EN. 520. 650 HW4

Qihua Gong

1. a) the optimal policy should get to state 3 as soon as possible and pay less cost to reach state 1 and 2. But if both state 1 and 2 aim directly to state 3, it has low probability and large cost. It can try action B in state 1 and action A in state 2, it's a better way of cost.

1. b)

$u \leftarrow \langle -1, -2, 0 \rangle$, $P \leftarrow \langle b, b \rangle$

$u_1 = -1 + 0.2u_3 + 0.8u_1$, $u_2 = -2 + 0.2u_3 + 0.8u_2$, $u_3 = 0$

$u_1 = -5$, $u_2 = -10$

In state 1: $\sum_j T(1, a, j) u_j = 0.6 \times (-10) + 0.4 \times (-5) = -8$

$\sum_j T(1, b, j) u_j = 0.2 \times 0 + 0.8 \times -5 = -4$

action B is perfered

In state 2 : $\sum_j T(1, a, j) u_j = 0.6 \times (-5) + 0.4 \times (-10) = -7$

$\sum_j T(1, b, j) u_j = 0.2 \times 0 + 0.8 \times (-10) = -8$

action a is perfered

now $u_1 = -5$ and $u_2 = -2 + 0.6u_1 + 0.4u_2$, $u_3 = 0$

$0.6u_2 = -2 + 0.6 \times (-5)$

$u_2 = -8.3$

now state 1 : $\sum_j T(1, a, j) u_j = 0.6 \times -8.3 + 0.4 \times -5 = -6.98$

$\sum_j T(1, b, j) u_j = 0.2 \times 0 + 0.8 \times (-5) = -4$

action b is still perfered

state 2 : $\sum_j T(1, a, j) u_j = 0.6 \times (-5) + 0.4 \times (-8.3) = -6.32$

$\sum_j T(1, b, j) u_j = 0.2 \times 0 + 0.8 \times (-8.3) = -6.64$

action a is still perfered

so state 2 move to state 1 and in state 1 to move to state 3

c) it will change the form to : $u_1 = -1 + 0.4 u_1 + 0.6 u_2$

$$u_2 = -2 + 0.6 u_1 + 0.24 u_2$$

$$u_3 = 0$$

we can solve this function of u_1, u_2

the value will tend to $-\infty$

the discounting will not help. The discount factor will affect the policy and results. It will cause we can choose action b in state 2. The discount short-term cost will outweights the discount long-term cost of action b, and it will finally repeatedly leaveing the agent in state 2.

2. Alexnet: numbers of parameters: 61 M parameters

training and test times: seperate from different computers

reference: 90 epochs trained for 6 days on two GTX 580 GPU

performance: first use of ReLu, heavy data augmentation, dropout 0.5, 8 layers, 16.4 score on Imagenet board, more than 75% accuracy

VGG : numbers of parameters : 138 million parameters.

training and test times: for resnet 50 runs to 76% + top 1 accuracy on 90 epochs in roughly 7.3 hours on a v2-8 TPU device.

performance: 2nd in classification, 1 st in localization, has better feature generalization, 19 layers, 7.3 scores, around 70% accuracy but large operations. and parameters

Resnet: numbers of parameters: 11 million parameters.

training and test times: newe training time of 224 seconds with validation accuracy of 75.03% using Tesla V100 GPUs

performance: ultra-deep in Imagine net 152-layer nets, the deeper networks achieve lower training error, 73% accuracy, with 152 layers

googlenet: number of parameters: 7 million parameters.

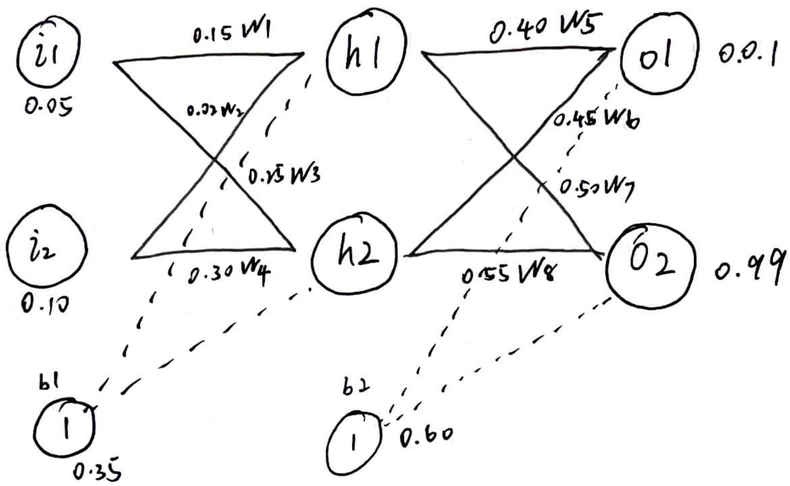
training and test times: close with the time as VGG, it needs to train 22 layers which more than vgg16

performance: 22 layers, 6.7 score in imagenet scale, 70% lower accuracy, less parameters and operations, is most efficient network architecture.

3. The idea of dropout is to removing neurons or input node in a neural network. The process searching the best dropout way is when in a large enough network, At each iteration step of the algorithm, We first retain each node together with it's income and outcome connections with probability P and remove the node with probability 1-P. Then train the remaining nodes according to the selected algorithm. So, in each step, a different subnetwork is trained. After the train is over, each parameter is multiplied by probability P and then one combines the trained subnetworks by an averaging rationale.

To find a best neural architecture, we need to find it in different use. A basic neural network need a input layer, a first hidden layer and second hidden layer. Each layer formed a different representation of input and different levels of abstraction. Usually we can use three layers to form a simple task but when meet more complex tasks, we need maybe various layers to learn the complex concepts. For more efficient, we expect compact representation to result in better generalization performance. Also, for a class of deep networks and target functions, one needs a substantially smaller number of nodes to achieve a predefind accuracy compared to a shallow one. So, the way to find a best neural network is base on the problem we need to solve. We should choose a network with approate layers, efficient parameters and quick runing time.

4.



first, forward pass

The net input for h_1 is calculated as the sum of the product of each weight value and input value and bias value

$$\text{net input } h_1 = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1$$
$$= 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35 \times 1 = 0.3775$$

then use sigmoid function to calculate the output of h_1

$$\text{out } h_1 = \frac{1}{1+e^{-net\ h_1}} = \frac{1}{1+e^{-0.3775}} \approx 0.5933$$

so $\text{input } h_2 = 0.25 \times 0.05 + 0.30 \times 0.1 + 0.35 \times 1 = 0.3925$

$$\text{out } h_2 = \frac{1}{1+e^{-0.3925}} = 0.59688$$

$$\text{input } O_1 = W_5 \times \text{out } h_1 + W_6 \times \text{out } h_2 + b_2 \times 1$$
$$= 0.4 \times 0.5933 + 0.45 \times 0.59688 + 0.6 \times 1 = 1.10591$$

$$\text{out } O_1 = \frac{1}{1+e^{-net\ o_1}} = \frac{1}{1+e^{-1.10591}} = 0.7514$$

same $\text{input } O_2 = 0.5 \times 0.5933 + 0.55 \times 0.59688 + 0.6 \times 1 = 1.2249$

$$\text{out } O_2 = \frac{1}{1+e^{-1.2249}} = \cancel{0.0235} \; 0.77293$$

Then calculate error $E_{total} = \sum \frac{1}{2}(\text{target} - \text{output})^2$

$$E_{o_1} = \frac{1}{2}(0.01 - 0.75136)^2 = 0.2748$$

$$E_{O_2} = \frac{1}{2}(0.99 - 0.77293)^2 = 0.02356$$

$$E_{total} = E_{o_1} + E_{O_2} = 0.27481 + 0.02356 = 0.29837$$

→ next

finally Use the back propagation to update each weight in the network.

update $w_5^* = w_5 - n \times \left(\frac{\partial E\ total}{\partial w_5}\right)$

the rate of change error

$$\frac{\partial E total}{\partial w_5} = \frac{\partial E total}{\partial out\ o1} \times \frac{\partial out\ o1}{\partial net\ o1} \times \frac{\partial net\ o1}{\partial w_5}$$

$$\frac{\partial E total}{\partial out\ o1} = -(target\ o1 - out\ o1) = -(0.01 - 0.75136) = 0.741365$$

$$\frac{\partial out\ o1}{\partial net\ o1} = out\ o1\ (1 - out\ o1) = 0.751365 \times (1 - 0.751365) = 0.18682$$

$$\frac{\partial net\ o1}{\partial w_5} = 1 \times out\ h1 \times w_5^{(1-1)} + 0 + 0 = out\ h1 = 0.59326$$

$$\therefore \frac{\partial E total}{\partial w_5} = 0.741365 \times 0.18682 \times 0.59326 = 0.082167$$

update $w_5^* = 0.4 - 0.5 \times 0.082167 = 0.35892$

as the same level backward. $w_6^* = 0.408666$

$w_7^* = 0.511301$

$w_8^* = 0.561370$

then go to next nevel layer, start with w_1

$$w_1^* = w_1 - n \times \left(\frac{\partial E\ total}{\partial w_1}\right)$$

$$\frac{\partial E total}{\partial w_1} = \frac{\partial E total}{\partial out\ h_1} \times \frac{\partial out\ h_1}{\partial net\ h_1} \times \frac{\partial net\ h_1}{\partial w_1}$$

$$\frac{\partial E total}{\partial out\ h_1} = \frac{\partial E_{o_1}}{\partial out\ h_1} + \frac{\partial E_{o_2}}{\partial out\ h_1} = \frac{\partial E_{o_1}}{\partial net\ o1} \times \frac{\partial net\ o1}{\partial out\ h_1} + \frac{\partial E_{o1}}{\partial net\ o1} \times \frac{\partial net\ o1}{\partial out\ h_1}$$

$= 0.741365 \times 0.186815 + 0.1384985 \times 0.40$

$=$ [illegible]

$0.0553994 25 + (-0.0190491) = 0.03635$

$$\frac{\partial out\ h1}{\partial net\ h1} = out\ h1\ (1 - out\ h1) = 0.593269\ (1 - 0.59327) = 0.2413$$

$$\frac{\partial net\ h1}{\partial w_1} = \cancel{net\ h}\ i_1 = 0.05$$

$$\therefore w_1^* = w_1 - n \times \frac{\partial E total}{\partial w_1} = 0.15 - 0.5 \times 0.03635 \times 0.2413 \times 0.05$$

$= 0.15 - 0.5 \times 0.0004385 = 0.14978$

as the same $w_2^* = 0.1995 6143$

$w_3^* = 0.2497 5114$

$w_4^* = 0.29950229$

addition:

$$W_7^* = W_7 - n \times \left(\frac{\partial E_{total}}{\partial W_7}\right)$$

$$\frac{\partial E_{total}}{\partial W_7} = \frac{\partial E_{total}}{\partial out\ O_2} \times \frac{\partial Out\ O_2}{\partial net\ O_2} \times \frac{\partial net\ O_2}{\partial W_7}$$

$$\frac{\partial E_{total}}{\partial out O_2} = -(target\ O2 - out\ O_2) = -(0.99 - 0.77293) = -0.21707$$

$$\frac{\partial Out O_2}{\partial net O_2} = out O_2 (1 - out O_2) = 0.77293(1 - 0.77293) = 0.17551$$

$$\frac{\partial net O_2}{\partial W_7} = 1 \times out\ h_1 \times W_7^{(1-1)} + 0 + 0 = out h_1 = 0.59327$$

$$W_7^* = 0.5 - 0.5 \times (-0.21707 \times 0.17551 \times 0.59327) = 0.511301$$

$$W_3^* = W_3 - n \times \frac{\partial E_{total}}{\partial W_3}$$

$$\frac{\partial E_{total}}{\partial W_3} = \frac{\partial E_{total}}{\partial out\ h_2} \times \frac{\partial out h_2}{\partial net h_2} \times \frac{\partial net h_2}{\partial W_3}$$

$$\frac{\partial E_{total}}{\partial out h_2} = \frac{\partial E_{O_1}}{\partial out h_2} + \frac{\partial E_{O_2}}{\partial out\ h_2}$$

$$= \frac{\partial E_{O1}}{\partial net\ O_2} \times \frac{\partial net O_2}{\partial out h_2} + \frac{\partial E_{O_2}}{\partial net O_2} \times \frac{\partial net\ O_2}{\partial out h_2}$$

$$= \frac{\partial E_{O1}}{\partial out h_2} \times \frac{\partial out h_2}{\partial net O_2} \times \frac{\partial net O_2}{\partial out h_2} + \frac{\partial E_{O_2}}{\partial out h_2} \times \frac{\partial out h_2}{\partial net O_2} \times \frac{\partial net O_2}{\partial out h_2}$$

$$= -0.21707 \times 0.17551 \times 0.55 + -0.21707 \times 0.17551 \times 0.55 = 0.041371$$

$$\frac{\partial out h_2}{\partial net h_2} = out h_2 (1 - out h_2) = (0.59688) \times (1 - 0.59688) = 0.240614$$

$$\frac{\partial net h_2}{\partial W_3} = i_1 = 0.05$$

$$\therefore W_3^* = 0.25 - 0.5 \times 0.041371 \times 0.240614 \times 0.05 = 0.24975114$$