به نام خداوند جان و خرد

Deep Learning
Homework-3

نیمسال دوم 1402-1401

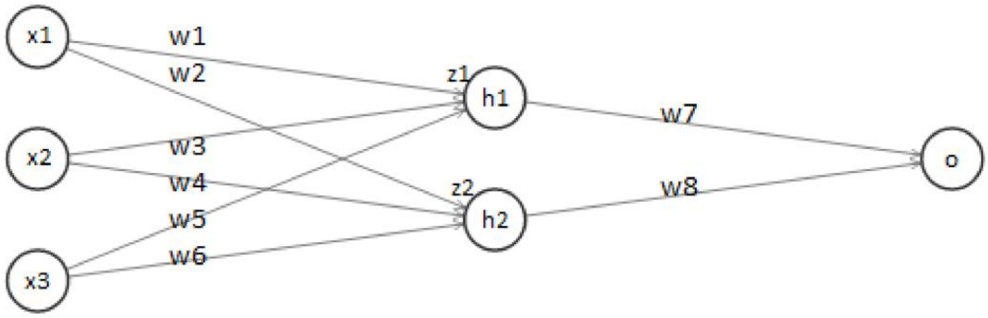
استاد درس: دکتر منصور رزقی آهق

دانشجو: بهار مهدوی
40152521337

Tarbiat Modares University

تمرین 1:

الف) شبکه عصبی زیر را با وزن های اولیه داده شده درنظر بگیرید و به روش‌های بهینه سازی SGD و Adam برای یک تکرار وزن ها را به روش پس انتشار به روز رسانی کنید.



Input Layer $\in R^3$ **Hidden Layer $\in R^2$** **Output Layer $\in R^1$**

جدول یک. مقدار دهی اولیه وزن ها

W1	W2	W3	W4	W5	W6	W7	W8
0.9	0.8	0.7	0.6	0.5	0.4	0.3	-0.4

جدول دو. مقدار دهی اولیه ورودی ها و خروجی

O	X1	X2	X3
0	1.5	1	1

جدول سه. توابع فعالساز مربوط به هر گره

h1	h2	O
Tanh	LeakyReLU(0.2)	Sigmoid

بر اساس شبکه عصبی مذکور خواهیم داشت:

$$y = \delta(o)$$

$$o = W_7 tanh + W_8 LeakyReLu(0.2)$$

$$z_1 = W_1x_1 + W_3x_2 + W_5x_3$$
$$z_2 = W_2x_1 + W_4x_2 + W_6x_3$$

Backpropagation در SGD:

در SGD داریم:

$$W_{inew} = W_{iold} - \eta \nabla W_i L$$

بنابراین در مشتق‌گیری Backpropagation به روش chain rule خواهیم داشت:

$$W_{8new} = W_{8old} - \eta \frac{\partial L}{\partial W_8}$$
$$W_{8new} = W_{8old} - \eta \left(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_8}\right)$$
$$W_{8new} = W_{8old} - (\delta - y) \times max(0.2z_2, z_2)$$

$$W_{7new} = W_{7old} - \eta \frac{\partial L}{\partial W_7}$$
$$W_{7new} = W_{7old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_7})$$
$$W_{7new} = W_{7old} - (\delta - y) \times tanh(z_1)$$

$$W_{6new} = W_{6old} - \eta \frac{\partial L}{\partial W_6}$$
$$W_{6new} = W_{6old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_6})$$

$$W_{6new} = W_{6old} - (\delta - y) \times W_8 \times \{0.2 \ \ if \ z_2 < 0, 1 \ \ if \ z_2 \geq 0\} \times x_3$$

$$W_{5new} = W_{5old} - \eta \frac{\partial L}{\partial W_5}$$
$$W_{5new} = W_{5old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_5})$$
$$W_{5new} = W_{5old} - (\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_3$$

$$W_{4new} = W_{4old} - \eta \frac{\partial L}{\partial W_4}$$

$$W_{4new} = W_{4old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_4})$$

$$W_{4new} = W_{4old} - (\delta - y) \times W_8 \times \{0.2 \;\; if \; z_2 < 0, 1 \;\; if \; z_2 \geq 0\} \times x_2$$

$$W_{3new} = W_{3old} - \eta \frac{\partial L}{\partial W_3}$$

$$W_{3new} = W_{3old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_3})$$

$$W_{3new} = W_{3old} - (\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_2$$

$$W_{2new} = W_{2old} - \eta \frac{\partial L}{\partial W_2}$$

$$W_{2new} = W_{2old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_2})$$

$$W_{2new} = W_{2old} - (\delta - y) \times W_8 \times \{0.2 \;\; if \; z_2 < 0, 1 \;\; if \; z_2 \geq 0\} \times x_1$$

$$W_{1new} = W_{1old} - \eta \frac{\partial L}{\partial W_1}$$

$$W_{1new} = W_{1old} - \eta (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_1})$$

$$W_{1new} = W_{1old} - (\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_1$$

Backpropagation در ADAM:

در ADAM داریم:

$$W_{inew} = W_{iold} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

بنابراین در مشتق‌گیری Backpropagation به روش chain rule خواهیم داشت:

$$W_{8new} = W_{8old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{8new} = W_{8old} - \frac{\in \frac{\partial L}{\partial W_8}}{\frac{\partial L}{\partial W_8} + \delta}$$

$$W_{8new} = W_{8old} - \frac{\in \left(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_8}\right)}{\left(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_8}\right) + \delta}$$

$$W_{8new} = W_{8old} - \frac{\in \left((\delta - y) \times max(0.2z_2, z_2)\right)}{\left((\delta - y) \times max(0.2z_2, z_2)\right) + \delta}$$

$$W_{7new} = W_{7old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{7new} = W_{7old} - \frac{\in \frac{\partial L}{\partial W_7}}{\frac{\partial L}{\partial W_7} + \delta}$$

$$W_{7new} = W_{7old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_7})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial W_7}) + \delta}$$

$$W_{7new} = W_{7old} - \frac{\in ((\delta - y) \times tanh(z_1))}{((\delta - y) \times tanh(z_1)) + \delta}$$

$$W_{6new} = W_{6old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{6new} = W_{6old} - \frac{\in \frac{\partial L}{\partial W_6}}{\frac{\partial L}{\partial W_6} + \delta}$$

$$W_{6new} = W_{6old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_6})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_6}) + \delta}$$

$$W_{6new} = W_{6old} - \frac{\in ((\delta - y) \times W_8 \times \{0.2 \ \ if \ z_2 < 0, 1 \ \ if \ z_2 \geq 0\} \times x_3 \ )}{((\delta - y) \times W_8 \times \{0.2 \ \ if \ z_2 < 0, 1 \ \ if \ z_2 \geq 0\} \times x_3 \ ) + \delta}$$

$$W_{5new} = W_{5old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{5new} = W_{5old} - \frac{\in \frac{\partial L}{\partial W_5}}{\frac{\partial L}{\partial W_5} + \delta}$$

$$W_{5new} = W_{5old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_5})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_5}) + \delta}$$

$$W_{5new} = W_{5old} - \frac{\in ((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_3)}{((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_3) + \delta}$$

$$W_{4new} = W_{4old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{4new} = W_{4old} - \frac{\in \frac{\partial L}{\partial W_4}}{\sqrt{\frac{\partial L}{\partial W_4}} + \delta}$$

$$W_{4new} = W_{4old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_4})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_4}) + \delta}$$

$$W_{4new} = W_{4old} - \frac{\in ((\delta - y) \times W_8 \times \{0.2 \ \ if \ z_2 < 0, 1 \ \ if \ z_2 \geq 0\} \times x_2)}{((\delta - y) \times W_8 \times \{0.2 \ \ if \ z_2 < 0, 1 \ \ if \ z_2 \geq 0\} \times x_2) + \delta}$$

$$W_{3new} = W_{3old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{3new} = W_{3old} - \frac{\in \frac{\partial L}{\partial W_3}}{\frac{\partial L}{\partial W_3} + \delta}$$

$$W_{3new} = W_{3old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_3})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_3}) + \delta}$$

$$W_{3new} = W_{3old} - \frac{\in ((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_2)}{((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_2) + \delta}$$

$$W_{2new} = W_{2old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{2new} = W_{2old} - \frac{\in \frac{\partial L}{\partial W_2}}{\frac{\partial L}{\partial W_2} + \delta}$$

$$W_{2new} = W_{2old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_2})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial LeakyReLu(0.2)} \times \frac{\partial LeakyReLu(0.2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_2}) + \delta}$$

$$W_{2new} = W_{2old} - \frac{\in ((\delta - y) \times W_8 \times \{0.2 \;\; if \; z_2 < 0, 1 \;\; if \; z_2 \geq 0\} \times x_1)}{((\delta - y) \times W_8 \times \{0.2 \;\; if \; z_2 < 0, 1 \;\; if \; z_2 \geq 0\} \times x_1) + \delta}$$

$$W_{1new} = W_{1old} - \frac{\in \hat{s}_{new}}{\sqrt{\hat{r}_{new}} + \delta}$$

$$W_{1new} = W_{1old} - \frac{\in \frac{\partial L}{\partial W_1}}{\frac{\partial L}{\partial W_1} + \delta}$$

$$W_{1new} = W_{1old} - \frac{\in (\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_1})}{(\frac{\partial L}{\partial \delta} \times \frac{\partial \delta}{\partial o} \times \frac{\partial o}{\partial tanh} \times \frac{\partial tanh}{\partial z_1} \times \frac{\partial z_1}{\partial W_1}) + \delta}$$

$$W_{1new} = W_{1old} - \frac{\in ((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_1)}{((\delta - y) \times W_7 \times \left(1 - tanh(z_1)\right)^2 \times x_1) + \delta}$$

ب) با طرح یک مثال غیر عددی نشان دهید که پس انتشار چگونه می تواند از مشتق گیری‌های تکراری پرهیز کند و سرعت بهینه سازی را بالا ببرد.

فرض کنیم شبکه زیر را داریم:

$$z_j^{(k)} = W_{j,:}^{k} h^{(k-1)} + b_j^{(k)}$$

$$W^{(k)} \in R^{nk-1 \times nk}, \quad h^{(k)}, b^{(k)} \in R^{nk}$$

$$\hat{y} = h_L = o\left(z^{(L)}\right): linear, softmax$$

$$L = (y - \hat{y})^2$$

اگر این شبکه با سه لایه پنهان و W(1)، W(2)، W(3)، W(4) و b(1)، b(2)، b(3)، b(4) باشد، اگر بخوایم مشتق $\frac{\partial L}{\partial W_4}$ ، $\frac{\partial L}{\partial W_3}$ ، $\frac{\partial L}{\partial W_2}$ و $\frac{\partial L}{\partial W_1}$ را بدست بیاوریم، خواهیم داشت:

$$\frac{\partial L}{\partial W_1}=\frac{\partial L}{\partial o}\times\frac{\partial o}{\partial z_3}\times\frac{\partial z_3}{\partial h_3}\times\frac{\partial h_3}{\partial z_2}\times\frac{\partial z_2}{\partial h_2}\times\frac{\partial h_2}{\partial z_1}\times\frac{\partial z_1}{\partial h_1}\times\frac{\partial h_1}{\partial W_1}$$

$$\frac{\partial L}{\partial W_2}=\frac{\partial L}{\partial o}\times\frac{\partial o}{\partial z_3}\times\frac{\partial z_3}{\partial h_3}\times\frac{\partial h_3}{\partial z_2}\times\frac{\partial z_2}{\partial h_2}\times\frac{\partial h_2}{\partial W_2}$$

$$\frac{\partial L}{\partial W_3}=\frac{\partial L}{\partial o}\times\frac{\partial o}{\partial z_3}\times\frac{\partial z_3}{\partial h_3}\times\frac{\partial h_3}{\partial W_3}$$

$$\frac{\partial L}{\partial W_4}=\frac{\partial L}{\partial o}\times\frac{\partial o}{\partial W_4}$$

بنابراین، بخشی از محاسبه $\frac{\partial L}{\partial W_4}$ ، $\frac{\partial L}{\partial W_3}$ و $\frac{\partial L}{\partial W_2}$ در محاسبه $\frac{\partial L}{\partial W_1}$، بخشی از محاسبه $\frac{\partial L}{\partial W_4}$ و $\frac{\partial L}{\partial W_3}$ در محاسبه $\frac{\partial L}{\partial W_2}$ و بخشی از محاسبه $\frac{\partial L}{\partial W_4}$، در محاسبه $\frac{\partial L}{\partial W_3}$ بکار رفته است. به این ترتیب یک روش بهینه برای جلوگیری از مشتق‌گیری تکراری و کاهش بار محاباتی شبکه روش پس انتشار (*Backpropagation*) است که از آخر مشتق‌گیری انجام شده و از مقادیر محاسبه شده در لایه‌های قبل استفاده می‌گردد و سرعت بهینه‌سازی را افزایش می‌دهد.

ج) بررسی کنید روش Newton’s method چه مزایایی دارد و نرخ یادگیری (learning rate) دراستفاده از آن به چه صورت است؟ چرا این روش در یادگیری ماشین چندان مورد توجه نیست؟

روش Newton's method یک روش بهینه‌سازی است که در حل مسائل بهینه‌سازی غیرخطی استفاده می‌شود. این روش به دنبال یافتن نقطه‌ی بیشینه یا کمینه‌ی یک تابع هدف است. برای بدست آوردن آن از تقریب درجه دوم سری تیلور استفاده می‌کنیم:

$$f(x+\alpha p)\approx f(x)+\alpha p^T\nabla f(x)+\frac{\alpha^2}{2!}p^T\nabla^2 f(x)p$$

و خواهیم داشت:

$$x^{k+1}=x^k-H(xk)^{-1}\nabla f(x^k)$$

مزایای استفاده از روش Newton's method عبارتند از:

۱. روش Newton's method به سرعت به نقطه‌ی بهینه همگرا می‌شود و با تعداد کمی تکرار می‌تواند نتیجه‌ی بهینه را بدهد.
۲. این روش به دقت بالایی در یافتن نقطه‌ی بهینه دست می‌یابد و می‌تواند به نتیجه‌ی دقیق‌تری نسبت به روش‌های دیگر برسد.
۳. روش Newton's method برای حل مسائل بهینه‌سازی غیرخطی بسیار مناسب است و می‌تواند در مسائل پیچیده‌تری نسبت به روش‌های دیگر عملکرد بهتری داشته باشد.

نرخ یادگیری در روش Newton's method به صورت ثابت تعیین نمی‌شود. در واقع، در هر مرحله از روش، ماتریس هسیان تابع هدف برای محاسبه‌ی قدم بعدی استفاده می‌شود. بنابراین، نرخ یادگیری در این روش به صورت خودکار توسط روش تعیین می‌شود و نیازی به تنظیم دستی ندارد.

Newton's method برای یافتن راه‌حل‌های بهینه جهانی، الگوریتمی به مراتب برتر از Gradient Descent اســـت، زیرا روش نیوتن می‌تواند تضـــمینی برای حل آن ارائه دهد. این روش ثابت اســـت و مهم‌تر از همه در مراحل بســـیار کمتری همگرا می‌شـــود. اما علت اینکه الگوریتم‌های بهینه‌ســـازی مرتبه دوم، مانند Newton's method، به اندازه Stochastic Gradient Descent در مســـائل یادگیری ماشین به طور گسترده مورد استفاده قرار نمی‌گیرند این است که Gradient descent یک تابع را با استفاده از دانش مشتق آن به حداکثر می رساند در حالی که Newton's method، یک الگوریتم ریشه یابی بوده و یک تابع را با استفاده از دانش مشتق دوم آن به حداکثر می رساند. زمانی که مشتق دوم شناخته شده باشد و محاسبه آن آسان باشد، Newton's method می تواند سریعتر عمل کند. با این حال، بیان تحلیلی برای مشتق دوم اغلب پیچیده یا غیرقابل حل است و به محاسبات زیادی نیاز دارد. روش‌های عددی برای محاسبه مشتق دوم نیز به محاسبات زیادی نیاز دارند (اگر مقادیر N پارامتر برای محاسبه مشتق اول مورد نیاز باشد، N^2 پارامتر برای مشتق دوم مورد نیاز است). بنابراین محاسبه‌ی مشتق دوم ماتریس Hessian در هر مرحله از روش نیاز به محاسبات پیچیده‌ای دارد که می‌تواند در مسائل با داده‌های بزرگ و با پارامترهای زیاد مثل یادگیری عمیق مشکل ساز شود و در هر تکرار زمان زیادی برده و حافظه فشرده شود. در حالی که محاسبه Hessian هزینه‌بر است، معکوس کردن آن یا حل حداقل مربعات آن اغلب حتی بدتر است. روش Newton's method همچنین برای حل مسائلی که تابع هدف غیرمشتق‌پذیر است، نامناسب است و ممکن است به جواب‌های نامطلوب منجر شود. همچنین، این روش حساس به نقطه‌ی شروع بوده و ممکن است در صورتی که نقطه‌ی شروع نزدیک به نقطه‌ی بهینه نباشد، به نتیجه‌ی نامطلوبی برسد. پس به طور کلی، روش Newton's method در مســائل ســاده‌تر به خوبی عمل می‌کند، اما در مســائل پیچیده‌تر و در یادگیری ماشــین که معمولاً با داده‌های بزرگ و پیچیده و با تعداد پارامترهای بالا سروکار دارد، ممکن است عملکرد ضعیفی داشته باشد. بنابراین، روش‌های دیگری مانند روش‌های گرادیانی معمولاً در یادگیری ماشین مورد توجه بیشتری قرار می‌گیرند.

د) مزیت های روش های بهینه سازی Adam و RMSprop را شرح دهید.

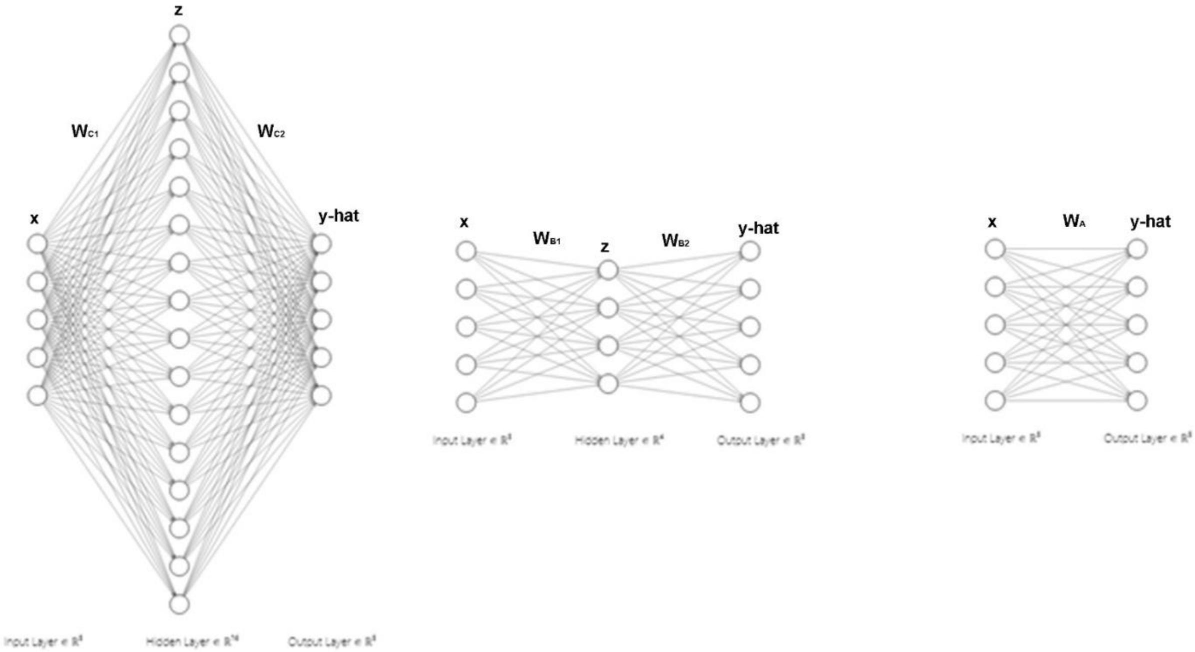
به طور کلی، Adam و RMSprop دو روش بهینه‌سازی قوی و محبوب هستند که در یادگیری عمیق بسیار مورد استفاده قرار می‌گیرند. این روش‌ها برای بهبود سرعت و کیفیت یادگیری در شبکه‌های عصبی عملکرد خوبی دارند و هر کدام از این آن‌ها ویژگی‌های خاص خود را دارند و در مسائل مختلف می‌توانند عملکرد بهتری نسبت به یکدیگر داشته باشند.

Adam با استفاده از ترکیبی از روش‌های Gradient descent و معیارهای مربوط به Momentum، سرعت همگرایی بالایی را دارد. این روش با کمک معیارهای Momentum به صورت خودکار و مستقل نرخ یادگیری را برای هر پارامتر تنظیم می‌کند و تطبیقی با تغییرات مسئله مثل تفاوت و عدم بالانس در میزان گرادیان پارامترها ایجاد می‌کند. این ویژگی باعث می‌شـود که Adam بتواند در مســائلی که نرخ یادگیری متفاوت برای هر پارامتر مناسب است، عملکرد بهتری داشته باشد. همچنین Adam تاریخچه گرادیان را در حافظه نگه می‌دارد و از آن در محاســبات بعدی اســتفاده می‌کند. این ویژگی باعث می‌شــود که Adam بتواند در توابعی که گرادیان‌ها از یک زمان به زمان دیگر تغییرات زیادی دارند، بهبود قابل توجه‌ای را به دست آورد.

RMSprop با اســتفاده از روشــی به نام تقسـیم بر مجذور گرادیان، تغییرات نرخ یادگیری را کاهش می‌دهد. این ویژگی باعث می‌شــود که RMSprop در مسائلی که نرخ یادگیری باید به طور متناسب با گرادیان‌ها تنظیم شود، بهبود قابل توجه‌ای را به دست آورد. RMSprop هم مانند Adam تاریخچه گرادیان را در حافظه نگه می‌دارد و از آن در محاسبات بعدی استفاده می‌کند، به این ترتیب به مسائلی که در طول زمان تغییر می‌کنند، کارایی خوبی دارند. RMSprop همچنین قابلیت تنظیم پارامترهای مختلفی مانند نرخ یادگیری، نرخ کاهش تغییرات نرخ یادگیری و تعداد تکرارها را دارد و این ویژگی باعث می‌شود که بتواند در مسائل مختلف و با تنظیمات مختلف عملکرد بهتری داشته باشد.

تمرین 2:

در سه شبکه زیر تابع هدف و مجموعه داده آموزشی یکسان هستند و همچنین توابع فعال ساز همگی خطی هستند. تعداد نرون‌های میانی در مدل ب، چهار گره و در شکل ج، شانزده گره می باشند. با فرض تعریف وظیفه رگرسیون برای آموزش شبکه، فرض کنید برای داده (xi, yi) پاسخ y^i باشد. کیفیت پاسخ‌های خروجی را قیاس کنید و توجیه ریاضی ارائه دهید. اگر شرطی برای تعداد داده ها وجود دارد بیان کنید.



(ج) (ب) (الف)

شبکه الف:

در شبکه الف، 5 نورون در لایه ورودی و 5 نورون در لایه خروجی وجود دارد که اگر بر طبق تعداد فیچرهای لایه خروجی، فرض کنیم linear classification پنج کلاسه داریم، اگر x_i عضو کلاس C_j باشد y_i ما یک بردار 5 تایی خواهد بود که j امین عنصر آن یک و مابقی آن صفر است. $\hat{y}_i$ برابر حاصل ضرب پارامتر w در x است که پارامترهای w عضو یک ماتریس 5*5 است. بنابراین در ماتریسی که 5 سطر و 5 ستون دارد حداکثر Rank، 5 می‌شود:

$$y(x_i, y_i) \quad if \quad x_i \in C_j \qquad y_i = \begin{pmatrix} 0 \\ \dots \\ j \\ \dots \\ 0 \end{pmatrix} \qquad \hat{y}_i = \begin{pmatrix} w_1^T \\ w_2^T \\ \dots \\ w_5^T \end{pmatrix} x \rightarrow \hat{y}_i = w_A x \quad w_A \in R^{5*5} \quad Rank(w_A)$$

$$\leq 5$$

$$w_A = \{w_A | Rank(w_A) \leq 5 \quad w_A \in R^{5*5}\}$$

شبکه ب:

اگر به شبکه الف یک لایه پنهان $(x \in R^4)$ اضافه گردد شبکه ب بدست می‌آید. در این صورت همانطور که میبینیم حداکثر Rank، 4 میشود:

$$\hat{y}_i = w_{B2}z \quad w_{B2} \in R^{5*4} \quad Rank(w_{B2}) \le 4$$
$$z = w_{B1}x \quad w_{B1} \in R^{4*5} \quad Rank(w_{B1}) \le 4$$
$$\hat{y}_i = w_{B2}w_{B1}x \quad if \;\; w_{B2}w_{B1} = \overline{w_B} \quad \overline{w_B} \in R^{5*5} \quad Rank(\overline{w_B}) \le 4$$

$$w_B = \{\overline{w_B} | \overline{w_B} = w_{B2}w_{B1} \quad w_{B1} \in R^{4*5} \;\; \& \;\; w_{B2} \in R^{5*4}\}$$

Rank4 نمیتواند Rank5 را پوشش دهد. به این ترتیب حتی در بهینه ترین شرایط شبکه ب، قدرت کمتری نسبت به شبکه الف و ج خواهد داشت.

شبکه ج:

اگر به شبکه الف یک لایه پنهان ($x \in R^{16}$) اضافه گردد شبکه ج بدست می‌آید. در این صورت هم همانطور که میبینیم حداکثر Rank، 5 میشود:

$$\hat{y}_i = w_{C2}z \quad w_{C2} \in R^{5*16} \quad Rank(w_{C2}) \le 5$$
$$z = w_{C1}x \quad w_{C1} \in R^{16*5} \quad Rank(w_{C1}) \le 5$$
$$\hat{y}_i = w_{C2}w_{C1}x \quad if \;\; w_{C2}w_{C1} = \overline{w_C} \quad \overline{w_C} \in R^{5*5} \quad Rank(\overline{w_C}) \le 5$$

$$w_C = \{\overline{w_C} | \overline{w_C} = w_{C2}w_{C1} \quad w_{C1} \in R^{16*5} \;\; \& \;\; w_{C2} \in R^{5*16}\}$$

پس در بهینه ترین شرایط شبکه ج، برای داده های train مشابه شبکه الف جواب خواهد داد. ولی چون در مسئله، Active function خطی در نظر گرفته شده است و چون با عمیق کردن شبکه تعداد پارامترها افزایش یافته است، در صورتی که تعداد داده ها کافی نباشد، برای داده های test برای شبکه ج احتمال Over fiting مطرح خواهد شد و این امکان وجود دارد که حتی نتیجه شبکه ج را در مقایسه با شبکه الف خرابتر کند. اگر Active function، غیر خطی در نظر گرفته میشد شرایط عوض میشد و عمیق تر شدن شبکه می‌توانست در داده های test نتیجه بهتری را ایجاد نماید.

پس به طور خلاصه میشه ثابت کرد که w_B زیر مجموعه w_C است و w_C با w_A برابر است:

$$w_B \subset w_C = w_A$$

تمرین 3:

الف) فرض کنید در یک زیرمسئله ، پس از یک لایه پیچشی با مشخصات زیر در یک مدل یک لایه AveragePool1D داریم و در مدل دیگر یک لایه MaxPool1D. به صورت مجزا برای هر یک از مدل‌ها، ضمن انجام مشتق گیری و بهینه سازی مدل به صورت دستی و عددی، به صورت کلی نیز توضیح دهید عملیات پس انتشار (Back Propagation) و به روز رسانی در این لایه‌ها به چه صورت رخ می دهد ؟ (فقط یک مرحله به روز رسانی شود).

- Modules:

o A: Conv1D (Input-Channel: 2, Output Channel: 1, kernel size: 2, Padding=0, Stride:1)

o B: Conv1D (Input-Channel: 2, Output Channel: 1, kernel size: 2, Padding=0, Stride:1)

o C: AveragePool1D (Kernel-Size:2, Padding:0, stride:1)

o D: MaxPool1D (Kernel-Size:2, Padding:0, stride:1)

- Data1:

1 2 -3 2

Channel1

1 2 -2 1

Channel2

- Data2:

4 2 -2 2

Channel1

3 2 -7 2

Channel2

Taski :min || out2 - out1|| , outj = Modeli(Dataj) [forj = 1, 2]

Model 1: Data -> A -> C -> out

Model 2: Data -> B -> D -> out

$$\boldsymbol{Model1}\text{: Conv1D}\left(\boldsymbol{C_{inp}}\text{: }\boldsymbol{D1} = \left[\begin{bmatrix} 1 \\ 2 \\ -3 \\ 2 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \\ -2 \\ 1 \end{bmatrix}\right] \ \boldsymbol{or}\ \boldsymbol{D2} = \left[\begin{bmatrix} 4 \\ 2 \\ -2 \\ 2 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \\ -7 \\ 2 \end{bmatrix}\right], \boldsymbol{C_{out}} = \ 1, \boldsymbol{k} = \ 2, \mathrm{P} = 0, \mathrm{S} = 1\right)$$

$$\rightarrow \boldsymbol{Ave\ Pooling}\ (\boldsymbol{k} = 2, \boldsymbol{p} = 0, \boldsymbol{s} = 1)$$

در این مدل، هدف ما کمینه کردن فاصله بین خروجی $\boldsymbol{Model1}$ با داده $\boldsymbol{D1}$ و داده $\boldsymbol{D2}$ است:

$$\boldsymbol{Task1}\text{: }\boldsymbol{min}\left\|\boldsymbol{out}_{\boldsymbol{D2}}^{\boldsymbol{model1}} - \boldsymbol{out1}_{\boldsymbol{D1}}^{\boldsymbol{model1}}\right\|_2^2$$

در لایه Conv1D، مشتقات جزئی تابع خطا نسبت به وزن‌ها و بازدهی لایه‌ها محاسبه می‌شوند و وزن‌ها با استفاده از الگوریتم به‌روزرسانی (مثلاً روش Gradient Descent) به‌روز می‌شوند.

$$\boldsymbol{Z}(\boldsymbol{1}, \boldsymbol{j}) = \sum_{\boldsymbol{c}=\boldsymbol{1}} \sum_{\boldsymbol{k}} \boldsymbol{x}(\boldsymbol{c}, \boldsymbol{j} + \boldsymbol{k}) \times \boldsymbol{W}(\boldsymbol{1}, \boldsymbol{c}, \boldsymbol{k}) + \boldsymbol{b}(\boldsymbol{1}, \boldsymbol{j})$$

چون AveragePool1D یک لایه غیر قابل آموزش است، هیچ وزن قابل به‌روزرسانی در این لایه وجود ندارد. بنابراین، مشتق آن برابر با صفر است و به لایه قبلی، یعنی Conv1D، منتقل می‌شود.

پس، از محاسبه خروجی $Model1$، با استفاده از تابع loss و روش Back-Propagation، مشتقات جزئی تابع خطا نسبت به وزن‌ها و بازدهی لایه‌ها محاسبه می‌شوند

$$Model2\text{: Conv1D}\left(C_{inp}\text{: } D1 = \begin{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -3 \\ 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -2 \\ 1 \end{bmatrix} \end{bmatrix} \; or \; D2 = \begin{bmatrix} \begin{bmatrix} 4 \\ 2 \\ -2 \\ 2 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ -7 \\ 2 \end{bmatrix} \end{bmatrix}, C_{out} = 1, k = 2, \mathrm{P} = 0, \mathrm{S} = 1\right)$$
$$\rightarrow Max\ Pooling\ (k = 2, p = 0, s = 1)$$

در این مدل، هدف ما کمینه کردن فاصله بین خروجی $Model1$ با داده $D1$ و داده $D2$ است:

$$Task2\text{: } min\left\| out_{D2}^{model2} - out1_{D1}^{model2} \right\|_2^2$$

در لایه Conv1D، مشتقات جزئی تابع خطا نسبت به وزن‌ها و بازدهی لایه‌ها محاسبه می‌شوند و وزن‌ها با استفاده از الگوریتم به‌روزرسانی (مثلاً روش Gradient Descent) به‌روز می‌شوند.

$$Z(1, j) = \sum_{c=1} \sum_{k} x(c, j + k) \times W(1, c, k) + b(1, j)$$

چون MaxPool1D یک لایه غیر قابل آموزش است، هیچ وزن قابل به‌روزرسانی در این لایه وجود ندارد. بنابراین، مشتق آن برابر با صفر است و به لایه قبلی، یعنی Conv1D، منتقل می‌شود.

پس، از محاسبه خروجی $Model2$، با استفاده از تابع loss و روش Back-Propagation، مشتقات جزئی تابع خطا نسبت به وزن‌ها و بازدهی لایه‌ها محاسبه می‌شوند

ب) این زیرمسئله را با پیاده سازی به کمک pytorch حل کنید و نزدیکی پاسخ‌ها را بررسی کنید.

کد مرتبط با تمرین 3:

- **DL-HW3-Q03 -Mahdavi.ipynb**

تمرین 4:

الف) شبکه زیر را بر روی مجموعه داده CIFAR10 آموزش دهید. میزان دقت و هزینه برای داده های آموزشی و آزمایشی را به صورت زوج در دو نمودار (یک نمودار برای دقت و یک نمودار برای هزینه) نشان دهید و مقادیر را

نیز چاپ کنید همچنین زمان اجرای آموزش را نیز ذخیره کنید و در ادامه موارد بالا را برای پنج آزمایش زیر را در مسئله دخیل کنید.

Epoch-number = 20 | Optimizer : SGD | lr = 0.001 | momentum = 0.9 | batchsize = 32 | loss : Cross Entropy

شبه کد برای مدل:

```
model: Forward(
Conv2d(in-Channel=3, out-channel=32, kernelSize=3, padding=1),
ReLU,
Conv2d(32, 64, kernelSize=3, stride=1, padding=1),
ReLU,
MaxPool2d(kernelSize=2, stride=2),
BatchNorm2d,
Conv2d(64, 128, kernelSize=3, stride=1, padding=1),
ReLU,
Conv2d(128, 128, kernelSize=3, stride=1, padding=1),
ReLU,
MaxPool2d(kernelSize=2, stride=2),
BatchNorm2d,
Conv2d(128, 256, kernelSize=3, stride=1, padding=1),
ReLU,
Conv2d(256, 256, kernelSize=3, stride=1, padding=1),
ReLU,
MaxPool2d(kernelSize=2, stride=2),
BatchNorm2d,
Flatten,
Linear(? to 1024),
ReLU,
Linear(1024 to 512),
ReLU,
Linear(512 to 10))
```

کد مربوطه:

- DL-HW3-Q04A0-Mahdavi.ipynb

1) منظم ساز نرم یک با ضریب منظم سازی های 1e-3 و 1e-2

کد مربوطه:

- **DL-HW3-Q04A1-Mahdavi.ipynb**

2) منظم ساز نرم دو با ضریب منظم سازی های 1e-3 و 1e-2

کد مربوطه:

- **DL-HW3-Q04A2-Mahdavi.ipynb**

3) افزودن داده به روش Data Augmentation با کمک عملیات های زیر:

1. چرخاندن ۹۰ درجه 2. یکی از عملیات های شیفت دادن یا تغییر روشنایی "یا برش تغییر همراه با تغییر اندازه"

torchvision.transforms: RandomResizedCrop/ RandomRotation / …

کد مربوطه:

- **DL-HW3-Q04A3-Mahdavi.ipynb**

4) افزودن دراپ اوت با نرخ 0.5 به لایه‌های خطی و با ضریب 0.15 برای سایر لایه‌ها (تحقیق کنید، برای یک لایه پیچشی، چه تفاوتی بین nn.Dropout و nn.Dropout2d وجود دارد؟)

یک Dropout نرمال در طول training، به طور تصادفی برخی از عناصر تانسور ورودی را با احتمال p با استفاده از نمونه هایی از توزیع برنولی صفر می‌کند. هر کانال در هر forward به طور مستقل صفر می شود.

Dropout2d به طور تصادفی کل کانال‌های یک ماتریس را صفر می‌کند. هر کانال به طور مستقل در هر فوروارد با احتمال p با استفاده از نمونه هایی از توزیع برنولی صفر می شود.

کد مربوطه:

- **DL-HW3-Q04A4-Mahdavi.ipynb**

5) روش Early-stopping

کد مربوطه:

- **DL-HW3-Q04A5-Mahdavi.ipynb**

ب) بررسی کنید که افزودن لایه‌هایی خطی مشابه و پشت سر هم از ۵۱۲ گره به ۵۱۲ گره در محل مناسب و با فعال سازهای برای هر لایه، می‌تواند باعث دیده شدن بیش برازش بیشتر در نمودارهای خواسته شده برای داده‌های آموزشی و آزمایشی در محل اول (بدون منظم سازی) شود؟

کد مربوطه:

- **DL-HW3-Q04B-Mahdavi.ipynb**

تمرین 5:

به صورت تئوری نشان دهید که Dropout مشابه Ensemble Methods عمل می کند. (منبع صفحه 262-263 کتاب Goodfellow)

اگر فرض کنیم نقش مدل خروجی یک توزیع احتمال است، در روش bagging، هر مدل یک توزیع $p^{(i)}(y|x)$ تولید میکند. پیشبینی ensemble میانگین حسابی تمامی این توزیع ها می‌شود:

$$\frac{1}{k}\sum_{i=1}^{k} p^{(i)}(y|x)$$

در روش dropout، هر مدل فرعی توسط یک بردار ماسک μ یک توزیع احتمالی $p(y|x,\mu)$ را تعریف می‌کند. میانگین حسابی روی این ماسک‌ها به این صورت است:

$$\sum_{\mu} p(\mu)p(y|x,\mu)$$

که $p(\mu)$ توزیع احتمالی است که برای نمونه μ در زمان training استفاده می‌گردد. چون این Sum شامل تعداد زیادی عبارت است، ارزیابی آن غیر قابل حل است مگر در مواردی که ساختار مدل اجازه ساده‌سازی بدهد. ولی تاکنون در مدل‌های شبکه عمیق همچین چیزی دیده نشده است. پس در عوض میتوانیم استنباط را با نمونه گیری تقریب بزنیم و برای بدست آوردن عملکردی مناسب، خروجی میانگین 10-20 ماسک را در نظر بگیریم.

با این حال یک رویکرد بهتر هم هست که به ما امکان میدهد تخمین مناسبی از prediction کل ensemble داشته باشیم. با cost فقط یک forward propagation. برای این کار به جای استفاده از میانگین هندسی، میانگین حسابی توزیع‌ها را بدست می‌آوریم. برای گارانتی اینکه نتیجه یک توزیع احتمالی باشد ما این شرط را میذاریم که هیچ یک از مدل‌های فرعی احتمال صفر را نسبت نمی‌دهد. توزیع احتمالی unnormalize به طور مستقیم با میانگین هندسی تعریف می‌شود:

$$\tilde{P}ensemble(y|x) = \sqrt[2^d]{\prod_{\mu} p(y|x,\mu)}$$

که در آن d تعداد نودهایی است که ممکنه dropped شوند. اینجا از توزیع یکنواخت بر روی μ استفاده شده ولی استفاده از توزیع های غیر یکنواخت هم امکانپذیر است. برای انجام پیشبینی باید ensemble را renormalize کنیم:

$$Pensemble(y|x) = \frac{\tilde{p}ensemble(y|x)}{\sum_{y'} \tilde{p}ensemble(y'|x)}$$

ما می‌توانیم $Pensemble$ را با ارزیابی $p(y|x)$ در یک مدل تخمین بزنیم. یک مدل با تمام واحدها ولی حذف وزن‌ها و با ضرب احتمال به واحدها که انگیزه این کار برای بدست آوردن دقیق ارزش خروجی از روی آن واحدها است که به این رویکرد قانون weight scaling inference می‌گویند.

اگر یک رگرسیون softmax را در نظر بگیریم، classifier با n متغیر ورودی که با بردار v نشان داده می‌شود:

$$P(Y = y|v) = softmax(W^T v + b)_y$$

می‌توان با ضرب عنصر ورودی با یک بردار باینری d به خانواده مدل فرعی index کنیم:

$$P(Y = y|v; d) = softmax(W^T (d \odot v) + b)_y$$

Ensemble predictor با renormalize، میانگین هندسی روی همه پیشبینی‌های اعضای ensemble تعریف می‌شود:

$$Pensemble(Y = y|v) = \frac{\widetilde{p}ensemble(Y = y|v)}{\sum_{y'} \widetilde{p}ensemble(Y = y'|v)}$$

$$\widetilde{P}ensemble(Y = y|v) = \sqrt[2n]{\prod_{d \in \{0,1\}^n} P(Y = y|v; d)}$$

برای اینکه ببینیم قانون weight scaling inference دقیق است می‌توانیم $\widetilde{P}ensemble$ را ساده کنیم:

$$\widetilde{P}ensemble(Y = y|v) = \sqrt[2n]{\prod_{d \in \{0,1\}^n} softmax(W^T (d \odot v) + b)_y}$$

$$\widetilde{P}ensemble(Y = y|v) = \sqrt[2n]{\prod_{d \in \{0,1\}^n} \frac{exp(W_{y;:}^T (d \odot v) + b_y)}{\sum_{y'} exp(W_{y;:}^T (d \odot v) + b_{y'})}}$$

$$\widetilde{P}ensemble(Y = y|v) = \frac{\sqrt[2n]{\prod_{d \in \{0,1\}^n} exp(W_{y;:}^T (d \odot v) + b_y)}}{\sqrt[2n]{\prod_{d \in \{0,1\}^n} \sum_{y'} exp(W_{y;:}^T (d \odot v) + b_{y'})}}$$

از آنجایی که $\widetilde{P}$ نرمال می‌شود می‌توانیم با اطمینان از ضرب در عواملی که نسبت به y ثابت هستند چشم پوشی کنیم:

$$\widetilde{P}ensemble(Y = y|v) \propto \sqrt[2n]{\prod_{d \in \{0,1\}^n} exp(W_{y;:}^T (d \odot v) + b_y)}$$

$$\widetilde{P}ensemble(Y = y|v) = (\frac{1}{2^n}) \sum_{d \in \{0,1\}^n} W_{y;:}^T (d \odot v) + b_y$$

$$\widetilde{P}ensemble(Y = y|v) = exp(\frac{1}{2} W_{y;:}^T \mathrm{v} + b_y)$$

با جایگزینی این به معادله:

$$Pensemble(Y=y|v)=\frac{\tilde{p}ensemble(Y=y|v)}{\sum_{y'}\tilde{p}ensemble(Y=y'|v)}$$

یک softmax classifier با وزن‌های $\frac{1}{2}W$ بدست می‌آید.

تمرین 6:

نشان دهید لایه پیچشی زیر مشابه یک MLP روی هر پیکسل عمل می کند:

Conv2D (Input-Channel: M, Output-Channel: N, kernel size: 1, Padding: 0, Stride: 1)

از آنجایی که zero-padding و kernel size=1 و اسکالر است این لایه پیچشی (Pooling layer) در کانال c ام به صورت زیر است:

$$z(i,:,:)=\sum_{j=1}^{M}W(i,j,:,:)*H(j,:,:)=\begin{pmatrix}\sum_{j=1}^{M}x_{11}^{j}\times W_{j}^{c} & \cdots & \sum_{j=1}^{M}x_{1s_2}^{j}\times W_{j}^{c}\\ \vdots & \ddots & \vdots\\ \sum_{j=1}^{M}x_{s_11}^{j}\times W_{j}^{c} & \cdots & \sum_{j=1}^{M}x_{s_1s_2}^{j}\times W_{j}^{c}\end{pmatrix}$$

که در آن j کانال ورودی تا M و i کانال خروجی تا N است و اندازه تصاویر s_1 و s_2 فرض شده‌اند.
به این ترتیب هر عنصر خروجی برای کانال c ام، به صورت یک MLP ساخته می‌شود که در آن اندازه تصویر خروجی با توجه به سایر کرنل و سایر تنظیمات لایه پیچشی تغییری نمی‌کند.

تمرین 7:

الف) شبکه زیر را بر روی مجموعه داده IRIS آموزش دهید. خطا و دقت را برای داده های آموزشی وآزمایشی (تست) گزارش کنید.

ب) سپس یک تابع برای کلاس شبکه بنویسید که مشتقات پس انتشار توسط شما دستی نوشته شده باشد و بدون نیاز به backward و توابع هزینه آماده در پایتورچ (pytodch) با کمک آن تابع درون کلاسی خروجی محاسبه گردد.

با توجه به loss شبکه که Cross-Entropy است و خروجی لایه آخر softmax خواهیم داشت:

$$Loss(y_n,\hat{y}_n)=-\sum_{i=1}y_i\,log\left(\frac{expx_4^i}{\sum_{j=1}expx_4^j}\right)=-\sum_{i=1}y_i\,(x_4^i-log\left(\sum\nolimits_{j=1}expx_4^j\right))$$

به این ترتیب مشتق‌های loss function نسبت به پارامترها را بدست می‌آوریم:

$$\frac{\partial L_n}{\partial W_{j,i}^4} = \frac{\partial L_n}{\partial x_j^4} \times \frac{\partial x_j^4}{\partial W_{j,i}^4}$$

$$\frac{\partial L_n}{\partial W_{j,i}^4} = -\sum_{i=1} y_i \left(\frac{\partial L_n}{\partial x_j^4} x_4^i - \frac{\partial L_n}{\partial x_j^4} log(\sum_{m=1} expx_4^m)\right) \times \frac{\partial x_j^4}{\partial W_{j,i}^4}$$

$$x_j^4 = \sum_{i=1} W_{j,i}^4 h_i^3 \;\rightarrow\; \frac{\partial L_n}{\partial W_{j,i}^4} = -\sum_{i=1} y_i \left(\frac{\partial L_n}{\partial x_j^4} x_4^i - \frac{\partial L_n}{\partial x_j^4} log(\sum_{m=1} expx_4^m)\right) \times h_i^3$$

$$\frac{\partial L_n}{\partial W_{j,i}^4} = -\sum_{i=1} y_i \left(\delta_{k,j} - \frac{1}{\sum_{m=1} expx_4^m)} \times \left(expx_4^j\right)\right) \times h_i^3 , \qquad \delta_{k,j} = \{1\ if\ k = j, 0\ if\ k \neq j\}$$

$$\frac{\partial L_n}{\partial W_{j,i}^4} = -(\sum_{i=1} y_i \delta_{k,j} - \sum_{i=1} y_i(\widehat{y_\iota})) \times h_i^3 = (-y_i + \widehat{y_\iota}\left(\sum_{i=1} y_i\right)) \times h_i^3 = (\widehat{y_\iota} - y_i) \times h_i^3$$

$$\frac{\partial L_n}{\partial W_{j,i}^3} = \frac{\partial L_n}{\partial h_j^3} \times \frac{\partial h_j^3}{\partial x_j^3} \times \frac{\partial x_j^3}{\partial W_{j,i}^3}$$

$$x_j^3 = \sum_{i=1} W_{j,i}^3 h_i^2 \;\rightarrow\; \frac{\partial x_j^3}{\partial W_{j,i}^3} = h_i^2$$

$$h_j^3 = LeakyReLU\left(x_j^3\right) \rightarrow \frac{\partial h_j^3}{\partial x_j^3} = LeakyReLU'\left(x_j^3\right)$$

$$\frac{\partial L_n}{\partial W_{j,i}^3} = (\sum\nolimits_{k=1}^{3} (\widehat{y_k} - y_k) \times W_{k,j}^4) \times LeakyReLU'\left(x_j^3\right) \times h_i^2$$

$$\frac{\partial L_n}{\partial W_{j,i}^2} = \frac{\partial L_n}{\partial h_j^2} \times \frac{\partial h_j^2}{\partial x_j^2} \times \frac{\partial x_j^2}{\partial W_{j,i}^2}$$

$$x_j^2 = \sum_{i=1} W_{j,i}^2 \, h_i^1 \ \rightarrow \ \frac{\partial x_j^2}{\partial W_{j,i}^2} = h_i^1$$

$$h_j^2 = LeakyReLU\left(x_j^2\right) \rightarrow \frac{\partial h_j^2}{\partial x_j^2} = LeakyReLU'\left(x_j^2\right)$$

$$\frac{\partial L_n}{\partial W_{j,i}^2} = (\sum\nolimits_{k=1}^{3} (\widehat{y_k} - y_k) \times \sum\nolimits_{p=1}^{5} W_{k,p}^4) \times LeakyReLU'\left(x_j^3\right) \times W_{p,j}^3)) \times LeakyReLU'\left(x_j^2\right) \times h_i^1$$

$$\frac{\partial L_n}{\partial W_{j,i}^1} = \frac{\partial L_n}{\partial h_j^1} \times \frac{\partial h_j^1}{\partial x_j^1} \times \frac{\partial x_j^1}{\partial W_{j,i}^1}$$

$$x_j^1 = \sum_{i=1} W_{j,i}^1 \, X_i \ \rightarrow \ \frac{\partial x_j^1}{\partial W_{j,i}^1} = X_i$$

$$h_j^1 = LeakyReLU\left(x_j^1\right) \rightarrow \frac{\partial h_j^1}{\partial x_j^1} = LeakyReLU'\left(x_j^1\right)$$

$$\frac{\partial L_n}{\partial W_{j,i}^1} = (\sum\nolimits_{k=1}^{3} (\widehat{y_k} - y_k) \times \left(\begin{array}{c} \sum\nolimits_{p=1}^{5} W_{k,p}^4 \times LeakyReLU'^{\left(x_j^3\right)} \times \\ \left(\sum\nolimits_{m=1}^{5} W_{p,m}^3 \times LeakyReLU'\left(x_j^2\right) \times W_{m,j}^2 \right) \end{array} \right)) \times LeakyReLU'\left(x_j^1\right) \times X_i$$

ج) در مورد روش های مقدار دهی اولیه‌ی Xavier initialize و He initialize تحقیق کنید. با فرض جایگزینی فعالساز شبکه با فعالساز sigmoid هر دو روش را اعمال و نتایج را قیاس کنید. همچنین فعال سازهای متناسب با آنها (شهود آزمایشی و عملی) را تحقیق کنید. بر اساس مقاله زیر:

He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing humanlevel performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

مقدار دهی اولیه در شبکه‌های عصبی عملیاتی است که در آن وزن‌ها و بایاس‌ها در لایه‌های مختلف شبکه با مقادیر اولیه تعیین می‌شوند. این مقادیر اولیه می‌توانند تأثیر قابل توجهی بر عملکرد شبکه داشته باشند. در سال‌های اخیر، دو روش مقدار دهی اولیه به نام‌های Xavier و He توسط Xavier Glorot و Kaiming He معرفی شدند که و به دلیل ویژگی‌های خاصی که دارند، می‌توانند بهبود قابل توجهی در آموزش شبکه‌ها ایجاد کنند. در مقاله ارائه شده توسط He و همکارانش در سال 2015، یک روش مقداردهی اولیه برای شبکه‌های عصبی با استفاده از فعالسازی ReLU معرفی شده است. این روش معروف به He initialization است. همچنین، در مقاله اولیه Xavier

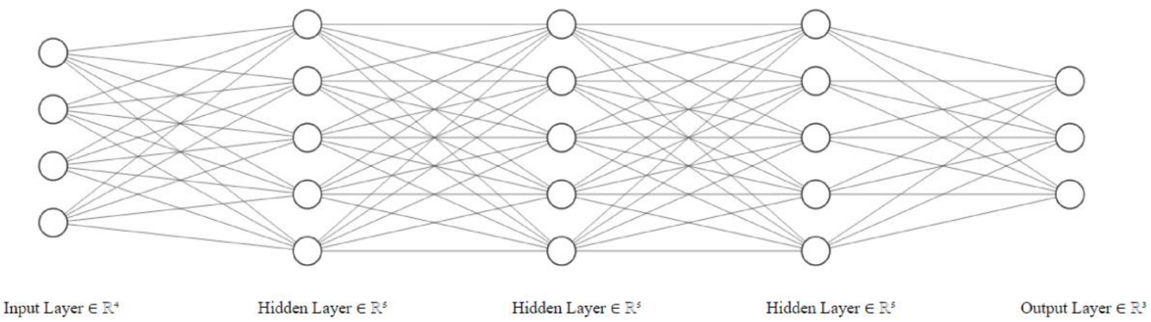
Glorot و همکارانش در سال 2010، یک روش مشابه برای مقداردهی اولیه در شبکه‌های عصبی با استفاده از فعالسازی sigmoid و tanh معرفی شد. این روش معروف به Xavier initialization است.

Xavier Initialization: روش مقدار دهی اولیه Xavier بر اساس ایده‌ای از نظر آماری به وجود آمده است. در این روش، وزن‌ها به صورت تصادفی از یک توزیع نرمال با میانگین صفر و واریانس برابر با n/1 خروجی لایه قبلی نمونه‌برداری می‌شوند. برای لایه‌های با تابع فعالسازی خطی، این روش به خوبی عمل می‌کند. با این حال، وقتی از فعالسازهایی مانند ReLU استفاده می‌شود، توزیع خروجی لایه قبلی غیر خطی است و در نتیجه Xavier Initialization به طور معمول به عنوان روشی کارا برای مقداردهی اولیه در این حالت محسوب نمی‌شود.

He Initialization: روش مقداردهی اولیه He بر اساس نظریه تحلیلی استوار بر ایده‌ای از نظر آماری استوار است. در این روش، وزن‌ها به صورت تصادفی از یک توزیع نرمال با میانگین صفر و واریانس برابر با n/2 خروجی لایه قبلی نمونه‌برداری می‌شوند. این روش برای فعالسازی‌های خطی و غیر خطی (مانند ReLU) عملکرد بسیار خوبی دارد. در مقاله مذکور، نویسندگان اثبات کردند که استفاده از He Initialization باعث می‌شود تا شبکه‌ها با فعالسازی ReLU به طور معمول بهبود عملکردی نسبت به روش‌های دیگر داشته باشند.

استفاده از فعالسازهای مختلف می‌تواند تأثیر زیادی بر کارایی مقداردهی اولیه داشته باشد. مقدار دهی اولیه Xavier برای فعالساز sigmoid مناسب است ولی در صورت جایگزینی فعالساز sigmoid با فعالساز ReLU، Xavier Initialization می‌تواند باعث ایجاد مشکل vanishing gradient شود، زیرا خروجی لایه قبلی توزیعی نرمال نیست و واریانس آن متفاوت است. از طرف دیگر، He Initialization بهبود قابل توجهی در عملکرد شبکه با فعالساز ReLU ایجاد می‌کند، زیرا با توجه به توزیع خروجی لایه قبلی، مقادیر اولیه وزن‌ها برای استفاده از ReLU مناسب تر هستند. به طور خلاصه، Xavier initialization با فعالسازی sigmoid و He initialization با فعالسازی ReLU بهترین عملکرد را دارند ولی به طور کلی، انتخاب مقداردهی اولیه مناسب و فعالساز مناسب بستگی به معماری شبکه، نوع عملکرد و ویژگی‌های داده‌ها دارد. همچنین، ممکن است نیاز باشد تا با امتحان و خطا بهترین ترکیب مقداردهی اولیه و فعالساز را برای هر مورد خاص تعیین کرد.

د) آزمایش بخش اول را طوری انجام دهید که در هزارمین لایه عملیات متوقف شود و موارد ضروری جهت آموزش (شامل مقادیر و ماژول های loss و optimizer و model weights) ذخیره گردند و مجددا با فراخوانی موارد ذخیره شده با شروع دوباره برنامه، آموزش ادامه یابد.



Weight initialize: sample from UNIFORM distribution (low = 0, high = 1)

Split rate for Test size to Train size = 0.2

Active Functions: Leaky-Relu (0.2) for hidden layers , Softmax for output layer

Loss : Cross Entropy

Optimizer: Adam (lr:0.001)

Epochs: 2000

Nodes in each hidden layer: 5

Base number of hidden layers: 3

کد مرتبط با تمرین 7:

- **DL-HW3-Q07-Mahdavi.ipynb**

تمرین 8:

الف) با مراجعه به مقاله زیر توضیح دهید که نرمال سازی Batch Normalization چه عملی انجام می دهد و چگونه با کاهش Internal Covariate Shift باعث سرعت بخشیدن به آموزش مدل می شود؟ چهار مزیت استفاده از این روش را با رجوع به مقاله مرتبط توضیح دهید:

Ioffe, S., Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). pmlr.

عملکرد نرمال‌سازی Batch Normalization:

Batch Normalization معمولاً در لایه‌های میانی شبکه‌های عصبی اعمال می‌شود و به طور خاص در معماری‌های مبتنی بر شبکه‌های عصبی کانولوشنی (Convolutional Neural Networks) استفاده می‌شود، اما می‌تواند در شبکه‌های عصبی fully connected نیز مورد استفاده قرار گیرد.

در فاز آموزش، Batch Normalization، در هر لایه از یک شبکه عصبی عمیق، ورودی‌های یک batch از داده‌ها را متناسب با میانگین و انحراف معیار آن batch، نرمال‌سازی می‌کند. برای این منظور با محاسبه میانگین و انحراف معیار داده‌های آن batch، داده‌های ورودی را متناسب با یک توزیع نرمال با میانگین صفر و واریانس یک مقیاس‌بندی و مرکزبندی می‌کند. این عمل باعث می‌شود تا ورودی‌های لایه بعدی به طور مشابهی توزیع شوند، شبکه بتواند با مقادیر اولیه‌ی نزدیک به صفر کار و همچنین باعث ایجاد بیشترین مشتق‌پذیری در تابع هدف شود. به این ترتیب، آموزش مدل شبکه بهبود یافته و بسیار سریع‌تر انجام می‌گیرد. در عمل، برای هر وزن (weight) و بایاس (bias) در لایه مورد نظر، پارامترهایی به نام γ (scaling factor) و β (shift factor) بر اساس $y = \gamma\hat{x} + \beta$ بهینه‌سازی می‌شوند. پس از اعمال Batch Normalization در یک لایه، خروجی نرمال‌شده لایه قبلی با استفاده از این پارامترها تغییر شکل داده می‌شود. این عمل برای هر نورون در لایه انجام می‌شود و باعث می‌شود تا شبکه به‌طور کلی به صورت موثرتری آموزش ببیند و داده‌های جدید را پیش‌بینی کند. در فاز آزمون، از میانگین و انحراف معیاری استفاده می‌شود که در طول آموزش برای تمام batch ها محاسبه شده و به عنوان یک مقدار ثابت ذخیره می‌شوند. این کار باعث می‌شود تا نتایج آزمون پایدارتر و قابل تکرارتر شوند. در کل، مقاله Batch Normalization را به عنوان یک روش موثر برای سرعت بخشی به آموزش شبکه‌های عمیق معرفی کرده است. با کاهش تغییرات توزیع داده‌ها و استفاده از میانگین و انحراف معیار، Batch Normalization بهبود قابل توجهی در پایداری، سرعت آموزش، عملکرد شبکه‌های عمیق و تعمیم‌پذیری آن‌ها به داده‌های جدید تست ایجاد می‌کند.

تأثیر کاهش Internal Covariate Shift:

Internal Covariate Shift به تغییرات آماری در توزیع ورودی‌های لایه‌های مختلف شبکه‌های عمیق در هر مرحله از آموزش اشاره دارد که به دنبال تغییر پارامترهای لایه‌ها و عبور داده‌ها از توابع فعالسازاتفاق می‌افتد. این امر با نیاز به نرخ‌های یادگیری پایین‌تر و

مقداردهی اولیه پارامترها، سرعت، پایداری و کیفیت آموزش شبکه آموزش را کاهش داده و آموزش مدل‌هایی با غیرخطی‌های اشباع‌کننده (saturating nonlinearities) را بسیار سخت می‌کند. استفاده از Batch Normalization، تغییرات در توزیع ورودی‌های هر لایه در طول آموزش یا Internal Covariate Shift را کاهش داده و توزیع ورودی‌های لایه‌های بعدی را به یک توزیع ثابت نزدیک می‌کند. به این ترتیب، نیاز به تنظیم مجدد نرخ یادگیری در هر مرحله کاهش یافته، همچنین از مشکل اشباع شدن (saturating) توابع فعال‌سازی در شبکه‌های عمیق جلوگیری می‌شود و به طور کلی باعث می‌شود تا فرآیند آموزش سریع‌تر و پایدارتر انجام گیرد.

مزایای استفاده از Batch Normalization:

1. Batch Normalization همگرایی شبکه در مراحل آموزش را افزایش داده و موجب کاهش زمان و سرعت بخشی به آموزش مدل‌های شبکه‌های عصبی عمیق می‌شود. با مجموعه‌سازی ورودی‌های هر Batch، Batch Normalization مقیاس داده‌ها را تغییر نمی‌دهد و بنابراین شبکه بسیار سریع‌تر آموزش داده می‌شود. همچنین استفاده از Batch Normalization، نیاز به نرخ یادگیری کم را از بین برده و این امکان را می‌دهد که از نرخ های یادگیری بسیار بالاتری استفاده گردد که باز منجر به سرعت دادن به آموزش مدل می‌شود.

2. استفاده از Batch Normalization، نیاز به انتخاب دقیق و تنظیم پارامترهای شبکه به طور مستقل به منظور آموزش موثر را کاهش داده، همچنین حساسیت مدل را به مقداردهی اولیه و تغییرات کوچک در داده‌های ورودی که در طول آموزش ایجاد می‌شود را کمتر می‌کند. این به معنای کاهش وابستگی شبکه به تنظیم‌های اولیه و پارامترهای مربوط به هر لایه است که مدل را پایدارتر می‌کند.

3. در شبکه‌های عمیق با افزایش عمق شبکه، مشکلاتی مانند گرادیان ناپایدار، کاهش گرادیان (Vanishing Gradient) و اشباع توابع فعال‌سازی وجود دارد. با استفاده از Batch Normalization، توزیع ورودی‌ها به لایه‌های عمیق‌تر بهبود یافته و از اشباع شدن توابع فعال‌سازی در این لایه‌ها جلوگیری می‌شود. این موضوع باعث می‌شود که شبکه‌های عمیق با کیفیت‌تری آموزش داده شده و عملکرد و دقت در زمان تست بهبود یابد.

4. استفاده از Batch Normalization، در برخی موارد نیاز به Dropout را از بین می برد.

ب) تحقیق کنید Batch Normalization باید قبل دراپ اوت (dropout) باشد یا بعد از آن؟ وضعیت این دو نسبت به فعالساز چگونه است؟ این نتیجه گیری طبق تئوری است یا مشاهده عملی؟

در حالت کلی، وقتی از Batch Normalization و Dropout به طور همزمان در یک شبکه عصبی استفاده می‌شود، معمولاً Batch Normalizationقبل از Dropout قرار می‌گیرد هرچند که بر اساس این مقاله بکارگیری Batch Normalization، در برخی موارد نیاز به Dropout را از بین برده و مانع منظم‌سازی بیش از اندازه می‌شود ولی مطالعات دیگر نشان داده‌اند که استفاده از Batch Normalization قبل از Dropout می‌تواند بهبود عملکرد شبکه را تسهیل کند.

Batch Normalization به عنوان یک عمل پیش‌پردازشی (preprocessing)، روشی است که در هر لایه از شبکه، بعد از عملیات خطی (مانند ضرب ماتریسی) و قبل از تابع فعال‌سازی اعمال می‌شود و از طریق محاسبه میانگین و انحراف معیار، تغییرات ورودی را کاهش داده و باعث نرمال‌سازی داده‌های ورودی در هر batch می‌شود. بنابراین خروجی از Batch Normalization به عنوان ورودی برای تابع فعال‌سازی استفاده می‌شود. این کار باعث کاهش واریانس گرادیان‌ها شده و از این رو، Batch Normalization می‌تواند به طور مؤثری در پیشگیری از gradient vanishing/exploding عمل کرده و به آموزش سریعتر و افزایش پایداری شبکه عصبی کمک کند.

از طرفی، Dropout یک روش منظم‌سازی (regularization) برای کاهش بیش‌برازش (overfitting) است که در آن، با احتمال مشخص (معمولاً بین 0.2 تا 0.5) تعدادی از نورون‌ها در هر لایه از شبکه عصبی به صورت تصادفی غیرفعال می‌شوند. این باعث می‌شود که شبکه

در طول فرآیند آموزش بجای وابستگی بیش‌اندازه به نورون‌های خاص، یادبگیرد وزن‌ها را بین تمام واحدها تقسیم کرده و باعث افزایش توانایی شبکه در تعمیم‌پذیری به داده‌های تست شود. این فرایند نیز قبل از تابع فعال‌سازی انجام می‌شود. یعنی خروجی از Dropout به عنوان ورودی برای تابع فعال‌سازی استفاده می‌گردد.

به این ترتیب Dropout با حذف تصادفی برخی از واحدها و اطلاعات، که احتمال وقوع آن به برخی از ویژگی‌ها وابسته است، میانگین و واریانس Batchها را تغییر می‌دهد بنابراین قرارگیری آن قبل از Batch Normalization ممکن است باعث نقض فرضیات Batch Normalizationشده و موجب شود که Batch Normalization توانایی خود را در نرمال‌سازی داده‌ها از دست داده و نتایج و عملکردش را تضعیف کند. بنابراین، به لحاظ تئوری و بر اساس نتایج عملی معمولا Batch Normalization قبل از Dropout پیشنهاد می‌شود. هر چند که ممکن است روی مجموعه داده خاص و بر اساس اندازه، عملکرد و یا سایر پارامترهای یک شبکه خلاف آن نیز جواب دهد.

ج) بررسی کنید Batch Normalization در داده های آزمایشی (تست) چگونه مورد استفاده قرار می‌گیرد و محاسبه می‌شود.

Batch Normalization در زمان فرآیند آموزش شبکه استفاده می‌شود و در هر لایه میانگین (mean) و واریانس (variance) هر Batch داده را محاسبه کرده و با استفاده از آن‌ها، ورودی‌ها را نرمال می‌کند. مقادیر نرمال‌سازی شده با استفاده از ضریب اسکیل (scale factor) و تغییر (shift factor) مقیاس‌بندی و تبدیل می‌شوند. این ضریب‌ها به صورت پارامترهای آموزشی در هنگام آموزش شبکه یاد گرفته می‌شوند. اما در فاز آزمایش (تست) شبکه، Batch Normalization غیرفعال است، به این معنی که در این مرحله، میانگین و واریانس بر اساس داده‌های آزمایش محاسبه نمی‌شود، بلکه از مقادیر میانگین و واریانس کلی داده‌های فاز آموزش برای نرمال سازی ورودی‌ها استفاده می‌شود. یعنی در فاز آزمایش، Batch Normalization به عنوان یک لایه عبور می‌کند و ورودی‌ها را بدون تغییر به لایه بعدی ارسال می‌کند و داده‌های آزمایشی با استفاده از ضریب اسکیل و تغییر fit شده در مرحله آموزش تبدیل می‌شوند. در اصل، Batch Normalization در زمان آزمایش نیز مزایایی مانند بهبود پایداری شبکه و کاهش تغییرات ناشی از Batchهای کوچک داده‌ها را دارد. با این حال، ممکن است برای تمامی نقاط داده‌های آزمایشی، میانگین و واریانس محاسبه شده درست نباشد. از این رو، Batch Normalization در زمان آزمایش معمولاً با مقادیر متوسط و واریانسی که در طول فرایند آموزش محاسبه شده‌اند، کار می‌کند.

Batch Normalization در PyTorch به صورت پیش‌فرض از مقادیر میانگین و واریانس تخمینی محاسبه شده در طول آموزش استفاده می‌کند و اگر بخواهیم در فاز آزمایش (model.eval()) از مقادیر میانگین و واریانس تخمینی استفاده نشود، می‌توان از پارامتر `track_running_stats=False` در ایجاد ماژول Batch Normalization استفاده کرد تا مقادیر میانگین و واریانس را به صورت دستی تنظیم کنیم.

د) نشان دهید که یک مدل Maxout با فقط دو واحد پنهان می‌تواند هر تابع دلخواه پیوسته از $v \in R^n$ را تخمین بزند. مطابق رفرنس زیر:

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y. (2013, May). Maxout networks. In International conference on machine learning (pp. 1319-1327). PMLR.

در مقاله "Maxout Networks" نوشته شده توسط Goodfellow و همکارانش در سال 2013، نشان داده شده است که یک مدل عصبی Maxout با دو واحد پنهان می‌تواند هر تابع پیوسته f(v) از فضای ورودی $v \in R^n$ را با دقت خوبی تخمین زده و بزرگترین مقدار خروجی

از یک مجموعه ورودی را انتخاب و نتیجه‌ی این تخمین را به عنوان خروجی مدل در نظر بگیرد. مدل Maxout یک مدل عصبی است که از ترکیب چندین واحد Maxout با دو لایه fully connected تشکیل شده است و در هر لایه از آن از تابع Maxout استفاده می‌شود. Maxout یک نوع لایه فعالسازی است که می‌تواند به عنوان بخشی از معماری شبکه عصبی استفاده شود. این لایه توانایی بالقوه برای تقریب هر تابع پیوسته را دارد. اصل ایده در Maxout این است که از بین تعدادی تابع خطی ساده، برای انتخاب بزرگترین خروجی استفاده می‌شود.

در حالت ساده، فرض می‌کنیم تابع دلخواه پیوسته $f \rightarrow \mathbb{R}\mathbb{R}^n$ است. برای تخمین این تابع با استفاده از Maxout با دو واحد پنهان، می‌توانیم از دو لایه خطی استفاده کنیم که هر کدام دارای یک واحد خروجی هستند. در واقع، هر کدام از این دو لایه می‌توانند تابعی خطی از v را تخمین بزنند. بنابراین، می‌توانیم دو لایه Maxout را به صورت زیر تعریف کنیم:

$$h_1 = max(w_1^T v + b_1, W_2^T v + b_2)$$

$$h_2 = max(w_3^T v + b_3, W_4^T v + b_4)$$

در اینجا، w_i و b_i پارامترهای لایه Maxout هستند و i از 1 تا 4 می‌تواند مقادیر 1 یا 2 را بگیرد. حالا، با داشتن این دو لایه Maxout، می‌توانیم تابع دلخواه f را به صورت زیر تخمین بزنیم:

$$\hat{f}(v) = max(w_1^T v + b_1, W_2^T v + b_2) \times max(w_3^T v + b_3, W_4^T v + b_4)$$

از آنجا که Maxout می‌تواند هر تابع پیوسته‌ای را با دقت دلخواه تخمین بزند، می‌توانیم نتیجه بگیریم که مدل Maxout با فقط دو واحد پنهان هر تابع دلخواه پیوسته‌ای از v را تخمین می‌زند.

ه) توضیح دهید batch normalization در یک لایه پیچشی دو بعدی $tesnor \in R^{(N,C,H,W)}$ چگونه عمل می‌کند؟

Batch Normalization در یک Convolutional Layer دو بعدی با ورودی $tesnor \in R^{(N,C,H,W)}$ به این صورت عمل می‌کند که در ابتدا، Batch Normalization میانگین و واریانس ورودی‌های لایه را بر اساس minibatch جاری محاسبه می‌کند. در اینجا N تعداد نمونه‌ها در batch size، C تعداد کانال‌ها یا ویژگی‌های ورودی، H و W اندازه بعد ورودی (ارتفاع و عرض) هستند. سپس، مقادیر میانگین و واریانس به دست آمده را برای استانداردسازی ورودی‌های لایه استفاده می‌کند. برای هر ویژگی C و هر نمونه N در batch size، ورودی استانداردشده $\hat{x}$ با استفاده از رابطه زیر محاسبه می‌شود:

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\delta_c^2 - \epsilon}}$$

در این رابطه، $\hat{x}$ نشان‌دهنده مقدار استاندارد شده ورودی است، x ورودی اصلی است، μ میانگین ورودی‌های کانال C است و δ^2 واریانس ورودی‌های کانال C است. همچنین، $\in$ یک عدد کوچک است که به منظور جلوگیری از تقسیم بر صفر در صورتی که واریانس بسیار کوچک باشد، به آن اضافه می‌شود.

پس از استانداردسازی ورودی‌ها، مقیاس‌بندی و تغییر مقیاس انجام می‌شود. این عمل با استفاده از دو پارامتر قابل یادگیری، یعنی وزن (γ (scaling factor)) و بایاس (β (shift factor)) صورت می‌گیرد. به ازای هر ویژگی C و هر نمونه N در batch size، خروجی نهایی y با استفاده از رابطه زیر محاسبه می‌شود:

$$y = \gamma\hat{x} + \beta$$

در اینجا، y نشان دهنده خروجی نهایی است که به لایه بعدی ارسال می‌شود، γ وزن و β بایاس برای کانال Cهستند.

ی) تفاوت‌های batch normalization و Layer normalization را بررسی کنید.

Batch Batch Normalization و Layer Normalization هر دو تکنیک‌های مهم در شبکه‌های عصبی هستند که به استقرار توزیع ورودی‌ها و کاهش وابستگی به مقیاس و توزیع خاص ورودی‌ها کمک می‌کنند. با این حال، تفاوت‌های مهمی بین این دو وجود دارد.

در Batch Normalization، مقیاس نرمال‌سازی براساس مقادیر میانگین و واریانس در طول batch صورت می‌گیرد. به عبارت دیگر، میانگین و واریانس بر اساس داده‌های ورودی در هر batch استانداردسازی می‌شوند و در جهت بعد batch است. در Layer Normalization، مقیاس نرمال‌سازی براساس مقادیر میانگین و واریانس در طول یک لایه صورت می‌گیرد و میانگین و واریانس بر اساس featureهای (کانال‌ها) هر نقطه در فضای ورودی در هر لایه استانداردسازی می‌شوند. در Batch Normalization، نتیجه وابسته batch است و بر اساس داده‌های موجود در هر batch متفاوت است. در Layer Normalization، نتیجه وابسته به ورودی هر نقطه (کانال) در لایه است و بر اساس featureهای هر نقطه در فضای ورودی تعیین می‌شود.

این تفاوت‌ها در عملکرد و توانایی این دو روش موثر است. به عنوان مثال، Batch Normalization معمولاً در دسته‌بندی‌های با مقیاس متفاوت به ویژه در شبکه‌های پیچشی که با تصاویر به خوبی عمل می‌کند و کمک به تسریع آموزش شبکه می‌کند. در حالتی که، Layer Normalization برای داده‌های با ساختار زمانی مناسب است و می‌تواند به استقرار توزیع ورودی‌ها در سطح هر نقطه در لایه کمک کند و در مواردی که ارتباط بین ویژگی‌ها در طول زمان یا فضا مهم است، استفاده می‌شود. به این ترتیب Batch Normalization معمولاً در Convolutional Layers و Fully Connected Layers استفاده می‌شود در حالی که Layer Normalization معمولاً در RNNها محبوب هستند و در شبکه‌هایی مانند LSTM (Long Short-Term Memory) و GRU استفاده می‌شود.

و) بررسی کنید که دراپ اوت (dropout) در شبکه CNN به چه صورت مورد استفاده است؟ (در لایه پیچشی)

Dropout یک تکنیک مهم در شبکه‌های عصبی است که به کنترل بیش برازش (overfitting) و افزایش تعمیم‌پذیری شبکه کمک می‌کند. در شبکه‌های CNN نیز می‌توان از Dropout در لایه‌های پیچشی استفاده کرد به این ترتیب که پس از عمل Convolution و قبل از اعمال فعالساز غیرخطی (مثل ReLU)، Dropout را اعمال می‌کنیم. Dropout به طور تصادفی برخی از ویژگی‌های (نرون‌ها) خروجی را در طول آموزش غیرفعال (صفر) می‌کند. این عمل موجب ایجاد انواع مختلف زیرمجموعه‌ها از ویژگی‌ها می‌شود، که به شبکه کمک می‌کند تا به صورت متوسط وابستگی‌های مفید و قوی‌تری را یاد بگیرد و از بیش برازش به داده‌های آموزش جلوگیری کند. در مرحله آزمایش، Dropout غیرفعال می‌شود و همه ویژگی‌ها به طور کامل فعال هستند. این به شبکه کمک می‌کند که در آزمون، خروجی‌های قابل پیش‌بینی تولید کند.

بسته به نوع مسئله گاهی نیاز است که کل کانال را صفر کنیم، در این صورت از Dropout دوبعدی استفاده می‌گردد. این مورد در صورتی که تعداد کانال ها کم باشد مناسب نبوده ولی در موارد large-scale می‌تواند استفاده شود. همچنین Dropout در لایه‌های پیچشی ممکن است در بعضی موارد منجر به از دست رفتن اطلاعات مکانی مرتبط باشد، بنابراین باید با دقت و با توجه به ویژگی‌های خاص مسئله و معماری شبکه استفاده شود. همچنین، مقدار dropout_rate باید به طور آزمایشی انتخاب شود تا بهترین نتایج را بر اساس مسئله بدست آورد.

تمرین 9:

دو شبکه زیر را که در آنها لایه‌ها تماما خطی با توابع فعالساز غیرخطی هستند، درنظر بگیرید. تفاوت دو شبکه در تابع فعالساز است. تعداد گره‌های لایه ورودی و خروجی را متناسب با تعداد ویژگی‌ها و تعداد کلاس‌های مربوط به داده و لایه‌های پنهان توضیح داده شده پیدا کنید و با هدف کلاس بندی، شبکه‌ها را با تنظیمات و مفروضات مشخص شده آموزش دهید. ضمن گزارش خطا و هزینه برای داده‌های آموزشی و آزمایشی، مقدار نهایی (پس از اتمام تکرار ها) نرم یک و نرم دوم گرادیان وزن‌های هر لایه در طول گذار به سمت لایه خروجی در یک نمودار برحسب شماره لایه یادداشت کنید و برای دو شبکه قیاس کنید. این آزمایش برای شبکه‌های مشابه اما با تعداد لایه‌های پنهان ۳ و ۹ و ۱۳ و ۱۷ و ۱۹ لایه تکرار شود.

Dataset: Breast Cancer (Use scikit-learn)

Train/test Split: ratio of (number of test samples / number of train samples) = 0.1

Optimizer: Adam, Learning rate = 0.001

Loss: Cross-Entropy

Number of epochs: 50

Number of nodes in each hidden layer: 5

Active functions:

for model A: Sigmoid

for model B: ReLU

در قدم بعدی با افزودن batchnorm در هر لایه برای هر دو مدل آزمایش را تکرار کنید و نتایج را قیاس کنید.

کد مرتبط با تمرین 9:

- **DL-HW3-Q09-Mahdavi.ipynb**

تمرین 10:

الف) شبکه زیر را با خواسته‌های مشخص شده پیاده سازی کنید و با تغییر میزان اندازه دسته آموزشی batch size با اندازه های ۱ و ۸ و ۳۲ و ۵۰، اعوجاج در نمودارهای هزینه - تکرار و خطا - تکرار را برای داده های آموزشی در هردسته (و در هر تکرار) و برای داده‌های آزمایشی در حین تکرار های آموزش (در هر تکرار) قیاس کنید. (برای حصول نمودار های مطلوب برای حالت تست، عملیات مربوط به تست، باید در حین آموزش در هر تکرار اجرا شود). (تابع هزینه کراس انتروپی (Cross Entropy) مطلوب است اما با درنظر گرفتن فعالساز لایه خروجی، باید از تابع دیگری استفاده شود تا برآیند فعالساز خروجی واین تابع خطا معادل تابع هزینه کراس انتروپی تعریف شده در پایتورچ شود.) برای تمرین بیشتر می توانید میزان مصرف حافظه (GPU/CPU) و سرعت اجراها را نیز مقایسه کنید.

Data: MNIST (from torchvision[URL: https://pytorch.org/vision/stable/index.html])
Optimizer:
Epochs: 15
Loss: Cross-Entropy

- DL-HW3-Q10A-Mahdavi.ipynb

ب) با کمک torch.optim.lr_scheduler و با درنظر گیری آهنگ مناسب (به عنوان مثال تغییر آهنگ نرخ یادگیری با ضریب، gamma=0.9) شبکه را با تنظیمات بخش الف اما با batchsize=50 طوری آموزش دهید که نرخ یادگیری در هر تکرار نسبت به تکرار قبلی کاهش یابد و نتایج مطلوب را برای داده‌های آموزشی و آزمایشی گزارش کنید.

- DL-HW3-Q10B-Mahdavi.ipynb

ج) شبکه را با batchsize=32 و با سایر تنظیمات مشابه بخش الف ولی با توابع هزینه زیر آموزش دهید و نتایج را قیاس کنید: (با کمک توابع هزینه از torch.nn)

nn.MultiLabelSoftMarginLoss() ,
nn.MultiMarginLoss() ,
nn.PoissonNLLLoss() ,
nn.GaussianNLLLoss() ,
nn.MultiLabelMarginLoss()

- DL-HW3-Q10C-Mahdavi.ipynb

د) فرض کنید که یک مســـئله کلاس بندی با دو کلاس دارید. این مســـئله را می‌توان با دو گره خروجی و یا یک گره خروجی پیاده سازی کرد؟ چه تفاوتی وجود دارد؟ درصورت پاسخ مثبت با فرض اینکه هدف از کلاس بندی مجموعه داده بالا صـرفادو کلاسـه باشـد(یک کلاس برای حضـور داده در کلاس اول (صـفر بودن رقم دسـتنویس) و یک کلاس برای عدم حضـور در کلاس اول (عدم صـفر بودن رقم دسـتنویس)) با این دو دیدگاه کلاس بندی را انجام دهید و نتایج را مقایسه کنید.

- DL-HW3-Q10D-Mahdavi.ipynb

ه) دربخش د، با توجه به بالانس نبودن تعداد داده‌ها چه معیارهایی برای ارزیابی‌های مشـــابه دقت (Accuracy) برای تحلیل بهتر می‌توان در نظر گرفت؟ در صورت لزوم در پیاده سازی و نتایج خود این موضوع را درنظر بگیرید.

- DL-HW3-Q10E-Mahdavi.ipynb

معماری شبکه:

Layer1	Conv2d (in-channels=1, out-channels=16, kernel-size=3, padding=1), ReLU, MaxPool2d (kernel-size=2, stride=2)
Layer2	Conv2d (in-channels=16, out-channels=32, kernel-size=3, padding=1), ReLU, MaxPool2d (kernel-size=2, stride=2)
-	flatten: view(-1, ?)
Layer3	linear(? to 1024), ReLU, dropout
Layer4	linear(1024 to 10), log-softmax

تمرین 11:

الف) با کتابخانه opencv-python فیلترهای زیر را بر تصویر ضمیمه شده Pic1 اعمال کنید، نتیجه را نمایش دهید و تحلیل انجام دهید.

1. Sharpen kernel

0	-1	0
-1	5	-1
0	-1	0

2. Laplacian kernel

0	1	0
1	-4	1
0	1	0

3. Emboss kernel

-2	-1	0
-1	1	1
0	1	2

4. Outline kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

5. Bottom sobel

-1	-2	-1
0	0	0
1	2	1

6. Right sobel

-1	0	1
-2	0	2
-1	0	1

7. Top sobel

1	2	1
0	0	0
-1	-2	-1

8. Difference

-1	-1	-1
-1	8	-1
-1	-1	-1

9. Weighted average

1	1	1
1	8	1
1	1	1

ب) تصویر ضمیمه شده Pic2 را به عنوان ماسک درنظربگیرید و با کمک opencv-python تطبیق الگو با تصویر Pic1 را به روش‌های زیر انجام دهید و نتایج را مشاهده کنید.

1) Cross Correlation
2) Normalized Cross Correlation
3) Correlation Coeffcient
4) Normalized Correlation Coeffcient
5) Square Difference
6) Normalized Square Difference

ج) بررسی کنید که چگونه یک فیلتر پیچشی مشتق گیر تک بعدی [-1 ;2; -1] را برای فضای دو بعدی (ماتریسی) باید نوشت.

د) در مورد نحوه کار فیلتر های Canny و bilateralFilter تحقیق کنید و این دو فیلتر را نیز مشابه بخش الف بر تصویر مربوطه اعمال کنید.

*تصاویر از مجموعه داده زیر انتخاب شده اند:

Dataset: ”iphone2dslr_flower”. This dataset is part of the CycleGAN datasets, originally hosted here: https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

کد مرتبط با تمرین 11:

- **DL-HW3-Q11-Mahdavi.ipynb**

تمرین 12:

جدول زیر مربوط به یک شبکه عصبی شامل لایه‌های پیچشی و خطی است. با فرض اینکه تعداد کلاس‌های داده ورودی ۱۰۰۰ مورد است جدول را پر کنید. مدل در حال انجام وظیفه کلاس بندی است.

Layer name (type)	Specification	Output dim	Number of parameters
INPUT	-	(3,227,227)	-
convolution layer	in=3 out=96 k=11 p=0 s=4	(96,55,55)	(3*96*11*11)+96=34944
*Max Pooling	in=96 out=96 k=3 p=0 s=2	(96,27,27)	0
convolution layer	in=96 out=256 k=1 p=0 s=1	(256,27,27)	(96*256*1*1)+256=24832
Max Pooling	in=256 out=256 k=3 p=0 s=2	(256,13,13)	0
convolution layer	in=256 out=384 k=3 p=1 s=1	(384,13,13)	(256*384*3*3)+384=885120
convolution layer	in=384 out=384 k=3 p=1 s=1	(384,13,13)	1327488
convolution layer	in=384 out=256 k=3 p=1 s=1	(256,13,13)	(384*256*3*3)+256=884992
Max Pooling	in=256 out=256 k=3 p=0 s=2	(256,6,6)	0
Linear (flatten)	from 6*6*256=9216 to 4096	-	9216*4096=37748736
linear	from 4096 to 4096	-	4096*4096=16777216
linear	from 4096 to 1000	1000 (#class)	4096*1000=4096000

توجه شود به عنوان مثال (3,10,20) به معنی تنسور با ۳ کانال و طول و عرض ۱۰و ۲۰ می‌باشد. همچنین p به معنی پدینگ و s به معنی طول گام و k به معنی اندازه فیلتر و in به معنی تعداد کانال‌های ورودی و out تعداد کانال‌های خروجی است.اندازه فیلتر پولینگ نیز با k نمایش داده می شود. در شمارش در اینجا تعداد پارامترهی مربوط به بایاس نیز محسوب شده است به این طریق که برای هر کانال خروجی تنها یک واحد پارامتر بایاس وجود دارد. در صورت عدم شمارش بایاس، در سطر هفتم تعداد پارامترها به 1327104 می‌رسد.

برای حل جدول از فرمول زیر استفاده گردید:

$$\frac{n+2p-k}{s}+1$$

n: size, s: stride, p: padding, k: kernel size

تعداد پارامترها برابر است با:

(in * out * k * k) + out (bias)

تمرین 13:

به صورت تئوری ارتباط Early Stopping و نرم دو l2-Norm را بررسی کنید. (منبع صفحه 250-253 کتاب Goodfellow)

(مدل را خطی ساده با بهینه ساز ساده GD و با تابع هزینه درجه دوم quadratic فرض کنید.)

به منظور مقایسه Early Stopping با منظم‌سازی کلاسیک L2، یک تنظیم ساده را بررسی می‌کنیم که در آن تنها پارامترها وزن‌های خطی هستند (θ = w). ما می‌توانیم از بسط سری تیلور، تابع هزینه J را با تقریب درجه دوم (quadratic) حول مقدار بهینه تجربی وزن‌های w* مدل کنیم:

$$\hat{J}(\theta)=J(w^{*})+\frac{1}{2}(w-w^{*})^{T}H(w-w^{*})$$

که در آن H ماتریس هسیین J با توجه به w است که در w* ارزیابی می شود. با توجه به این فرض که w* حداقل J(w) است، می دانیم که H نیمه معین مثبت است. تحت یک تقریب محلی سری تیلور، گرادیان به صورت زیر داده می شود:

$$\nabla_{w}\hat{J}(w)=H(w-w^{*})$$

ما قصد داریم مسیری که بردار پارامتر در طول آموزش دنبال می‌شود را مطالعه کنیم. برای سادگی، بردار پارامتر اولیه را روی مبدا تنظیم می‌کنیم، یعنی $w^{(0)}=0$ و رفتار تقریبی نزول گرادیان در J را با تجزیه و تحلیل گرادیان نزول در $\hat{J}$ مطالعه می‌کنیم:

$$w^{(\tau)}=w^{(\tau-1)}-\epsilon\nabla_{w}\hat{J}\left(w^{(\tau-1)}\right)=w^{(\tau-1)}-\epsilon H\left(w^{(\tau-1)}-w^{*}\right)$$

$$w^{(\tau)}-w^{*}=(I-\epsilon H)\left(w^{(\tau-1)}-w^{*}\right)$$

اکنون این عبارت را در فضـــای بردارهای ویژه H بازنویســـی می‌کنیم، با بهره برداری از تجزیه ویژه $H: H = Q\Lambda Q^{T}$، که در آن Λ یک ماتریس diagonal و Q مبنای orthonormal بردارهای ویژه (eigenvectors) است.

$$w^{(\tau)}-w^{*}=(I-\epsilon Q\Lambda Q^{T})\left(w^{(\tau-1)}-w^{*}\right)$$

$$Q^T(w^{(\tau)} - w^*) = (I - \epsilon\Lambda)Q^T(w^{(\tau-1)} - w^*)$$

با فرض $w^{(0)}=0$ و اینکه ϵ برای تضمین $|1-\epsilon\lambda_i| < 1$، به اندازه کافی کوچک انتخاب شود، مسیر پارامتر در طول training پس از به روز رسانی پارامتر τ به شرح زیر است:

$$Q^T w^{(\tau)} = [I - (I - \epsilon\Lambda)^\tau]Q^T w^*$$

حال، عبارت $Q^T\widetilde{w}$ در معادله $\widetilde{w} = Q(\Lambda + \alpha I)^{-1}\Lambda Q^T w^*$ برای منظم سازی L2 را می‌توان به صورت زیر مرتب کرد:

$$Q^T\widetilde{w} = (\Lambda + \alpha I)^{-1}\Lambda Q^T w^* = [I - (\Lambda + \alpha I)^{-1}\alpha]Q^T w^*$$

با مقایسه دو معادله بالا، می بینیم که اگر هایپرپارامترهای ϵ ، α و τ به گونه ای انتخاب شوند که:

$$(I - \epsilon\Lambda)^\tau = (\Lambda + \alpha I)^{-1}\alpha$$

پس منظم‌سازی L2 و early stopping می‌تواند معادل (حداقل در تقریب درجه دوم تابع هدف) باشد. فراتر از این، با گرفتن لگاریتم و استفاده از بسط سری برای $\log(1+x)$، می‌توان نتیجه گرفت که اگر همه λ_i کوچک باشند (یعنی $\epsilon\lambda i \ll 1$ و $\lambda i/\alpha \ll 1$) پس:

$$\tau \approx \frac{1}{\epsilon\alpha}$$
$$\alpha \approx \frac{1}{\tau\epsilon}$$

یعنی، تحت این مفروضات، تعداد تکرارهای training τ نقشی را به طور معکوس با پارامتر تنظیم L2 ایفا می کند، و معکوس $\tau\epsilon$ نقش ضریب فروپاشی وزن را بازی می کند.

تمرین 14:

الف) می‌دانیم که یک لایه پیچش دو بعدی به صورت زیر قابل نوشتن است. یک لایه Separable Convolution است، در قالب مشابه Point-wise convolution و Depth-wise convolution را که شامل دو بخش بنویسید.

$$Z(i,:,:) = \sum_{j=1}^{nk} W(i,j,:,:) * H(j,:,:) + B(i,:,:)$$

W: Weights of 2d-convolution

H: Previous Layer

Z: Next Layer

B: Bias

یک لایه Separable Convolution متشـکل از دو بخش Depth-wise Convolution و Point-wise Convolution اسـت که به صورت متوالی اعمال می‌شوند:

1. Depth-wise Convolution:

در این بخش، برای هر کانال ورودی، یک فیلتر کوچک به ابعاد $(k \times k)$ اعمال می‌شـود. این فیلتر به طور جداگانه روی هر کانال ورودی اعمال می‌شود و خروجی آن یک feature map است. بنابراین، اگر ورودی شامل c کانال باشد، خروجی این بخش شامل c feature map جداگانه است.

$$D(i,:,:) = \sum_{j=1}^{nk} H(j,:,:) * W^{D}(i,j,:,:) + B^{D}(i,:,:)$$

2. Point-wise Convolution:

در این بخش، یک فیلتر با ابعاد (1×1) استفاده می‌شود.

$$P(i,:,:) = \sum_{j=1}^{nk} W^{P}(j,:,:) * D(i,:,:) + B^{P}(i,:,:)$$

به این ترتیب، با ترکیب این دو بخش، لایه Separable Convolution تشـکیل می‌شـود که به صـورت متوالی و به طور جداگانه روی هر کانال ورودی، عملیات Depth-wise Convolution و Point-wise Convolution را انجام می‌دهد:

$$Z(i,:,:) = \sum_{j=1}^{nk} W^{P}(j,:,:) * (H(j,:,:) * W^{D}(i,j,:,:) + B^{D}(i,:,:)) + B^{P}(i,:,:)$$

ب) وزن معادل یک لایه پیچش را W فرض می کنیم که $W \in R^{N\times M\times L\times K}$ می‌باشد. اگر به صورت رنـک پایین این تنسور را حاصل ضرب چهار بردار یک بعدی درنظر بگیریم به طوریکه:

$$W_{i,j,l,k} = a_i b_j c_l d_k$$

آنگاه پیچش را به صـورت تنوری در این لایه با کمترین محاسـبات بنویسـید، مشـتقات مربوط به پس انتشــار آن را محاسبه کنید و مزیت‌های این مدل سازی را شرح دهید.

در یک لایه پیچش، ورودی و وزن‌ها را به صورت تنسورهای چهار بعدی می‌گیریم. بر اساس فرض مسئله اگر وزن معادل لایه پیچش را W در نظر بگیریم که $W \in R^{N\times M\times L\times K}$ اسـت. N تعداد فیلترها، M و L ابعاد فیلترها و K تعداد کانال‌های ورودی (عمق ورودی) را نشــان می‌دهد.

اگر به صورت رنک پایین این تنسور (رنک (t) را حاصل ضرب چهار بردار یک بعدی درنظر بگیریم، می‌توان W را به صورت ضرب خارجی این بردارها به این صورت نوشت:

$$W_{i,j,l,k} = \sum_{j=1}^{t} a_i \odot b_j \odot c_l \odot d_k$$

که:

$$a_i \in R^N, b_j \in R^M, c_l \in R^L, d_k \in R^k$$

در این حالت، پیچش با ورودی z و وزن W به صورت زیر تعریف می‌شود:

$$z_{i,j} = \sum \sum H(i+l, j+k) * W_{i,j,l,k}$$

برای محاسبه مشتق پس‌انتشار، می‌توانیم از قاعده زنجیره استفاده کنیم:

$$\frac{\partial z_{i,j}}{\partial H(i+l, j+k)} = W_{i,j,l,k}$$

$$\frac{\partial z_{i,j}}{\partial W_{i,j,l,k}} = H(i+l, j+k)$$

از مزیت‌های این مدلسازی با استفاده از وزن‌های پیچش و استفاده از ویژگی‌های کمتر این است که استفاده از وزن‌های پیچش به جای وزن‌های کامل، تعداد پارامترها را به طور قابل توجهی کاهش می‌دهد. این امر باعث می‌شود که مدلسازی مقیاس‌پذیرتر و قابل اجرا بر روی داده‌های بزرگتر شود همچنین با اشتراک وزن‌ها مدل می‌تواند با تعداد کمتری پارامتر، اطلاعات بیشتری را استخراج کند.

ج) قطعه کد مربوط به بخش دوم را بنویسید.

کد مرتبط با تمرین 14:

- DL-HW3-Q14-Mahdavi.ipynb

د) مشتقات پس انتشار برای یک لایه پیچشی ساده را بنویسید.

یک لایه پیچشی یک بعدی ساده شامل یک convolution را در نظر می‌گیریم که شامل یک تصویر در ورودی (X)، وزن لایه (W) بایاس b، لایه خروجی $\hat{y}$ خطی و تابع هزینه L2-norm است:

$$\hat{y} = WX + b$$

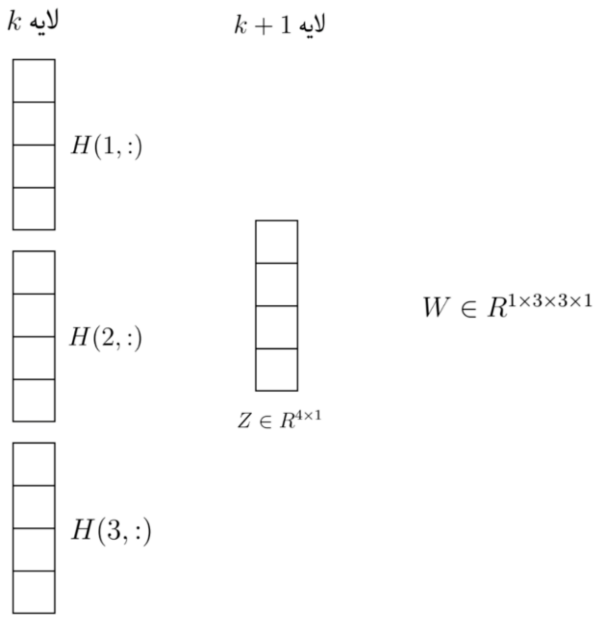
$$L = (y - \hat{y})^2$$

مشتقات پس از انتشار برای **convolution** آن:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W} = -2(y - \hat{y}) \times X$$

تمرین 15:

در شکل زیر دو لایه k و k+1 از یک شبکه پیچشی یک بعدی را مشاهده می کنید.



ابتدا مشتقات مربوط پس انتشار پیچش بین این دو لایه محاسبه کنید و محاسبه مشتق را در این مرحله به صورت یک پیچش مدل کنید. سپس با فرض $Z \in R^{2\times1}$ باشد نوع پیچش را مشخص کنید و مشتقات مربوط به آنرا محاسبه کنید.

در اینجا **Padding** یک داریم و برای محاسبه مشتقات مربوط پس انتشار پیچش کافیست از ترانهاده پیچش استفاده کنیم:

$$k_1 = \begin{bmatrix} \frac{\partial L}{\partial H_{11}} \\ \frac{\partial L}{\partial H_{12}} \\ \frac{\partial L}{\partial H_{13}} \\ \frac{\partial L}{\partial H_{14}} \end{bmatrix} = \begin{bmatrix} w_{12} & w_{13} & 0 & 0 \\ w_{13} & w_{12} & w_{11} & 0 \\ 0 & w_{13} & w_{12} & w_{11} \\ 0 & 0 & w_{13} & w_{12} \end{bmatrix} \times \begin{bmatrix} \frac{\partial L}{\partial Z_1} \\ \frac{\partial L}{\partial Z_2} \\ \frac{\partial L}{\partial Z_3} \\ \frac{\partial L}{\partial Z_4} \end{bmatrix}$$

$$\boldsymbol{k_2} = \begin{bmatrix} \frac{\partial L}{\partial H_{21}} \\ \frac{\partial L}{\partial H_{22}} \\ \frac{\partial L}{\partial H_{23}} \\ \frac{\partial L}{\partial H_{24}} \end{bmatrix} = \begin{bmatrix} w_{22} & w_{23} & 0 & 0 \\ w_{23} & w_{22} & w_{21} & 0 \\ 0 & w_{23} & w_{22} & w_{21} \\ 0 & 0 & w_{23} & w_{22} \end{bmatrix} \times \begin{bmatrix} \frac{\partial L}{\partial Z_1} \\ \frac{\partial L}{\partial Z_2} \\ \frac{\partial L}{\partial Z_3} \\ \frac{\partial L}{\partial Z_4} \end{bmatrix}$$

$$\boldsymbol{k_3} = \begin{bmatrix} \frac{\partial L}{\partial H_{31}} \\ \frac{\partial L}{\partial H_{32}} \\ \frac{\partial L}{\partial H_{33}} \\ \frac{\partial L}{\partial H_{34}} \end{bmatrix} = \begin{bmatrix} w_{32} & w_{33} & 0 & 0 \\ w_{33} & w_{32} & w_{31} & 0 \\ 0 & w_{33} & w_{32} & w_{31} \\ 0 & 0 & w_{33} & w_{32} \end{bmatrix} \times \begin{bmatrix} \frac{\partial L}{\partial Z_1} \\ \frac{\partial L}{\partial Z_2} \\ \frac{\partial L}{\partial Z_3} \\ \frac{\partial L}{\partial Z_4} \end{bmatrix}$$