Security and Privacy in Machine Learning

Homework 2

Spring 2023

Javad Hezareh

98101074

Contents

1 Optimization

(a) Learning rate will affect the convergence and also convergence rate. If LR be large, optimization might not converge to a local minima and diverge. If this value be very small, the convergence rate will be very slow. So it's crucial to choose learning rate not that slow to not have a very slow convergence and not that large to not diverge and have a good convergence rate.

Using momentum helps to gain a speed in directions that we have almost stable gradient. This will help the optimization process not affected by oscillations and reach the local minima faster.

(b) During optimization we might get to landscapes that are flat or fluctuate a lot. In these surfaces using the initial learning rate might not work very well. Our learning rate might be too small for passing flat surfaces or too large to cautiously probe ups and downs of landscape in fluctuating surfaces. To solve this problem one can use learning rate schedulers in order to change the learning rate value during training process.

(c) By decreasing learning rate during training one can ensure that the closer we are to the optimal point, the more precise we update the parameters. This helps convergence and not to diverge from the optimal point or oscillate around that.

There are several different schemes to decrease the learning rate during training. One can divide learning rate each iteration by a constant factor or a factor that increase linearly with iteration number. There is also a method that decrease learning rate by a factor every few epochs. In another method the learning rate can be decreased exponentially. One needs to decide based on the problem specification or the data characteristics to choose the best scheme among all.

(d) Increasing learning rate during training benefits the optimization process especially in landscapes that have multiple local minima. By increasing learning rate, we can pass some local minima and finally find a better solution for our problem that leads to better generalization.

(e) One of the key differences between SGD and Adam is in the way they use learning rate. In vanilla SGD learning rate is constant and equal for all parameters. However, in Adam optimization learning rate is chosen adaptively that can be adjusted for each parameter separately. This helps optimization converge to a better optimal point. The other difference is that Adam optimization benefits from momentum while vanilla SGD don't.

2 Convolution2D

(a) Properties of network:

Layer	Property				Input dim		Output dim		Parameters		Cost
	C	Kernel	Pad	Stride	C	W × H	C	W × H	Weighs	Biases	(MFLOPS)
C_1	10	9×9	0	1×1	3	256×256	10	248×248	$3 \times 9 \times 9 \times 10$	10	150
`ReLU`	-	-	-	-	-	-	-	-	-	-	-
C_2	20	7×7	0	2×2	10	248×248	20	121×121	$10 \times 7 \times 7 \times 20$	20	143
C_3	40	5×5	0	1×1	20	121×121	40	117×117	$20 \times 5 \times 5 \times 40$	40	273
`ReLU`	-	-	-	-	-	-	-	-	-	-	-
`Max Pooling`	-	2×2	0	2×2	40	117×117	40	58×58	0	0	0
C_4	50	3×3	0	2×2	40	58×58	50	28×28	$40 \times 3 \times 3 \times 50$	50	14
`Flatten`	-	-	-	-	50	28×28	-	39200×1	0	0	0
FC	-	-	-	-	-	39200×1	-	1×1	39200×1	1	0.04

(b) Using more non-linearity helps our network be able to model more complex functions. Therefore adding activation function after C_2 and C_4 will improve the network power in modeling more complex functions.

3 Convolution1D

(a) After convolution the output will be:

$$\boldsymbol{y} = \begin{pmatrix} x_1 w_1 + x_2 w_2 \\ x_2 w_1 + x_3 w_2 \\ x_3 w_1 + x_4 w_2 \\ x_4 w_1 + x_5 w_2 \end{pmatrix}$$

We can rewrite this equation using matrix multiplication:

$$\boldsymbol{y} = \begin{pmatrix} w_1 & w_2 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & 0 \\ 0 & 0 & 0 & w_1 & w_2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

(b) As we can see in the above matrix, most of the elements are zero and hence the convolution matrix is sparse. It will reduce the calculation and also as we can see we have parameter sharing in this matrix.

(c) For a 4×1 convolution filter the output will be 2×1, therefore:

$$\boldsymbol{y} = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 & 0 \\ 0 & w_1 & w_2 & w_3 & w_4 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

In this case the convolution matrix is not that sparse and hence one can not benefits from sparse matrix advantages.

4 BatchNormalization

(a) Using batch normalization will help model to improve its performance and stability. By normalizing inputs of layers and decreasing the internal covariant shift, model will be less sensitive to the specific input values which will lead to less overfitting and better generalization. Also using batch normalization will enable us to use larger learning rate and train deeper models that can have better performance.

(b) For β using chain rule we have:

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_i} \frac{\partial y_i}{\partial \beta}$$

$$= \boxed{\sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_i}}$$

For γ also using chain rule we have:

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_i} \frac{\partial y_i}{\partial \gamma}$$

$$= \boxed{\sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_i} \hat{x}_i}$$

For x_i we have:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial \mathcal{L}}{\partial y_j} \frac{\partial y_j}{\partial \hat{x}_j} \frac{\partial \hat{x}_j}{\partial x_i}$$

We know $\frac{\partial y_j}{\partial \hat{x}_j} = \gamma$, for the last term we use chain rule and total derivative:

$$\frac{\partial \hat{x}_j}{\partial x_i} = \frac{d\hat{x}_j}{dx_i} + \frac{d\hat{x}_j}{d\mu} \frac{d\mu}{dx_i} + \frac{d\hat{x}_j}{d\sigma^2} \frac{d\sigma^2}{dx_i}$$

$$= \mathbb{1}_{(i=j)} \frac{1}{\sqrt{\sigma^2 + \epsilon}} + \frac{-1}{n\sqrt{\sigma^2 + \epsilon}} + \frac{-(x_j - \mu)(x_i - \mu)}{n(\sigma^2 + \epsilon)^{3/2}}$$

Using the relation in above summation we have:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \gamma \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_j} \frac{1}{n\sqrt{\sigma^2 + \epsilon}} \left[n\mathbb{1}_{(i=j)} - 1 - \frac{(x_j - \mu)}{\sqrt{\sigma^2 + \epsilon}} \frac{(x_i - \mu)}{\sqrt{\sigma^2 + \epsilon}} \right]$$

$$= \frac{\gamma}{n\sqrt{\sigma^2 + \epsilon}} \sum_{i=1}^{n} \frac{\partial \mathcal{L}}{\partial y_j} \left[n\mathbb{1}_{(i=j)} - 1 - \hat{x}_j \hat{x}_i \right]$$

$$= \frac{\gamma}{n\sqrt{\sigma^2 + \epsilon}} \left[n \frac{\partial \mathcal{L}}{\partial y_i} - \sum_{j=1}^{n} \frac{\partial \mathcal{L}}{\partial y_j} - \hat{x}_i \sum_{j=1}^{n} \frac{\partial \mathcal{L}}{\partial y_j} \hat{x}_j \right]$$

$$= \boxed{\frac{\gamma}{n\sqrt{\sigma^2 + \epsilon}} \left[n \frac{\partial \mathcal{L}}{\partial y_i} - \frac{\partial \mathcal{L}}{\partial \beta} - \hat{x}_i \frac{\partial \mathcal{L}}{\partial \gamma} \right]}$$

5 Pooling

(a) After convolution filter we know that $o_i = w_1 x_i + w_2 x_{i+1}$. Therefore using chain rule we have:

$$\text{In normal network}: \begin{cases} \dfrac{\partial \mathcal{L}}{\partial w_1} = \displaystyle\sum_{i=1}^{4} \frac{\partial \mathcal{L}}{\partial o_i} x_i \\ \dfrac{\partial \mathcal{L}}{\partial w_2} = \displaystyle\sum_{i=1}^{4} \frac{\partial \mathcal{L}}{\partial o_i} x_{i+1} \end{cases}$$

(b) Using average pooling we have:

$$\begin{cases} y_1 = \frac{o_1 + o_2}{2} \\ y_2 = \frac{o_3 + o_4}{2} \end{cases}$$

Therefore using chain rule for w_1 we have:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial \mathcal{L}}{\partial y_1}\frac{\partial y_1}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial y_2}\frac{\partial y_2}{\partial w_1} \\ &= \frac{\partial \mathcal{L}}{\partial y_1}\frac{x_1 + x_2}{2} + \frac{\partial \mathcal{L}}{\partial y_2}\frac{x_3 + x_4}{2} \end{aligned}$$

With similar calculation we can find $\partial \mathcal{L}/\partial w_2$:

$$\text{Using average pooling}: \begin{cases} \dfrac{\partial \mathcal{L}}{\partial w_1} = \dfrac{\partial \mathcal{L}}{\partial y_1}\dfrac{x_1 + x_2}{2} + \dfrac{\partial \mathcal{L}}{\partial y_2}\dfrac{x_3 + x_4}{2} \\ \dfrac{\partial \mathcal{L}}{\partial w_2} = \dfrac{\partial \mathcal{L}}{\partial y_1}\dfrac{x_2 + x_3}{2} + \dfrac{\partial \mathcal{L}}{\partial y_2}\dfrac{x_4 + x_5}{2} \end{cases}$$

(c) Using max pooling and assuming $o_1 > o_2$ and $o_3 > o_4$ we will have:

$$\begin{cases} y_1 = \max(o_1, o_2) = o_2 \\ y_2 = \max(o_3, o_4) = o_3 \end{cases}$$

Now using chain rule:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial \mathcal{L}}{\partial y_1}\frac{\partial y_1}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial y_2}\frac{\partial y_2}{\partial w_1} \\ &= \frac{\partial \mathcal{L}}{\partial y_1} x_2 + \frac{\partial \mathcal{L}}{\partial y_2} x_3 \end{aligned}$$

Similarly for $\partial \mathcal{L}/\partial w_2$:

$$\text{Using max pooling}: \begin{cases} \dfrac{\partial \mathcal{L}}{\partial w_1} = \dfrac{\partial \mathcal{L}}{\partial y_1} x_2 + \dfrac{\partial \mathcal{L}}{\partial y_2} x_3 \\ \dfrac{\partial \mathcal{L}}{\partial w_2} = \dfrac{\partial \mathcal{L}}{\partial y_1} x_3 + \dfrac{\partial \mathcal{L}}{\partial y_2} x_4 \end{cases}$$