Assignment 4: BP

LIAO,WEN

December 11, 2018

1 Task Description

Please derive a backpropagation process

1. for the multi-layer neural network with one hidden layer, where data are in a d-dimensional feature space with C classes. Loss functions can use L2 distance or cross entropy.
2. for the LeNet-5 CNN.
3. for Vanilla recurrent neural networks with T time steps.

2 Multilayer Perceptron

2.1 Notations

Consider a multilayer perceptron with one hidden layer. We'll use the following notations to describe the archietecture of a feedforward neural network.

1. N: dimension of the input layer
2. M: dimension of the hidden layer
3. C: dimension of the output layer
4. x_i: the i-th dimension of the input layer
5. z_k: the k-th dimension of the hidden layer
6. y_j: the j-th dimension of the output layer

2.2 Loss Function

Let $\{(x_n, y_n)\}_{n=1}^{N'}$ be the training dataset. For a sample (x_n, y_n), the loss function is defined as

$$L(\hat{y_n}, y_n) = -\sum_{i=1}^{N} y_{nj} log(\hat{y}_{nj})$$

The loss function is defined as

$$L(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}) = \sum_{n=1}^{N'} L(\hat{y}_n, y_n) + \frac{\lambda}{2} tr\{W^{(1)T} W^{(1)} + W^{(2)T} W^2\}$$

2.3 Forward Propagation

Let (x, y) be a sample. In the forward propagation phase,we calculate the loss function in the following way

$$z_k = f_1(\sum_{i=1}^{N} w_{ki}^{(1)} x_i + b_k^{(1)})$$

$$y_i = f_2(\sum_{k=1}^{M} w_{jk}^{(2)} z_k + b_k^{(2)})$$

$$L(\hat{y_n}, y_n) = -\sum_{i=1}^{N} y_{nj} log(y_j)$$

2.4 Backward Propagation

For each iteration, we update the parameters of the model by computing the gradient of the loss function.

$$w_{ki}^{(1)} \leftarrow w_{ki}^{(1)} + \eta \frac{\partial L}{\partial w_{ki}^{(1)}} + \lambda w_{ki}^{(1)}$$

$$w_{jk}^{(2)} \leftarrow w_{jk}^{(2)} + \eta \frac{\partial L}{\partial w_{jk}^{(2)}} + \lambda w_{jk}^{(2)}$$

$$b_k^{(1)} \leftarrow b_k^{(1)} + \eta \frac{\partial L}{\partial b_k^{(1)}}$$

$$b_j^{(2)} \leftarrow b_j^{(1)} + \eta \frac{\partial L}{\partial b_j^{(1)}}$$

The gradients are calculated on a computational graph in the following way.

$$\frac{\partial L}{\partial y_i} = \frac{y_{nj}}{y_j}$$

$$\frac{\partial L}{\partial b_k^{(2)}} = \sum_{j=1}^{C} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial b_k^{(2)}} = \sum_{j=1}^{C} \frac{\partial L}{\partial y_j} f_2'(\sum_{k=1}^{M} w_{jk}^{(2)} z_k + b_k^{(2)})$$

$$\frac{\partial L}{\partial w_{jk}^{(2)}} = \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial w_{jk}^{(2)}} = \frac{\partial L}{\partial y_j} f_2'(\sum_{k=1}^{M} w_{jk}^{(2)} z_k + b_k^{(2)}) z_k$$

$$\frac{\partial L}{\partial z_k} = \sum_{j=1}^{C} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial z_k} = \sum_{j=1}^{C} \frac{\partial L}{\partial y_j} w_{jk}^{(2)}$$

$$\frac{\partial L}{\partial b_i^{(1)}} = \sum_{k=1}^{M} \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial b_i^{(1)}} = \sum_{k=1}^{M} \frac{\partial L}{\partial z_k} f_1^{'}(\sum_{i=1}^{N} w_{ki}^{(1)} x_i + b_i^{(1)})$$

$$\frac{\partial L}{\partial w_{ki}^{(1)}} = \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial w_{ki}^{(1)}} = \frac{\partial L}{\partial z_k} f_1^{'}(\sum_{i=1}^{N} w_{ki}^{(1)} x_i + b_i^{(1)}) x_i$$

3 LeNet-5 CNN

3.1 Archietecture of LeNet-5 RNN

Figure 1 shows the archietecure of LeNet-5 CNN.The are two convolutional layers (CONV), two pooling layers (POOL) and three fully connected layers (FC).

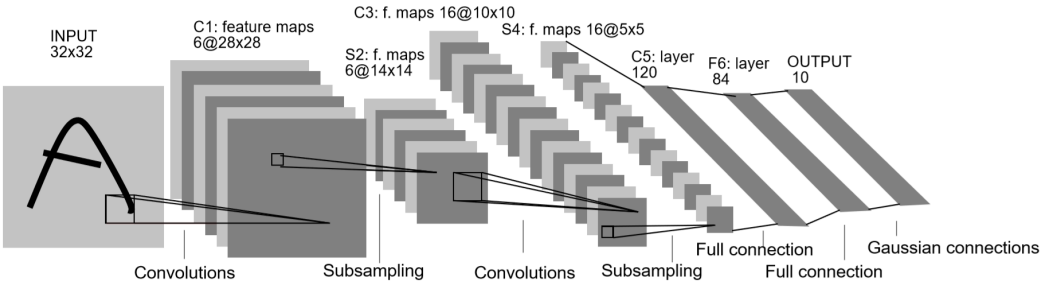

Figure 1: Archietecture of LeNet-5

3.2 Notations

The way of computing the gradients in fully connected layer is actually similar to the way we compute that in a multilayer perceptron. Therefore, we mainly focus on the CONV and POOL layers. Here are notations for further discussion.

1. $\tilde{X}^{(l)} \in \mathbb{R}^{N^{(l)} \times M^{(l)} \times D^{(l)}}$: the output of the l-th CONV block before the activation function
2. $X^{(l)} \in \mathbb{R}^{N^{(l)} \times M^{(l)} \times D^{(l)}}$: the output of the l-th CONV block. This means the input layer when $l = 0$
3. $\tilde{Y}^{(l)} \in \mathbb{R}^{N^{'(l)} \times M^{'(l)} \times D^{(l)}}$: the l-th CONV layer before the activation function

4. $Y^{(l)} \in \mathbb{R}^{N'^{(l)} \times M'^{(l)} \times D^{(l)}}$: the l-th CONV layer
5. $T^{(l)} \in \mathbb{R}^{D^{(l-1)} \times D^{(l)}}$: the link table of the l-th layer
6. $W^{(l)} \in \mathbb{R}^{n^{(l)} \times m^{(l)} \times D^{(l-1)} \times D^{(l)}}$: the filter of the l-th CONV layer
7. $b^{(l)} \in \mathbb{R}^{D^{(l)}}$: the bias of the l-th CONV layer
8. $w^{(l)} \in \mathbb{R}^{D^{(l)}}$: the weight of the l-th pooling layer
9. $a^{(l)} \in \mathbb{R}^{D^{(l)}}$: the bias of the l-th pooling layer
10. f_l: the activation function of the i-th CONV layer
11. g_l: the pooling function of the i-th pooling layer,

3.3 Forward Propagation

In the forward propagation phase, the CONV layers are computed in the following way

$$\tilde{Y}^{(l,d)} = \sum_{p, T^{(l)}_{p,d}=1} W^{(l,p,d)} \otimes X^{(l-1,p)} + b^{(l,d)}$$

$$Y^{(l,d)} = f_l(\tilde{Y}^{(l,d)})$$

where $\otimes$ refers to the convolution operation. The POOL layers are calculated as follows

$$\tilde{X}^{(l,d)} = \omega^{(l,d)} MeanPool(Y^{(l,p)}) + a^{(l,d)}$$

$$X^{(l,d)} = g_l(\tilde{X}^{(l,d)})$$

3.4 Backward Propagation

1. The way to calculate $\frac{\partial L}{\partial X^{(2)}}$ and the gradients of parameters in the FC layers is similar to what we have done with a multilayer perceptron, and thus requires more ink than is available for a detailed discussion.

2. Given $\frac{\partial L}{\partial X^{(l)}}$, $\frac{\partial L}{\partial \omega^{(l)}}$ can be computed in the following way.

$$\frac{\partial L}{\partial \omega^{(l,d)}} = sum(\frac{\partial L}{\partial X^{(l,d)}} \odot g_l^{'}(\tilde{X}^{(l,d)}) \odot MeanPool(Y^{(l,p)}))$$

   where sum refers to the sum of all elements of a matrix and $\odot$ means elementwise product of two matrices. $\frac{\partial L}{\partial a^{(l,d)}}$ can be computed in the following way.

$$\frac{\partial L}{\partial a^{(l,d)}} = sum(\frac{\partial L}{\partial X^{(l,d)}} \odot g_l^{'}(\tilde{X}^{(l,d)}))$$

$\frac{\partial L}{\partial Y^{(l)}}$ can be computed in the following way.

$$\frac{\partial L}{\partial Y_{i,j}^{(l,d)}} = \frac{\omega^{(l,d)}}{4}(\frac{\partial L}{\partial X^{(l,d)}} \odot g_l^{'}(\tilde{X}^{(l,d)}))_{[\frac{i+1}{2}],[\frac{j+1}{2}]}$$

3. Given $\frac{\partial L}{\partial Y^{(l)}}, \frac{\partial L}{\partial \tilde{Y}^{(l)}}$ can be computed in the following way.

$$\frac{\partial L}{\partial \tilde{X}^{(l)}} = \frac{\partial L}{\partial Y^{(l)}} \odot f_l^{'}(\tilde{X}^{(l)})$$

From that we can compute the following gradients

$$\frac{\partial L}{\partial W^{(l,p,d)}} = \frac{\partial L}{\partial \tilde{X}^{(l,d)}} \otimes X^{(l-1,p)}$$

for $T_{p,d}^{(l)} = 1$.

$$\frac{\partial L}{\partial b^{(l,d)}} = sum(\frac{\partial L}{\partial \tilde{X}^{(l,d)}})$$

$$\frac{\partial L}{\partial X^{(l-1,d)}} = \sum_{p,T_{p,d}^{(l)}=1} rot180(W^{(l,p,d)}) \tilde{\otimes} \frac{\partial L}{\partial \tilde{X}^{(l,d)}}$$

where $rot180$ means the operation of rotating a matrix by 180° and $\tilde{\otimes}$ means the wide convolution operation.

4 Vanilla RNN

4.1 Notations

Let's define some notations for the following discussion.

1. $x_t \in \mathbb{R}^N$: the input layer at time t
2. $h_t \in \mathbb{R}^M$: the hidden layer at time t
3. $y_t \in \mathbb{R}^C$: the output layer at time t
4. f: activation function at the hidden layer, usually a sigmoid or hyperbolic tangent function
5. g: mapping function at the ouput layer

4.2 Loss Function

Let's consider a synchronous seq2seq model. Suppose $(x_{1:T}, y_{1:T})$ is a sample, and $\hat{y}_{1:T}$ is the output at time t. The loss function at time t is defined as

$$L_t = L(y_t, \hat{y}_t)$$

which is usually a cross entropy loss. The overall loss function is then defined as

$$L = \sum_{t=1}^{T} L_t$$

4.3 Forward Propagation

The output layer can be computed in the following way

$$z_t = Uh_{t-1} + Wx_t + b$$

$$h_t = f(z_t)$$

$$y_t = g(h_t)$$

where $U \in \mathbb{R}^{M \times M}$, $W \in \mathbb{R}^{N \times M}$, $b \in \mathbb{R}^M$ are parameters of the model.

4.4 Backward Propagation: Backpropagation Through time (BPTT)

Let's take $\frac{\partial L}{\partial U}$ as the example.

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial U}$$

$\frac{\partial L_t}{\partial U}$ can be computed in the following way

$$\frac{\partial L_t}{\partial U_{ij}} = \sum_{k=1}^{t} \frac{\partial^+ z_k}{\partial U_{ij}} \frac{\partial L_t}{\partial z_k}$$

$\delta_{t,k}$ is defined as follows

$$\delta_{t,k} = \frac{\partial L_t}{\partial z_k}$$

which can be computed via the recurrent formula

$$\begin{aligned} \delta_{t,k} &= \frac{\partial h_k}{\partial z_k} \frac{\partial z_{k+1}}{\partial h_k} \frac{\partial L_t}{\partial z_{k+1}} \\ &= diag(f^{'}(z_k)) U^T \delta_{t,k+1} \end{aligned}$$

And $\frac{\partial^+ z_k}{\partial U_{ij}}$ can be calculated as

$$(\frac{\partial^+ z_k}{\partial U_{ij}})_p = [h_{k-1}]_j I(i = p)$$

Combine these formulas we get the gradient

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} h_{k-1}^T$$

Other gradients can be calculated in very similar ways. Here are the results.

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} x_k^T$$

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k}$$

5 References

1. https://nndl.github.io