Weekly Work Report

Yufeng Jiang

VISION@OUC

July 22, 2018

1 Research problem

I am working on the paper named discriminative region proposal adversarial networks for high-quality image-to-image translation and written by Chao Wang *et al.* [3]. Helping teacher to write the patent of this paper and running codes written by myself as the auxiliary experiment are main tasks needing me to devote myself to do.

2 Research approach

In this week, I am still doing the basic work like watching the deep learning videos [2] and GAN videos [1] to build my strong fundation.

3 Research progress

I have learned the first part of the deep learning coarses [2] before this week. The first part mainly teaches the basic of neural network containing bilinary classificaiton, logistic regression, gradient descent, cost function, loss function and vectorization. And then the teacher named Andrew extends the scope to one hidden layer neural network and deep neural networks that are established at logistic regression *et al.*

4 Progress in this week

In this week, I have watched the second and fourth part of deep learning videos [2] and the first and second videos of GAN [1]. And I have translated the paper [3] into Chinese for the teacher Wang who writes the patent of this paper, and explained the working principle and formula mearnings to teacher.

4.1 Setting up your ML application

For a sample dataset, it will be divided into three parts: train set, development set and test set. The development set and test set have been changed more smaller because of the increase of data. We usually use the train set to train our models to make it more efficient and better. The purpose of development set is validating the most efficient algorithm among some different algorithms. And the test set is used to evaluate the result of our models.

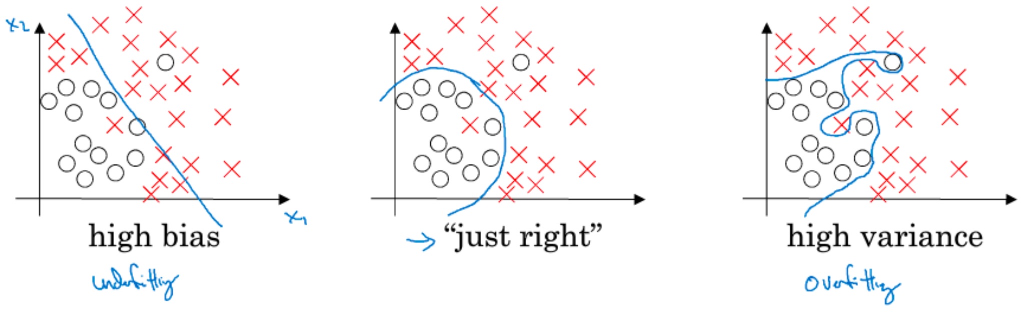

Figure 1: Bias and variance

From the aspect of the bias-variance tradeoff, our models can become high bias (see Fig. 1) if we have less times to train models. However, if our models have been set so compicated that there are some noise at train set, models can become overfitting, and experience high variance at developement set. In view of high variance problem, regularizaiton that is added to cost function as a regularization term (see Eq. 1) has been proposed to solve it.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} l(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{m}||w||_2^2 \tag{1}$$

where $\frac{\lambda}{m}||w||_2^2$ is the $L2$ regularization of logistic regression, and it can be explained as:

$$\frac{\lambda}{m}||w||_2^2 = \frac{\lambda}{2m}\sum_{j=1}^{n_x} w_j^2 = \frac{\lambda}{2m} w^\top w \tag{2}$$

Similarly to vectorization, this regularization term can also be added to neural network, just as shown at Eq. 3.

$$J(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} l(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||w^{[l]}||_F^2 \tag{3}$$

where $||w^{[l]}||_F^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} (w_{ij}^{[l]})^2$. After adding the regularization term, the gradient updating equation becomes Eq. 5:

$$dW^{[l]} = (form\_backprop) + \frac{\lambda}{m} W^{[l]} \tag{4}$$
$$W^{[l]} := W^{[l]} - \alpha dW^{[l]} \tag{5}$$

From the understanding of intuition, at the case of the regularization factor is set big enough, the value of weight matrix W will be set close to 0 to minimize the cost function. It is equivalent to eliminate the effects of many neurons, so the big neural network will become a smaller network (see Fig. 2)

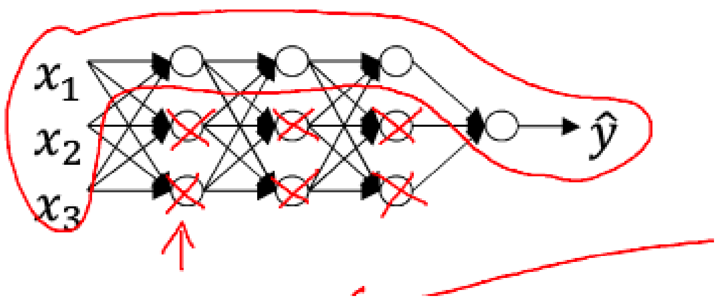

Figure 2: How does regularization prevent overfitting?

Another common method of regularization to prevent high variance is *Dropout*. The main thought is setting a probability of random elimination for each neuron node at the *Dropout* layers at neural network. Thus, we get a small scale network with less nodes to train just shown at Fig. 3.

The approach to achieve is inverted dropout, and the main code is:

$$a3 \quad /= \quad keep\_prob \tag{6}$$

Notation: It is better not using *Dropout* at test stage, beacuse it can make predicted results become randomly.

When training a neural network, one of the techniques that will speed up our training is if we normalize inputs. This method can be decomposed into two steps: **(i)** zero out the mean, **(ii)** normalize

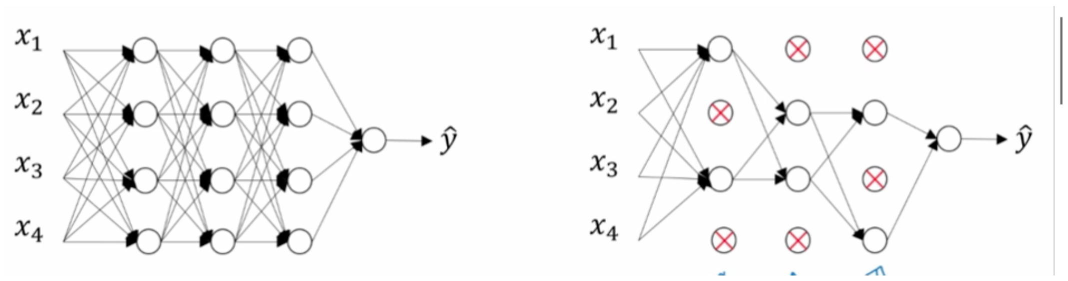

Figure 3: Dropout regularization.

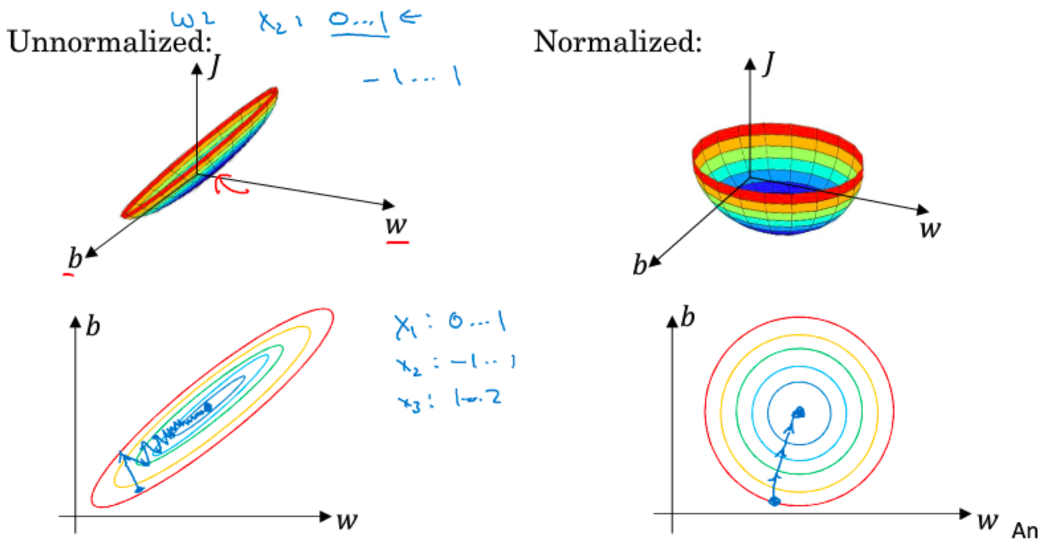

Figure 4: Why normalize inputs?

the variances. First, we need to compute the mean value of all sample data at each feature and minus the mean value to get a symmeticr distribution:

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \tag{7}$$

$$x := x - \mu \tag{8}$$

The second step contains two aspects:

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} x^{(i)^2} \tag{9}$$

$$x = x/\sigma \tag{10}$$

It can be seen at Fig. 4 that the shape of cost function has much differences as we use normalization or not. We may have a lot of iterations to get the most global optimal solution if we set a smaller learning rate. But if we use the normalization, we can find the most global optimal solution as a little iteration wherever location.

4.2 Optimization algorithms

When we use gradient descent method at the whole trainnig set, we need to train the whole training dataset to get once gradient descent. To improve the disadvange about slower proceeding speed, we can divide the train set into some parts called $Mini-batch$, and train one part every time.

There are three methods to accelerate the learning algorithms: $Momentum$, $RMSprop$ and $Adam$. The basic thought of $Momentum$ gradient descent is computing the exponential weighted average of gradient, and using this gradient to update weight. The traditional method like gradient descent produces a larger fluctuation to slow down the rate of gradient descent. And the other shortcoming is that we only use a smaller learning rate to iterate. We implement algorithm by:

$$\begin{aligned} v_{dw} &= \beta v_{dw} + (1-\beta)dW \\ v_{db} &= \beta v_{db} + (1-\beta)db \\ W &= W - \alpha v_{dw} \\ b &= b - \alpha v_{db} \end{aligned} \tag{11}$$

The explination of it is shown at Fig. 5.

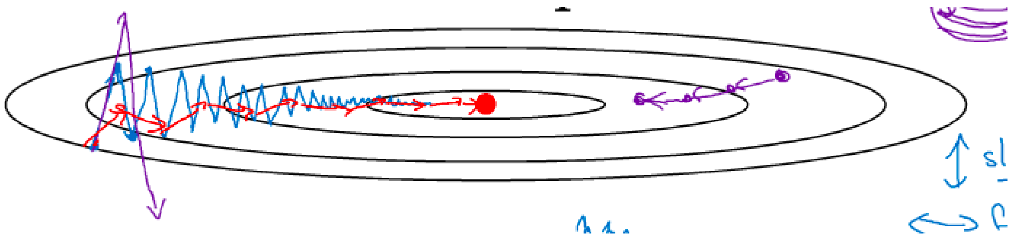

Figure 5: Momentum vs gradient descent.

The $RMSprop$ can reduce the large fluctuation at some dimension gradients updating. The basic thought of adaptive moment estimation (Adam) optimization algorithm is combining the $Momentum$ and $RMSprop$ to become a suitable for different depth learning structures.

4.3 Hyperparameter tuning

In the logistic regression, normalizing inputs can speed up the training model. Similarly to a deeper layer neural network, we can still normalize the $z^{[l]}$ before the activation function at hidden layers. The implement of batch norm can be explained at:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_i z^{(i)} \\ \sigma^2 &= \frac{1}{m}\sum_i (z^{(i)}-\mu)^2 \\ z^{(i)}_{norm} &= \frac{z^{(i)}-\mu}{\sqrt{\sigma^2+\varepsilon}} \end{aligned} \tag{12}$$

After these steps, we have normalized $z^{(i)}$, so each component of $z^{(i)}$ contains an average of 0 and a variance of 1. But, we don not hope units of hidden layers are always like so. So we can use Eq. 13 to make $z^{(i)}$ can be a random value.

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta \tag{13}$$

The main though of $softmax$ is computing the probabilities that the whole sum is 1 by computing the vector $z^{[l]}$, and logistic regression is generalized by softmax regression from bilinary classificaiton problem to multi classification problem.

$$\begin{aligned} z^{[l]} &= W^{[l]} z^{[l-1]} + b^{[l]} \\ t &= e^{(z^{[l]})} \\ a^{[l]} &= \frac{e^{z^{[l]}}}{\sum_{j=1}^{4} t_i} \\ a_i^{[l]} &= \frac{t_i}{\sum_{j=1}^{4}} t_i \end{aligned} \tag{14}$$

4.4 Convolutional neural networks

The convolution at neural networks is using filter to multiply the corresponding factor and then sum all results. The detail can be seen at Fig. 6.

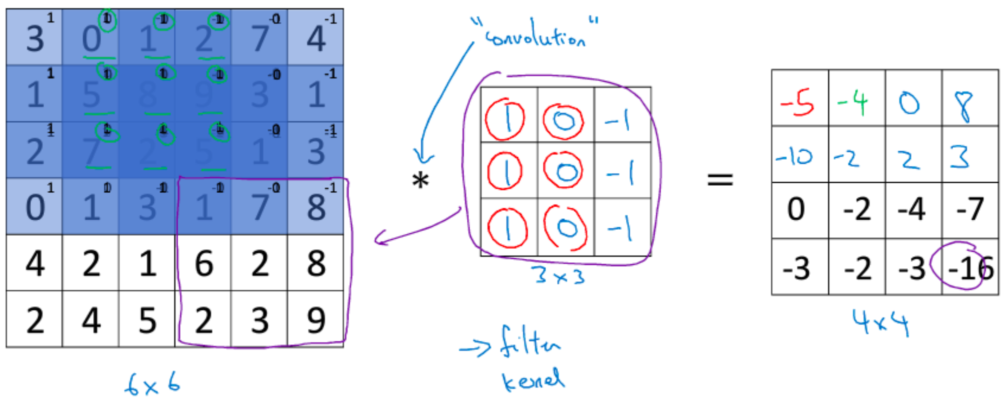

Figure 6: Vertical edge detection

The size of the right matrix is gotted by:

$$\lfloor \frac{n+2p-f}{s} + 1 \rfloor \times \lfloor \frac{n+2p-f}{s} + 1 \rfloor \tag{15}$$

where p means *padding* that is added before convolution to enclose pixels at corner and edge, so the image does not change its shape and loss the information of corner and edge. The s is *stride*, it represents the length of moving each convolution operation. The principle is adapted to the cube convolution. The specific steps are shown at Fig. 7.

A whole convolutional neural network is constitutes by three parts: convolution (CONV), pooling (POOL) and fully connected (FC). The pooling layer has two traditional methods: max pooling that takes the maximum value of each frame, and average pooling that takes the average value of each frame.

4.5 Case studies

Traditional convolutional neural networks contain LeNet-5, AlexNet and VGGNet. The size of image has narrowed as the depth of neural network deepens, while the number of channels increases. The LeNet-5 mainly handles gray images and the AlexNet handles big colorful images directly. The convolution layers and pooling layers have the same convolutional core with 3×3 and $stride = 1$, $SAME$ convolution and 2×2, $stride = 2$ pooling.

The ResNet is sturctured by residual block. The big innovation adds a connection from $a^{[l]}$ to $a^{[l+2]}$ named "short cut" or "skip connection" (see Fig. 8).

The 1×1 convolution equals to apply a fully connected neurla network to n_c units on a slice.

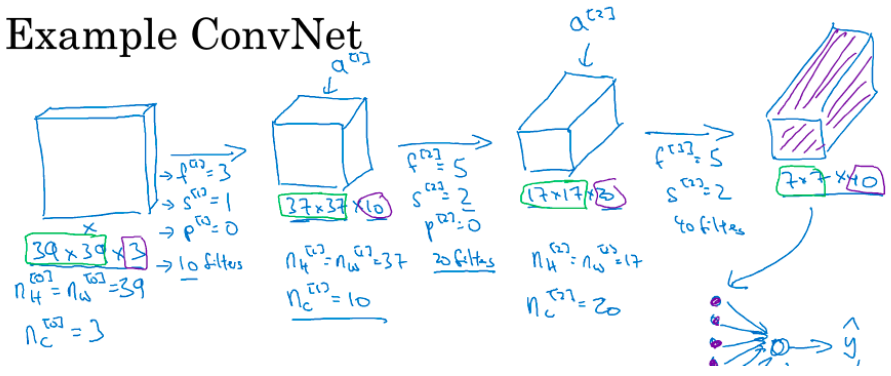

Figure 7: Example ConvNet.

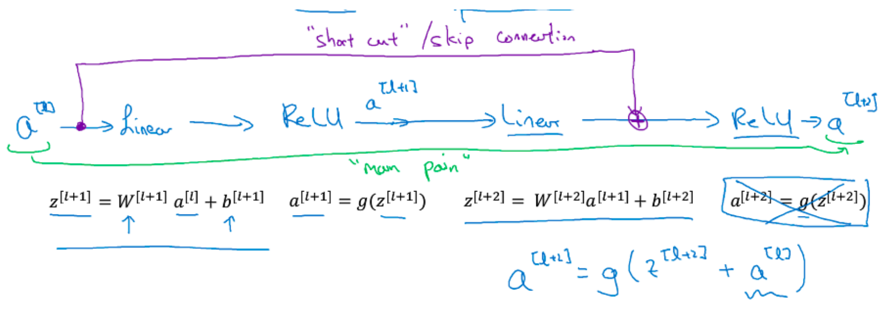
"short cut" / skip connection
$a^{[l]}$
Linear
ReLU
$a^{[l+1]}$
Linear
ReLU
$a^{[l+2]}$
"main path"
$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$
$a^{[l+1]} = g(z^{[l+1]})$
$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$
~~$a^{[l+2]} = g(z^{[l+2]})$~~
$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

Figure 8: Residual block.

4.6 Object detection

If we want to determine the object localization, the label of y can be:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Supporsing the problem is a four classification problem. P_c corresponds to whether there is an object. And b_x, b_y, b_h, b_w can get a boundary frame. The c_1, c_2, c_3 represent whether there is pedestrain, car and motor separately. The objection detection adapts the sliding windows algorithm that traversing each area of the image at a fixed stride. But this method has a disadvange is that it can not output the most accurate bounding box. Based on this motivation, **YOLO** algorithm is researched to find the bounding box accurately.

Notation: The key step of using **YOLO** algorithm to assign object to a lattice is observing the midpoint of the object, and assign object to the lattice contains the midpoint. **Non-max suppression (NMS)** can make sure that the algorithm detect each object for one time. However, if we want to detect many objects at the same time, we can use **anchor box** just as shown at Fig. 10.

4.7 Face recognition

For a face recognition system, we need to realize the face recognition only through a previous face image. *Similarity* function which inputs two images and outputs the degree of difference between these two

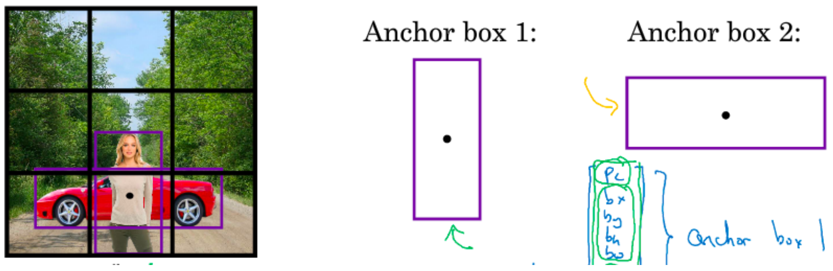

Figure 9: Overlapping objects.

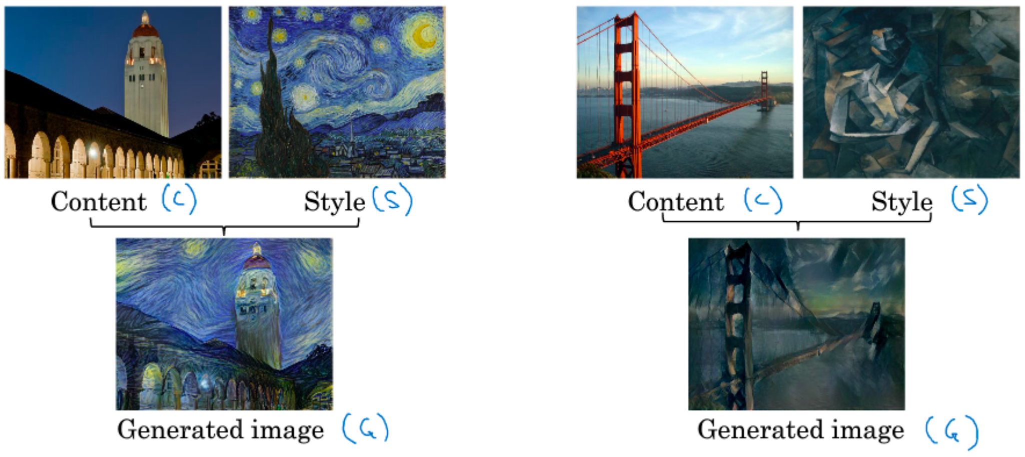

Figure 10: Neural style transfer.

images. If these two images are from a same person, $d(img1, img2) \leq \tau$ and function outputs "same". Otherwise, it will output "different". Putting similarity function into neural network, this neural network will become a *Siamese* network. It represents two images norm:

$$d(x_1, x_2) = ||f(x_1) - f(x_2)||_2^2 \tag{16}$$

Sometimes, we want to recognize three images containing anchor (A), positive (P) and negative (N) whether are from a same people. The *Triplet* loss function based on these three images (see Eq. 17) and the cost function of the whole network is Eq. 18.

$$L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0) \tag{17}$$

$$J = \sum_{i=1}^{m} L(A^{(i)}, P_{(i)}, N^{(i)}) \tag{18}$$

As for neural style transfer, our goal is generating the final style transfered image (G) through content image (C) and style image (S). The cost function J definided at Eq. 19 is used to evaluate the result of generated image (see Fig. 10).

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G) \tag{19}$$

where $J_{content}(C, G)$ represents the similary degree between the generated image and the content image, and $J_{style}(S, G)$ represents the similary degree between the generated image and the style image.

4.8 The introduction of GAN and ECCV paper

A generative adversarial networks (GAN) contains two modules: generator and discriminator. The purpose of generator is generating a synthesized image close to the real image and make the discriminator output value close to 1. But, the discriminator wants to discriminate the input image whether is a real image or not correctly.

In this paper, authors explore a noval method to generate a high-quality image-to-image translation based on GAN. There are a generator and two discriminator named patch discriminator (PatchD) and reviser separately.

Inputting a semantic segmentation graph, and then generator (G) generates a fake image mapped by a proposed image patch gotted from patch discriminator (D) onto a discriminative region. Then authors mask the corresponding real image using the discriminative region to obtain a fake-mask real image. Reviser (R) is used to provide constructive revisions to generator through distinguishing the real image from the fake-mask real image.

5 Plan

2018.07.23—2018.07.24 Doing the code exercise of the second and fourth part at deeplearning.ai.

2018.07.25—2018.07.29 Watching the GAN videos, and see codes written from Chao Wang.

References

[1] bilibili. https://www.bilibili.com/video/av24011528/. 1

[2] Neural Networks & Deep Learning. https://mooc.study.163.com/course/2001281002#/info. 1

[3] C. Wang, H. Zheng, Z. Yu, Z. Zheng, Z. Gu, and B. Zheng. Discriminative region proposal adversarial networks for high-quality image-to-image translation. In *ECCV*, 2018. 1