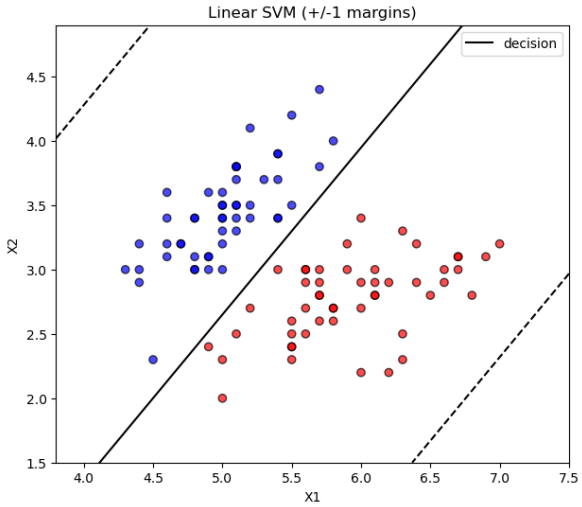

Figure 4: Updated n-epoch 10000

In Figure 5, we have updated our n-epoch to 10000 and our margin to 100. We see that our margin has shrunk significantly and that we are successfully classifying our data. This confirms that the more we increase our C hyperparameter the smaller our margin becomes. We could also change our learning rate to see if there is a more optimal boundary line, because we know that the learning rate updates our w and b parameters in the gradient algorithm.

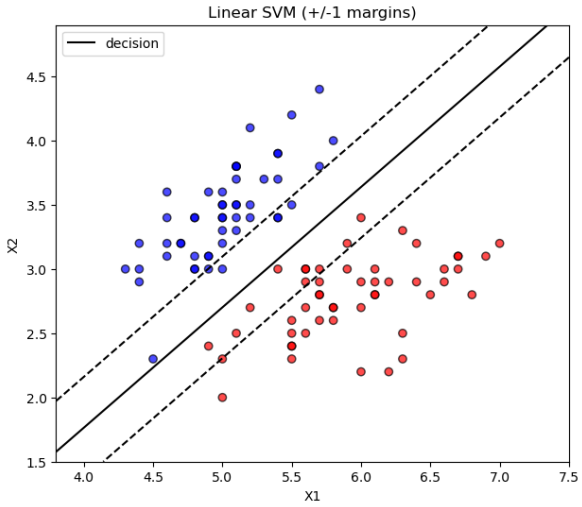

Figure 5: Margin = 100, n-epoch = 10000

7.2 Make-Blob Dataset

We will now make an analysis of the make-blob dataset. The dataset has a few parameters that we can change such as the number of samples in the clusters, and the clusters' standard deviation(std). The smaller the std the more separated our clusters are. We can create what looks like a hard margin SVM with a smaller margin.

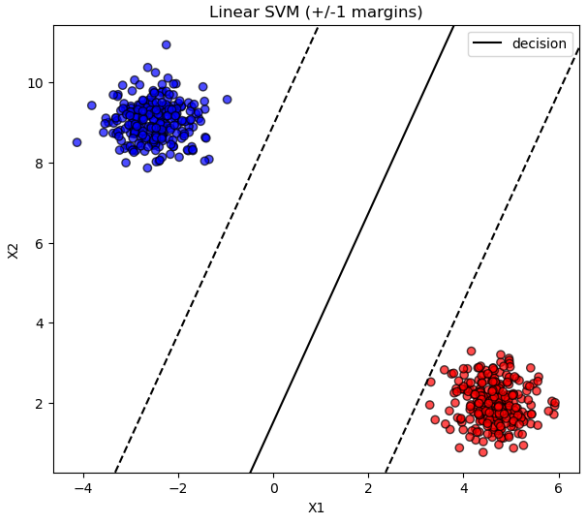

Figure 6: Hard Margin like boundary

But this isn't very interesting and it doesn't really tell us if our model would work with similar data. We will now increase the std to see how our model will place the decision boundary based on the fact that there will be many outliers. In Figure 7, we see that the two clusters merge together with many outliers. We can see that our model does a great job at setting the decision boundary, even when we vary the C parameter. The model is also allowing a lot of slack so that many points can be misclassified.

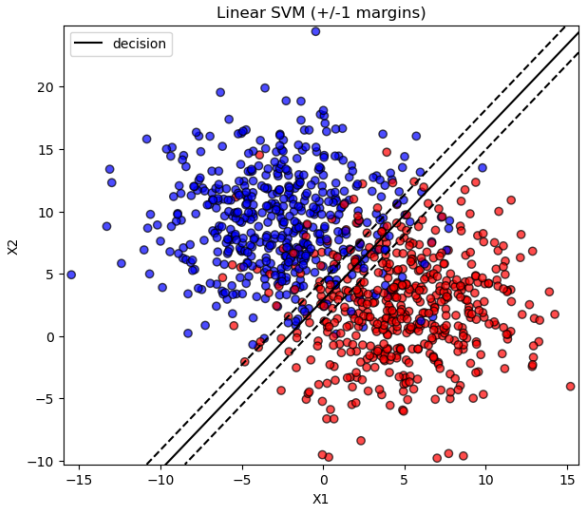

Figure 7: Hard Margin like boundary

In Figure 8, we changed the learning rate and the amount of samples in each cluster to see the effects on our model. We observed that increasing the learning rate made our decision boundary less accurate. The learning rate when dealing with margin violations is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial J}{\partial \mathbf{w}}$$

However Figure 8 shows that if the learning rate is too high, it will ignore the information our batch would have given us. Stepping over many points and not updating correctly. That is why, the learning rate is very important and that tuning the learning rate to a smaller is better than a larger value for the sake of accuracy but not for time. It is also important to not make it too small because it might stop our model too early.

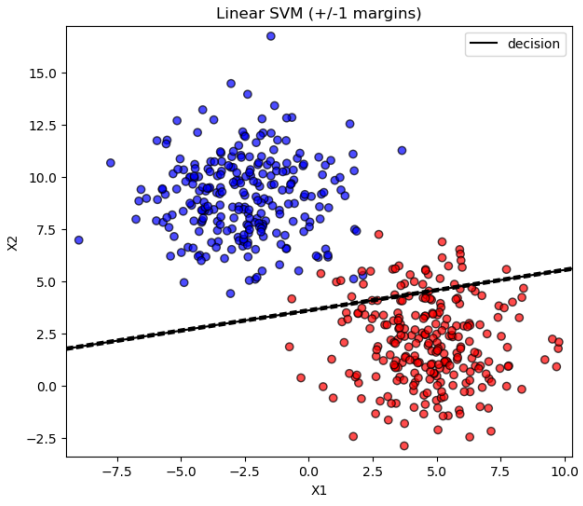

Figure 8: High Learning Rate

8 Comparison with Scikit functions

Comparing the custom implementation to the Scikit-learn-LinearSVC implementation, we have discovered the LinearSVC class is normalizng the bias, which leads to slightly different resoults. For this reason we have also decided to compare our implementation to the SGDClassifier scikit linear model,

which is not normalizing the bias. It uses different parameters than the other 2 functions, for this reason we have used Gemini 3 pro LLM to create the code necessary to generate the results using the SGDCClassifier function. The 3 clasifiers were then compared on the blob dataset. The acuracy is very similair in those 3 compared SVC implementation

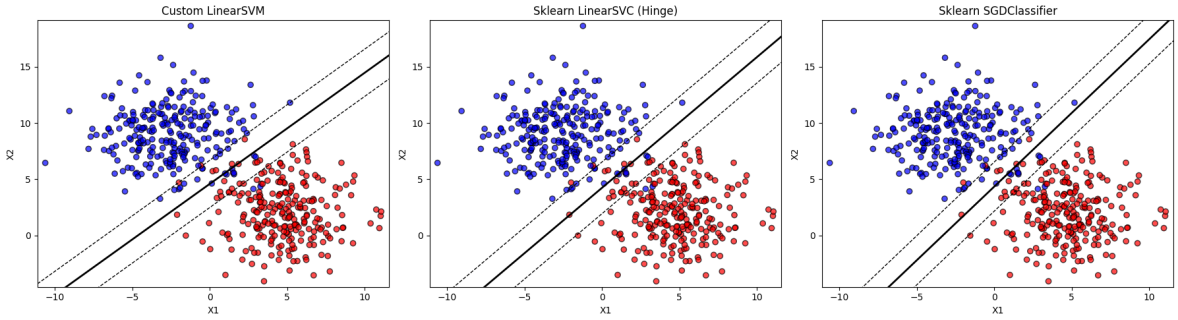

Figure 9: Model Comparison

Table 1: Model Accuracy Comparison

Model	**Accuracy**
Custom LinearSVM	0.9680
LinearSVC (Hinge)	0.9720
SGDClassifier	0.9680

9 Hand Derived Gradient Descent

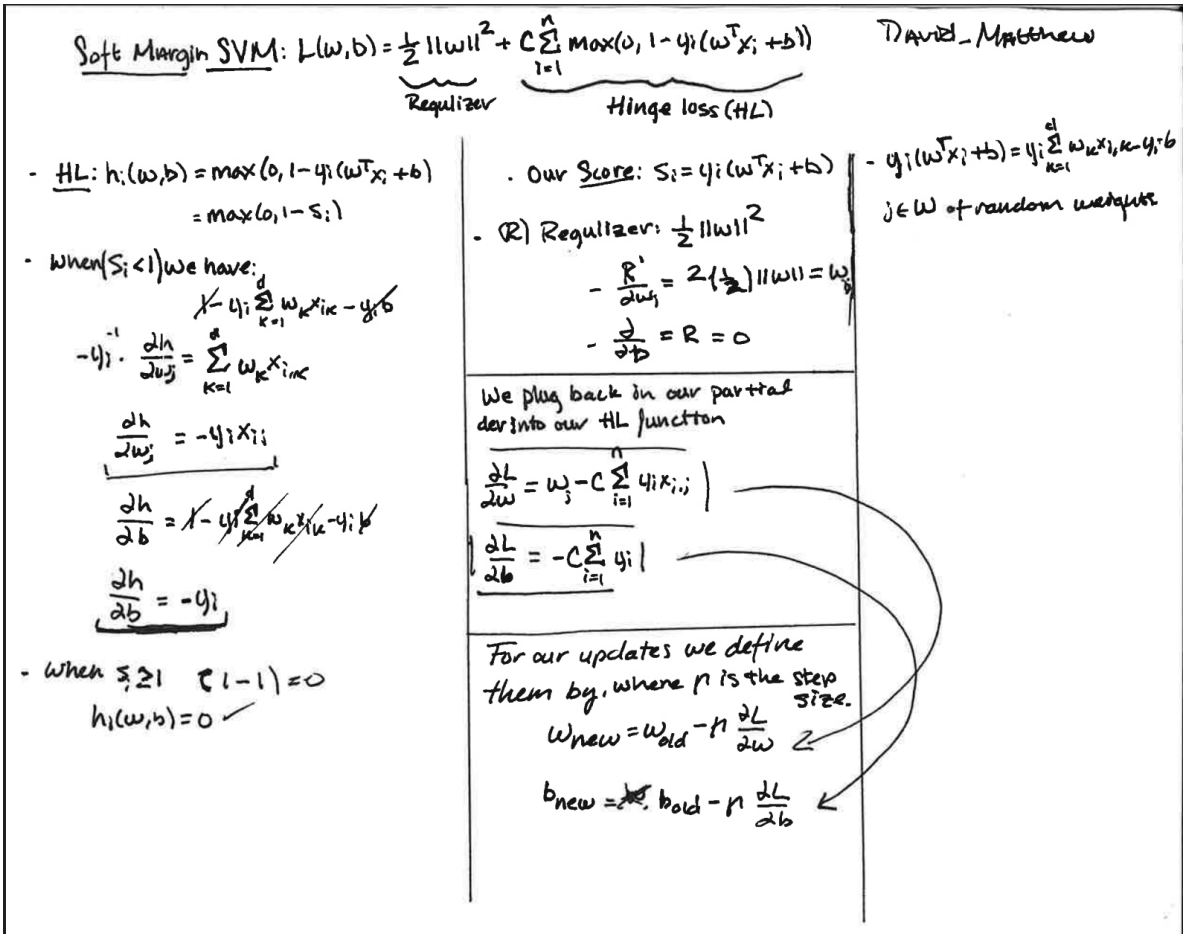

Figure 10: Hinge Loss Gradients

For correctly classified points that lie within the margin, the Hinge Loss is **0** as stated on the bottom left. Because there is no loss, the gradient is just $\mathbf{w}$, and the update rule will just be $\mathbf{w}_{t+1} = \mathbf{w}_t(1-\eta)$. For the points that are misclassified or in the margin violation, the Hinge Loss is positive ($L_H > 0$). The gradients correction term is $(\mathbf{w} - Cy_i\mathbf{x}_i)$, meaning the final update rule becomes $\mathbf{w}_{t+1} = (1-\eta)\mathbf{w}_t + \eta C y_i \mathbf{x}_i$. It shrinks the weights for margin maximization but also pushes the decision boundary away from the error that occurred.

10 Checking Math and code

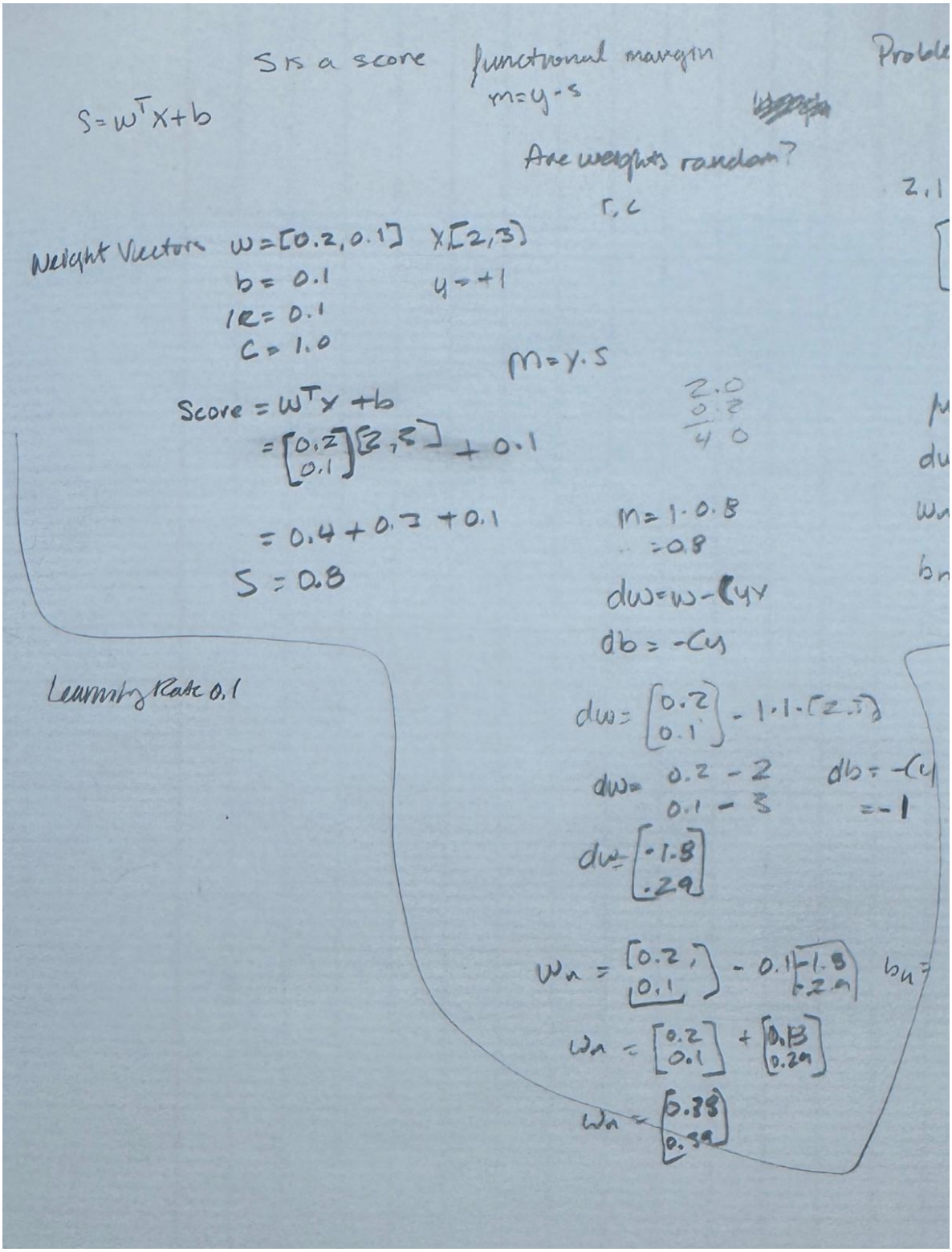

Figure 11: Updating Loss

11 Personal Reflection

While implementing the batch gradients, we ran into a discrepancy of if our regularization gradient should be normalized by dividing it by our batch size. We found that we had very similar results when we didn't normalize. However many SVM libraries normalized their batch gradients so we did as well, but we would like to state that there is probably not difference.

12 AI disclousure

Artificial intelligence tools were used as general help during the research on this topic. They were also used to check the correctness of code implementation or of Math derivations. However, AI was not used for writing this report and was not used in any capacity for writing the SVM library. AI was used to generate parts of the testing notebook, but strict instructions on what exactly it should create were given. For most of our background math work we are providing scans of by hand derivations, or we are using the suggested book.

13 Contributions

David and Matthew each did half of the work.

14 Citation and References List

All equations of math are from Mathematics for Machine Learning [1].

References

[1] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.