CS231n Summary

Chengzong Cai

January 2019

1 Image Classification

1.1 kNN(k Nearest Neighbor)

This algorithm is a data driven method.Compute the l1(or l2,etc) distance between the given image and training image,then pick out the most common label of nearest k training images as the label of given image.

This method is naive without a training process and it's performance is so poor to address practical problems.For example, if you are dressed in a leopard skirt, this algorithm definitely can't tell you from a real leopard.It just simply compares images and images without learning the nature of images. Hence once the lighting condition or rotation angle of images change even a little, this algorithm won't work anymore.

1.2 SVM Linear Classification

This algorithm get a score of each label by using a weight matrix to dot product with the given image, and use SVM loss(hinge loss) to evaluate the performance and optimize the weight matrix.The working procedure is described in the following formulas.

$$S = W^T x + b$$
$$y_{pred} = argmax_j S_j$$
$$L_x = \sum_i max(0, S_i - S_y + 1)$$

Surprisingly, the algorithm works well on CIFAR-10 with the accuracy near 40%.The visualizing shows us that $W_i looks similar to the images with label i$.

2 Loss and Optimization

2.1 Loss Function

Loss function is used to evaluate the distance between output and the ground truth.Basically,the performance of a classifier becomes higher when the value of

loss function gets lower.In practical application, we use a set of loss function to fit different tasks.Choosing a appropriate loss function may make the training process much faster and easier.
Frequently used loss functions:

$$\text{Hinge loss:} L = \sum_i max(0, S_i - S_y + 1)$$
$$\text{Softmax loss:} L = -log(e_y^S / \sum_j e^{S_j})$$
$$\text{Cross entropy loss:} L = \sum_i y_i log(y_{ipred} + (1-y) log(1 - y_{ipred}))$$
$$\text{MSE loss:} L = \sum_j (y_i^2 - y_{ipred}^2)$$

2.2 Optimization and back propagation

Due to the connection between loss function and performance,we take it's gradient to update the parameters and get a higher performance.

2.2.1 Gradient Descend and Back Propagation

As we know, the value of function changes largestly when the change of parameters follows the direction of gradient.Naturely the first idea that come to us might be compute the formula describing the gradient of each parameter.When the scale of parameter is small and the relationship among them is simple,this idea may be feasible.However the pratically used classifier usually get a lot layers of parameters.Both the relationship and scale is too complex to directly compute gradient formulas.
So the scientist found another way to compute the gradient and that way is back propagation.This idea comes from simple mathematical principle(the chain rule).

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial a} \frac{\partial a}{\partial b}$$

The above formula figures out that the gradient flows from upside to the downside.To get the gradient of each parameter,all we need to do is multiplying the upper gradient that flows here and the local gradient.
Now we get the gradient,finding a proper way to do gradient descending is another problem.So I collect some most commonly used optimizer.

$$\text{BGD:} \theta = \theta - \alpha \bigtriangledown_\theta L(\theta)$$
$$\text{SGD:} \theta = \theta - \alpha \bigtriangledown_\theta L(\theta, X_i, Y_i)$$
$$\text{Momentum:} v_t = \gamma v_{t-1} + \alpha \bigtriangledown_\theta L(\theta)$$
$$\theta = \theta - v_t$$
Nestrov accelarated momentum:
$$v_t = \gamma v_{t-1} + \alpha \bigtriangledown_{\theta - \gamma v_{t-1}} L(\theta)$$
$$\theta = \theta - v_t$$
Adam:
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \bigtriangledown_\theta L$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \bigtriangledown_\theta L^2$$
$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} - \epsilon} \hat{m}_t$$

Using Nestrov accelerated momentum and Adam make the training process much faster and more steadily even with a larger learning rate α.

3 Neural Networks

Being inspired by the biological neural system,scientists designed artificial neural networks(ANN) in order to achieve artificial intelligence. A modern artificial neural network contains lots layers of nodes.,Just like the biological neurons do,these nodes get input data from lower layers,they weight and bias those inputs then output after a certain activation function.The activation function is the point,it provides the non-linearity to the neural network,allowing it to fit a certain function.After a proper training process,it's powerful fitting ability allows it to deal with some tasks that once can be done only by humans.Now we've got lots variations of neural networks designed to deal with certain tasks like Convolutional Neural Network(CNN) in computer vision, Recurrent Neural Network(RNN) in sequential data processing and so on.

3.1 Training A Neural Network

3.1.1 Activation Function

In my opinion, activation function is one of the most essential part in the neural network, providing non-linearity that gives neural network the ability of fitting a certain function.Choosing a proper activation function may lead to a faster and easier training process.We used to apply tanh and sigmoid function as the activation function.However both of them would be saturated with large inputs,bringing a extremely small gradient that could block even stop the training process.So now we often use the variations of ReLU(Using vanilla ReLU leads to dead ReLU problem) like leaky ReLU as the activation function.

3.1.2 Data Preprocessing and Initialization

Generally speaking,before being sent to train a neural network,image data will be preprocessed to be zero-centered.Because if data vector were all positive or negative,allowed updating direction of weight matrix will be also all positive or negative,which leads to a longer and less effective training process.
Data preprocessing also includes PCA,whitening,and many other methods,but for images they are not commonly used.
Weight initialization is another problem.In practice we frequently use and Xavier initialization.

Xavier Initialization:$W_i = Random(outputsize, inputsize)/\sqrt{inputsize}$

3.1.3 Batch Normalization

Batch normalization is born to address internal coviarate shift.In my opinion,the core idea of batch normalization is to limit the mean and variance in a proper interval of input of every layer to make sure all layers do really converges.

$$\text{Batch Normalization:}\hat{x} = \frac{x-\bar{x}}{Var(x)}$$
$$y = \gamma\hat{x} + \beta$$

The above formulas is all BN does,where γ and β is parameter for NN to learn.

3.1.4 Regularization

Regularization is a method to address overfitting problem.In practical application,we use dropout,dropconnect,BN,weight regularization,data augment and so on.In my opinion, dropout, dropconnect and data augment have something in common,that is forcing neural network to learn from not local feature but the entirety.Weight regularization is following another line of thought that is to decrease the coefficient of each term especially the higher ones would prevent overfitting.And personally speaking,I think BN makes NN to learn from the structure of the input of each layer,not including their mean and variance.

3.1.5 Transfer Learning

Basically speaking, the lower layers of different neural network have the something in common that is to extract features.So we can apply pre-trained model on our tasks.All we need to do is directly use pre-trained model and fine tune the higher layer to fit our learning problems.

3.2 CNN

Convolutional neural networks is inspires from biological vision processing system.Different from the other kinds of neural networks, CNN have convolutional layers and pooling layers.A convolutional layer contains a set of filters and biases.Filters are used to do convolution with the image,finding certain features.If the feature that filter finds does appear in the image, the output value of corresponding area in the output image would be high.A pooling layer is just simple shrink the size of input data in order to save the major feature, decrease parameters,and avoid overfitting.The working process of CNN can be imaged that as the data flows deeper and deeper,the level of feature those layers exact becomes higher and higher,and finally the level of feature is so high that the network realized that in the image there does exist something.

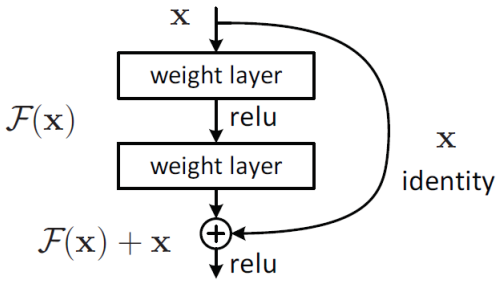

Figure 1: ResNet

3.2.1 ResNet and Bottleneck layer

With the net becoming deeper and deeper,the gradient from the top will be hard to flow to the bottom of the network ,causing a less effective training process.To handle this problem,Kaiming He invited the ResNet.
The core idea of ResNet is that the result of the former layer is close to the background truth,and the deeper layer just need to slightly correct the given result.So in ReNet, the former result is added to the output of the next layer(or using bottleneck layers when the channel is different between output and the input),creating a gradient super highway which makes the training much faster and easier and allowing a much deeper network.In 2015 ImageNet,ResNet even beat human in CIFAR10 dataset.And so far,many researchers uses ResNet in their AI projects like AlphaGo and Mask R-CNN.

3.2.2 Upsampling:Using transposed convolution

Different from traditional convolution,transpose convolution upsamples a feature map to generate an image.In the transpose convolution layer,the input give weights for filters.And then put those weighted filters in different place of each output channel(sum if they overlap) by following given strides and padding.And mathematically speaking,transpose convolution operation is just doing a matrix multiplication between transposed filter matrix and given inputs. The upsampling feature makes transposed convolution frequently be applied in generative models,style transfer and segmentation which output high resolution images.

3.3 RNN

Different from images,sequential data cares more about relationship of context like language and audio data,and the length of the input and output we need could be variable,with which ordinary NN are not capable. So here comes RNN,it get input data per time step and then compute the hidden state not only with the input data,but also with the former hidden state.And it's input and output size can be variable.Thus,RNN is qualified for dealing with sequential data tasks like voice recognition and language translation.

The training process of RNN is also quite different from ordinary NN.The back propagation of RNN is called BPTT(Back Propagation Through Time).As the figure shows,each output has something to do with the former inputs through hidden states.For example in Figure 2.

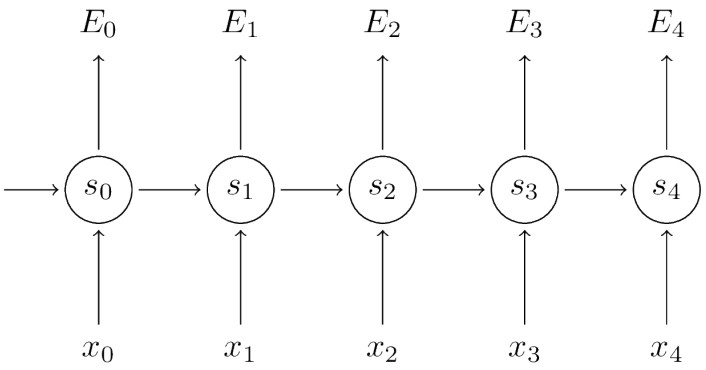

Figure 2: RNN Architecture

$$\frac{\partial E_4}{\partial x_0} = \frac{\partial E_4}{s_4} \prod_{i=0}^{3} \frac{\partial s_{i+1}}{\partial s_i} \frac{\partial s_0}{\partial x_0}$$

The internal production is the partial between the i_{th} hidden state and the former one. However,BPTT of the original RNN would cause gradient vanishing and gradient explosion due to the internal series multiplication of gradient.This problem is addressed by the variations of RNN like LSTM.

3.3.1 LSTM

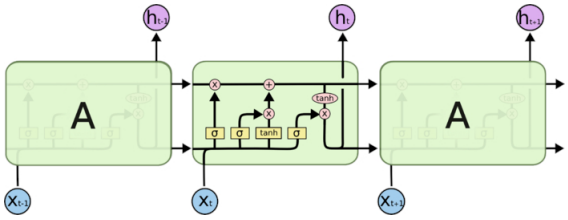

Figure 3: LSTM

As the Figure 2 shows,at each time step,LSTM concatenate the current input X_i and the former hidden state H_{i-1},and then compute $_1$,$_2$,$Tanh_1$.Then $_1$ does a pointwise multiplication with cell state C_{i-1},deciding whether C_{i-1} will be forgotten or not.And the following step is to do a pointwise multiplication between $_2$ and $Tanh$ and update C_i with the result.Lastly LSTM compute $_3$

using $[X_i, H_{i-1}]$ and $Tanh_2$ using the updated cell state C_i,and do a pointwise multiplication between them to generate the current hidden state and output H_i.This RNN model solved the gradient vanishing/explosion problem, the gradient flows fluently along the the cell state branch.And it and it's variation have become the most applied RNN model.

4 Some Questions

Q1:My interested researching area is about generative models.Is there any mini projects about that for me to train my coding ability?
Q2:How can I become a qualified researcher like my seniors and those essay writers.I sometimes feel great pressure when I'm reading essays and the corresponding codes because totally understanding those essays and codes is still not easy for me.