Deep Learning: Homework #2

Due on April 28, 2021

Professor Zhen Li

Peng Deng

Problem 1

Equivariance is an appealing property when design neural network operations. It means that transforming the input image (e.g., translation) will also transform the output feature maps similarly after certain operations.

Formally, denote the image coordinate by $x \in \mathbb{Z}^2$, and the pixel values at each coordinate by a function $f : \mathbb{Z}^2 \to \mathbb{Z}^K$, where K is the number of image channels. A convolution filter can also be formulated as a function $w : \mathbb{Z}^2 \to \mathbb{Z}^K$. Note that f and w are zero outside the image and filter kernel region, respectively. The convolution operation (correlation indeed for simplicity) is thus defined by

$$[f * w](x) = \sum_{\mathbf{y} \in \mathbb{Z}^2} \sum_{k=1}^{K} f_k(y) w_k(y - x). \tag{1}$$

(a). Let L_{t} be the translation $x \to x + t$ on the image or feature map, i.e., $[L_{\mathrm{t}} f](x) = f(x - t)$. Prove that convolution has equivariance to translation:

$$[[L_{\mathrm{t}} f] * w](x) = [L_{\mathrm{t}}[f * w]](x)$$

which means that first translating the input image then doing the convolution is equivalent to first convolving with the image and then translating the output feature map. (Hints: Use the formula (1) for the proof.)

(b). Let $L_{\mathbf{R}}$ be the 90° -rotation on the image or feature map, where

$$\mathbf{R} = \begin{bmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{bmatrix}$$

then $[L_{\mathrm{R}} f](x) = f\left(\mathbf{R}^{-1} x\right)$. However, convolution is not equivariant to rotations, i.e., $[L_{\mathrm{R}} f] * w \neq L_{\mathrm{R}}[f * w]$, which is illustrated by Figure 1 ((a) is not equivalent to (b) rotated by 90°). In order to establish the equivalence, the filter also needs to be rotated (i.e. (b) is equivalent to (c) in Figure 1). Prove that:

$$[[L_{\mathrm{R}} f] * w](x) = L_{\mathbf{R}}\left[f * [L_{\mathbf{R}^{-1}} w]\right](x)$$

(Hints: Use the formula (1) for the proof.)

To make convolution equivariant to rotations, we need to extend the definition of convolution and transformation. Recall a group $(G, \otimes)$ in algebra is a set G, together with an binary operation $\otimes$, which satisfies four requirements:

Closure $a \otimes b \in G, \forall a, b \in G$

Associativity $(a \otimes b) \otimes c = a \otimes (b \otimes c), \forall a, b, c \in G$.

Identity element There exists a unique $e \in G, e \otimes a = a \otimes e = a, \forall a \in G$.

Inverse element $\forall a \in G, \exists a^{-1} \in G, a \otimes a^{-1} = a^{-1} \otimes a = e$.

We can formulate 90° -rotation and translation by a group $(G, \otimes)$ consisting of

$$g(r, u, v) = \begin{bmatrix} \cos(r\pi/2) & -\sin(r\pi/2) & u \\ \sin(r\pi/2) & \cos(r\pi/2) & v \\ 0 & 0 & 1 \end{bmatrix}$$

where $r \in \{0, 1, 2, 3\}$ and $(u, v) \in \mathbb{Z}^2 . G = \{g\}$ and $\otimes$ is matrix multiplication. Translation is a special case of G when $r = 0$ (i.e., $g(0, u, v)$) and rotation is a special case of G when $u = v = 0$ (i.e., $g(r, 0, 0)$).

A key concept is to extend the definition of both the feature f and the filter w to G. Imagine the feature map is duplicated four times with rotation of $0°, 90°, 180°$, and $270°$. Then $f(g)$ is the feature values at particular rotated pixel coordinate, and the convolution operation becomes

$$[f * w](g) = \sum_{\mathbf{h} \in G} \sum_{k=1}^{K} f_k(\mathbf{h}) w_k\left(g^{-1}\mathbf{h}\right).$$

Problem 1 continued on next page...

A rotation-translation $u \in G$ on the feature map is thus $[L_u f](g) = f\left(u^{-1}g\right)$. Prove that under such extensions, the convolution is equivariant to rotation-translation:

$$[[L_u f] * w](g) = L_u[f * w](g)$$

c. Briefly explain how to implement this group convolution with traditional convolution and by rotating the feature map or filter. (Hints: Please read the paper "Group Equivariant Convolutional Networks").

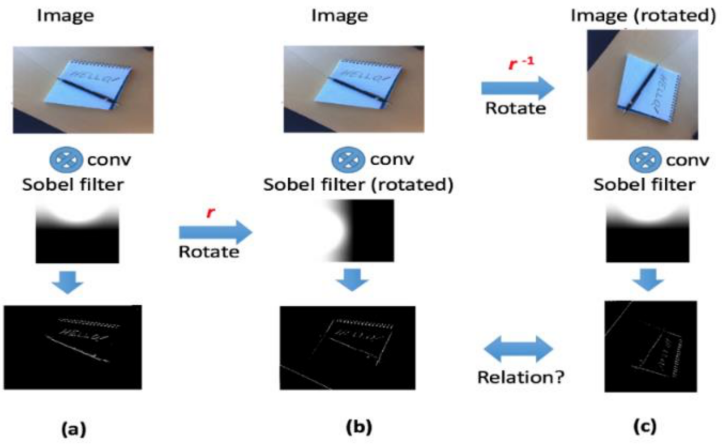


Figure 1 Equivariance relationship between convolution and rotation. (a) An image is convolved with a Sobel filer to detect horizontal edges. (b) The filter is rotated counterclockwise and then convolves the original image. (c) The image is first rotated clockwise, then it is convolved with the filter.

Solution

Subproblem(a)

$$\begin{aligned}
[[L_{\mathbf{t}} f] * w](\boldsymbol{x}) &= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} [L_{\mathbf{t}} f]_k(\boldsymbol{y}) w_k(\boldsymbol{y}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}-\mathbf{t}) w_k(\boldsymbol{y}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}) w_k(\boldsymbol{y}+\boldsymbol{t}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}) w_k(\boldsymbol{y}-(\boldsymbol{x}-\boldsymbol{t})) \\
[L_{\mathbf{t}}[f * w]](\boldsymbol{x}) &= [f * w](\boldsymbol{x}-\boldsymbol{t}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}) w_k(\boldsymbol{y}-(\boldsymbol{x}-\boldsymbol{t}))
\end{aligned} \tag{2}$$

Thus, we finished the proof.

Subproblem(b)

$$
\begin{aligned}
[[L_{\mathbf{R}} f] * w](\mathbf{x}) &= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K}[L_{\mathbf{R}} f]_k(\boldsymbol{y}) w_k(\boldsymbol{y}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k\left(\mathbf{R}^{-1}\boldsymbol{y}\right) w_k(\boldsymbol{y}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}) w_k(\mathbf{R}\boldsymbol{y}-\boldsymbol{x}) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y}) w_k\left(\mathbf{R}\left(\boldsymbol{y}-\mathbf{R}^{-1}\boldsymbol{x}\right)\right) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y})\left([L_{\mathbf{R}^{-1}} w]_k\left(\boldsymbol{y}-\mathbf{R}^{-1}\boldsymbol{x}\right)\right) \\
L_{\mathbf{R}}\left[f * [L_{\mathbf{R}^{-1}} w]\right](\boldsymbol{x}) &= [f * [L_{\mathbf{R}^{-1}} w]]\left(\mathbf{R}^{-1}\boldsymbol{x}\right) \\
&= \sum_{\boldsymbol{y}\in\mathbb{Z}^2}\sum_{k=1}^{K} f_k(\boldsymbol{y})\left([L_{\mathbf{R}^{-1}} w]_k\left(\boldsymbol{y}-\mathbf{R}^{-1}\boldsymbol{x}\right)\right)
\end{aligned}
\tag{3}
$$

Thus, we finished the proof.

Subproblem(c)

$$
\begin{aligned}
[[L_{\mathbf{u}} f] * w](\mathbf{g}) &= \sum_{\mathbf{h}\in G}\sum_{k=1}^{K}[L_{\mathbf{u}} f]_k(\mathbf{h}) w_k\left(\mathbf{g}^{-1}\mathbf{h}\right) \\
&= \sum_{\mathbf{h}\in G}\sum_{k=1}^{K} f_k\left(\mathbf{u}^{-1}\mathbf{h}\right) w_k\left(\mathbf{g}^{-1}\mathbf{h}\right) \\
&= \sum_{\mathbf{h}\in G}\sum_{k=1}^{K} f_k(\mathbf{h}) w_k\left(\mathbf{g}^{-1}\mathbf{u}\mathbf{h}\right) \\
&= \sum_{\mathbf{h}\in G}\sum_{k=1}^{K} f_k(\mathbf{h}) w_k\left(\left(\mathbf{u}^{-1}\mathbf{g}\right)^{-1}\mathbf{h}\right) \\
&= [f * w]\left(\mathbf{u}^{-1}\mathbf{g}\right) = [L_{\mathbf{u}}[f * w]](\mathbf{g})
\end{aligned}
\tag{4}
$$

Thus, we finished the proof.

To implement this group convolution, we just need to

(a) duplicate and rotate the input image by $\{0°, 90°, 180°, 270°\}$, which leads to four times of input channels;

(b) duplicate and rotate the original filters by $\{0°, 90°, 180°, 270°\}$.

Problem 2

A recurrent neural network is shown in Figure 2.

$$
\begin{aligned}
h_t &= \tanh\left(\mathbf{W}_{xh} x_t + \mathbf{W}_{hh} h_{t-1} + b_h\right) \\
z_t &= \operatorname{softmax}\left(\mathbf{W}_{hz} h_t + b_z\right)
\end{aligned}
$$

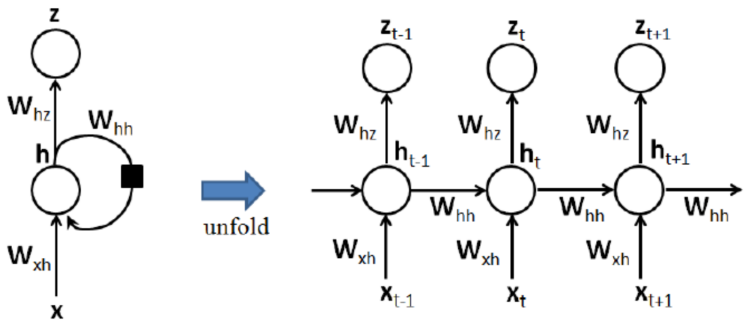


Figure 2 RNN fold and unfold structure.

The total loss for a given input/target sequence pair (x, y), measured in cross entropy

$$L(x, y) = \sum_t L_t = \sum_t -\log z_{t,y_t}$$

In the lecture, we provide the general idea of how to calculate the gradients $\frac{\partial L}{\partial W_{hz}}$ and $\frac{\partial L}{\partial W_{hh}}$. Please provide the details of the algorithms and equations, considering the mapping and cost functions provided above.

Solution

Part (1)

Let $\boldsymbol{u_t} = \boldsymbol{W}_{hz}\boldsymbol{h}_t + \boldsymbol{b}_z$, so $\boldsymbol{z}_t = \text{softmax}(\boldsymbol{u}_t)$. Let $\boldsymbol{y}_t$ be the one-hot representation of label at time t. Then we have

$$\begin{aligned} \frac{\partial L_t}{\partial \boldsymbol{u}_t} &= \frac{\partial L_t}{\partial \boldsymbol{z}_t} \cdot \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{u}_t} = \boldsymbol{z}_t - \boldsymbol{y}_t \\ \Longrightarrow \frac{\partial L_t}{\partial \boldsymbol{u}_{t,i}} &= \boldsymbol{z}_{t,i} - \boldsymbol{y}_{t,i} \end{aligned} \tag{5}$$

Then, we can derive $\frac{\partial \boldsymbol{u}_{t,i}}{\partial \boldsymbol{W}_{hz,ij}}$ as follow

$$\frac{\partial \boldsymbol{u}_{t,i}}{\partial \boldsymbol{W}_{hz,ij}} = \boldsymbol{h}_{t,j} \tag{6}$$

Then, we can derive $\frac{\partial L_t}{\partial \boldsymbol{W}_{hz}}$

$$\begin{aligned} \frac{\partial L_t}{\partial \boldsymbol{W}_{hz,ij}} &= \frac{\partial L_t}{\partial \boldsymbol{u}_{t,i}} \cdot \frac{\partial \boldsymbol{u}_{t,i}}{\partial \boldsymbol{W}_{hz,ij}} \\ &= \left(\boldsymbol{z}_{t,i} - \boldsymbol{y}_{t,i}\right) \cdot \boldsymbol{h}_{t,j} \\ \Longrightarrow \frac{\partial L_t}{\partial \boldsymbol{W}_{hz}} &= \left(\boldsymbol{z}_t - \boldsymbol{y}_t\right) \cdot \boldsymbol{h}_t^\top \end{aligned} \tag{7}$$

Finally, we can derive $\frac{\partial L}{\partial \boldsymbol{W}_{hz}}$ as follow

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{W}_{hz}} &= \frac{\partial \sum_t L_t}{\partial \boldsymbol{W}_{hz}} \\ &= \sum_t \frac{\partial L_t}{\partial \boldsymbol{W}_{hz}} \\ &= \sum_t \left(\boldsymbol{z}_t - \boldsymbol{y}_t\right) \cdot \boldsymbol{h}_t^\top \end{aligned} \tag{8}$$

Part (2)

Let $\boldsymbol{v}_t = \mathbf{W}_{xh}\boldsymbol{x}_t + \mathbf{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h$, then we have

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_{hh}} &= \sum_t \frac{\partial L}{\partial \boldsymbol{h}_t} \cdot \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{W}_{hh}} \\
\Longrightarrow \frac{\partial L}{\partial \boldsymbol{W}_{hh,ij}} &= \sum_t \frac{\partial L}{\partial \boldsymbol{h}_{t,i}} \cdot \frac{\partial \boldsymbol{h}_{t,i}}{\partial \boldsymbol{W}_{hh,ij}} \\
&= \sum_t \frac{\partial L}{\partial \boldsymbol{h}_{t,i}} \cdot \frac{\partial \boldsymbol{h}_{t,i}}{\partial \boldsymbol{v}_{t,i}} \cdot \frac{\partial \boldsymbol{v}_{t,i}}{\partial \boldsymbol{W}_{hh,ij}} \\
&= \sum_t \frac{\partial L}{\partial \boldsymbol{h}_{t,i}} \cdot \left(1 - \boldsymbol{h}_{t,i}^2\right) \cdot \boldsymbol{h}_{t-1,j}
\end{aligned} \tag{9}$$

We can derive $\frac{\partial L}{\partial \boldsymbol{h}_t}$ as follow

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{h}_t} &= \frac{\partial L}{\partial \boldsymbol{h}_{t+1}} \cdot \frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{h}_t} + \frac{\partial L}{\partial \boldsymbol{u}_t} \cdot \frac{\partial \boldsymbol{u}_t}{\partial \boldsymbol{h}_t} \\
&= \frac{\partial L}{\partial \boldsymbol{h}_{t+1}} \cdot \frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{v}_{t+1}} \cdot \frac{\partial \boldsymbol{v}_{t+1}}{\partial \boldsymbol{h}_t} + \left(\boldsymbol{z}_t - \boldsymbol{y}_t\right)^\top \boldsymbol{W}_{hz} \\
&= \frac{\partial L}{\partial \boldsymbol{h}_{t+1}} \cdot \operatorname{diag}\left(1\text{-}\boldsymbol{h}_{t+1}^2\right) \cdot \boldsymbol{W}_{hh} + \left(\boldsymbol{z}_t - \boldsymbol{y}_t\right)^\top \boldsymbol{W}_{hz}
\end{aligned} \tag{10}$$

The term $\frac{\partial L}{\partial \boldsymbol{h}_{t,i}}$ in equation 9 can be derived from equation 10. Besides, the term is a recursive term.

Problem 3

Consider an episodic, deterministic chain MDP with n = 7 states assembled in a line. The possible actions are $a \in \{-1, 1\}$, and the transition function is deterministic such that $s' = s + a$. Note that as an exception, taking $a = -1$ from $s = 1$ keeps us in $s = 1$, and taking $a = -1$ from $s = 7$ keeps us in $s = 7$.
We have a special goal state, $g = 4$, such that taking any action from g ends the episode with a reward of $r = 0$. From all other states, any action incurs a reward of $r = -1$. We let discount factor $\gamma = 1/3$.

The chain MDP is pictured in Figure 3, with the goal state s_4 shaded in.

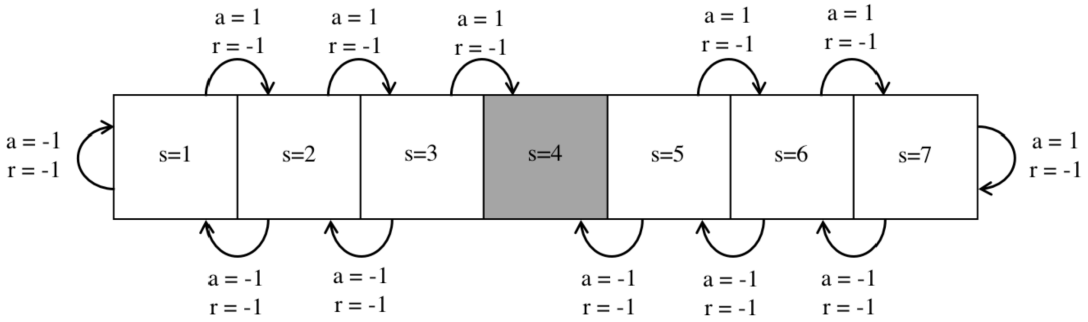


Figure 3 MDP procedure with seven states.

By inspection, we see that $V^* = -|s - 4|$

a. We would like to perform tabular Q-learning on this chain MDP. Suppose we observe the following 4 step trajectory (in the form (state, action, reward)):

$$(3, -1, -1), (2, 1, -1), (3, 1, -1), (4, 1, 0)$$

Suppose we initialize all Q values to 0 . Use the tabular Q-learning update to give updated values for

$$Q(3,-1), Q(2,1), Q(3,1)$$

assuming we process the trajectory in the order given from left to right. Use the learning rate $\alpha = 1/2$.

[Hints: Using $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a' \in \{1,-1\}} Q\left(s', a'\right) - Q(s,a)\right)$]

Now, we are interested in performing linear function approximation in conjunction with Qlearning. In particular, we have a weight vector $w = [w_0, w_1, w_2]^T \in \mathbb{R}^3$. Given some state s and action $a \in \{-1, 1\}$, the featurization of this state, action pair is: $[s, a, 1]^T$. To approximate the Q - values, we compute

$$\hat{q}(s, a; w) = [w_0, w_1, w_2]\,[s, a, 1]^T = w_0 * s + w_1 * a + w_2$$

Given the parameters w and a single sample (s, a, r, s'). The loss function we will minimize is

$$J(w) = \left(r + \gamma \max_{a'} \hat{q}\left(s', a'; w^-\right) - \hat{q}(s, a; w)\right)^2$$

where $\hat{q}\left(s', a'; w^-\right)$ is a target network parameterized by fixed weights w^-.

b. Suppose we currently have a weight vectors $w = [-1, 1, 1]^T$ and $w^- = [1, -1, -2]^T$, and we observe a sample $(s = 2, a = -1, r = -1, s' = 1)$. Perform a single gradient update to the parameters w given this sample. Use the learning rate $\alpha = 1/4$. Write out the gradient $\nabla_w J(w)$ as well as the new parameters w'.

c. The optimal Q function $Q^*(s, a)$ is exactly representable by some neural network architecture N. Suppose we perform Q-learning on this MDP using the architecture N to represent the Q-values. Suppose we randomly initialize the weights of a neural net with architecture N and collect infinitely many samples using an exploration strategy that is greedy in the limit of infinite exploration (GLIE). Are we guaranteed to converge to the optimal Q function $Q^*(s, a)$? Explain it.

Solution

Subproblem(a)

Given (s, a, r, s'), we use the update equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a' \in \{-1,1\}} Q\left(s', a'\right) - Q(s,a)\right)$$

Using this equation with $\alpha = \frac{1}{2}, \gamma = \frac{1}{3}$, we have:

$$Q(3,-1) \leftarrow 0 + \frac{1}{2}\left(-1 + \frac{1}{3}\max_{a'} Q\left(2, a'\right)\right) = \frac{1}{2}(-1+0) = -\frac{1}{2}$$
$$Q(2,1) \leftarrow 0 + \frac{1}{2}\left(-1 + \frac{1}{3}\max_{a'} Q\left(3, a'\right)\right) = \frac{1}{2}(-1+0) = -\frac{1}{2}$$
$$Q(3,1) \leftarrow 0 + \frac{1}{2}\left(-1 + \frac{1}{3}\max_{a'} Q\left(4, a'\right)\right) = \frac{1}{2}(-1+0) = -\frac{1}{2}$$

Subproblem(b)

We have $\nabla_w J(w)$ as follow

$$\begin{aligned}\nabla_w J(w) &= -2\left(r + \gamma \max_{a'} \hat{q}\left(s', a'; w^-\right) - \hat{q}(s, a; w)\right) \nabla_w \hat{q}(s, a; w) \\ &= -2\left(r + \frac{1}{3}\max_{a'} \left(w^-\right)^T \begin{bmatrix} s' \\ a' \\ 1 \end{bmatrix} - w^T \begin{bmatrix} s \\ a \\ 1 \end{bmatrix}\right) \begin{bmatrix} s \\ a \\ 1 \end{bmatrix}\end{aligned}$$

Using this, the parameter update with a single sample (s, a, r, s') is:

$$\begin{aligned} w' &\leftarrow w - \alpha \nabla_w J(w) \\ &= w + \frac{1}{2}\left(r + \frac{1}{3}\max_{a'}\left(w^-\right)^T \begin{bmatrix} s' \\ a' \\ 1 \end{bmatrix} - w^T \begin{bmatrix} s \\ a \\ 1 \end{bmatrix}\right)\begin{bmatrix} s \\ a \\ 1 \end{bmatrix} \end{aligned}$$

Using the sample $(2, -1, -1, 1)$ and the particular values of w and w^- yields:

$$\begin{aligned} w' &\leftarrow \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + \frac{1}{2}\left(-1 + \frac{1}{3}\max_{a'}\begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}^T \begin{bmatrix} 1 \\ a' \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}\right)\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + \frac{1}{2}\left(-1 + \frac{1}{3}\max_{a'}(1 - a' - 2) - (-2 - 1 + 1)\right)\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 1/2 \\ 3/2 \end{bmatrix} \end{aligned}$$

Subproblem(c)

Because this method of Q-learning involves function approximation, bootstrapping, and off-policy training, we are not guaranteed to converge to anything, which includes no guarantee of converging to the optimal Q function.

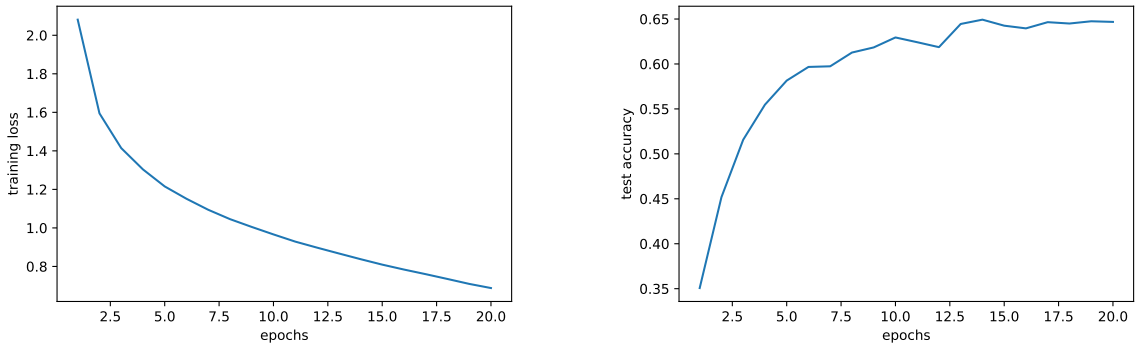
Problem 4

Solution

Subproblem(a)

The codes please see the file `CIFAR_A2.ipynb`.

We set `batch_size=16`, and `epoch=20`, we can derive the training curve and test accuracy curve as Figure 4. As we can see, the test accuracy is about 65% after 20 epochs.



(a) The training curve (b) The test accuracy curve

Figure 4 The plot of the training curve and test accuracy curve

Then, we can visualize the filters learned in the first convolutional layer as Figure 5. We do normalization to the weights in order to make the range of the weights in $[0, 1]$.

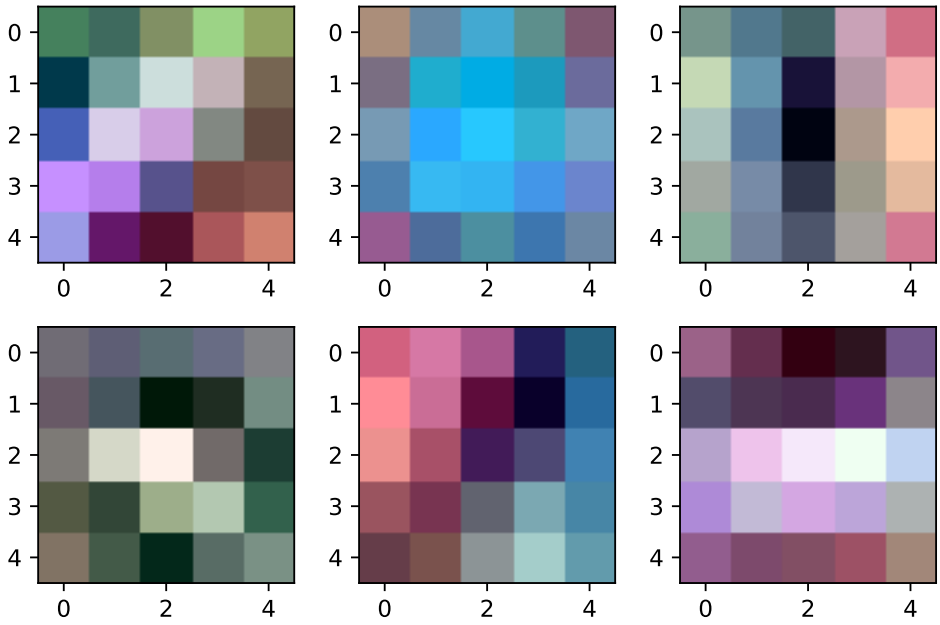


Figure 5 The filters learned in the first convolutional layer

Subproblem(b)

• We prove $\mathrm{Var}[y_l] = n_l \mathrm{Var}[w_l] \mathrm{E}\left[x_l^2\right]$ as follow

$$\begin{aligned}
\mathrm{Var}[y_l] &= \mathrm{Var}[\boldsymbol{W}_{l,i} \cdot \boldsymbol{x}_l] \\
&= n_l \cdot \mathrm{Var}[w_l \cdot x_l] \\
&= n_l \cdot \left(\mathrm{E}\left[(w_l \cdot x_l)^2\right] - (\mathrm{E}[w_l \cdot x_l])^2\right) \\
&= n_l \cdot \left(\mathrm{E}\left[(w_l \cdot x_l)^2\right] - (\mathrm{E}[w_l] \cdot \mathrm{E}[x_l])^2\right) \\
&= n_l \cdot \mathrm{E}\left[(w_l \cdot x_l)^2\right] \\
&= n_l \cdot \mathrm{E}\left[w_l^2\right] \cdot \mathrm{E}\left[x_l^2\right] \\
&= n_l \cdot \mathrm{Var}[w_l] \cdot \mathrm{E}\left[x_l^2\right]
\end{aligned} \tag{11}$$

• We prove $\mathrm{E}\left[x_l^2\right] = \frac{1}{2}\mathrm{Var}[y_{l-1}]$ as follow. If we let w_{l-1} have a symmetric distribution around zero and $b_{l-1} = 0$, then y_{l-1} has zero mean and has a symmetric distribution around zero. We let $P(x)$ to denote the cdf of x_l, $p(x)$ to denote the pdf of x_l, and $Q(y)$ to denote the cdf of y_{l-1}, $q(y)$ to denote the pdf of y_{l-1}. Because the activation function f is ReLu, so that we have

$$P(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{2} & x = 0 \\ Q(x) & x > 0 \end{cases} \tag{12}$$

$$p(x) = q(x) \quad (x > 0)$$

Then, we can have

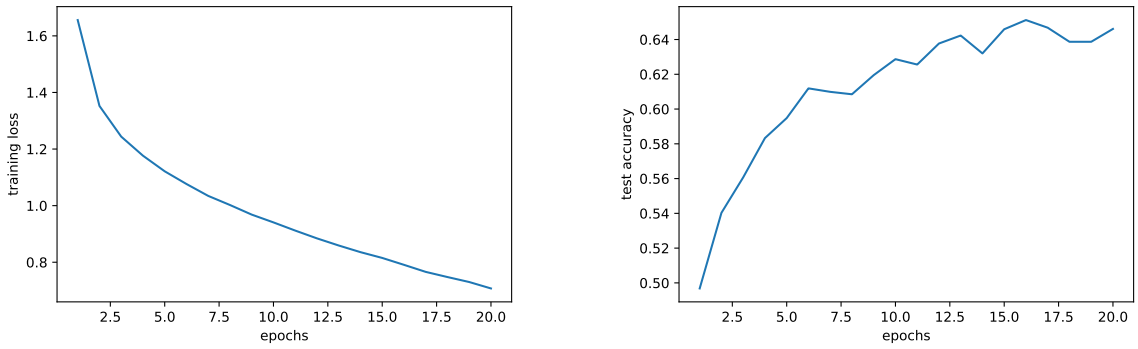
$$\begin{aligned}
\mathrm{E}\left[x_l^2\right] &= \int_{-\infty}^{+\infty} x_l^2 \cdot p\left(x_l\right) dx_l \\
&= 0^2 \cdot \frac{1}{2} + \int_{0}^{+\infty} x_l^2 \cdot p\left(x_l\right) dx_l \\
&= \int_{0}^{+\infty} x_l^2 \cdot q\left(x_l\right) dx_l
\end{aligned} \tag{13}$$

We can also have

$$\begin{aligned}
\frac{1}{2}\mathrm{Var}\left[y_{l-1}\right] &= \frac{1}{2}\mathrm{E}\left(y_{l-1}^2\right) \\
&= \frac{1}{2}\int_{-\infty}^{+\infty} y_{l-1}^2 q\left(y\right) dy \\
&= \frac{1}{2}\cdot 2\int_{0}^{+\infty} y_{l-1}^2 q\left(y\right) dy \\
&= \int_{0}^{+\infty} y_{l-1}^2 q\left(y\right) dy
\end{aligned} \tag{14}$$

By combination of equation 13 and 14, we can see that $\mathrm{E}\left[x_l^2\right] = \frac{1}{2}\mathrm{Var}\left[y_{l-1}\right]$. Thus, we finished the proof.

Then, we use the weight initialization strategy, and get the training curve and test accuracy curve as Figure 6. As we can see, the test accuracy is still about 65% after 20 epochs, which seems no improvement. The reason is maybe that the original initialization is good enough.



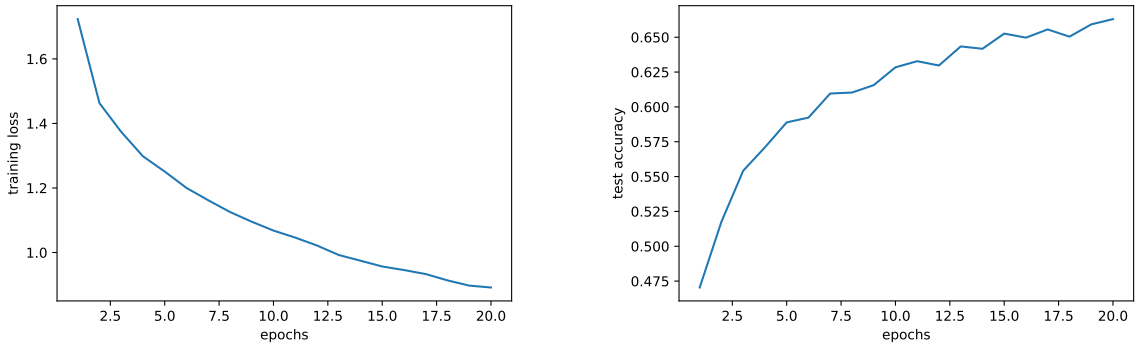
(a) The training curve

(b) The test accuracy curve

Figure 6 The plot of the training curve and test accuracy curve with kaiming initialization

Subproblem(c)

By using batch normalization, we can get the training curve and test accuracy curve as Figure 7. As we can see, the test accuracy reaches 66% after 20 epochs. Besides, the trend seems to increase continuously, so we believe that the test accuracy will be higher after more epochs. Thus, the batch normalization is really helpful.

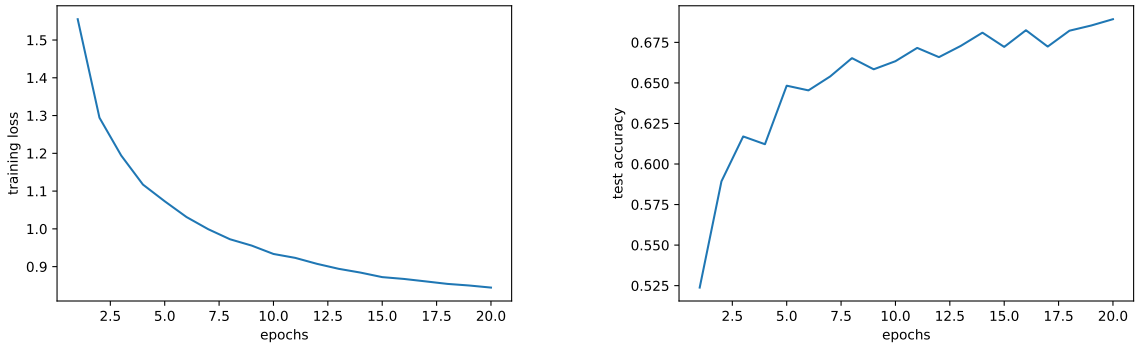


(a) The training curve (b) The test accuracy curve

Figure 7 The plot of the training curve and test accuracy curve with kaiming initialization and batch normalization

Subproblem(d)

We try to ues Adam as the optimizer, and we can get the training curve and test accuracy curve as Figure 8. As we can see, the test accuracy reaches 69% after 20 epochs, which is higher than before. Thus, we can see Adam optimizer if very helpful.



(a) The training curve (b) The test accuracy curve

Figure 8 The plot of the training curve and test accuracy curve with kaiming initialization and batch normalization, and Adam optimizer