Machine Learning II

- Assignment #2 -

Course	Department	Student ID	Name
SOI1010	Automotive Engineering	2021048140	Dongmin Kim

November 4, 2025

Colab Notebook Link

https://colab.research.google.com/drive/1VGd6Avy2Bh6pARnp-tYwyd3dvvJR_zly?usp=sharing

Probelm #1

(a) Load CIFAR10 dataset as follows:

- **Dataset:** CIFAR-10 (60,000 RGB images, 10 classes, 32x32 resolution)

(b) Visualize at least one image for each class.

- They have 10 classes:
  {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}

(c) Split the trainset into training set and validation set with 90% : 10% ratio

- The dataset was split into **training (90%)** and **validation (10%)** sets using random_split()
- The test set was used for evaluation.

(d) Choose any two classes. Then, make a SVM classifier

- We chose two classes, {3: cat, 5: dog}, from the CIFAR-10 dataset for binary classification.
- Implemented a **Soft-Margin Binary SVM** using PyTorch
- Hinge Loss definded as:

$$L = \frac{1}{N}\sum_i \max(0, 1 - y_i(w^\top x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

(e) Train for 10 epochs with batch size 64

- Epochs: 10
- Batch size: 64
- Learning Rate: 1e-3
- Weight Decay: 1e-4
- Weight decay term added for regularization.

Result:

- **Validation Accuracy: 57.96%**
- **Test Accuracy: 59.00%**

(f) Perform data normalization

- Normalization equation:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

(g) Again, train for 10 epochs with batch size 64 after data normalization.

- Trainsets Mean and Std: [0.4914, 0.4822, 0.4465], [0.2470, 0.2435, 0.2616]
- Epochs: 10
- Batch size: 64
- Learning Rate: 1e-3
- Weight Decay: 1e-4

Result:

- **Validation Accuracy: 60.04%**
- **Test Accuracy: 62.35%**

Observation:

- After applying data normalization, the **validation accuracy** improved from **57.96%** to **60.04%**, and the **test accuracy** increased from **59.00%** to **62.35%**, showing about a **2–3%p** performance gain.

- This improvement occurred because normalization reduced the scale differences among features, allowing the SVM to learn a more balanced decision boundary. By centering the data around zero and scaling to unit variance, the optimization process became more stable, leading to faster convergence and better generalization performance.

- Therefore, data normalization is an essential preprocessing step for improving SVM training stability and accuracy.

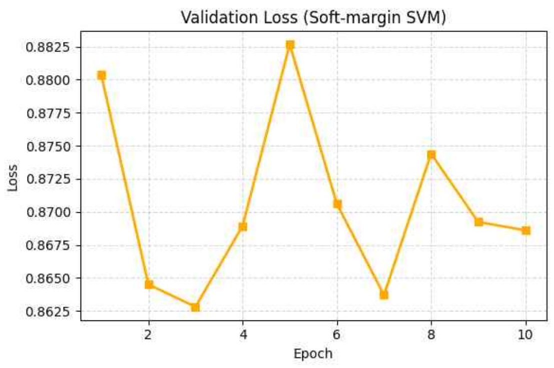
(h) What are the hyperparameters you can tune?

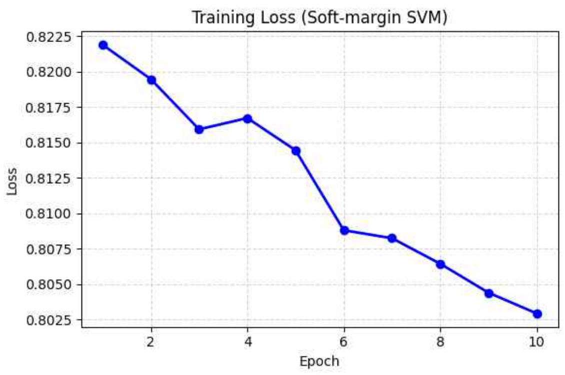
- Learning Rate
- Weight Decay
- Epochs
- Batch Size (Optional)

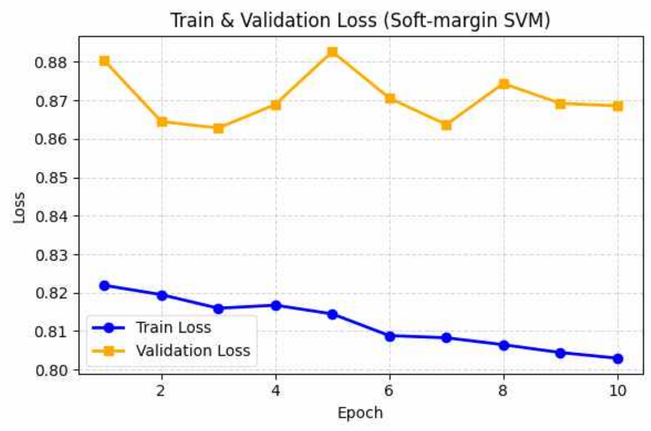
(i) Try to obtain find optimal hyperparameters.

1. Epochs

- We can choose the best epochs by looking the loss graph







- As shown in the loss graph, the training loss continuously **decreases throughout all 10 epochs**, indicating that the model is still learning useful patterns from the training data.

- Although the validation loss fluctuates slightly, it remains relatively stable without a significant upward trend, meaning that severe overfitting has not yet occurred.

- Therefore, **choosing 10 epochs** is a reasonable balance between sufficient training progress and acceptable generalization performance.

2. Learning Rate and Weight Decay

LR \ WD	0.0001	0.0002	0.0003	0.0004	**0.0005**
0.001	58.75	60.44	59.25	60.53	60.04
0.002	60.34	59.25	58.36	60.63	60.44
0.003	59.84	54.90	58.06	60.24	**61.03**
0.004	60.24	59.84	58.95	58.16	57.07
0.005	55.00	58.95	56.97	57.76	58.85

- Through hyperparameter tuning, we found that a **learning rate** of **0.003** and a **weight decay** of **0.0005** produced the best overall results (validation accuracy).

- This configuration maintained stable training, avoided overfitting, and achieved the highest validation accuracy among all tested settings.

j) What is the final test accuracy?

- **Learning Rate**: 0.002
- **Weight Decay**: 0.0005
- **Validation Accuracy**: **58.46%**
- **Test Accuracy: 60.05%**

- The validation accuracy differs slightly from the previous experiments. This variation is mainly due to the fact that the random split seed was fixed after training, rather than before. Because the dataset was divided differently for each run, the composition of the training and validation sets changed, leading to small fluctuations in performance.

- Additionally, minor randomness from mini-batch sampling and the stochastic nature of optimization can also contribute to this difference.

- The **limitation of Colab's TPU runtime** duration prevented retraining with a consistent random seed, which explains the inconsistency across runs.

Problem #2: Multiclass Classification via multiple one-vs-all soft-margin SVM

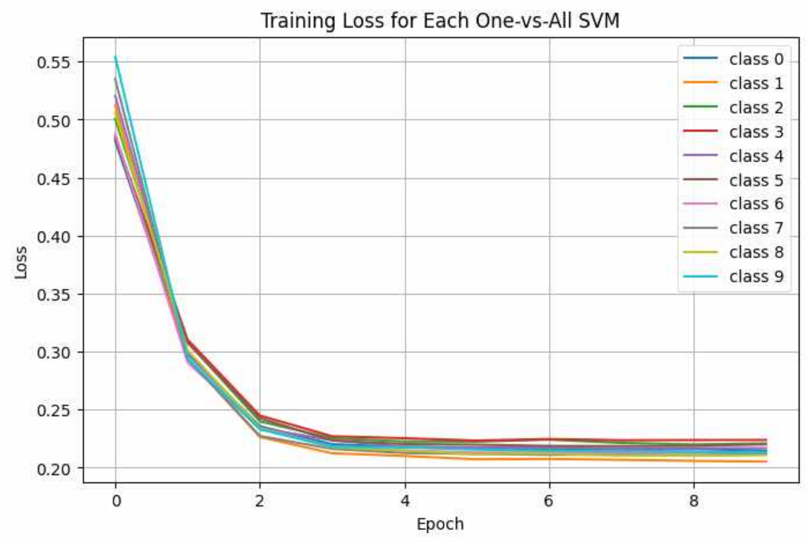
(a) Perform multiclass classification using multiple one-vs-all soft-margin SVM

	Validation Accuracy
Class 0	90.46%
Class 1	90.50%
Class 2	85.32%
Class 3	89.92%
Class 4	89.34%
Class 5	89.86%
Class 6	89.92%
Class 7	89.44%
Class 8	87.70%
Class 9	**90.78%**

- In this approach, ten binary SVM classifiers were trained independently, where each classifier distinguishes one target class (positive) from all remaining classes (negative).

- The **validation accuracies** for all classes are fairly consistent **(85–91%)**, though slight variations arise due to differences in class complexity and visual similarity. This shows that the one-vs-all SVM performs uniformly across classes but is still limited by the linear separability of raw pixel features.

- Since finding the **optimal hyperparameters** (Learning Rate and Weight Decay) for all ten classifiers would require evaluating 5 × 5 combinations for each class—a total of 250 validation results—the process would be **computationally expensive** and **overly complex.** Therefore, **the hyperparameter search will be conducted only for class 9,** which achieved the highest validation accuracy **(90.78%)**, and the resulting optimal values will be used for all classes.



- In addition, The results show that the model performs stably across most parameter ranges, but too high a learning rate or excessive regularization can degrade performance.

- As shown in the training loss curves, all ten one-vs-all SVMs exhibited a rapid loss reduction within the first few epochs and then converged smoothly around a **stable value near 0.20**.

- Since the loss no longer decreased significantly after approximately the fifth epoch, the model had already reached convergence and additional training would bring only marginal improvement.

- Therefore, setting the total number of **epochs** to **10** was a reasonable choice.

- The consistent convergence behavior among all SVMs also confirms that the chosen hyperparameters allowed the models to learn efficiently and stably.

(b) Perform hyperparameter search for each SVM separately

LR \ WD	0.0001	0.0002	0.0003	0.0004	0.0005
0.001	90.66	90.66	90.66	90.66	90.66
0.002	90.66	90.66	90.66	90.54	90.66
0.003	90.64	89.56	90.56	90.58	89.76
0.004	86.18	90.88	**91.12**	90.48	90.78
0.005	90.00	90.68	90.36	89.96	78.10

- From the hyperparameter search conducted on **class 9**, the best validation accuracy of **91.12%** was achieved with a **learning rate** of **0.004** and **weight decay** of **0.0003**. These optimal values were then applied to all one-vs-all classifiers.

(c) What is the final test accuracy?

- The final test accuracy of **25.42%** is **relatively low** compared to typical convolutional-network baselines on CIFAR-10. This outcome is mainly due to the limitations of the implemented one-vs-all linear SVMs.

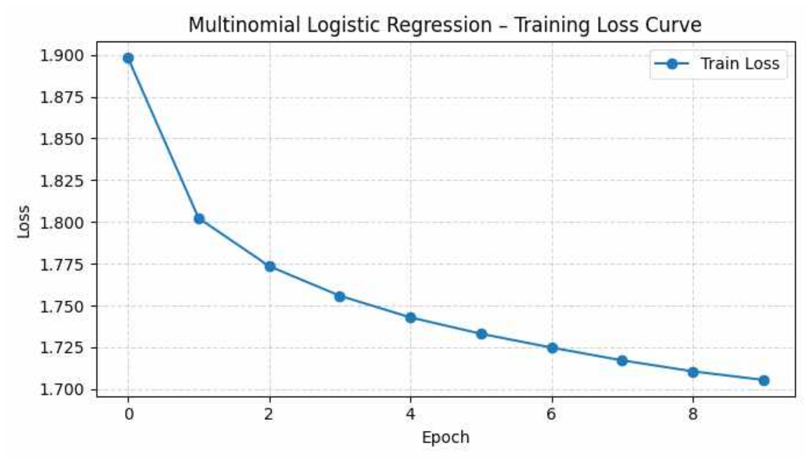
- First, each classifier was trained only with a single linear decision boundary, **without any kernel mapping or feature extraction,** which restricts its ability to separate complex image patterns in the CIFAR-10 dataset. Second, the input images were directly flattened into 3,072-dimensional vectors, so **all spatial and local correlations among pixels were lost**. Third, the same hyperparameters (learning rate and weight decay) were applied to all ten binary classifiers for computational efficiency, which may not be optimal for every class. Lastly, the model could easily **overfit** to training data but generalize poorly to unseen test images.

- Overall, these design constraints explain why the final performance remained around 25%, even though the training and validation accuracies showed consistent convergence.

Problem #3: Multiclass Classification via multinomial/multiclass logistic regression

(a) Implement your own multinomial/multiclass logistic regression from scratch.

Epoch	Training Loss	Validation Accuracy
1	1.8983	90.46%
2	1.8023	90.50%
3	1.7736	85.32%
4	1.7559	89.92%
5	1.7429	89.34%
6	1.7331	89.86%
7	1.7248	89.92%
8	1.7172	89.44%
9	1.7106	87.70%
10	1.7055	90.78%



- With learning rate **1e-3**, weight decay **1e-4**, and **10** epochs, the model converged from a training loss of **1.8983** to **1.7055**, achieving a peak **validation accuracy** of **40.00%.**

(b) Perform hyperparameter search

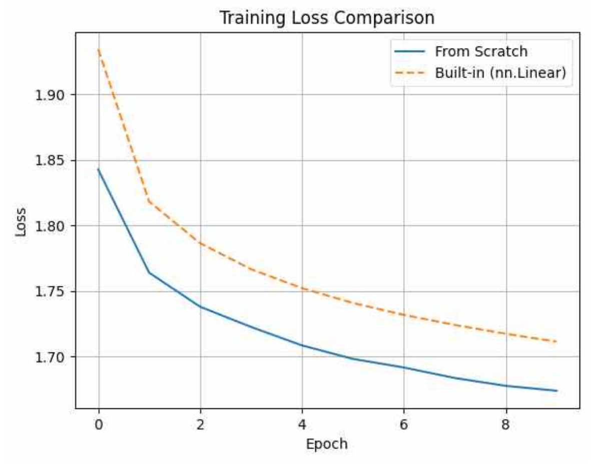
LR \ WD	**0.0001**	0.0002	0.0003	0.0004	0.0005
0.001	**40.64**	40.50	40.08	39.90	39.76
0.002	39.48	40.00	38.86	38.66	40.02
0.003	38.86	40.08	39.08	38.16	40.30
0.004	39.52	38.10	37.40	38.52	38.56
0.005	37.52	39.16	37.92	36.96	34.86

- Through hyperparameter tuning, the best performance was achieved with a learning rate of **0.001** and weight decay of **0.0001.** Under this configuration, the model reached a validation accuracy of **40.64%** after **10** epochs, with the training loss decreasing from **1.8983** to **1.7055.**

(c) What is the final test accuracy?

- When evaluated on the test set, the multinomial logistic regression model achieved a **test accuracy** of **40.80%.**

(d) Compare against a model implemented using functions from PyTorch libraries



Module	**Val Acc (%)**	**Test Acc (%)**
PyTorch	38.60	39.26
Manual	**40.38**	**40.80**

- For comparison, the **PyTorch implementation** of multinomial logistic regression achieved a **validation accuracy** of **38.60%** and a **test accuracy** of **39.26%.**

- In contrast, the manually implemented version achieved higher performance, with a **validation accuracy** of **40.38%** and a **test accuracy** of **40.80%.**

- This result demonstrates that the custom implementation functions correctly and slightly outperforms the baseline in both validation and test performance.