ID2223
Scalable Machine Learning and Deep Learning

Review Questions 3

FALL2021

Yuchen Gao yuchenga@kth.se
Weikai Zhou weikai@kth.se

November 27, 2021

KTH EECS Embedded Systems

1 Is it OK to initialize all the weights of a neural network to the same value as long as that value is selected randomly using He initialization? Is it okay to initialize the bias terms to 0?

It is not OK to initialize all the weights of a neural network to the same value. Because the initial parameters need to break symmetry between different units. For example, two hidden units with the same activation function connected to the same inputs must have different initial parameters to compute a different function. If the weights are the same, the output will also be the same, and training will serve no purpose. Therefore, the weights should not be initialized the same.

It is OK to initialize the bias terms to 0. Because symmetry is broken by the random weights. And in normal cases, we actually initialize the bias with 0.

2 In which cases would you want to use each of the following activation functions: ELU, leaky ReLU, ReLU, tanh, logistic, and softmax?

In general, considering the accuracy, logistic $<$ tanh $<$ ReLU $<$ leaky ReLU $<$ ELU. However, when considering the runtime performance, leaky ReLU works better than ELU. Then we consider each activation function separately.

- ELU: It tends to produce more accurate results. However, the time of training may be longer. Therefore, we generally choose it when we care more about the accuracy.
- leaky ReLU: It solves the problem of dying ReLU. But it can't be used for the complex Classification. And we should choose it when we care more about the runtime efficiency.
- ReLU: It solves vanishing gradient problem, meaning that we can use it in deep feed forward neural networks. And it is less computationally expensive compared with tanh, logistic and softmax. But it has the problem of dying ReLU, which is that some neurons stop outputting anything other than 0. And it is usually used within hidden layers of a neural network model.
- tanh: The gradient of tanh is stronger than sigmoid. It has the vanishing gradient problem. Its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity.
- logistic: It outputs the value between 0.0 and 1.0. And it is usually used in binary classification.
- softmax: It calculates the probabilities of each target class over all possible target classes. It is usually used in multiclass classification.

3 What is batch normalization and why does it work?

It is a technique to address the problem that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. It makes the learning of layers in the network more independent of each other. It adds an operation in the model just before the

activation function of each layer. It zero-centers and normalizes the inputs, and then scales and shifts the result.

First, it estimates the inputs' mean and standard deviation.

$$\mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B} x^{(i)},$$

$$\sigma_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}(x^{(i)} - \mu_B)^2,$$

where m_B is the number of instances in the mini-batch.

Then, we can zero-center and normalize it with

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}},$$

where ϵ is a tiny number to avoid division by zero.

And we can scale and shift it by

$$z^{(i)} = \gamma\hat{x}^{(i)} + \beta,$$

where γ is the scaling parameter vector for the layer and β is the shifting parameter (offset) vector for the layer.

After the above process, it changes the distribution of the inputs, and makes the neural network faster and stable.

4 Does dropout slow down training? Does it slow down inference (i.e., making predictions on new instances)?

It does slow down training. When dropping some neuron, they will not be updated during the backpropagation process, thus the convergence is slowed down. Therefore, the training speed is slowed down.

It does not slow down the inference. During inference time, dropout is not applied, but we need to multiply the weights with $(1 - \textit{dropout rate})$ to make them close to that during training time. Therefore, the inference will not be slowed down.

5 Consider a CNN composed of three convolutional layers, each with 3 × 3 filters, a stride of 2, and SAME padding. The lowest layer outputs 100 feature maps, the middle one outputs 200, and the top one outputs 400. The input images are RGB images of 200 × 300 pixels. What is the total number of parameters w in the CNN?

The first layer : We have 3 channels, 3×3 filters, so we have $3 \times 3 \times 3 + 1 = 28$ weights; For 100 feature maps, we have $100 \times 28 = 2800$ parameters;

The second layer: We have $(3 \times 3 \times 100 + 1) \times 200 = 180200$ parameters;

The third layer: We have $(3 \times 3 \times 200 + 1) \times 400 = 720400$ parameters;

To sum up: $720400 + 180200 + 2800 = 903400$ weights;

6 **Consider a CNN with one convolutional layer, in which it has a 3 × 3 filter (Figure 1) and a stride of 2. Please write the output of this layer for the given input image (the left image in the following figure)?**

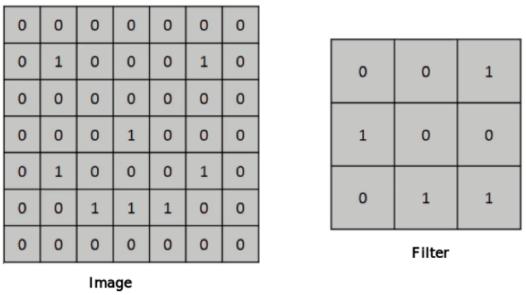

Figure 1: Filter for question 6

The output of the layer (Figure 2):

0	0	0
1	0	1
0	1	1

Figure 2: Answer for question 6