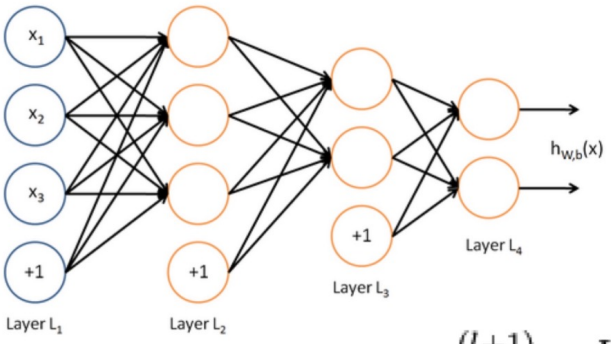
Deep Learning:Algorithm and Application: Homework #1

Due on March 28, 2017 at 23:59pm

Hu Tao

Problem 1

Derivation of backprop for MLP with ReLU, Sigmoid,and try an implementation.
Solution



$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

基本传播过程:

$$z^{(2)} = w^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = w^{(2)}a^{(2)} + b^{(2)}$$
$$\text{其中, } h_{w,b}(x) = a^{(3)} = f(z^{(3)})$$

变量定义:

$w^{(i)}$:第i层的w权重
$b^{(i)}$:第i层的偏置b权重
$z^{(i)}$:通过w,b计算以后的结果
$a^{(i)}$:通过w,b加权计算，然后通过非线性计算f以后的结果,其中$a^{i+1} = x^i$

要优化的目标函数如下:

$$J(W,b;x,y) = \frac{1}{2} \parallel h_{w,b}(x) - y \parallel_2^2$$
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} J(w,b;x,y) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2$$
$$= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2} \parallel h_{w,b}(x) - y \parallel_2^2 + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2$$

如果要进行梯度下降，总体上需要进行如下计算:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(w,b)$$
$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(w,b)$$
$$\frac{\partial}{\partial W_{ij}^{(l)}}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial W_{ij}^{(l)}}J(w,b,x^{(i)},y^{(i)}) + \lambda W_{ij}^{l}$$
$$\frac{\partial}{\partial b_i^{(l)}}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial b_i^{(l)}}J(w,b,x^{(i)},y^{(i)})$$

令$\delta_i^{(n_l)} = \frac{\partial}{\partial Z_i^{n_l}}J(W,b;x,y)$ 则:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial Z_i^{n_l}}J(W,b;x,y)$$
$$= \frac{\partial}{\partial Z_i^{n_l}}\frac{1}{2}\sum_{j=1}^{S^{n_l}}(y_j - a_j^{n_l})^2$$

$$
\begin{aligned}
&= \frac{\partial}{\partial Z_i^{n_l}} \frac{1}{2} \sum_{j=1}^{S^{n_l}} (y_j - f(Z_j^{n_l}))^2 \\
&= -(y_i - f(Z_i^{n_l})) f^{'}(Z_i^{(n_l)}) \\
&= -(y_i - a_i^{(n_l)}) f^{'}(Z_i^{(n_l)})
\end{aligned}
$$

上面的推导可以让我们从网络的底部（接近loss function的那一边）那一层求出δ.
下面推导如何递推求δ:

$$
\begin{aligned}
\delta_i^{n_l-1} = \frac{\partial}{\partial Z_i^{n_l-1}} J(W,b;x,y) &= \frac{\partial}{\partial Z_i^{n_l-1}} \partial 12 \parallel y - h_{w,b}(x) \parallel_2^2 \\
&= \frac{\partial}{\partial Z_i^{n_l-1}} \frac{1}{2} \sum_{j=1}^{s_{n_l}} (y_i - a_j^{n_l})^2 \\
&= \frac{1}{2} \sum_{j=1}^{s_{n_l}} \frac{\partial}{\partial Z_i^{n_l-1}} (y_i - a_j^{n_l})^2 \\
&= \sum_{j=1}^{S_{nl}} -(y_j - f(z_j^{nl}) \frac{\partial}{\partial Z_i^{n_l-1}} f(z_j^{nl}) \\
&= \sum_{j=1}^{S_{nl}} -(y_j - f(z_j^{nl}) \frac{\partial}{\partial Z_i^{n_l-1}} f^{'}(z_j^{nl}) \frac{\partial Z_j^{nl}}{Z_i^{nl-1}} \\
&= \sum_{j=1}^{S_{nl}} \delta_j^{nl} \frac{\partial Z_j^{nl}}{Z_i^{nl-1}} \\
&= \sum_{j=1}^{S_{nl}} \delta_j^{nl} \frac{\partial}{Z_i^{nl-1}} \sum_{k=1}^{S_{nl-1}} f(Z_k^{nl-1}) W_{jk}^{nl-1} \\
&= \sum_{j=1}^{S_{nl}} \delta_j^{nl} W_{ji}^{nl} f^{'}(Z_i^{nl-1}) \\
&= (\sum_{j=1}^{S_{nl}} \delta_j^{nl} W_{ji}^{nl}) f^{'}(Z_i^{nl-1})
\end{aligned}
$$

这样就完成的δ的推导。有了每一层的δ以后，就很好进行梯度下降了。

$$
\frac{\partial}{\partial W_{ij}^{(l)}} J(w,b;x,y) = \frac{\partial}{\partial Z_i^{(l+1)}} * \frac{\partial Z_i^{(l+1)}}{\partial W_{ij}^{(l)}} = a_j^{(l)} \delta_i^{(l+1)} \quad \frac{\partial}{\partial b_{ij}^{(l)}} J(w,b;x,y) = \delta_i^{(l+1)}
$$

最终:

$$
\begin{aligned}
W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{l}} J(w,b) \\
b_i^{(l)} &= b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{l}} J(w,b)
\end{aligned}
$$

最终的梯度下降流程如下:
1,前向传播

$$
\begin{aligned}
f(z) &= \frac{1}{1+exp(-z)} \\
z^{(l+1)} &= w^{(l)} a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

2,计算输出层（第nl层）的各个神经元的$\delta_i^{(nl)}$:

$$
\delta_i^{nl} = -(y_i - a_i^{nl}) * f^{'}(z_i^{(l)})
$$

3,从输出层向输入层递推计算每一层l神经元的$\delta_i^{(l)}$:

$$
\delta_i^{(l)} = (\sum_{j=1}^{S_l+1} w_{ji}^{(l)} \delta_i^{(l+1)}) * f^{'}(z_i^{(l)})
$$

4,计算各层$\frac{\partial}{\partial W_{ij}^{(l)}} J(w,b;x,y)$和$\frac{\partial}{\partial b_{ij}^{(l)}} J(w,b;x,y)$的值:

$$
\begin{aligned}
\frac{\partial}{\partial W_{ij}^{(l)}} J(w,b;x,y) &= a_j^{(l)} \delta_i^{(l+1)} \\
\frac{\partial}{\partial b_{ij}^{(l)}} J(w,b;x,y) &= \delta_i^{(l+1)}
\end{aligned}
$$

5,计算各层$\frac{\partial}{\partial W_{ij}^{(l)}} J(w,b)$和$\frac{\partial}{\partial b_i^{(l)}} J(w,b)$的值:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(w,b,x^{(i)},y^{(i)}) + \lambda W_{ij}^{l}$$
$$\frac{\partial}{\partial b_{i}^{(l)}} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(w,b,x^{(i)},y^{(i)})$$

6,对各层的$W_{ij}^{(l)}$和$b_i^{(l)}$进行更新:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{l}} J(w,b)$$
$$b_{i}^{(l)} = b_{i}^{(l)} - \alpha \frac{\partial}{\partial b_{i}^{l}} J(w,b)$$

7,直到J(w,b)足够小:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} J(w,b;x,y) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2$$

上面是使用sigmoid激活函数进行的推导，使用ReLu，LeakyReLu的推导类似，唯一的区别在于激活函数处的求导，其他地方可以依据链式法则照着推导。另外我使用了mlp中的sigmoid激活函数实现了异或计算，执行"python nn.py"即可。

Problem 2

Run a Mnist Classifer based on tensorflow

Solution

just run the python script by "python multilayer_perceptron.py".

During the script, we construct the model with layer input layer size 784(flattend by the input image),two intermediate layers with size 256,and output size 10(equal to the final class size).

and then we define the loss and optimizer.

finally we set the batch size = 100,and output epoch of the result.after the training.the test data inference result will print as follows.

```
Epoch: 0001 cost= 212.434582488
Epoch: 0002 cost= 43.698304957
Epoch: 0003 cost= 27.303758452
Epoch: 0004 cost= 19.144715998
Epoch: 0005 cost= 13.905557823
Epoch: 0006 cost= 10.521803028
Epoch: 0007 cost= 7.942286914
Epoch: 0008 cost= 6.117937308
Epoch: 0009 cost= 4.404222389
Epoch: 0010 cost= 3.519790992
Epoch: 0011 cost= 2.575894599
Epoch: 0012 cost= 1.885512865
Epoch: 0013 cost= 1.441038678
Epoch: 0014 cost= 1.136289034
Epoch: 0015 cost= 0.908172189
Optimization Finished!
Accuracy: 0.9474

Process finished with exit code 0
```