CS587 - Neural Networks & Learning of Hierarchical Representation

Spring Semester 2024
Assignment 1

Papageridis Vasileios - 4710
csd4710@csd.uoc.gr

March 19, 2024

Introduction

In this assignment we have to tackle the task of image classification for the CIFAR10 dataset, which has 60,000 color images of size 32x32 pixels in 10 classes. The goal is to develop a linear classifier that accurately predicts the class of each image.
To solve this problem, we will also have to implement a simple version of the mini-batch Stochastic Gradient Descent (SGD) to train our model. Mini-batch SGD is one of the optimization gradient descent methods. This is a modification of conventional SGD, updating model parameters using not every training sample or the whole training dataset, but a small random subset of it.
Lastly, in order to control overfitting we are going to measure the model's performance for different regularization techniques. In this assignment we are going to use the L1 and L2 regularization.

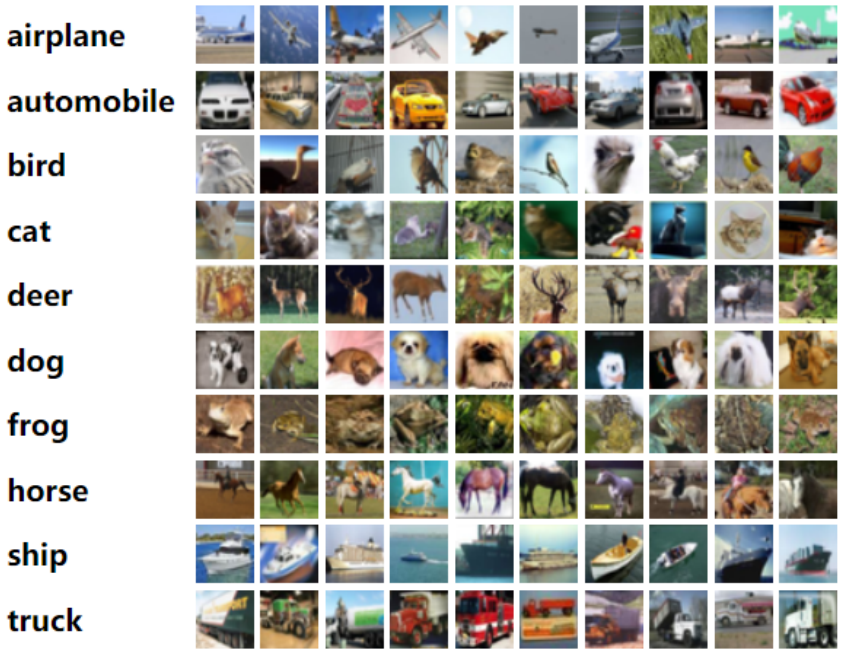

Figure 1: Sample images from the CIFAR-10 dataset, depicting the ten different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

1 Model Foundations

This section describes the theoretical foundation underlying the approach to solve the classification problem.

1.1 Model Description

The linear classifier is defined as: $f(x; W, b) = Wx + b$, where:

- $x \in \mathbb{R}^D$ represents a vectorized image from the CIFAR10 dataset, flattened from its original 32x32x3 structure into a single-dimensional array.
- $W \in \mathbb{R}^{10 \times D}$ is the weight matrix, with each row corresponding to one of the ten classes.
- $b \in \mathbb{R}^{10}$ is the bias vector, which adjusts the output scores for each class.

In this framework, the classifier takes an input image x, applies a linear transformation using W and b, and produces a set of scores for each class. The predicted class for the image is then the one with the highest score.

1.2 Loss Function

We measure the performance of our classifier with a multi-class hinge loss, often referred to as 'max-margin' loss because of regularization to avoid overfitting. From this point on, the empirical loss function may be derived as follows:

$$L(W, b) = \lambda R(W) + \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 + \max_{j \neq y_i}(W_j x_i + b_j) - (W_{y_i} x_i + b_{y_i}))$$

where:

- $R(W)$ is the regularization term, which can be either L_1 norm ($\sum |W_{i,j}|$) or L_2 norm ($\sum W_{i,j}^2$) of the weights.
- λ is the regularization strength, a hyperparameter that controls the trade-off between increasing the margin size and ensuring the classifier does not overfit.
- N is the number of training samples.
- The summation term computes the hinge loss, encouraging correct classification with a margin of at least 1.

1.3 Gradient Descent Update Rule

$$W \leftarrow W - \gamma \left(\lambda \nabla R(W) + \frac{1}{M} \sum_{i=1}^{M} \nabla_W \text{loss}(f(x_i; W, b), y_i) \right)$$

$$b \leftarrow b - \gamma \left(\frac{1}{M} \sum_{i=1}^{M} \nabla_b \text{loss}(f(x_i; W, b), y_i) \right)$$

where:

- M is the size of the mini-batch.
- γ is the learning rate, a hyperparameter that influences the speed and quality of the convergence.

2 Gradient Computation

In this section, we will detail the mathematically derived gradients used to update the steps in our mini-batch Stochastic Gradient Descent (SGD) algorithm.

2.1 Partial Derivatives of the Hinge Loss

The hinge loss incorporates a margin that enforces a separation between the score of the correct class and the scores of incorrect classes. For a given class j, the partial derivatives of the loss with respect to the scores s_j are defined as follows:
1. For the correct class ($j = y_i$):

$$\frac{\partial \text{loss}}{\partial s_{y_i}} = \begin{cases} -1 & \text{if } 1 + \max_{j \neq y_i} s_j - s_{y_i} > 0 \\ 0 & \text{otherwise} \end{cases}$$

2. For incorrect classes ($j \neq y_i$):

$$\frac{\partial \text{loss}}{\partial s_j} = \begin{cases} 0 & \text{if } j \text{ is not the maximizer or } 1 + s_j - s_{y_i} \leq 0 \\ 1 & \text{if } j \text{ is the maximizer and } 1 + s_j - s_{y_i} > 0 \end{cases}$$

2.2 Gradient with Respect to Weights W and Bias b

Using the chain rule, we translate these score gradients into gradients with respect to the weights and bias:
For the weights W:

$$\nabla_W \text{loss}_i = \begin{cases} -\mathbf{x}_i & \text{if } j = y_i \text{ and margin violated} \\ \mathbf{x}_i & \text{if } j \neq y_i \text{ is the maximizer and margin violated} \end{cases}$$

For the bias b, a similar logic applies, but without the $\mathbf{x}_i$ factor:

$$\nabla_b \text{loss}_i = \begin{cases} -1 & \text{if } j = y_i \text{ and margin violated} \\ 1 & \text{if } j \neq y_i \text{ is the maximizer and margin violated} \end{cases}$$

2.3 Final Gradients for a Mini-batch

The final gradients for the weights and bias are obtained by averaging over all samples in the mini-batch:

$$\nabla_W L = \frac{1}{N}\sum_{i=1}^{N} \nabla_W \text{loss}_i \quad \text{and} \quad \nabla_b L = \frac{1}{N}\sum_{i=1}^{N} \nabla_b \text{loss}_i$$

Finally, these gradients are used to update the model parameters in each iteration of the mini-batch SGD.

3 Results

In this section, we analyze the results obtained from the training of the Linear Classifier on the CIFAR-10 dataset using Stochastic Gradient Descent (SGD). The Linear Classifier was trained over six epochs with a learning rate of $1e-7$ and a regularization strength of $5e4$ using L2 regularization. The batch size was set to 200, and the validation loss was evaluated every 100 iterations.

3.1 Validation Loss

As shown in Figure 2, the validation loss decreases sharply during the initial iterations and then stabilizes as the number of iterations increases. This suggests that the model quickly learns from the training data, but as it converges to a solution, the rate of learning decreases and the loss levels off.

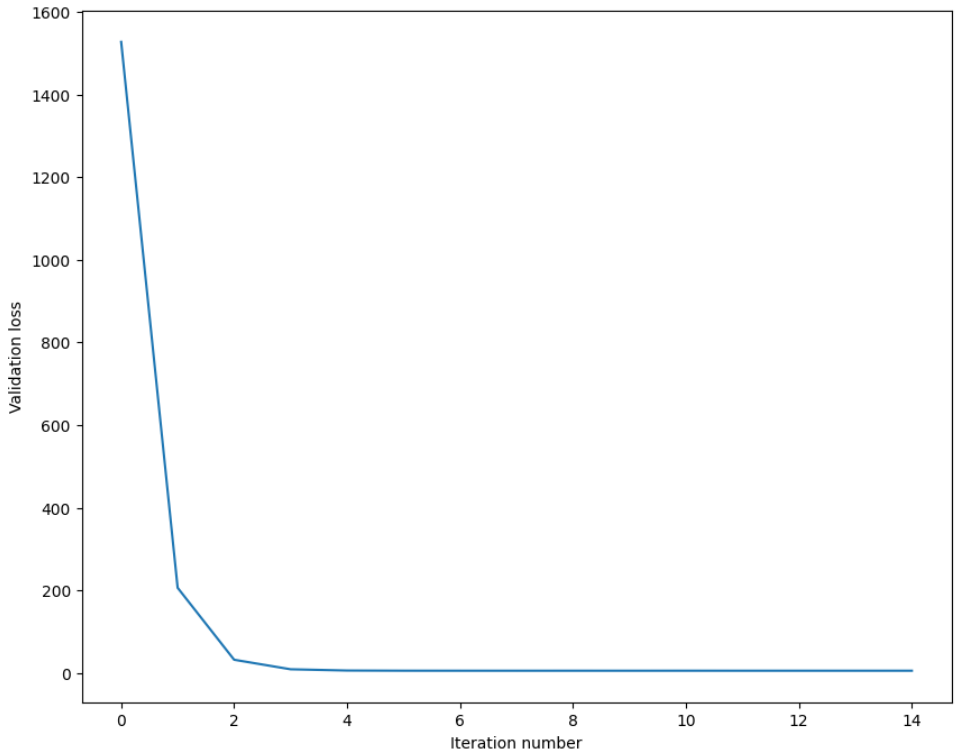

Figure 2: Validation loss as a function of iteration number during training of the Linear Classifier.

The rapid decrease in loss during the early iterations can be attributed to the large margin errors being corrected by the SGD algorithm.
The results demonstrate the effectiveness of SGD in optimizing the Linear Classifier for image classification tasks. The decrease in validation loss indicates that the model is learning generalizable features from the training data, which are useful in classifying unseen validation data.

4 Hyperparameter Tuning and Validation

In this section, we discuss the process and outcomes of hyperparameter tuning for the Linear Classifier on the CIFAR-10 dataset. The goal was to identify the best combination of learning rate, regularization strength, regularization type, and batch size to maximize the classification accuracy on the validation set.

4.1 Experimental Setup

To find the optimal set of hyperparameters, we explored eight different combinations, varying learning rates, regularization strengths, regularization types (L1 vs. L2), and batch sizes. The specific values tested were:

- Learning rates: $[1e-8, 1e-7, 3e-7, 3e-7, 5e-7, 8e-7, 1e-6, 1e-5]$
- Regularization strengths: $[1e4, 3e4, 5e4, 1e4, 8e4, 1e5, 5e4, 5e5]$
- Regularization types: $[1, 2, 1, 2, 1, 2, 1, 2]$ corresponding to L1 and L2 respectively
- Batch sizes: $[50, 100, 200, 400, 100, 200, 200, 400]$

Each combination was used to train a Linear Classifier for six epochs, and the classification accuracy was measured on both the training and validation datasets.

4.2 Results

After experimenting with all combinations, the highest validation accuracy achieved was 39.2%, with a corresponding training accuracy of 38.02%. These results were obtained using the following hyperparameters:

- Learning rate: $3e-07$
- Regularization strength: 10000.0 (L2 regularization)
- Batch size: 400

The combination of a higher learning rate with moderate regularization and a larger batch size proved to be most effective for this dataset. The higher batch size might have contributed to a more stable and accurate estimation of the gradient, leading to better convergence properties.

5 Testing and Visualization of the Final Classifier

Upon finalizing the hyperparameters based on validation performance, we proceeded to evaluate the best classifier on the test dataset and visualize the learned weights.

5.1 Training and Validation Losses

As illustrated in Figure 3, the training and validation losses decrease significantly with iterations, indicating a good fit to the data without obvious overfitting. The sharp decline in the initial iterations suggests that the learning rate is adequately chosen, allowing for rapid convergence to a lower loss value.

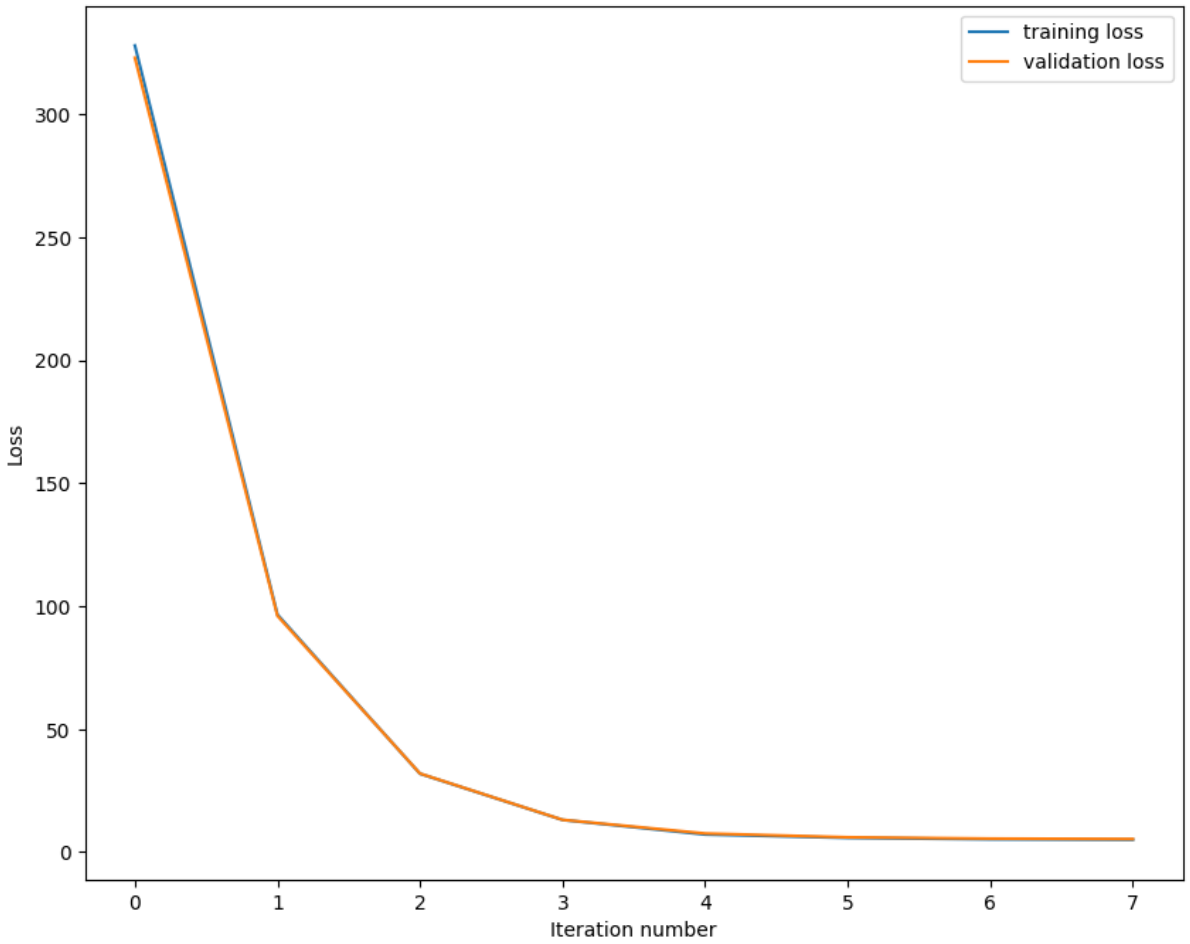

Figure 3: The plot of training and validation loss as a function of iteration number. The training loss (blue) and validation loss (orange) both decrease over time, indicating the learning process of the classifier.

5.2 Evaluation on Test Data

Finally, the last classifier, meaning the optimized one for hyperparameters based on the validation data, is then applied to the test set for its generalization ability. The test accuracy achieved was 37.3%—nothing revolutionary, but still modestly acceptable, considering that raw pixels from the dataset are being worked on by a linear classifier. This is to say that, even if the model is so simple, it still has the potential to capture important features for distinguishing the different categories of images.

5.3 Visualization of Learned Weights

As it turned out, visualizing the images might look not as clear as one would expect for a better model, say, a Convolutional Neural Network (CNN), since linear classifiers over raw pixel values need not produce weight maps that are always visually interpretable. The trends observed in the training process, combined with the final performance metrics. We visualize the weights of each class, reshaped and normalized according to the size and color scale of CIFAR-10 images, in Figure 4. These aren't real images from the dataset, but rather we can refer them as templates that the classifier has learned in order to identify each class.

- The "plane" weights show a vague shape that could resemble the aerial view of an airplane, with bluish tones possibly capturing the sky background.
- The "car" weights seem to highlight areas in the center, which could correspond to the typical position of cars in images and also a red tone which may indicates that there are a lot red cars in CIFAR-10 dataset.
- The "bird" weights are more diffused, but hints of sky could be interpreted in the pattern.
- The "cat" weights mix various colors, which could correspond to the various environments cats are found in, though a clear shape is hard to discern.
- The "deer" weights exhibit a mix of greens and browns, likely representing the animal and forest background.
- The "dog" weights show a less clear pattern, indicating diversity in appearance and background for this class.
- The "frog" weights have a strong green center, likely capturing the color of a frog.
- The "horse" weights seem to capturing the body shape of horses in different angles, as we can see a "double-headed" horse.
- The "ship" weights are dominated by blues, indicative of the sea typically found in ship images.
- The "truck" weights again focus possibly on the body of the truck, with some blue tones that might reflect sky backgrounds.

Some loose patterns could be identified, but in general, these visualizations are abstract. The reason for this abstraction lies in the linear nature of the model and, of course, in the complexity of natural images.
However, these visualizations are useful to have an idea about the overall regions of image space on which the classifier focuses for each of the classes. They can also indicate possible biases in the dataset, for example, if the class weight captures mainly the background and not the object.

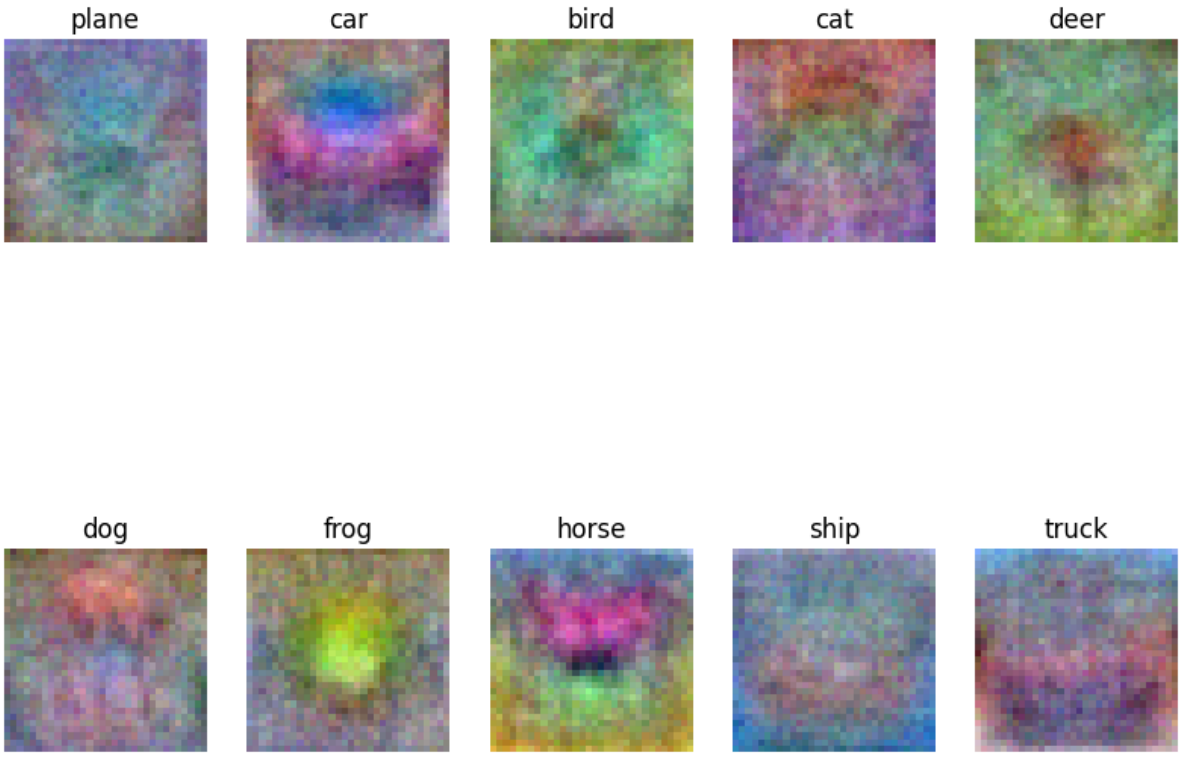

Figure 4: Visualization of the weights learned by the linear classifier for each class in the CIFAR-10 dataset. Each image represents the weights associated with one of the ten classes: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck.

References

[1] Justin Johnson, *EECS 498-007 / 598-005 Deep Learning for Computer Vision*, 2022, https://web.eecs.umich.edu/~justincj/teaching/eecs498/WI2022/.

[2] Simon J.D. Prince, *Understanding Deep Learning*, MIT Press, 2023, https://udlbook.github.io/udlbook/.

[3] Sebastian Nowozin and Christoph H. Lampert, *Structured Learning and Prediction in Computer Vision*, 2012, https://www.nowozin.net/sebastian/cvpr2012tutorial/.