DS-GA-1008: Deep Learning, Assignment 3

Jiayi Lu (jl6583)
Center for Data Science, New York University
726 Broadway, 7th floor, New York, NY 10012
jiayi.lu@nyu.edu

Abstract

This paper provides solutions to problems in assignment #3 of the 2016 spring deep learning course (instructed by Prof.Zaid Harchaoui) at Courant Institute of Mathematical Science

1 General Questions

(a)

If there are no non-linear activation between two linear modules, let's say $x_2 = W_1 x_1 + b_1, x_3 = W_2 x_2 + b_2$, then using substitution we can simplify the two-layer structure by replacing the two layers with a single layer module represented as $x_3 = W_2 W_1 x_1 + W_2 b_1 + b_2$

(b)

The dictionary of sparse coding can be viewed as the weight matrix of the output layer in autoencoders, however, in autoencoder, the weight matrix is learned from input data, while the sparse coding dictionary is either hard-coded or learned from encoded inputs.

2 Softmax regression gradient calculation

(a)

Let $a_i = (Wx + b)_i$, according to multivariate chain rule,

$$\frac{\partial \ell}{\partial W_{ij}} = \sum_k \frac{\partial \ell}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial a_i} \frac{\partial a_i}{\partial W_{ij}} \tag{1}$$

We also have

$$\frac{\partial \ell}{\partial \hat{y}_k} = \frac{\partial - \sum_i y_i \log \hat{y}_i}{\partial \hat{y}_k} = -\frac{y_k}{\hat{y}_k} \tag{2}$$

$$\tag{3}$$

Using the solution in Assignment 1,

$$\frac{\partial \hat{y}_k}{\partial a_i} = \begin{cases} \hat{y}_i \left(1 - \hat{y}_i\right), k = i \\ -\hat{y}_k \hat{y}_i, k \neq i \end{cases} \tag{4}$$

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

And $\frac{\partial a_i}{\partial W_{ij}} = x_j$, we could get:

$$\frac{\partial \ell}{\partial W_{ij}} = \sum_k \frac{\partial \ell}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial a_i} \frac{\partial a_i}{\partial W_{ij}} \tag{5}$$

$$= -y_i \left(1 - \hat{y}_i\right) x_j - \sum_{k \neq i} -y_k \hat{y}_i x_j \tag{6}$$

$$= x_j \left(\hat{y}_i - y_i\right) \tag{7}$$

(b)

In this case the loss function is $\inf$, and the gradient is $0, x_j$ or $-x_j$. And since $\hat{y}_{c_2} = 1, \exp(a_j) = 0$, a_j has to be $-\inf$.

3 Chain Rule

(a)

Let $m(x, y) = x^2 + \sigma(y), n(x, y) = 3x + y - \sigma(x)$, according to multivariate chain rule we have:

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial m}\frac{\partial m}{\partial x} + \frac{\partial f}{\partial n}\frac{\partial n}{\partial x} \\ \frac{\partial f}{\partial y} &= \frac{\partial f}{\partial m}\frac{\partial m}{\partial y} + \frac{\partial f}{\partial n}\frac{\partial n}{\partial y} \end{aligned} \tag{8}$$

Further, since we have:

$$\begin{aligned} &\frac{\partial m}{\partial x} = \frac{\partial x^2}{\partial x} = 2x, \frac{\partial m}{\partial y} = \frac{\partial \sigma(y)}{\partial y} \\ &\frac{\partial n}{\partial x} = 3 - \frac{\partial \sigma(x)}{\partial x}, \frac{\partial n}{\partial y} = 1 \end{aligned} \tag{9}$$

And $\frac{\partial f}{\partial m} = \frac{1}{n}, \frac{\partial f}{\partial n} = -\frac{m}{n^2}$ are gradOutputs of a division module with input m, n, we could compute the gradient using only input and gradOutputs of given modules.

(b)

We compute partial derivatives described in (a):

$$\begin{aligned} \frac{\partial f}{\partial m} &= \frac{1}{3x + y - \sigma(x)} = \frac{1}{3 - \frac{1}{1+e^{-1}}} \\ \frac{\partial f}{\partial n} &= -\frac{x^2 + \sigma(y)}{(3x + y - \sigma(x))^2} = -\frac{3/2}{\left(3 - \frac{1}{1+e^{-1}}\right)^2} \end{aligned} \tag{10}$$

Combining all previous evaluations we could get gradients at $x = 1, y = 0$ as follows:

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{3 + 5e - e^2}{2(3+2e)^2} \approx 0.065 \\ \frac{\partial f}{\partial y} &= -\frac{(1+e)(3+4e)}{4(3+2e)^2} \approx -0.181 \end{aligned} \tag{11}$$

4 Variants of Pooling

(a)

1. Max Pooling: its corresponding Torch7 implementation is SpatialMaxPooling
2. Average Pooling: its corresponding Torch module is SpatialAveragePooling
3. LP(p-norm) Pooling: in Torch it's implementation is SpatialLPPooling

(b)

Let P be the pooling region (a sub-matrix of input), r is the representative value of that region, we have:

1. Max Pooling: $r = \max_{1 \le i \le kW, 1 \le j \le kH}(P_{ij})$
2. Average Pooling: $r = \frac{1}{n} \sum_{i=1}^{kW} \sum_{j=1}^{kH} P_{ij}$
3. LP Pooling: $r = \left(\sum_{i=1}^{kW} \sum_{j=1}^{kH} \|P_{ij}\|^p \right)^{1/p}$

(c)

One of the major reason for using pooling is to capture transition invariant features, take max pooling as an example, suppose we have the two following input matrices with "L" shaped feature at different locations:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

After pooling on these inputs in the group of 2×2 we have:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

As we can see, the "L" shaped feature is captured in a smaller region, that is to say, pooling makes a kind of transitional invariant down sampling.

5 Convolution

(a)

Assume that there are no paddings and stride = 1, we would end up with 9 (3×3) values after forward propagation

(b)

The values generated are listed below,

109	92	72
108	85	74
110	74	79

Table 1: Output of convolution

(c)

Let E be the output of the layer above the convolutional layer, Y be the output matrix of the convolutional layer, X be the input matrix, K be the kernel matrix, according to multivariate chain

rule, we could write $\frac{\partial E}{\partial X}$ as follows:

$$\frac{\partial E}{\partial x_{ij}} = \frac{\partial E}{\partial y_{11}}\frac{\partial y_{11}}{\partial x_{ij}} + \cdots + \frac{\partial E}{\partial y_{33}}\frac{\partial y_{33}}{\partial x_{ij}} \tag{12}$$

$$= \sum_{m,n=1}^{3} \frac{\partial E}{\partial y_{mn}}\frac{\partial y_{mn}}{\partial x_{ij}} \tag{13}$$

Note that $\frac{\partial E}{\partial y_{mn}} = 1, \forall m, n = 1, 2, 3$, and,

$$y_{mn} = k_{11}x_{mn} + k_{12}x_{m,n+1} + k_{13}x_{m,n+2} + \tag{14}$$

$$\cdots \tag{15}$$

$$+ k_{31}x_{m+2,n} + k_{32}x_{m+2,n+1} + k_{33}x_{m+2,n+2} \tag{16}$$

Hence,

$$\frac{\partial E}{\partial x_{11}} = k_{11} \tag{17}$$

$$\frac{\partial E}{\partial x_{12}} = k_{11} + k_{12} \tag{18}$$

$$\frac{\partial E}{\partial x_{13}} = k_{11} + k_{12} + k_{13} \tag{19}$$

$$\cdots \tag{20}$$

The gradient backpropagated through this convolutional layer is therefore a 5×5 matrix looks like this:

$$\frac{\partial E}{\partial X} = \begin{pmatrix} k_{11} & k_{11}+k_{12} & k_{11}+k_{12}+k_{13} & k_{12}+k_{13} & k_{13} \\ k_{11}+k_{21} & \cdots & \cdots & \cdots & k_{13}+k_{23} \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ k_{31} & \cdots & \cdots & \cdots & k_{33} \end{pmatrix} \tag{21}$$

Actually, $\frac{\partial E}{\partial X}$ can be represented as the convolution of $\frac{\partial E}{\partial Y}$ added with zero padding of width 2 and K rotated by 180 degrees:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} -\text{convolve}- \begin{pmatrix} k_{33} & k_{32} & k_{31} \\ k_{23} & k_{22} & k_{21} \\ k_{13} & k_{12} & k_{11} \end{pmatrix} \tag{22}$$

The result of this convolution is:

$$\frac{\partial E}{\partial X} = \begin{pmatrix} 4 & 7 & 10 & 6 & 3 \\ 9 & 17 & 25 & 16 & 8 \\ 11 & 23 & 34 & 23 & 11 \\ 7 & 16 & 24 & 17 & 8 \\ 2 & 6 & 9 & 7 & 3 \end{pmatrix} \tag{23}$$

6 Optimization

(a)

According to the definition of autoencoder on Wikipedia, the reconstruction loss can be formulated as:

$$\ell(x, \hat{x}) = \|x - \hat{x}\|^2 \tag{24}$$

where $\hat{x} = \sigma_2(W_2\sigma_1(W_1x + b_1) + b_2)$

(b)

We can use chain rules to derive the derivatives, for simplicity we skip b_1, b_2 here by letting both of them equal to 0, let $h_1(x) = \sigma_1(W_1 x), h_2(x) = \sigma_2(W_2 x)$:

$$\frac{\partial \ell}{\partial {W_1}^{(ij)}} = \sum_k \frac{\partial \ell}{\partial \hat{x}_k} \frac{\partial \hat{x}_k}{\partial {h_1}^{(i)}} \frac{\partial {h_1}^{(i)}}{\partial {W_1}^{(ij)}} \tag{25}$$

$$\frac{\partial \ell}{\partial {W_2}^{(ij)}} = \frac{\partial \ell}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial {W_2}^{(ij)}} \tag{26}$$

(c)

$$W_1(t+1) \leftarrow W_1(t) - \alpha \frac{\partial \ell}{\partial W_1(t)} \tag{27}$$

$$W_2(t+1) \leftarrow W_2(t) - \alpha \frac{\partial \ell}{\partial W_2(t)} \tag{28}$$

(d)

Let $v_X = \eta v_X + \alpha \frac{\partial \ell}{\partial X(t)}, \eta \in (0, 1]$, we have

$$W_1(t+1) \leftarrow W_1(t) - v_{W_1} \tag{29}$$

$$W_2(t+1) \leftarrow W_2(t) - v_{W_2} \tag{30}$$

7 Top-k error

Top-k error is the fraction of test images for which the correct label is not among the five labels considered most probable by the model. The motivation for ImageNet to use top-5 and top-1 error is to better evaluate model performance with the large number of class labels to predict and the ambiguity between classes. In addition to the percentage of predictions that generated the exact target class label (which might be fairly small), we also want to know the performance of the model in generating predictions "similar" to the target class.

8 t-SNE

(a)

The crowding problem refers to the fact that when mapping clusters embedded in high-dimensional space to 2-d space, the volume of the spherical space around a certain point increases slower in 2-d space when distance increases, therefore, if we want to allocate enough space to visualize small distances, the moderate distant points has to be placed very far away, on the other way, if moderate distance are kept closer, small distance points will be crowded together.

To alleviate this problem, t-SNE used two techniques:

1. t-SNE minimizes a KL-divergence from high-dimensional edge strength to low-dimensional distance, by using KL-divergence, t-SNE imposes a bigger penalty for similar datapoints that are placed far apart than dissimilar points placed closely together.

2. t-SNE employs a heavy tail distribution (student-t) in low dimensional to avoid the situation when moderately dissimilar points are repulsed together.

(b)

Let $d_{ij} = -\|y_i - y_j\|^2$, we first compute:

$$\frac{\partial C}{\partial y_t} = -\sum_i \sum_j p_{ij} \frac{\partial \log q_{ij}}{\partial y_t} \tag{31}$$

$$= -\sum_{i,j} p_{ij} \frac{\partial d_{ij} - \partial \log \sum_{k \neq l} e^{d_{kl}}}{\partial y_t} \tag{32}$$

$$= -\sum_{i,j} p_{ij} \frac{\partial d_{ij}}{\partial y_t} - \frac{1}{\sum_{k \neq l} e^{d_{kl}}} \frac{\partial \sum_{k \neq l} e^{d_{kl}}}{\partial y_t} \tag{33}$$

And since

$$\frac{\partial d_{ij}}{\partial y_t} = \begin{cases} -2\left(y_i - y_j\right), \text{when} t = i \\ -2\left(y_j - y_i\right), \text{when} t = j \\ 0, \text{otherwise} \end{cases} \tag{34}$$

$$d_{ij} = d_{ji} \tag{35}$$

$$\frac{\partial \sum_{k \neq l} e^{d_{kl}}}{\partial y_t} = \frac{\partial \left(2 \sum_m e^{d_{tm}}\right)}{\partial y_t} = -4 \sum_m \left(y_t - y_m\right) e^{d_{tm}} \tag{36}$$

Combine the derivatives above, we get

$$\frac{\partial C}{\partial y_t} = -\sum_{i,j} p_{ij} \frac{\partial d_{ij}}{\partial y_t} - \frac{1}{\sum_{k \neq l} e^{d_{kl}}} \frac{\partial \sum_{k \neq l} e^{d_{kl}}}{\partial y_t} \tag{37}$$

$$= -2 \sum_m p_{tm} \left(-2\left(y_t - y_m\right)\right) - 4 \sum_{i,j} p_{ij} \sum_m \left(y_t - y_m\right) e^{d_{tm}} \tag{38}$$

$$= 4 \sum_m \left(p_{tm} - q_{tm}\right)\left(y_t - y_m\right) \tag{39}$$

9 Proximal gradient descent

(a)

$$\text{prox}_{h,t}(x) = \arg\min_z \frac{1}{2} \sum_i \left(z_i - x_i\right)^2 + t\left|z_i\right| \tag{40}$$

$$\frac{\partial \frac{1}{2} \sum_i \left(z_i - x_i\right)^2 + t\left|z_i\right|}{\partial z_i} = \begin{cases} z_i - x_i + t, z_i \geq 0 \\ z_i - x_i - t, z_i < 0 \end{cases} \tag{41}$$

Note that if $x_i - t < 0, z_i \geq 0$, or $x_i + t \geq 0, z_i < 0$, the derivative does not equals 0, hence the minimizer of this function is 0, which is the property of soft thresholding function.

(b)

ISTA minimizes $\frac{1}{2}\|z - Dx\|_2^2 + t\|z\|$, note that the first term is convex differentiable, and second term is convex, simple, non-differentiable, so it is a proximal gradient descent method.

(c)

let $f = \frac{1}{2}\|z - x\|_2^2 + th(z)$, we have

$$\partial f = z - x + t\partial h(z) \tag{42}$$

Since u minimizes f, we have $0 \in u - x + t\partial h(u)$, hence $\frac{x-u}{t} \in \partial h(u)$

References

[1] Lapin, M., Hein, M., & Schiele, B. (2015). *Loss Functions for Top-k Error: Analysis and Insights.* arXiv preprint arXiv:1512.00486.