دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین ۴

علی صداقی

۹۷۵۲۱۳۷۸

سوال ۱)

اگر داده ورودی Sparse می‌باشد استفاده از روش‌هایی که مبتنی بر نرخ یادگیری Adaptive هستند بهتر می‌باشد زیرا در این نوع داده‌ها واریانس و پراکندگی ویژگی‌ها به گونه‌ای نیست که طول قدم (Learning Rate) ثابتی در همه جهات داشته باشیم. Adam هم یک روش بر اساس نرخ یادگیری تطبیقی (Adaptive) است پس در داده‌های Sparse نتیجه خوبی می‌دهد. همچنین Adam با بهره بردن از روش Momentum امکان گیر کردن کمتری در نقاط زینی یا مینیمم محلی را دارد. تمامی این ویژگی‌ها باعث می‌شود در موارد زیر از Adam استفاده کنیم:

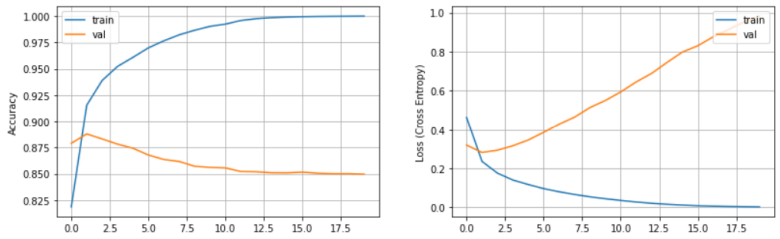
۱ داده ورودی Sparse است.

۲ می‌خواهیم به سرعت زیاد Converge کنیم.

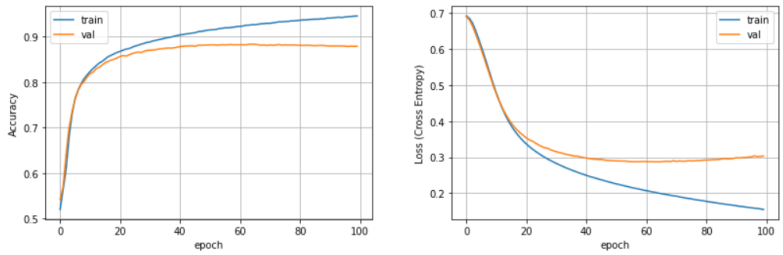
۳ مدل ما بسیار عمیق است.

معایب: همانطور که در اسلایدهای درس بررسی شد، Adam بسیار سریع Converge می‌شود ولی پس از آن شروع به Overfit می‌کند. در نتیجه آن عملکرد در بخش Generalization کاهش می‌یابد. پس Adam برای Generalization خیلی خوب نیست زیرا می‌تواند داده آموزشی را به سرعت حفظ کند.

- بهینه‌ساز Adam



- بهینه‌ساز SGD



با توجه به نمودارهای بالا Adam عملکرد بهتری برروی داده آموزشی (Convergence) دارد ولی در داده ارزیابی (Generalization) عملکرد جالبی ندارد. در صورتی که SGD خلاف آن است. این مورد را در نمودارهای ارائه شده در منبع ۲ نیز می‌توان مشاهده کرد.

✓ به طور کلی می‌توان گفت در روش‌های Adaptive الگوریتم Adam بهترین عملکرد را تحت هر شرایطی دارد. (در مقایسه با نسخه‌های AdaDelta، AdaGrad، RMSprop و ...) به‌دلیل استفاده از Momentum

- ✓ SGD ممکن است در داده آموزشی دچار Underfit شود (حتی با تعداد Epoch بالا) اما Adam به سرعت Fit می‌شود و در ادامه‌ی آن دچار Overfit می‌شود.
- ✓ SGD روی داده ارزیابی Generalization بهتری نسبت به Adam دارد.

با توجه به موارد بالا می‌توان از یک روش ترکیبی استفاده کرد. یعنی ابتدا با Adam شروع به آموزش می‌کنیم سپس تحت شرایط و نقطه‌ای خاص بهینه‌ساز را SGD می‌کنیم. این شرایط می‌تواند زمانی باشد که دیگر مقدار Validation Loss کاهش نمی‌یابد و شروع به افزایش می‌کند.

https://arxiv.org/abs/1712.07628

سوال ۲)

- Forward Pass:

$$h_1 = i_1 w_1 + i_2 w_2$$

$$h_2 = i_1 w_3 + i_2 w_4$$

$$z = h_1 w_5 + h_2 w_6$$

$$\hat{y} = a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$L(a, y) = (y - a)^2$$

- Backward pass for single example:

$$\frac{dL(a,y)}{da} = -2 \times (y - a)$$

$$\frac{da}{dz} = \frac{-(-e^{-z})}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \times \frac{e^{-z} + 1 - 1}{1 + e^{-z}} = \sigma(z) \times \big(1 - \sigma(z)\big) = a(1 - a)$$

$$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz} = -2 \times (y - a)a(1 - a) = -2a(1 - a)(y - a)$$

$$\frac{dL}{dw_6} = \frac{dL}{dz} \times \frac{dz}{dw_6} = -2a(1 - a)(y - a) \times h_2$$

$$\frac{dL}{dw_5} = \frac{dL}{dz} \times \frac{dz}{dw_5} = -2a(1 - a)(y - a) \times h_1$$

$$\frac{dL}{dh_2} = \frac{dL}{dz} \times \frac{dz}{dh_2} = -2a(1 - a)(y - a) \times w_6$$

$$\frac{dL}{dw_4} = \frac{dL}{dh_2} \times \frac{dh_2}{dw_4} = -2a(1 - a)(y - a) \times w_6 \times i_2$$

$$\frac{dL}{dw_3} = \frac{dL}{dh_2} \times \frac{dh_2}{dw_3} = -2a(1 - a)(y - a) \times w_6 \times i_1$$

$$\frac{dL}{dh_1} = \frac{dL}{dz} \times \frac{dz}{dh_1} = -2a(1 - a)(y - a) \times w_5$$

$$\frac{dL}{dw_2} = \frac{dL}{dh_1} \times \frac{dh_1}{dw_2} = -2a(1 - a)(y - a) \times w_5 \times i_2$$

$$\frac{dL}{dw_1} = \frac{dL}{dh_1} \times \frac{dh_1}{dw_1} = -2a(1 - a)(y - a) \times w_5 \times i_1$$

- Update Parameters:

$$w_i = w_i - \alpha \frac{dL}{dw_i}$$

$$i_1 = 3, i_2 = 5$$

$$w_i = \overline{0.i}, 1 \leq i \leq 6$$

$$\alpha = 0.1$$

❖ Epoch 1 :

$$h_1 = 3 \times 0.1 + 5 \times 0.2 = 1.3$$

$$h_2 = 3 \times 0.3 + 5 \times 0.4 = 2.9$$

$$z = 1.3 \times 0.5 + 2.9 \times 0.6 = 2.39$$

$$a = \frac{1}{1 + e^{-2.39}} = 0.916$$

$$L(a, y) = (y - a)^2 = (1 - 0.916)^2 = 0.00705$$

$$\frac{dL}{dz} = -2(0.916)(1 - 0.916)(1 - 0.916) = -0.01292$$

$$\frac{dL}{dw_6} = (-0.01292) \times (2.9) = -0.037468$$

$$\frac{dL}{dw_5} = (-0.01292) \times (1.3) = -0.016796$$

$$\frac{dL}{dw_4} = (-0.01292) \times (0.6) \times (5) = -0.03876$$

$$\frac{dL}{dw_3} = (-0.01292) \times (0.6) \times (3) = -0.023256$$

$$\frac{dL}{dw_2} = (-0.01292) \times (0.5) \times (5) = -0.0323$$

$$\frac{dL}{dw_1} = (-0.01292) \times (0.5) \times (3) = -0.01938$$

$$w_6 = 0.6 - 0.1 \times (-0.037468) = 0.6037468$$
$$w_5 = 0.5 - 0.1 \times (-0.016796) = 0.5016796$$
$$w_4 = 0.4 - 0.1 \times (-0.03876) = 0.403876$$
$$w_3 = 0.3 - 0.1 \times (-0.023256) = 0.3023256$$
$$w_2 = 0.2 - 0.1 \times (-0.0323) = 0.20323$$
$$w_1 = 0.1 - 0.1 \times (-0.01938) = 0.101938$$

❖ Epoch 2:

$$h_1 = 3 \times 0.101938 + 5 \times 0.20323 = 1.321964$$

$$h_2 = 3 \times 0.3023256 + 5 \times 0.403876 = 2.9263568$$

$$z = 1.321964 \times 0.5016796 + 2.9263568 \times 0.6037468 = 2.429980924$$

$$a = \frac{1}{1 + e^{-2.429980924}} = 0.919085$$

$$L(a, y) = (y - a)^2 = (1 - 0.919085)^2 = 0.006547237225$$

$$\frac{dL}{dz} = -2(0.919085)(1 - 0.919085)(1 - 0.919085) = -0.01203493505$$

$$\frac{dL}{dw_6} = (-0.01203493505) \times (2.9263568) = -0.03521851402$$

$$\frac{dL}{dw_5} = (-0.01203493505) \times (1.321964) = -0.01590975088$$

$$\frac{dL}{dw_4} = (-0.01203493505) \times (0.6037468) \times (5) = -0.03633026762$$

$$\frac{dL}{dw_3} = (-0.01203493505) \times (0.6037468) \times (3) = -0.02179816057$$

$$\frac{dL}{dw_2} = (-0.01203493505) \times (0.5016796) \times (5) = -0.03018840701$$

$$\frac{dL}{dw_1} = (-0.01203493505) \times (0.5016796) \times (3) = -0.01811304421$$

$$w_6 = 0.6037468 - 0.1 \times (-0.03521851402) = 0.6072686514$$
$$w_5 = 0.5016796 - 0.1 \times (-0.01590975088) = 0.5032705751$$
$$w_4 = 0.403876 - 0.1 \times (-0.03633026762) = 0.4075090268$$
$$w_3 = 0.3023256 - 0.1 \times (-0.02179816057) = 0.3045054161$$
$$w_2 = 0.20323 - 0.1 \times (-0.03018840701) = 0.2062488407$$
$$w_1 = 0.101938 - 0.1 \times (-0.01811304421) = 0.1037493044$$

❖ Evaluation:

$$h_1 = 3 \times 0.1037493044 + 5 \times 0.2062488407 = 1.342492117$$

$$h_2 = 3 \times 0.3045054161 + 5 \times 0.4075090268 = 2.951061382$$

$$z = 1.342492117 \times 0.5032705751 + 2.951061382 \times 0.6072686514 = 2.467723845$$

$$a = \frac{1}{1 + e^{-2.467723845}} = 0.921847$$

$$L(a, y) = (y - a)^2 = (1 - 0.921847)^2 = 0.006107891409$$

تحلیل: همانطور که در کلاس درس نیز بررسی شد تابع ضرر MSE برای مسئله دسته‌بندی باینری که تابع فعال‌سازی لایه آخر Sigmoid است به هیچ عنوان مناسب نیست. در واقع نمی‌توان با این تابع ضرر Likelihood را Max کرد. در سوال بالا این مشکل خود را در محاسبه مقدار زیر نشان داد:

$$\frac{dL}{dz} = -2a(1-a)(y-a)$$

این مقدار همواره عددی نزدیک به صفر می‌باشد و وقتی آن را در قاعده مشتق زنجیره‌ای ضرب می‌کنیم اثر گرادیان را محو می‌کند (Gradient Vanishing). این اتفاق هرچه به لایه‌های اولیه می‌رویم شدت بیشتری می‌یابد. با این حال حتی MSE نیز در کاهش دادن Loss موثر بود و مقدار Loss به صورت زیر تغییر کرد (سرعت بسیار پایین):

$$0.00705 \rightarrow 0.006547237225 \rightarrow 0.006107891409$$

$Y_{Predict} = 0.921847$

در واقع می‌توان گفت مقدار اولیه وزن‌ها در نقطه‌ای تقریبا همگرا بود و الوریتم در این نقطه *Convex* حرکت کمی داشت.

وزن‌ها هم به صورت زیر دچار تغییر شدند که تغییر کمی است:

$$w_6 = 0.6 \rightarrow 0.6072686514$$
$$w_5 = 0.5 \rightarrow 0.5032705751$$
$$w_4 = 0.4 \rightarrow 0.4075090268$$
$$w_3 = 0.3 \rightarrow 0.3045054161$$
$$w_2 = 0.2 \rightarrow 0.2062488407$$
$$w_1 = 0.1 \rightarrow 0.1037493044$$

سوال ۳) در حل این سوال منابع زیر مطالعه شدند و از آن‌ها کمک گرفته شد:

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

https://pytorch.org/docs/stable/generated/torch.optim.Adam.html#torch.optim.Adam

https://pytorch.org/vision/stable/transforms.html

https://pytorch.org/docs/stable/nn.html

https://madebyollin.github.io/convnet-calculator/

https://androidkt.com/convolutional-neural-network-using-sequential-model-in-pytorch/

همه‌ی لایه‌های Conv2d که قرار گرفته Channel ورودی را دو برابر می‌کند و در خروجی می‌فرستد. هم‌چنین با دادن آرگومان‌های مشترک زیر مقدار Height و Width همواره ثابت می‌ماند.

```
KERNEL_SIZE = 3
STRIDE = 1
PADDING = 1
```

$$In: (C_{in}, H_{in}, W_{in})$$
$$Out: (2 \times C_{in}, H_{in}, W_{in})$$

لایه‌های MaxPool2d نیز دارای آرگومان‌های زیر هستند و مقدار Height و Width را نصف می‌کنند. مقدار Channels حفظ می‌شود.

```
KERNEL_SIZE = 2
STRIDE = 2
```

$$In: (C_{in}, H_{in}, W_{in})$$
$$Out: \left(C_{in}, \frac{H_{in}}{2}, \frac{W_{in}}{2}\right)$$

✓ ورودی و خروجی هر لایه در نوتبوک به صورت کامنت شده موجود می‌باشد.

در نهایت هم ۳ لایه Dense با تعداد نورون ۵۱۲، ۲۵۶، ۱۰ قرار دادیم.

احتمال برای ۳ لایه Dropout به صورت 0.05 می‌باشد.

Augmentation: برای این کار از توابع Transforms زیر استفاده کردیم:

```
transforms.RandomHorizontalFlip(p=0.5),
transforms.RandomVerticalFlip(p=0.5),
transforms.RandomRotation(degrees=(0, 180)),
transforms.RandomGrayscale(p=0.1),
```

به عبارتی هر عکس را با احتمال ۵۰ درصد به صورت افقی برعکس می‌کنیم. همین کار را در جهت عمودی نیز داریم. عکس‌هارا به صورت شانسی بین ۰ تا ۱۸۰ درجه دوران می‌دهیم. با احتمال ۱۰ درصد بعضی تصاویر را خاکستری می‌کنیم.

کارهای بالا باعث می‌شود شبکه داده ورودی بیشتری داشته باشد و همچنین با اضافه شدن نویز امکان Overfit را کاهش می‌دهد. در کل Augmentation باعث کاهش bias و variance در شبکه می‌شود.

✓ در داده تست این Augmentation را دیگر نداریم زیرا نمی‌خواهیم روی آن آموزش بدهیم.

سایر هایپرپارامترهای شبکه:

```
BATCH_SIZE = 4
LEARNING_RATE = 0.001
EPOCHS = 2
```

با توجه به مستند تمرین از تابع ضرر زیر استفاده کردیم. که مناسب مسائل دسته‌بندی است.

```
criterion = nn.CrossEntropyLoss()
```

همچنین از بهینه‌ساز Adam به صورت زیر استفاده کردیم.

```
optimizer = optim.Adam(
  net.parameters(),
  lr=LEARNING_RATE,
  betas=(0.9, 0.999),
  eps=1e-08,
  weight_decay=0,
  amsgrad=False
)
```

مراحل آموزش را در دو حلقه طی کردیم. حلقه اول برای Epoch و حلقه دوم برای Batch.

✓ از GPU در هر دو فاز Train و Test استفاده کردیم.

✓ با توجه به اینکه گفت شد آرگومان‌های لایه‌های Conv2d دلخواه باشند، همچنین تعداد 2 اپاک نتوانستیم و نمیشد دقت خوبی بگیریم.

پس از آموزش مدل را در فایلی ذخیره کردیم. سپس با Load کردن مدل آن را بر روی داده آزمون ارزیابی کردیم.

نتایج به صورت زیر بود:

```
[1,  2000] loss: 2.207
[1,  4000] loss: 2.140
[1,  6000] loss: 2.094
[1,  8000] loss: 2.061
[1, 10000] loss: 2.024
[1, 12000] loss: 1.977
[2,  2000] loss: 1.964
[2,  4000] loss: 1.947
[2,  6000] loss: 1.929
[2,  8000] loss: 1.912
[2, 10000] loss: 1.907
[2, 12000] loss: 1.873
```

```
Accuracy: 33.360000%

Accuracy for class plane is: 23.2
Accuracy for class car   is: 36.6
Accuracy for class bird  is: 19.9
Accuracy for class cat   is: 15.0
Accuracy for class deer  is: 33.8
Accuracy for class dog   is: 12.9
Accuracy for class frog  is: 51.8
Accuracy for class horse is: 38.2
Accuracy for class ship  is: 51.0
Accuracy for class truck is: 52.4
```