EECS 598 Deep Learning

HW1

Chuan Cen / chuancen / Feb 13 2019

Q1.

1. Fully-connected layer

$$\frac{\partial \mathrm{L}}{\partial \mathrm{W}} = X\left(\frac{\partial L}{\partial Y}\right)^T$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{b}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Y}}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}} = W\frac{\partial \mathrm{L}}{\partial \mathrm{Y}}$$

2. ReLU

$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Y}} \odot I\{X > 0\}$$

3. Dropout

$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}} = \frac{\partial \mathrm{L}}{\partial \mathrm{Y}} \odot \mathrm{M}$$

4. Batch Norm

$$\frac{\partial \mu}{\partial X_i} = \frac{1}{n}$$

$$\frac{\partial \sigma}{\partial X_i} = \frac{\frac{1}{n}(X_i-\mu)}{\sqrt{\frac{1}{n}\sum_j\left(X_j-\mu\right)^2+\epsilon}} = \frac{1}{n\sigma}(X_i-\mu)$$

Define $\hat{X}_i = \frac{X_i-\mu}{\sigma}$, let's first compute $\frac{\partial L}{\partial \hat{X}_i}$:

$$\frac{\partial L}{\partial \hat{X}_i} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial \hat{X}_i} = \frac{\partial L}{\partial Y_i}\gamma$$

$\frac{\partial L}{\partial \sigma}$:

$$\frac{\partial L}{\partial \sigma} = \sum_{i=1}^n \frac{\partial L}{\partial \hat{X}_i}\frac{\partial \hat{X}_i}{\partial \sigma} = \sum_{i=1}^n(-1)\frac{\partial L}{\partial \hat{X}_i}\frac{X_i-\mu}{\sigma^2}$$

$\frac{\partial L}{\partial \mu}$:

$$\frac{\partial L}{\partial \mu} = \sum_{i=1}^{n} \frac{\partial L}{\partial \hat{X}_i}\frac{\partial \hat{X}_i}{\partial \mu} + \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial \mu}$$

$$= \sum_{i=1}^{n} \frac{\partial L}{\partial \hat{X}_i}\frac{(-1)}{\sigma} + \frac{-\frac{2}{n}\sum_{i=1}^{n}(X_i-\mu)}{\sqrt{\frac{1}{n}\sum_j\left(X_j-\mu\right)^2+\epsilon}}\sum_{i=1}^{n}(-1)\frac{\partial L}{\partial \hat{X}_i}\frac{X_i-\mu}{\sigma^2}$$

$$= \sum_{i=1}^{n} \frac{\partial L}{\partial \hat{X}_i}\frac{(-1)}{\sigma}$$

$\frac{\partial L}{\partial X_i}$:

$$\frac{\partial L}{\partial X_i} = \frac{\partial L}{\partial \hat{X}}\frac{\partial \hat{X}}{\partial X_i} + \frac{\partial L}{\partial \mu}\frac{\partial \mu}{\partial X_i} + \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial X_i}$$

$$= \frac{1}{\sigma}\frac{\partial L}{\partial \hat{X}_i} + \frac{1}{n}\sum_{j=1}^{n} \frac{\partial L}{\partial \hat{X}_j}\frac{(-1)}{\sigma} + \frac{1}{n\sigma}(X_i-\mu)\sum_{i=1}^{n}(-1)\frac{\partial L}{\partial \hat{X}_i}\frac{X_i-\mu}{\sigma^2}$$

$$= \frac{1}{\mathrm{n}\sigma}\left(n\frac{\partial L}{\partial \hat{X}_i} - \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{X}_j} - \hat{X}_i\sum_{j=1}^{n}\frac{\partial L}{\partial \hat{X}_j}\hat{X}_j\right)$$

$$= \frac{\gamma}{\mathrm{n}\sigma}\left(n\frac{\partial L}{\partial Y_i} - \sum_{\mathrm{j}=1}^{\mathrm{n}}\frac{\partial L}{\partial Y_j} - \left(\frac{X_i-\mu}{\sigma}\right)\sum_{\mathrm{j}=1}^{\mathrm{n}}\frac{\partial L}{\partial Y_j}\left(\frac{X_j-\mu}{\sigma}\right)\right)$$

5. Valid Conv

$\frac{\partial L}{\partial X_{n,c}}$:

$$\frac{\partial L}{\partial X_{n,c,i,j}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X_{n,c,i,j}}$$

$$= \frac{\partial L}{\partial Y_n}\frac{\partial Y_n}{\partial X_{n,c,i,j}},\ \textit{since only the } n'th \textit{ sample is relevant}$$

$$= \sum_f \frac{\partial L}{\partial Y_{n,f}}\frac{\partial Y_{n,f}}{\partial X_{n,c,i,j}}$$

$$= \sum_f \sum_{p,q} \frac{\partial L}{\partial Y_{n,f,p,q}}\frac{\partial Y_{n,f,p,q}}{\partial X_{n,c,i,j}}$$

$$= \sum_f \sum_{\mathrm{u,v}} \frac{\partial L}{\partial Y_{n,f,i-u+1,j-v+1}}\frac{\partial Y_{n,f,i-u+1,j-v+1}}{\partial X_{n,c,i,j}}$$

$$\text{where } \mathrm{u} = \mathrm{i}-\mathrm{p}+1, \mathrm{v} = \mathrm{j}-\mathrm{q}+1$$

$$= \sum_{\mathrm{f}}\sum_{\mathrm{u,v}} \frac{\partial L}{\partial Y_{n,f,i-u+1,j-v+1}} W_{f,c,u,v}$$

$$= \sum_{\mathrm{f}} W_{f,c} *_{full} \left(\frac{\partial L}{\partial Y_{n,f}}\right), \qquad \text{by definition of " } *_{\text{full}} \text{ ".}$$

$\frac{\partial L}{\partial W_{f,c}}$:

$$\frac{\partial L}{\partial W_{f,c,i,j}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial W_{f,c,i,j}}$$

$$= \frac{\partial L}{\partial Y_f}\frac{\partial Y_f}{\partial W_{f,c,i,j}} \text{ , } since\ only\ output\ channel\ f\ is\ relevant$$

$$= \sum_n \frac{\partial L}{\partial Y_{n,f}}\frac{\partial Y_{n,f}}{\partial W_{f,c,i,j}}$$

$$= \sum_{\mathrm{n}} \sum_{\mathrm{u,v}} \frac{\partial L}{\partial Y_{n,f,u,v}}\frac{\partial Y_{n,f,u,v}}{\partial W_{f,c,i+u-1,j+v-1}}$$

$$= \sum_{\mathrm{n}} \sum_{\mathrm{u,v}} \frac{\partial L}{\partial Y_{n,f,u,v}} X_{n,c,i+u-1,j+v-1}$$

$$= \sum_{\mathrm{n}} X_{n,c} *_{filt} \left(\frac{\partial L}{\partial Y_{n,f}}\right), \qquad \text{by definition of " } *_{\text{filt}} \text{ ".}$$

Q2.

3. Train a simple logistic classifier (with no hidden units). Report the test accuracy for the best model you identified.

test accuracy = 93.6%

4. Train a 2-layer neural network with logistic regression as the output layer. Identify and report an appropriate number of hidden units based on the validation set. Report the test accuracy for your best model.

hidden_dim = 1000, test accuracy = 94%

Q3.

3. Train a binary SVM classifier (with no hidden units). Report the test accuracy for the best model you identified.

test accuracy = 92.8%

4. Train a 2 layer neural network with hinge-loss as the output layer. Identify and report an appropriate number of hidden units based on the validation set. Report the best test accuracy for your best model.

hidden_dim = 1000, test accuracy = 94%

Q4.

3. Train a softmax multi-class classifier (with no hidden units) on MNIST dataset. Report the test accuracy.

test accuracy = 92%

4. Train a 2 layer neural network with softmax-loss as the output layer on MNIST dataset. Identify and report an appropriate number of hidden units based on the validation set (you can leave 1/10 data samples in the train dataset as validation dataset). Report the best test accuracy for your best model.

hidden_dim = 1000, test accuracy = 93.63%

Q5.

3. Train the CNN multi-class classifier on MNIST dataset. Identify and report an appropriate filter size of convolutional layer and appropriate number of hidden units in fully-connected layer based on the validation set (you can leave 1/10 data samples in the train dataset as validation dataset). Report the best test accuracy for your best model.

- Filter size: 7x7

- number of hidden units:100

- test accuracy: 98.39%

4. Train the CNN multi-class classifier with dropout and batch normalization on

MNIST dataset. Identify and report an appropriate architecture and appropriate rate of dropout based on the validation set (you can leave 1/10 data samples in the train dataset as validation dataset). Report the best test accuracy for your best model.

- Architecture: Conv-BN-Relu-Maxpool-Dropout-Fc-Relu-Fc-Softmax

- Rate of dropout: 0.8

- Test accuracy: 98.68%

Q6.

2. Test the model in test function and report the classification accuracy on the test set.

Test accuracy: 69%

3. **You are encouraged to try different optimizer, image preprocessing, activation function, convolutional feature size, learning rate strategy, architecture of the network to make the classification accuracy better on test set.**

By inserting a ReLU layer right after every Conv2d layer, and by increasing the filter number of every Conv2d layer to 4 times larger than the original one (i.e. 8→32, 64→256, etc.), we got ***75% accuracy on test set.***

The full architecture: *(see next page)*

```
self.conv = nn.Sequential(
        nn.Conv2d(3, 32, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2, padding=0),
        nn.Conv2d(32, 64, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2, padding=0),
        nn.Conv2d(64, 128, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.Conv2d(128, 128, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2, padding=0),
        nn.Conv2d(128, 256, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.Conv2d(256, 256, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2, padding=0),
        nn.Conv2d(256, 256, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.Conv2d(256, 256, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.MaxPool2d(2, padding=0)
)
self.fc = nn.Sequential(
        nn.Linear(256, 256, bias=True),
        nn.ReLU(),
        nn.Linear(256, 10, bias=True)
)
```

Q7. Short answer questions

1. Describe the main difficulty in training deep neural networks with the sigmoid non-linearity.

The outputs of sigmoid activation are not zero-centered and always positive. This means that the neurons of later layers would receive all positive data, which will cause problems when doing gradient descent. Consider a fully-connected layer $\mathrm{f} = \mathrm{w}^{\mathrm{T}}x + b$ which receives all positive data x, then the gradient on the weights w will during backpropagation become either all be positive, or all negative (depending on the sign of $\frac{\partial \mathrm{L}}{\partial \mathrm{f}}$). This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights, thus result in slow convergence.

2. Consider a linear layer W. During training, dropout with probability p is applied to the layer input x and the output is given by y = W (x ⊙ m) where m represents

the dropout mask. How would you use this layer during inference to reduce the train/test mismatch (i.e., what is the input/output relationship).

During inference, every element of x should multiply p to reduce the train/test mismatch: $y = W(x \cdot p)$.

3. If the training loss goes down quickly and then diverges during training, what hyper-parameters would you modify?

1) Reduce the learning rate. High learning rate result in initially quick loss drop and then divergence in later iterations. A lower learning rate would alleviate this problem.

2) Increase the batch size. A larger batch size results in a gradient direction of higher confidence and makes the training process more robust and less divergent.