Reading Report

2019.3.26

1 Lenet5

1.1 convolution

filters point multiplication the original imagine matrix and abstract features of the picture(like identity, eage detectiong or sharpen), which can be seemed as feature detectors. $x_{i,j}$ means the element of imagine the i th row and the jth column, $w_{m,n}$ means the element of filter the m th row and the n th column. w_b means the bias of filter. F means the width of the filter, f means activation function.

$$a_{i,j} = f(\sum_{m=0}^{F-1}\sum_{n=0}^{F-1} w_{m,n}x_{i+m,j+n} + w_b)$$

if the depth >1,

$$a_{i,j} = f(\sum_{d=0}^{D-1}\sum_{m=0}^{F-1}\sum_{n=0}^{F-1} w_{d,m,n}x_{d,i+m,j+n} + w_b)$$

1.2 activation function

ReLU like

(1) $Relu = Max\{x, 0\}$
ReLU stands for Rectified Linear Unit and is a non-linear operation. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear
(2) $LeakyReLU = max(x, \alpha x), \alpha \sim 0.01$
In case of dead neuro ReLU may cause.
(3) $VeryLeakyReLU = max(x, \alpha x), \alpha \sim \{0.1, 0.5\}$
introduce sparsity into the convolutional kernels by learning a basis of kernels, or by PCA

decompose which also treat channel as variables
(4) $RandomRelu = max(x, \alpha x), \alpha = random\{0.1, 0.5\}$
some inference about ReLU initialization
(5) $adaptivePiecewiseLinearUnitsy = max(x, 0) + \Sigma_{s=1}^{S} a_i^s max(0, -x + b_i^s)$
Designed a novel form of piecewise linear activation function that is learned independently for each neuron using gradient descent.

sigmoid like

(1)$sigmoid(x) = 1/(1 + e^{-x})$
sigmoid function has a special character $s'(x) = s(x) * (1 - s(x))$, but it has two problems.
1.the $s(x)$ is only positive or negative
2.gradient descent, if the depth of network is large, the gradient will become 0.
3.the amount of calculation is large
(2)$tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
faster than sigmoid and the center is 0.

1.2.1 softmax

softmax $S_i = \frac{e_i}{\Sigma_j e_j}$
used for multiclass problem, and the output number of a neuron is more than one.

1.3 The introduce

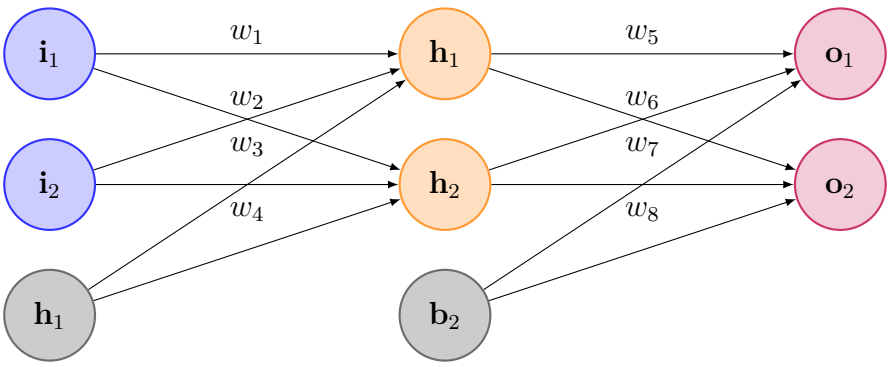

图 1: a basic 2 layer neuron network assumed

we structure a 3 layer neuron network. And I try to introduce the BackPropagation
1.the first layer is input layer

$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$

2.the second layer is hidden layer

$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1$ the output of second layer

$out_{h1} = activityfuction(net_{h1})$

now we seems $activityfuction(net_{h1})$ as $Act(net_{h1})$

square error: $E_{total} = \Sigma(target - out)^2$

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_5} &= \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5} \\ &= \frac{\partial \Sigma(target - out)^2}{\partial out_{o1}} * Act\text{'}(net_{o1}) * out_{h1} \\ &= (out_{o1} - target_{o1}) * Act\text{'}(net_{o1}) * out_{h1}\end{aligned}$$

if activity function is sigmoid, $Act'(net_{o1}) = out_{h1} * (1 - out_{h1} \leq 1$,so we can see that sigmoid function may cause gradient disappeared.

if activity function is ReLU,$Act'(net_{o1}) = \{0, 1\}$,so ReLU can increase the sparsity of data in case of overfitting

if activity function is Leaky ReLU,$Act'(net_{o1}) = \{\alpha, 1\}$, if ReLU cause too many dead neurons, we can try Leaky ReLU, and it is used well in acoustic system.

1.4 Pooling or Sub Sampling

(1)max pooling

max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks

(2)average pooling

(3)Stochastic pooling

(4)LP-Norm pooling

(5)Tree-Gated pooling

(6)sum of average and max pooling

compare to (1)removing each pooling layer and increase the stride of the convolutional layer that preceded it accordingly or (2)replacing the pooling layer by a normal convolution with stride larger than one. pooling layer should extract the mathematical features we need

1.5 Fully Connected layer

Every neuron in the previous layer is connected to every neuron on the next layer. now it can be changed by Global Average Pooling.

2 Novelty-Map

2.1 parameterless SOM

we always use SOM to calssify 2 dimensional space and SOM only impact the active neuron, and the learning rate always the same.
While parameterless SOM interact the other cells when the winning cell is decided. And the longer the distance of winning cell and the cell, the smaller the influence.

$$Argmin\|x - w_i\|$$

$$\triangle w_i(t) = \varepsilon(t) h_{i,c}(x(t) - w_i(t))$$

εalways decrease with time, and the $h_{i,c}$decrease with distance.
For example
$\varepsilon(t) = 0.1 * (0.99999)^t$
$h_{i,c} = e^{dist(i,c)^2}$

2.2 Novelty-Map

3 My questions and opinion

1.Actually I do not understand clearly about Novelty-Map, only consider it as
1.Select the nearest cell, one of its individuals chosen for activation(a cell is made up of many individuals)
2.perform the output of the chosen indiviaual's neural network.
3.the next is just like Q-learning but I am not sure.
4.accumulates the rewards
But I wonder if it will rely too much on original cells set.
2.Next I want to read some about dropout and the paper"Network In Network"
3.I have tried to imitate and build Lenet. But tensorflow seems special for storage. Once I complete LeNet, I will report.
4.I think I still need to read more about parameter design like the depth and convolution

size.