Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

Homework 4:

Theoretical Questions:

1. SVM with multiple classes:

> 1. **(15 points) SVM with multiple classes.** One limitation of the standard SVM is that it can only handle binary classification. Here is one extension to handle multiple classes. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and now let $y_1, \ldots, y_n \in [K]$, where $[K] = \{1, 2, \ldots, K\}$. We will find a separate classifier $\mathbf{w}_j$ for each one of the classes $j \in [K]$, and we will focus on the case of no bias ($b = 0$). Define the following loss function (known as the *multiclass hinge-loss*):
>
> $$\ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i) = \max_{j \in [K]} (\mathbf{w}_j \cdot \mathbf{x}_i - \mathbf{w}_{y_i} \cdot \mathbf{x}_i + \mathbb{1}(j \neq y_i)),$$
>
> where $\mathbb{1}(\cdot)$ denotes the indicator function. Define the following multiclass SVM problem:
>
> $$f(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i)$$
>
> After learning all the $\mathbf{w}_j$, $j \in [K]$, classification of a new point $\mathbf{x}$ is done by $\arg\max_{j \in [K]} \mathbf{w}_j \cdot \mathbf{x}$. The rationale of the loss function is that we want the "score" of the true label, $\mathbf{w}_{y_i} \cdot \mathbf{x}_i$, to be larger by at least 1 than the "score" of each other label, $\mathbf{w}_j \cdot \mathbf{x}_i$. Therefore, we pay a loss if $\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_j \cdot \mathbf{x}_i \leq 1$, for $j \neq y_i$.
>
> Consider the case where the data is linearly separable. Namely, there exists $\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*$ such that $y_i = argmax_y \mathbf{w}_y^* \cdot \mathbf{x}_i$ for all i. Show that any minimizer of $f(\mathbf{w}_1, \ldots, \mathbf{w}_K)$ will have zero classification error.

Solution:
In the case that the data is linearly separable, there exist weights $w_1^*, x_2^*, \ldots, w_k^*$ such that $y_i = \arg\max_y (w_y^* x_i)$ holds for every i.
Now, let's consider any minimizer of the multiclass SVM problem, denoted as $w_1', \ldots, w_k'$. My goal is to show that this minimizer will have zero classification error, meaning it will correctly classify all the data points.
For a given data point x_i, the multiclass hinge-loss function $\ell(w_1, \ldots w_k, x_i, y_i)$ is defined as:

$$\ell(w_1, \ldots w_k, x_i, y_i) = \max_{j \in [k]} \left(w_j x_i - w_{y_i} x_i + 1(j \neq y_i) \right)$$

Here, y_i is the true label of x_i, and j denotes the classes in $[k]$.

Since the data is linearly separable, for any data point x_i, we have:

$$w\, y_i \cdot x_i - w_j \cdot x_i > 1 \quad for\ all\ j \neq y_i$$

This inequality holds because the score of the true label, $wy_i \cdot x_i$, is always larger than the score of any other label, $w_j \cdot x_i$, by at least 1.

Now, let's consider the loss incurred for x_i when using the minimizer weights $w_1', \ldots, w_k'$. Using the definition of the multiclass hinge-loss, I have:

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

$$\ell(w_1', \ldots, w_k', x_i, y_i) = \max_{j \in [k]} \left(w_j{}' x_i - w_{y_i}{}' x_i + 1(j \neq y_i) \right)$$

Since the minimizer weights $w_1{}', \ldots, w_k{}'$ are obtained by minimizing the multiclass SVM problem, we can conclude that for each data point x_i, the loss incurred $\ell(w_1{}', \ldots, w_k{}', x_i, y_i)$ is minimized.

To minimize the loss, the following inequality must hold:

$$w' y_i \cdot x_i - w_j' \cdot x_i \geq 1 \quad for\ all\ j \neq y_i$$

This means that the score of the true label, $w' y_i \cdot x_i$, is larger by at least 1 than the score of each other label, $w_j' \cdot x_i$.

Combining this inequality with the linear separability assumption, I have:

$$(w y_i \cdot x_i - w_j \cdot x_i) - (w' y_i \cdot x_i - w_j' \cdot x_i) > 0 \quad for\ all\ j \neq y_i$$

Expanding the expression, I get:

$$w y_i \cdot x_i - w_j \cdot x_i - w' y_i \cdot x_i + w_j' \cdot x_i > 0$$

Now, let's consider the case where $j = y_i$:

$$w y_i \cdot x_i - w y_i \cdot x_i - w' y_i \cdot x_i + w' y_i \cdot x_i > 0$$

Simplifying further:

$$w' y_i \cdot x_i - w' y_i \cdot x_i > 0$$

This implies:

$$0 > 0$$

Since this inequality cannot hold, it means that the assumption $j = y_i$ is not possible, and hence we must have $j \neq y_i$.

Therefore, we can conclude that for any data point x_i, the loss incurred using the minimizer weights $w_1{}', \ldots, w_k{}'$ will always be zero:

$$\ell(w_1{}', \ldots, w_k{}', x_i, y_i) = \max_{j \in [k]} \left(w_j{}' x_i - w_{y_i}{}' x_i + 1(j \neq y_i) \right)$$

Since the loss is zero for all data points, it implies that the minimizer weights $w_1{}', \ldots, w_k{}'$ correctly classify all the data points without any misclassification. Hence, the minimizer of the multiclass SVM problem will have zero classification error when the data is linearly separable.

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

2. Expressivity of ReLU networks:

> 2. **(10 points) Expressivity of ReLU networks.** Consider the ReLU activation function:
>
> $$h(x) = \max\{x, 0\}$$
>
> Show that the maximum function $f(x_1, x_2) = \max\{x_1, x_2\}$ can be implemented using a neural network with one hidden layer and ReLU activations. You can assume that there is no activation after the last layer.
>
> (Hint: It is possible to implement the function using a hidden layer with 4 neurons. You may find the following identity useful: $\max\{x_1, x_2\} = \frac{x_1+x_2}{2} + \frac{|x_1-x_2|}{2}$.)

Solution:
Our goal is to use a neural network to implement the maximum function:

$$f(x_1, x_2) = \max\{x_1, x_2\}$$

Let's break the ReLU activation function into cases:

$$h(x) = \max\{x, 0\}$$

Case 1: $x \geq 0$ –

$$x = \max\{0, x\}, 0 = \max\{0, -x\} \rightarrow x = x - 0 = \max\{0, x\} - \max\{0, -x\}$$

$$x \geq -x \rightarrow \max\{x, -x\} = x = |x|$$

Case 2: $x < 0$ –

$$0 = \max\{0, x\}, -x = \max\{0, -x\} \rightarrow x - x = 0$$

$$-x \geq x \rightarrow \max\{x, -x\} = -x = |x|$$

Finally:

$$\frac{x_1 + x_2}{2} + \frac{|x_1 - x_2|}{2} = \frac{x_1 + x_2 + \max\{x_1 - x_2, x_2 - x_1\}}{2} = 0.5(\max\{2x_1, 2x_2\})$$
$$= \max\{x_1, x_2\}$$

3. Soft SVM with $\mathbb{l}^2$ penalty:

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

3. **(15 points) Soft SVM with ℓ^2 penalty.** Consider the following problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n$$

(a) Show that a constraint of the form $\xi_i \geq 0$ will not change the problem. Meaning, show that these non-negativity constraints can be removed. That is, show that the optimal value of the objective will be the same whether or not these constraints are present.

(b) What is the Lagrangian of this problem?

(c) Minimize the Lagrangian with respect to $\boldsymbol{w}, b, \boldsymbol{\xi}$ by setting the derivative with respect to these variables to 0.

(d) What is the dual problem?

Solution:

(a) To show that the non-negativity constraints $\xi_i \geq 0$ can be removed without changing the problem, we can observe that if any ξ_i is negative, it will only increase the objective value.
Assume we have a feasible solution (w^*, b^*, ξ^*) with $\xi_i < 0$ for some i. Now consider a modified solution (w', b', ξ') where $\xi_i' = 0$ and all other variables remain the same. Since the objective function includes a term $\frac{C}{2}\sum \xi_i^2$, by setting $\xi_i' = 0$, the objective value of the modified solution is reduced by $\frac{C}{2}(\xi_i')^2 = \frac{C}{2}(\xi_i)^2 > 0$.
Thus, we can always find a feasible solution with $\xi_i \geq 0$ that has the same or smaller objective value. Therefore, the non-negativity constraints $\xi_i \geq 0$ do not affect the optimal value of the objective.

(b) Considering:

$$f(w, \xi) = \frac{1}{2}||w||^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2$$
$$r_i(w, b, \xi_i) = -y_i(w^T x_i + b) - \xi_i + 1 \leq 0$$

The Lagrangian of the problem is given by:

$$L(w, b, \xi, \alpha, \gamma) = \frac{1}{2}\| w \|^2 + \frac{C}{2}\sum \xi_i^2 + \sum \alpha_i \left(1 - \xi_i - y_i(w^T x_i + b)\right)$$

(c) To minimize the Lagrangian with respect to w, b, and ξ, we set the partial derivatives with respect to these variables to zero.

$$\frac{\partial L}{\partial w} = w - \sum \alpha_i y_i x_i = 0$$

$$w = \sum \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C\xi_i - \alpha_i = 0$$

$$\xi_i = \frac{C}{\alpha_i}$$

(d) The dual problem is obtained by substituting the values of w and b from the Lagrangian into the Lagrangian function and simplifying. The dual problem is given by:

$$\max_{\alpha_i \geq 0, \gamma_i \geq 0} \min_{w,b,\xi} L(w, b, \xi, \alpha, \gamma)$$

subject to $\sum \alpha_i y_i = 0$ and $\alpha_i + \gamma_i = C\xi_i$ for all i

4. Gradient of cross-entropy loss over softmax:

> 4. **(15 points) Gradient of cross-entropy loss over softmax**. Let $\mathbf{y} \in \{0,1\}^d$ be a one-hot vector, i.e. $\mathbf{y}$ has a single entry which is 1 and the rest are 0. Consider the following loss function $\ell_{\mathbf{y}} : \mathbb{R}^d \to \mathbb{R}$:
>
> $$\ell_{\mathbf{y}}(\mathbf{w}) = -\mathbf{y} \cdot \log\left(\text{softmax}(\mathbf{w})\right),$$
>
> where $\text{softmax}(\mathbf{w}) = \frac{e^{\mathbf{w}}}{\sum_{j=1}^{d} e^{w_j}}$, is the softmax function (see slide 7 of lecture #7). In the above notation, the exponent and logarithm function operate elementwise when given a vector as an input. The function $\ell_{\mathbf{y}}$ is known as the *cross-entropy loss*, and you will encounter it in the programming assignment.
>
> Prove that the gradient of $\ell_{\mathbf{y}}$ with respect to $\mathbf{w}$ is given by:
>
> $$\nabla \ell_{\mathbf{y}}(\mathbf{w}) = \text{softmax}(\mathbf{w}) - \mathbf{y}$$

Solution:
To prove the gradient of ℓ_y with respect to w, I will use the chain rule and properties of the softmax function.
The cross-entropy loss $\ell_y(w)$ is defined as:

$$\ell_y(w) = -y \cdot log(softmax(w))$$

To find the gradient of ℓ_y with respect to w, I need to calculate the partial derivatives of ℓ_y with respect to each component of w. Let's consider the i-th component:

$$\left(\frac{\partial \ell_y}{\partial w_i}\right) = \left(\frac{\partial}{\partial w_i}\right)\left[-y \cdot log\big(softmax(w)\big)\right]$$

Using the chain rule, I can write this as:

$$\left(\frac{\partial \ell_y}{\partial w_i}\right) = -\frac{y_i}{softmax(w_i)} \cdot \left(\frac{\partial softmax(w_i)}{\partial w_i}\right)$$

Calculating $\left(\frac{\partial softmax(w_i)}{\partial w_i}\right)$, the partial derivative of $softmax(w_i)$ with respect to w_i:

$$softmax(w_i) = \frac{e_i^w}{\sum_j e_j^w}$$

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

Taking the derivative of $softmax(w_i)$ with respect to w_i:

$$\left(\frac{\partial softmax(w_i)}{\partial w_i}\right) = \left(\frac{\partial}{\partial w_i}\right)\left[\frac{e_i^w}{\sum_j e_j^w}\right]$$

I can simplify this using the quotient rule:

$$\left(\frac{\partial softmax(w_i)}{\partial w_i}\right) = \frac{\left(e_i^w * \left(\sum_j e_j^w\right) - e_i^w * e_i^w\right)}{\left(\sum_j e_j^w\right)^2}$$

Notice that $(\sum_j e_j^w)$ is simply the denominator of the softmax expression, which is $softmax(w_i)$ itself. Substituting that in, I get:

$$\left(\frac{\partial softmax(w_i)}{\partial w_i}\right) = \frac{e_i^w * softmax(w_i) - e_i^w * e_i^w}{\left(softmax(w_i)\right)^2}$$

We can further simplify this expression:

$$\left(\frac{\partial softmax(w_i)}{\partial w_i}\right) = e_i^w\left(\frac{softmax(w_i) - e_i^w}{\left(softmax(w_i)\right)^2}\right)$$

Now, substituting $\left(\frac{\partial softmax(w_i)}{\partial w_i}\right)$ back into the expression for $\left(\frac{\partial \ell_y}{\partial w_i}\right)$:

$$\left(\frac{\partial \ell_y}{\partial w_i}\right) = -\frac{y_i}{softmax(w_i)} * e_i^w * \frac{softmax(w_i) - e_i^w}{\left(softmax(w_i)\right)^2}$$

Simplifying:

$$\left(\frac{\partial \ell_y}{\partial w_i}\right) = -y_i * \left(1 - softmax(w_i)\right)$$

y is a one-hot vector, so $y_i = 1$ when i is the index of the non-zero entry in y. Otherwise, $y_i = 0$. Therefore, I can rewrite $\left(\frac{\partial \ell_y}{\partial w_i}\right)$ as:

$$\left(\frac{\partial \ell_y}{\partial w_i}\right) = softmax(w_i) - y_i$$

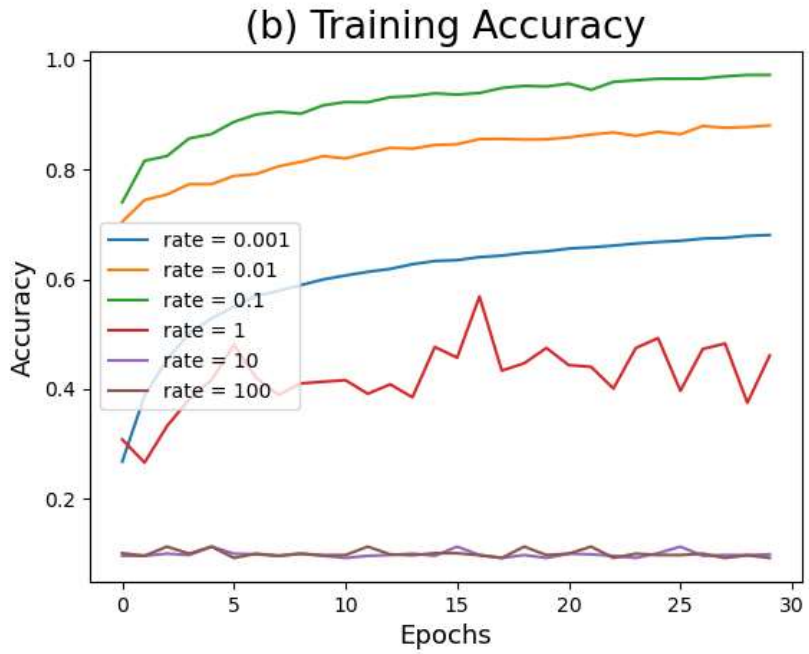
Since this holds for all components of w, I can conclude that the gradient of ℓ_y with respect to w is given by:

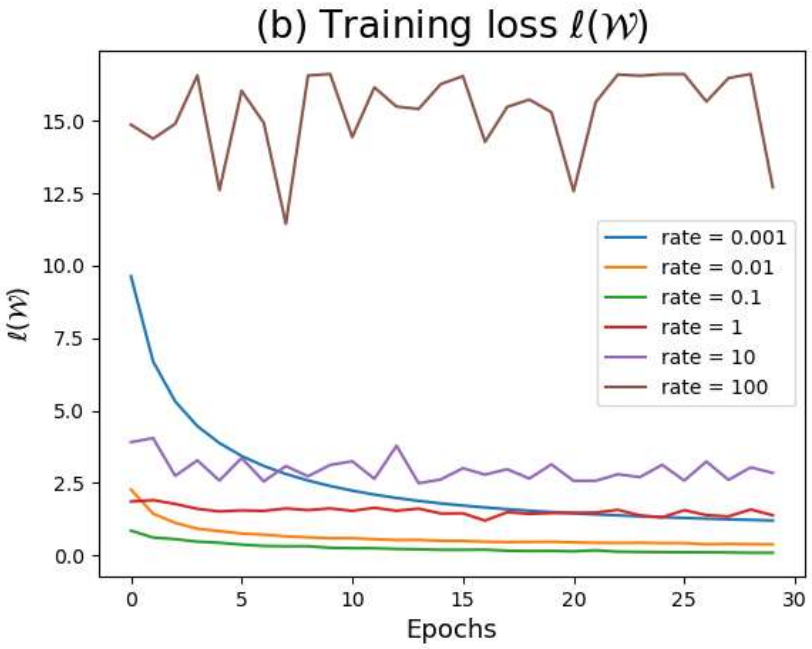
$$\nabla \ell_y(w) = softmax(w) - y \blacksquare$$

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

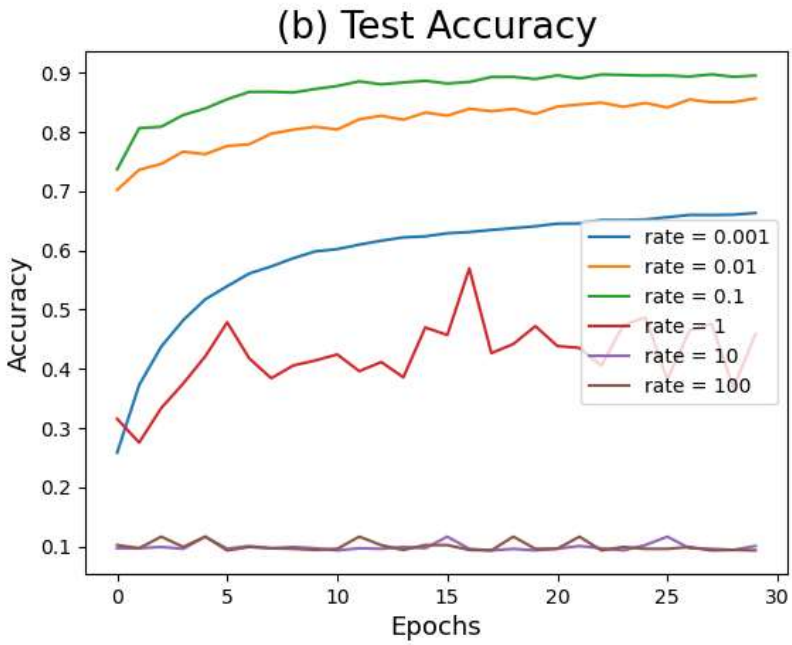
Coding Assignment:

1. Neural Networks:
   (b) Plots:





Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich



Regarding the phenomenon of using learning rates that are too small or too large, here's an explanation:

- Learning rate too small: When the learning rate is very small, the network updates its weights and biases very slowly. This leads to slow convergence or getting stuck in local minima. The training process may take a long time to reach an optimal solution, and the model may struggle to learn complex patterns in the data. The accuracy and loss curves show slow or limited improvement although seems like even the smallest tested learning rate wasn't as much excessively small as the largest learning rates were excessively large.
- Learning rate too large: Conversely, when the learning rate is excessively large, the network updates its weights and biases by large amounts in each iteration. This can cause overshooting the optimal solution and prevent convergence. The updates become unstable, leading to erratic behavior and divergence. The accuracy and loss curves may fluctuate or diverge entirely. The results do show this, as can be seen in the test accuracy graph, the two largest learning rates performed very badly, resulting with very low accuracy throughout the experiment.

(c) The test accuracy in the final epoch is 0.943 (the full results are attached in the appendix of this file).

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

2. (a) Those are the results I have got:

Optimizer Type	Learning Rate	Loss	Validation Accuracy	Test Accuracy
SGD	0.005	1.5785	42.90%	42.81%
SGD	0.005	1.1545	50.94%	50.58%
SGD	0.005	0.95	63.78%	63.35%
SGD	0.005	0.8295	61.28%	62.23%
SGD	0.005	0.7439	48.88%	47.86%
SGD	0.005	0.6837	53.22%	53.85%
SGD	0.005	0.6291	64.94%	64.71%
SGD	0.005	0.5746	61.40%	61.63%
SGD	0.005	0.5333	74.28%	74.43%
SGD	0.005	0.4939	75.52%	75.07%
Adam	5.00E-05	1.29	65.48%	64.97%
Adam	5.00E-05	0.8741	72.70%	72.24%
Adam	5.00E-05	0.7221	76.10%	75.73%
Adam	5.00E-05	0.6272	76.64%	76.30%
Adam	5.00E-05	0.552	78.32%	77.68%
Adam	5.00E-05	0.4949	78.98%	79.15%
Adam	5.00E-05	0.4447	79.92%	79.90%
Adam	5.00E-05	0.4054	81.04%	81.17%
Adam	5.00E-05	0.373	80.52%	80.53%
Adam	5.00E-05	0.3334	81.18%	81.71%

Analyzing these results:

SGD Optimizer with learning rate = 0.005:

- The training loss decreases gradually over the epochs, indicating that the model is learning from the data.
- The validation accuracy and test accuracy show an increasing trend, reaching around 75% at the end of training.

Adam Optimizer with learning rate = 5e-05:

- The training loss decreases more rapidly compared to the SGD optimizer, indicating faster convergence.
- The validation accuracy and test accuracy are consistently higher compared to the SGD optimizer, reaching around 81% at the end of training.

Introduction to Machine Learning (0368-3235) - Assignment 4
Lior Erenreich

Overall, using the Adam optimizer with a learning rate of 5e-05 yields better results compared to the SGD optimizer with a learning rate of 0.005. The Adam optimizer shows faster convergence and achieves higher accuracy on both the validation and test sets.

(b) Those are the results I have got:

Optimizer Type	Learning Rate	Loss	Validation Accuracy	Test Accuracy
Adam	0.005	4.288	12.92%	12.84%
Adam	0.005	2.2498	16.36%	16.40%
Adam	0.005	2.1538	15.04%	15.98%
Adam	0.005	2.135	16.80%	16.64%
Adam	0.005	2.1445	14.70%	15.02%
Adam	0.005	2.1146	17.02%	16.88%
Adam	0.005	2.1255	16.54%	16.20%
Adam	0.005	2.2486	15.30%	16.49%
Adam	0.005	1.9108	27.24%	28.40%
Adam	0.005	1.7545	34.54%	34.95%

My observations:

Adam Optimizer with learning rate = 0.005:

- The training loss decreases initially, but then it starts to fluctuate and even increases slightly over the epochs. This indicates that the model is unable to converge and might be overshooting the optimal solution.
- The validation accuracy and test accuracy show a low and fluctuating trend, ranging from around 12% to 34% throughout the training.

Compared to the previous case where the Adam optimizer was used with a learning rate of 5e-05, the training process with the higher learning rate of 0.005 is affected negatively. The model struggles to converge, resulting in higher training loss and lower accuracy on both the validation and test sets. It seems that the learning rate of 0.005 is too high for this model and dataset, causing instability and hindered learning.

(c) Those are the results I have got:

Epoch	Training Loss	Validation Accuracy	Test Accuracy
1	1.549	55.64%	55.41%
2	1.0797	65.32%	65.87%
3	0.8387	71.84%	70.58%

4	0.6641	74.50%	73.99%
5	0.5324	77.50%	76.42%
6	0.4104	78.28%	78.44%
7	0.2936	80.46%	78.36%
8	0.1859	79.72%	78.62%
9	0.1091	80.06%	79.42%
10	0.0741	78.52%	78.71%

The network is trained using the Adam optimizer with a learning rate of 5e-05, but the architecture does not include Batch Normalization or Dropout layers. Here are my observations:

The training process without Batch Normalization or Dropout layers shows the following characteristics:

- The training loss decreases steadily over the epochs, indicating that the model is learning and converging effectively.
- The validation accuracy and test accuracy show an increasing trend over the epochs, reaching around 78% accuracy on both sets by the end of training.

Comparison with the previous training process that included Batch Normalization and Dropout layers:

- The training loss evolves similarly, steadily decreasing in both cases. However, in this case without Batch Normalization or Dropout, the loss seems to decrease slightly faster.
- The additions of Batch Normalization and Dropout layers do seem to significantly accelerate the learning process in this particular case.

Overall, without Batch Normalization and Dropout layers, the model achieves an accuracy of approximately 78-80% on both the validation and test sets at the end of the training. This is a substantial improvement compared to the accuracy achieved with BatchNorm + Dropout, which was around 34%.

The absence of Batch Normalization and Dropout layers might have allowed the model to learn more freely and avoid potential over-regularization.

Appendix:

The full printed results on the first programming question:

Initial test accuracy: 0.0838

Epoch 1 test accuracy: 0.7066

Epoch 2 test accuracy: 0.784

Epoch 3 test accuracy: 0.793

Epoch 4 test accuracy: 0.8344

Epoch 5 test accuracy: 0.8462

Epoch 6 test accuracy: 0.8508

Epoch 7 test accuracy: 0.8674

Epoch 8 test accuracy: 0.8576

Epoch 9 test accuracy: 0.8652

Epoch 10 test accuracy: 0.8736

Epoch 11 test accuracy: 0.876

Epoch 12 test accuracy: 0.881

Epoch 13 test accuracy: 0.8796

Epoch 14 test accuracy: 0.8852

Epoch 15 test accuracy: 0.8812

Epoch 16 test accuracy: 0.8858

Epoch 17 test accuracy: 0.8784

Epoch 18 test accuracy: 0.8892

Epoch 19 test accuracy: 0.889

Epoch 20 test accuracy: 0.8922

Epoch 21 test accuracy: 0.8904

Epoch 22 test accuracy: 0.8854

Epoch 23 test accuracy: 0.8962

Epoch 24 test accuracy: 0.8938

Epoch 25 test accuracy: 0.8948

Epoch 26 test accuracy: 0.8958

Epoch 27 test accuracy: 0.8912

Epoch 28 test accuracy: 0.8966

Epoch 29 test accuracy: 0.8936

Epoch 30 test accuracy: 0.8912

-----(a)------

Initial test accuracy: 0.0696

Epoch 1 test accuracy: 0.707

Epoch 2 test accuracy: 0.7586

Epoch 3 test accuracy: 0.817

Epoch 4 test accuracy: 0.8204

Epoch 5 test accuracy: 0.8488

Epoch 6 test accuracy: 0.8614

Epoch 7 test accuracy: 0.8578

Epoch 8 test accuracy: 0.8618

Epoch 9 test accuracy: 0.8666

Epoch 10 test accuracy: 0.879

Epoch 11 test accuracy: 0.8802

Epoch 12 test accuracy: 0.8724

Epoch 13 test accuracy: 0.89

Epoch 14 test accuracy: 0.8806

Epoch 15 test accuracy: 0.887

Epoch 16 test accuracy: 0.8896

Epoch 17 test accuracy: 0.8894

Epoch 18 test accuracy: 0.8894

Epoch 19 test accuracy: 0.8966

Epoch 20 test accuracy: 0.899

Epoch 21 test accuracy: 0.8982

Epoch 22 test accuracy: 0.8988

Epoch 23 test accuracy: 0.8928

Epoch 24 test accuracy: 0.8996

Epoch 25 test accuracy: 0.8892

Epoch 26 test accuracy: 0.902

Epoch 27 test accuracy: 0.9034

Epoch 28 test accuracy: 0.9062

Epoch 29 test accuracy: 0.8998

Epoch 30 test accuracy: 0.9044

-----(b)------

Rate 0.001

Introduction to Machine Learning (0368-3235) - Assignment 4

Lior Erenreich

Initial test accuracy: 0.1058

Epoch 1 test accuracy: 0.2586

Epoch 2 test accuracy: 0.3728

Epoch 3 test accuracy: 0.4374

Epoch 4 test accuracy: 0.4822

Epoch 5 test accuracy: 0.5174

Epoch 6 test accuracy: 0.5396

Epoch 7 test accuracy: 0.561

Epoch 8 test accuracy: 0.573

Epoch 9 test accuracy: 0.5868

Epoch 10 test accuracy: 0.5984

Epoch 11 test accuracy: 0.6022

Epoch 12 test accuracy: 0.6098

Epoch 13 test accuracy: 0.6164

Epoch 14 test accuracy: 0.622

Epoch 15 test accuracy: 0.6238

Epoch 16 test accuracy: 0.629

Epoch 17 test accuracy: 0.631

Epoch 18 test accuracy: 0.6346

Epoch 19 test accuracy: 0.6376

Epoch 20 test accuracy: 0.6408

Epoch 21 test accuracy: 0.6452

Epoch 22 test accuracy: 0.6456

Epoch 23 test accuracy: 0.6514

Epoch 24 test accuracy: 0.6512

Epoch 25 test accuracy: 0.6522

Epoch 26 test accuracy: 0.656

Epoch 27 test accuracy: 0.66

Epoch 28 test accuracy: 0.6598

Epoch 29 test accuracy: 0.6604

Epoch 30 test accuracy: 0.6632

(b) Rate 0.01

Initial test accuracy: 0.1348

Epoch 1 test accuracy: 0.7022

Epoch 2 test accuracy: 0.7362

Epoch 3 test accuracy: 0.746

Epoch 4 test accuracy: 0.7668

Epoch 5 test accuracy: 0.7624

Epoch 6 test accuracy: 0.7764

Epoch 7 test accuracy: 0.7792

Epoch 8 test accuracy: 0.7972

Epoch 9 test accuracy: 0.8038

Epoch 10 test accuracy: 0.8088

Epoch 11 test accuracy: 0.8042

Epoch 12 test accuracy: 0.8214

Epoch 13 test accuracy: 0.8274

Epoch 14 test accuracy: 0.8206

Epoch 15 test accuracy: 0.8328

Epoch 16 test accuracy: 0.8276

Epoch 17 test accuracy: 0.8392

Epoch 18 test accuracy: 0.835

Epoch 19 test accuracy: 0.839

Epoch 20 test accuracy: 0.8306

Epoch 21 test accuracy: 0.8428

Epoch 22 test accuracy: 0.8464

Epoch 23 test accuracy: 0.8496

Epoch 24 test accuracy: 0.8424

Epoch 25 test accuracy: 0.849

Epoch 26 test accuracy: 0.8412

Epoch 27 test accuracy: 0.8548

Epoch 28 test accuracy: 0.8502

Epoch 29 test accuracy: 0.8504

Epoch 30 test accuracy: 0.8566

(b) Rate 0.1

Initial test accuracy: 0.1068

Epoch 1 test accuracy: 0.737

Epoch 2 test accuracy: 0.8064

Epoch 3 test accuracy: 0.8088

Epoch 4 test accuracy: 0.8286

Epoch 5 test accuracy: 0.8396

Epoch 6 test accuracy: 0.8552

Epoch 7 test accuracy: 0.8676

Epoch 8 test accuracy: 0.8678

Epoch 9 test accuracy: 0.8666

Epoch 10 test accuracy: 0.8726

Epoch 11 test accuracy: 0.8776

Epoch 12 test accuracy: 0.8854

Epoch 13 test accuracy: 0.8804

Epoch 14 test accuracy: 0.8836

Epoch 15 test accuracy: 0.8864

Epoch 16 test accuracy: 0.8818

Epoch 17 test accuracy: 0.8844

Epoch 18 test accuracy: 0.8928

Epoch 19 test accuracy: 0.893

Epoch 20 test accuracy: 0.8894

Epoch 21 test accuracy: 0.8956

Epoch 22 test accuracy: 0.8904

Epoch 23 test accuracy: 0.897

Epoch 24 test accuracy: 0.8962

Epoch 25 test accuracy: 0.8952

Epoch 26 test accuracy: 0.8954

Epoch 27 test accuracy: 0.8936

Epoch 28 test accuracy: 0.8972

Epoch 29 test accuracy: 0.8932

Epoch 30 test accuracy: 0.8952

(b) Rate 1

Initial test accuracy: 0.0882

Epoch 1 test accuracy: 0.3156

Epoch 2 test accuracy: 0.2756

Epoch 3 test accuracy: 0.3338

Epoch 4 test accuracy: 0.3752

Epoch 5 test accuracy: 0.421

Epoch 6 test accuracy: 0.4786

Epoch 7 test accuracy: 0.4178

Epoch 8 test accuracy: 0.3842

Epoch 9 test accuracy: 0.4058

Epoch 10 test accuracy: 0.4142

Epoch 11 test accuracy: 0.4244

Epoch 12 test accuracy: 0.3962

Epoch 13 test accuracy: 0.4116

Epoch 14 test accuracy: 0.3858

Epoch 15 test accuracy: 0.47

Epoch 16 test accuracy: 0.4572

Epoch 17 test accuracy: 0.5696

Epoch 18 test accuracy: 0.4266

Epoch 19 test accuracy: 0.442

Epoch 20 test accuracy: 0.4722

Epoch 21 test accuracy: 0.4386

Epoch 22 test accuracy: 0.4354

Epoch 23 test accuracy: 0.4056

Epoch 24 test accuracy: 0.474

Epoch 25 test accuracy: 0.487

Epoch 26 test accuracy: 0.3822

Epoch 27 test accuracy: 0.465

Epoch 28 test accuracy: 0.4764

Epoch 29 test accuracy: 0.368

Epoch 30 test accuracy: 0.4588

(b) Rate 10

Initial test accuracy: 0.1234

Epoch 1 test accuracy: 0.0972

Epoch 2 test accuracy: 0.0972

Epoch 3 test accuracy: 0.0994

Epoch 4 test accuracy: 0.0962

Epoch 5 test accuracy: 0.1168

Epoch 6 test accuracy: 0.0964

Epoch 7 test accuracy: 0.101

Epoch 8 test accuracy: 0.0972

Epoch 9 test accuracy: 0.0994

Epoch 10 test accuracy: 0.0972

Epoch 11 test accuracy: 0.0936

Epoch 12 test accuracy: 0.0972

Epoch 13 test accuracy: 0.0962

Epoch 14 test accuracy: 0.0994

Epoch 15 test accuracy: 0.0972

Epoch 16 test accuracy: 0.1168

Epoch 17 test accuracy: 0.0962

Epoch 18 test accuracy: 0.0936

Epoch 19 test accuracy: 0.0962

Epoch 20 test accuracy: 0.0936

Epoch 21 test accuracy: 0.0964

Epoch 22 test accuracy: 0.101

Epoch 23 test accuracy: 0.0972

Epoch 24 test accuracy: 0.0936

Epoch 25 test accuracy: 0.1026

Epoch 26 test accuracy: 0.1168

Epoch 27 test accuracy: 0.0972

Epoch 28 test accuracy: 0.0962

Epoch 29 test accuracy: 0.0944

Epoch 30 test accuracy: 0.101

(b) Rate 100

Initial test accuracy: 0.052

Epoch 1 test accuracy: 0.1026

Epoch 2 test accuracy: 0.0972

Epoch 3 test accuracy: 0.1168

Epoch 4 test accuracy: 0.0994

Epoch 5 test accuracy: 0.1168

Epoch 6 test accuracy: 0.0936

Epoch 7 test accuracy: 0.0994

Epoch 8 test accuracy: 0.0972

Epoch 9 test accuracy: 0.0964

Epoch 10 test accuracy: 0.0944

Epoch 11 test accuracy: 0.0962

Epoch 12 test accuracy: 0.1168

Epoch 13 test accuracy: 0.1024

Epoch 14 test accuracy: 0.0944

Epoch 15 test accuracy: 0.1026

Epoch 16 test accuracy: 0.1026

Epoch 17 test accuracy: 0.0944

Epoch 18 test accuracy: 0.0936

Epoch 19 test accuracy: 0.1168

Epoch 20 test accuracy: 0.0962

Epoch 21 test accuracy: 0.0964

Epoch 22 test accuracy: 0.1168

Epoch 23 test accuracy: 0.0936

Epoch 24 test accuracy: 0.0994

Epoch 25 test accuracy: 0.0962

Epoch 26 test accuracy: 0.0962

Epoch 27 test accuracy: 0.0994

Epoch 28 test accuracy: 0.0936

Epoch 29 test accuracy: 0.0944

Epoch 30 test accuracy: 0.0936

-----(c)------

Initial test accuracy: 0.0809

Epoch 1 test accuracy: 0.8164

Epoch 2 test accuracy: 0.8881

Epoch 3 test accuracy: 0.8887

Epoch 4 test accuracy: 0.912

Epoch 5 test accuracy: 0.9126

Epoch 6 test accuracy: 0.9219

Epoch 7 test accuracy: 0.9231

Epoch 8 test accuracy: 0.9219

Epoch 9 test accuracy: 0.926

Epoch 10 test accuracy: 0.9195

Epoch 11 test accuracy: 0.9271

Epoch 12 test accuracy: 0.9326

Epoch 13 test accuracy: 0.9336

Epoch 14 test accuracy: 0.9361

Epoch 15 test accuracy: 0.934

Epoch 16 test accuracy: 0.9333

Epoch 17 test accuracy: 0.9359

Epoch 18 test accuracy: 0.9323

Epoch 19 test accuracy: 0.9391

Epoch 20 test accuracy: 0.9376

Epoch 21 test accuracy: 0.9389

Epoch 22 test accuracy: 0.942

Epoch 23 test accuracy: 0.9417

Epoch 24 test accuracy: 0.9388

Epoch 25 test accuracy: 0.9363

Epoch 26 test accuracy: 0.9381

Epoch 27 test accuracy: 0.9455

Epoch 28 test accuracy: 0.9448

Epoch 29 test accuracy: 0.9415

Epoch 30 test accuracy: 0.943

Process finished with exit code 0