COMP9444 Final A

Yihan Xiao 2018

Section1. Perceptron and Backpropagation

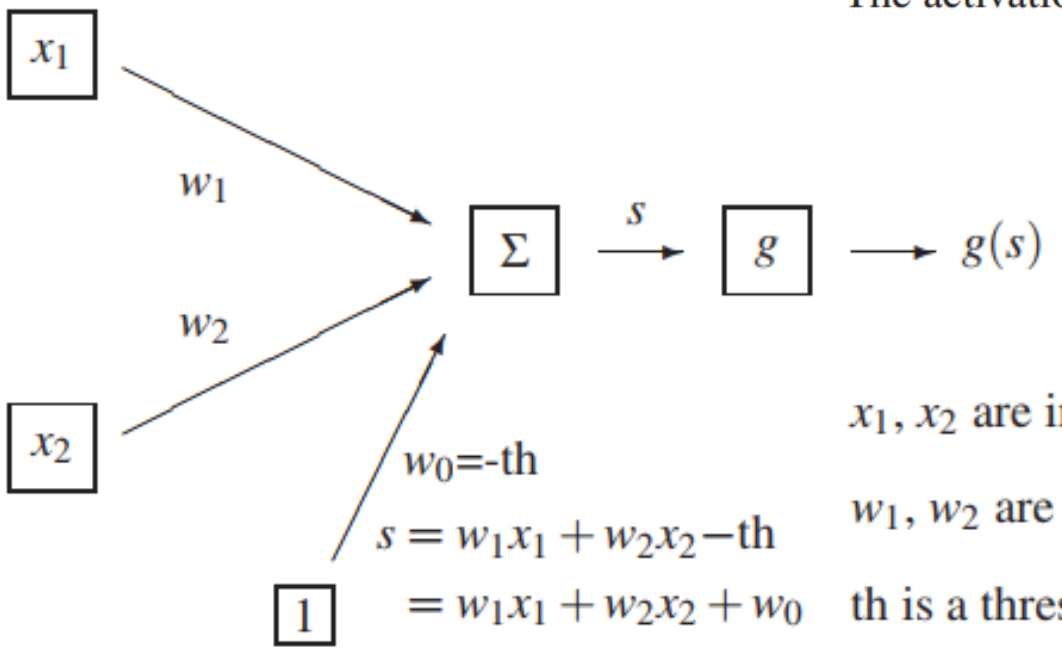
- Structure of perceptron

Weights can be positive or negative and may change over time (learning).

The input function is the weighted sum of the activation levels of inputs.

The activation level is a non-linear transfer function g of this input:

$$\text{activation}_i = g(s_i) = g(\sum_j w_{ij} x_j)$$



x_1, x_2 are inputs

w_1, w_2 are synaptic weights

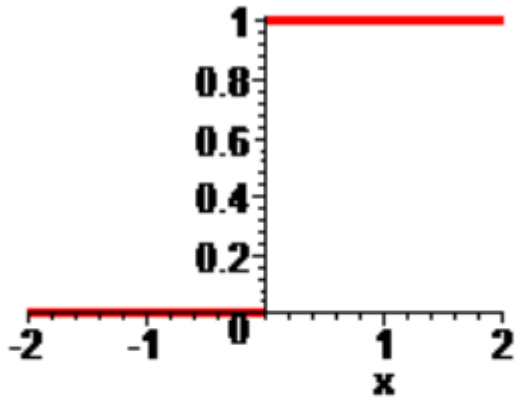
th is a threshold

w_0 is a **bias** weight

g is transfer function

Section1. Perceptron and Backpropagation

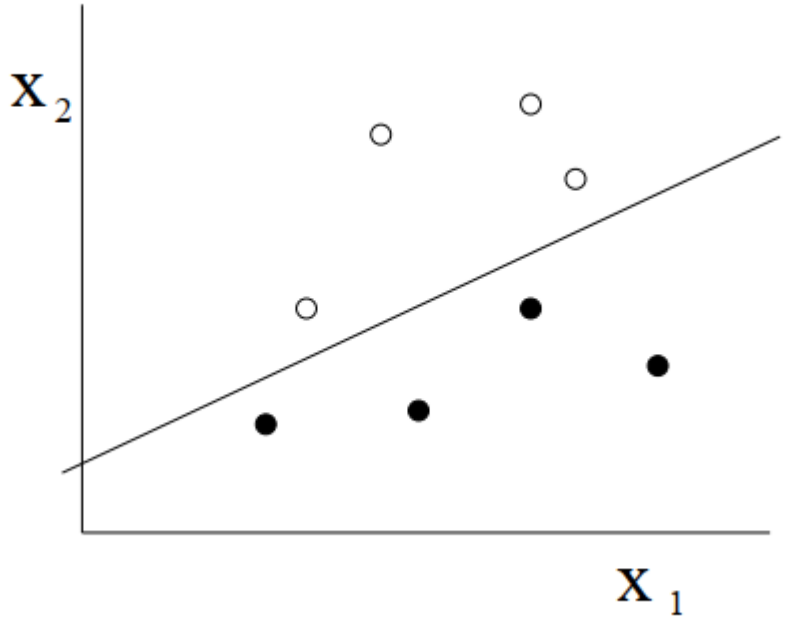
- Transfer function and linear separability



$$g(s) = \begin{cases} 1, & \text{if} \quad s \geq 0 \\ 0, & \text{if} \quad s < 0 \end{cases}$$

Step function

Question: what kind of functions can a perceptron compute?



Answer: linearly separable functions

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule

Adjust the weights as each input is presented.

recall: $s = w_1 x_1 + w_2 x_2 + w_0$

if $g(s) = 0$ but should be 1,

$$\begin{aligned} w_k &\leftarrow w_k + \eta x_k \\ w_0 &\leftarrow w_0 + \eta \end{aligned}$$

$$\text{so} \quad s \leftarrow s + \eta\left(1 + \sum_k x_k^2\right)$$

if $g(s) = 1$ but should be 0,

$$\begin{aligned} w_k &\leftarrow w_k - \eta x_k \\ w_0 &\leftarrow w_0 - \eta \end{aligned}$$

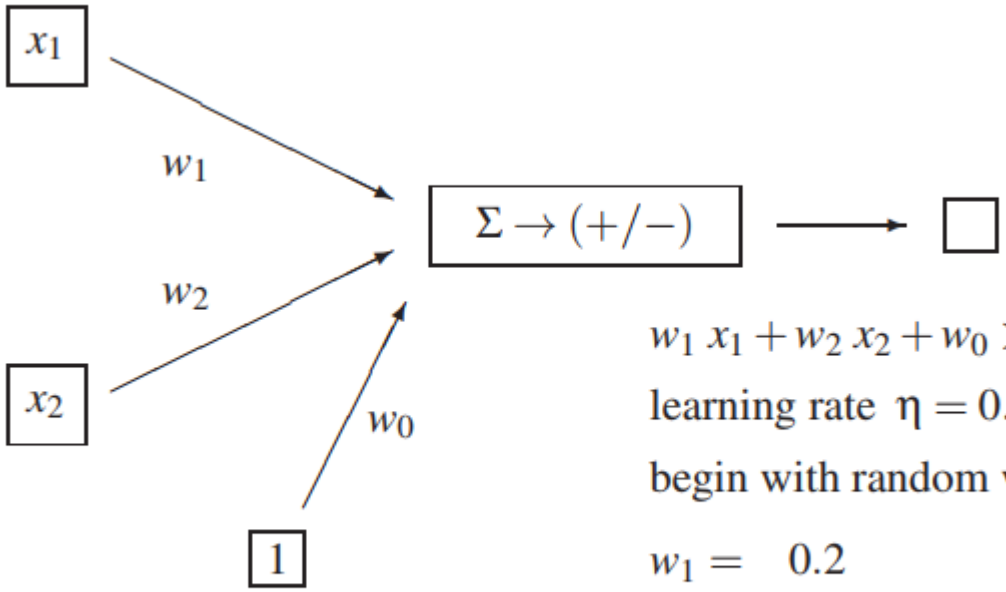
$$\text{so} \quad s \leftarrow s - \eta\left(1 + \sum_k x_k^2\right)$$

注意后面是加还是减是差值的正负决定的

$$w_k = w_k + \eta(y - \hat{y})x_k$$

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule - example



$w_1 x_1 + w_2 x_2 + w_0 > 0$

learning rate $\eta = 0.1$

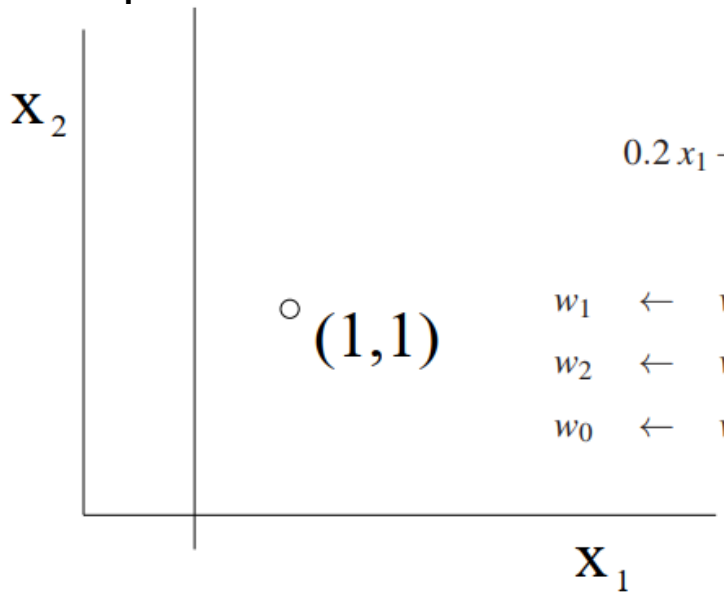
begin with random weigh

$w_1 = 0.2$

$w_2 = 0.0$

$w_0 = -0.1$

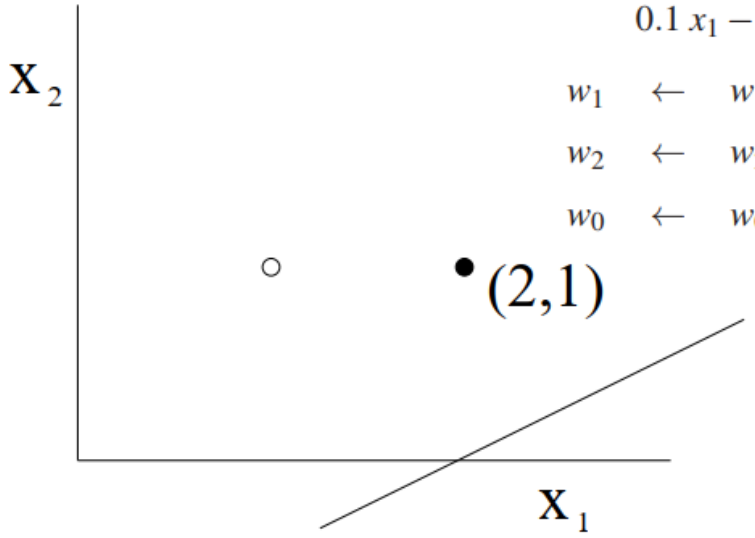
Step1



$0.2\,x_1 + 0.0\,x_2 - 0.1 > 0$

$$\begin{aligned} w_1 &\leftarrow w_1 - \eta\, x_1 = 0.1 \\ w_2 &\leftarrow w_2 - \eta\, x_2 = -0.1 \\ w_0 &\leftarrow w_0 - \eta = -0.2 \end{aligned}$$

Step2



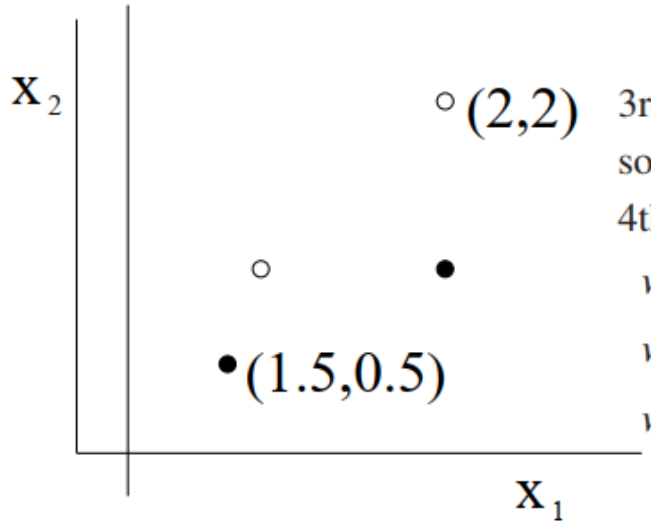
$0.1\,x_1 - 0.1\,x_2 - 0.2 > 0$

$$\begin{aligned} w_1 &\leftarrow w_1 + \eta\, x_1 = 0.3 \\ w_2 &\leftarrow w_2 + \eta\, x_2 = 0.0 \\ w_0 &\leftarrow w_0 + \eta = -0.1 \end{aligned}$$

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule - example

Step3



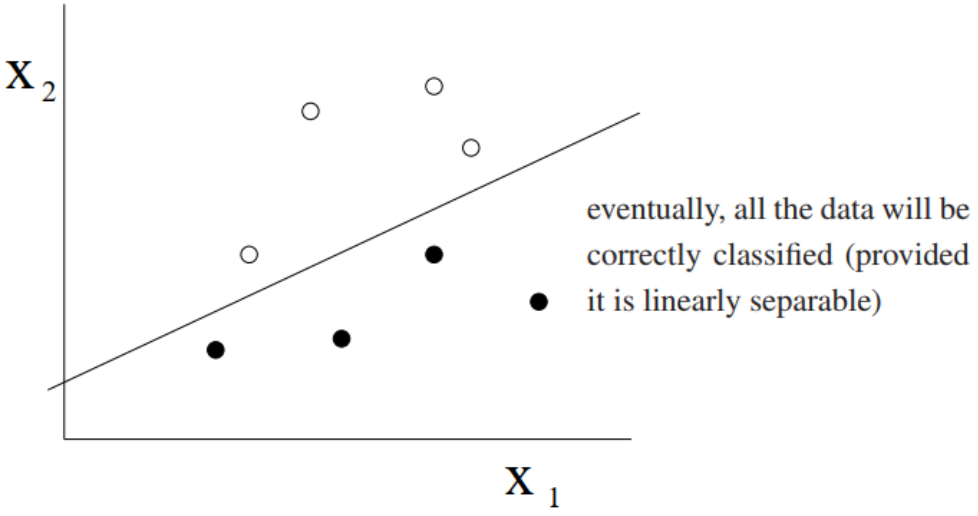
$$0.3\,x_1 + 0.0\,x_2 - 0.1 > 0$$

3rd point correctly classified, so no change

4th point:

$$\begin{aligned} w_1 &\leftarrow w_1 - \eta\,x_1 = 0.1 \\ w_2 &\leftarrow w_2 - \eta\,x_2 = -0.2 \\ w_0 &\leftarrow w_0 - \eta = -0.2 \end{aligned}$$

$$0.1\,x_1 - 0.2\,x_2 - 0.2 > 0$$

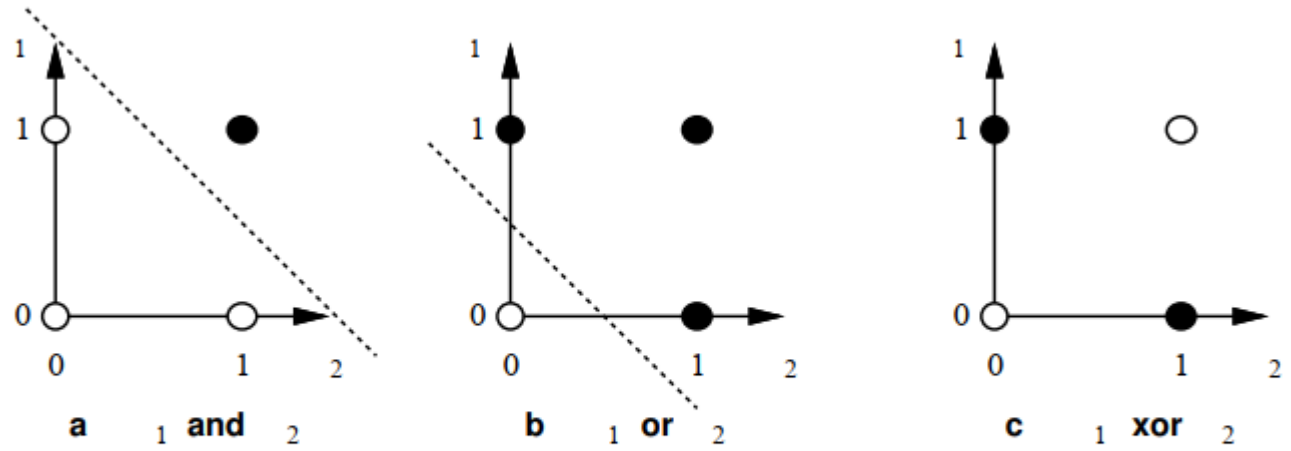


eventually, all the data will be correctly classified (provided it is linearly separable)

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule - Limitation

Problem: many useful functions are not linearly separable (e.g. XOR)



我们可以通过不断叠加perceptron的方式来解决线性不可分的问题

Possible solution:

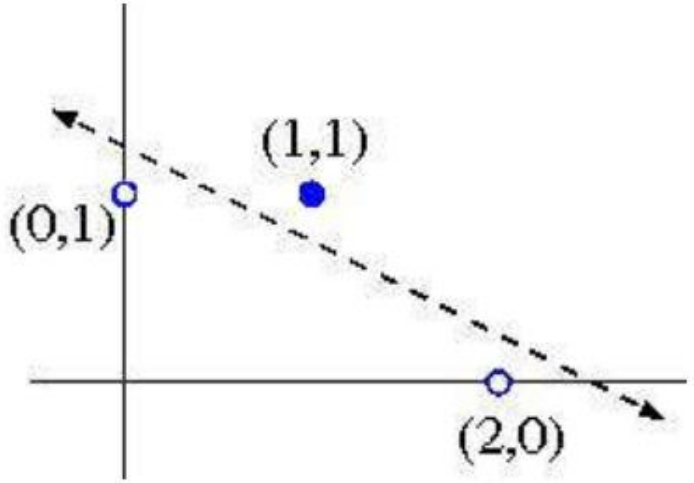
x_1 XOR x_2 can be written as: (x_1 AND x_2) NOR (x_1 NOR x_2)

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule – exercise 1

a. Construct by hand a Perceptron which correctly classifies the following data; use your knowledge of plane geometry to choose appropriate values for the weights w_0, w_1 and w_2.

Training Example	x_1	x_2	Class
a.	0	1	-1
b.	2	0	-1
c.	1	1	+1



求解方式：
先求出两个点之间的直线表达式，平移到第三个点求出第二根直线的表达式（斜率相同，只是常量b不同），再把两个常量相加求平均即可

This line has slope -1/2 and x_2-intersect 5/4, so its equation is:

$x_2 = 5/4 - x_1/2$,
i.e. $2x_1 + 4x_2 - 5 = 0$.

$w_0 = -5$
$w_1 = 2$
$w_2 = 4$

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule – exercise 1

b. Demonstrate the Perceptron Learning Algorithm on the above data, using a learning rate of 1.0 and initial weight values of

$w_0 = -1.5$
$w_1 = 0$
$w_2 = 2$

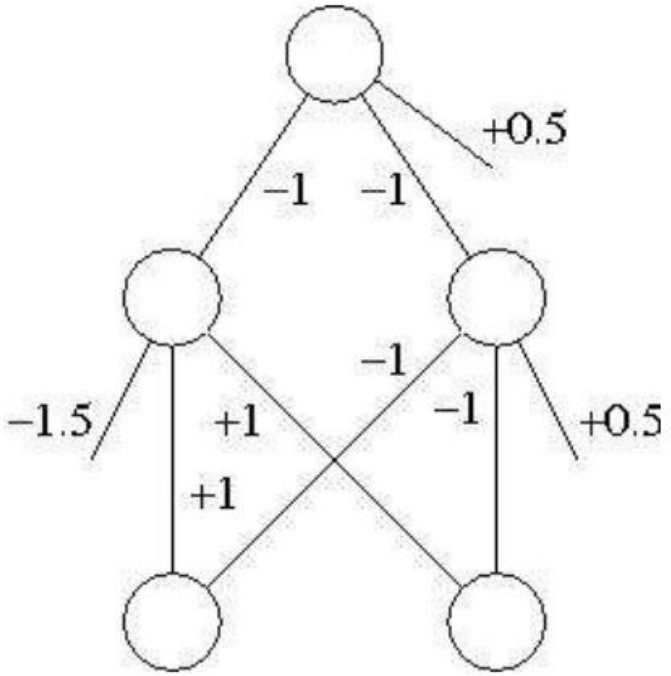
Iteration	w_0	w_1	w_2	Training Example	x_1	x_2	Class	$s=w_0+w_1x_1+w_2x_2$	Action
1	-1.5	0	2	a.	0	1	-	+0.5	Subtract
2	-2.5	0	1	b.	2	0	-	-2.5	None
3	-2.5	0	1	c.	1	1	+	-1.5	Add
4	-1.5	1	2	a.	0	1	-	+0.5	Subtract
5	-2.5	1	1	b.	2	0	-	-0.5	None
6	-2.5	1	1	c.	1	1	+	-0.5	Add
7	-1.5	2	2	a.	0	1	-	+0.5	Subtract
8	-2.5	2	1	b.	2	0	-	+1.5	Subtract
9	-3.5	0	1	c.	1	1	+	-2.5	Add
10	-2.5	1	2	a.	0	1	-	-0.5	None
11	-2.5	1	2	b.	2	0	-	-0.5	None
12	-2.5	1	2	c.	1	1	+	+0.5	None

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule – exercise 1

2. **XOR Network**

Construct by hand a Neural Network (or Multi-Layer Perceptron) that computes the XOR function of two inputs. Make sure the connections, weights and biases of your network are clearly visible.



常见的几种perceptron参数建议大家熟记

AND	$w_1 = w_2 = 1.0,$	$w_0 = -1.5$
OR	$w_1 = w_2 = 1.0,$	$w_0 = -0.5$
NOR	$w_1 = w_2 = -1.0,$	$w_0 = 0.5$

Section1. Perceptron and Backpropagation

- Perceptron Learning Rule – exercise plus

  Try to design a multilayer perceptron which has 3 inputs: x ,y and z.
  The output is 1 only when all 3 inputs are 0.

Section2. Backpropagation and Variations

- Learning Types

Supervised Learning 有监督学习

agent is presented with examples of inputs and their target outputs

Unsupervised Learning 无监督学习

agent is only presented with the inputs themselves, and aims to find structure in these inputs

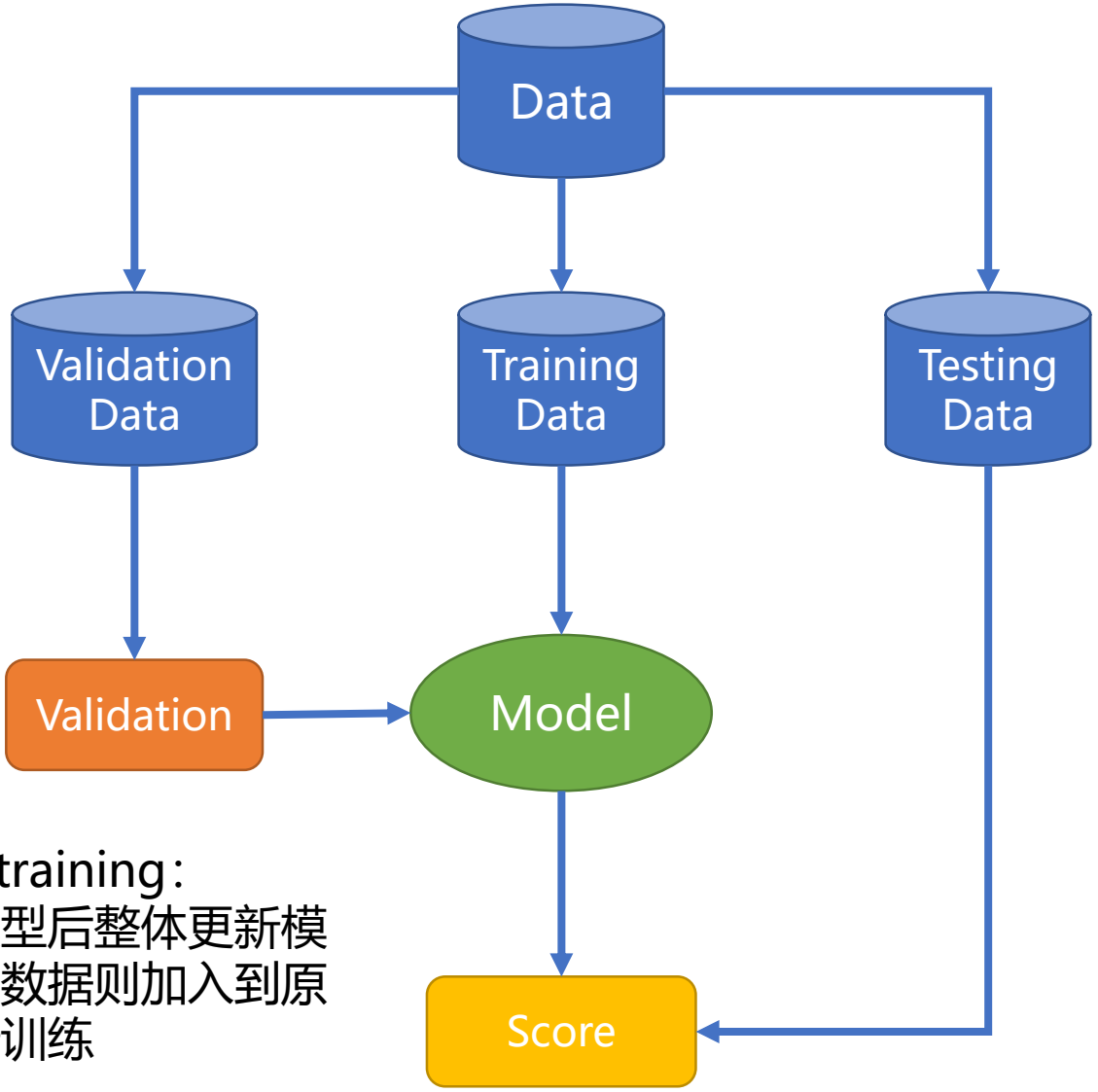
Reinforcement Learning 强化学习

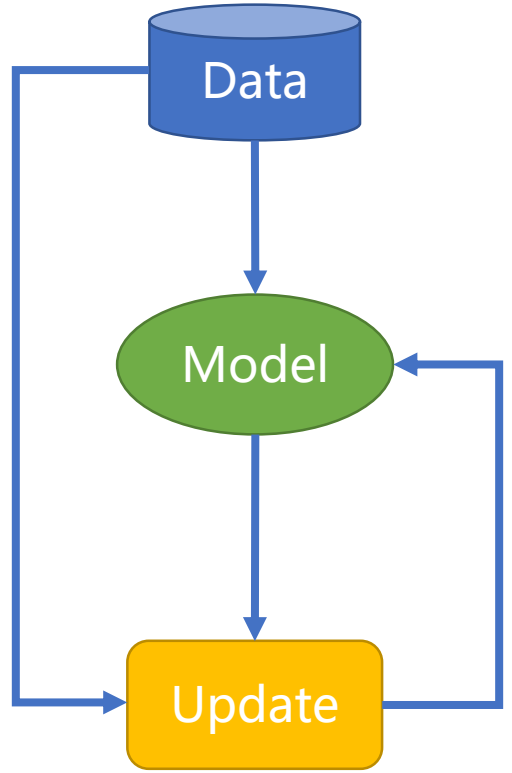
agent is not presented with target outputs, but is given a reward signal, which it aims to maximize

Section2. Backpropagation and Variations

一般常用的训练方式为
Offline(Batch) training

- Supervised Learning





Offline(Batch) training:
所有数据进入模型后整体更新模型，如果有新的数据则加入到原来的数据中重新训练

Online training:
每个数据进入模型后实时更新模型参数

Section2. Backpropagation and Variations

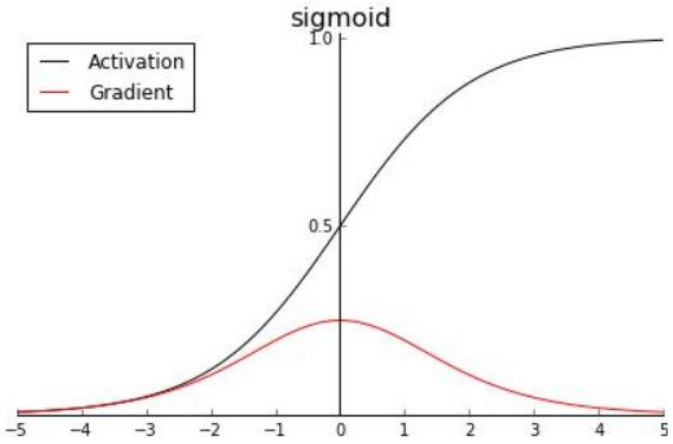
- Supervised Learning – Error function and Activation

Square Error $E = \frac{1}{2}\sum(z-t)^2$

Sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Sigmoid类型的函数梯度会消失

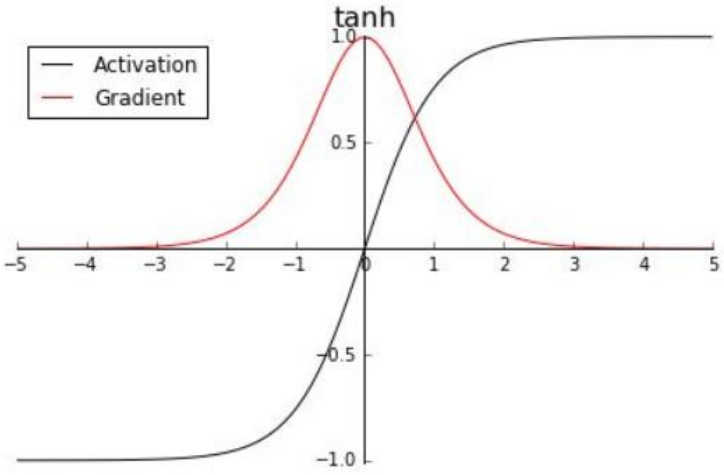


sigmoid及其梯度（红色曲线为梯度）

tanh

$$tanh(x) = 2\,sigmoid(2x) - 1$$

$$\frac{2}{1+e^{-2x}} - 1$$

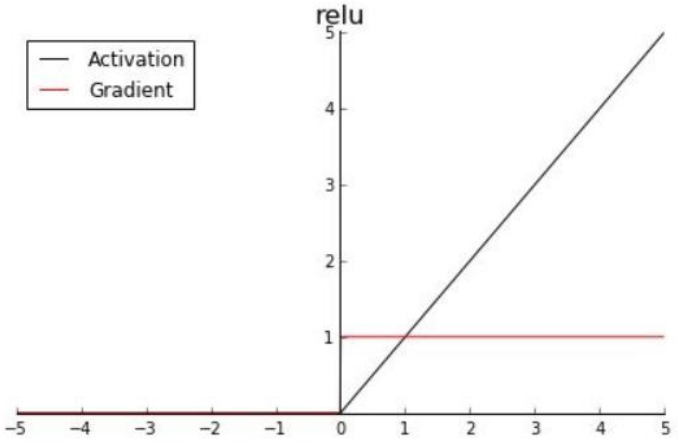


tanh及其梯度（红色曲线为梯度）

ReLU

$$\sigma(z) = max(0, z)$$

ReLU很好的保持了梯度，
可以避免梯度消失



ReLU及其梯度（红色折线为梯度）

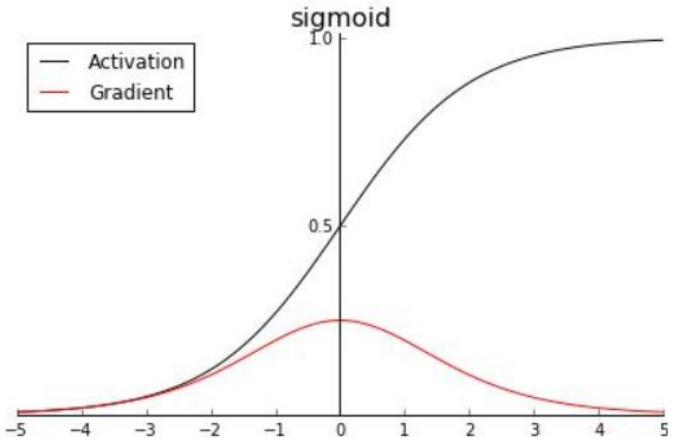
Section2. Backpropagation and Variations

- Supervised Learning – Error function and Activation

Square Error $E = \frac{1}{2}\sum(z-t)^2$

Sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

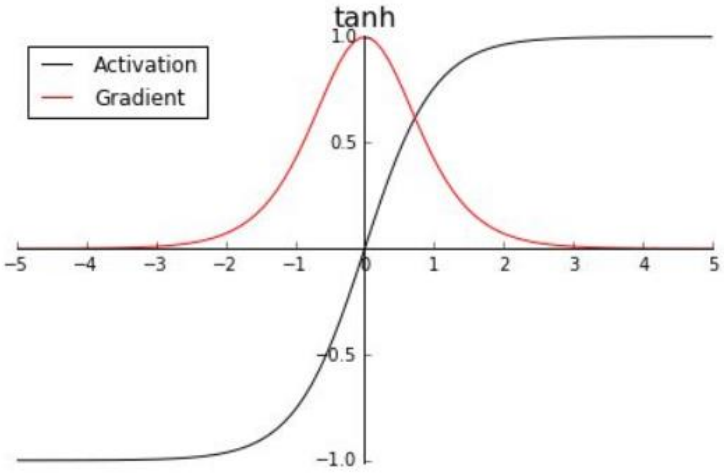


sigmoid及其梯度（红色曲线为梯度）

tanh

$$tanh(x) = 2\,sigmoid(2x) - 1$$

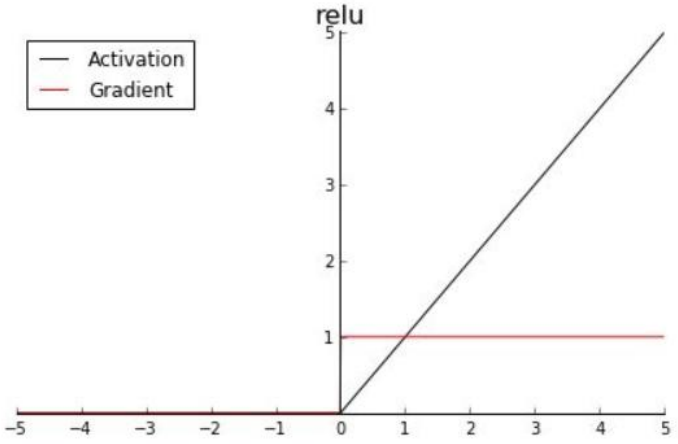
$$\frac{2}{1+e^{-2x}} - 1$$



tanh及其梯度（红色曲线为梯度）

ReLU

$$\sigma(z) = max(0, z)$$



ReLU及其梯度（红色折线为梯度）

Section2. Backpropagation and Variations

- Supervised Learning – Chain Rule

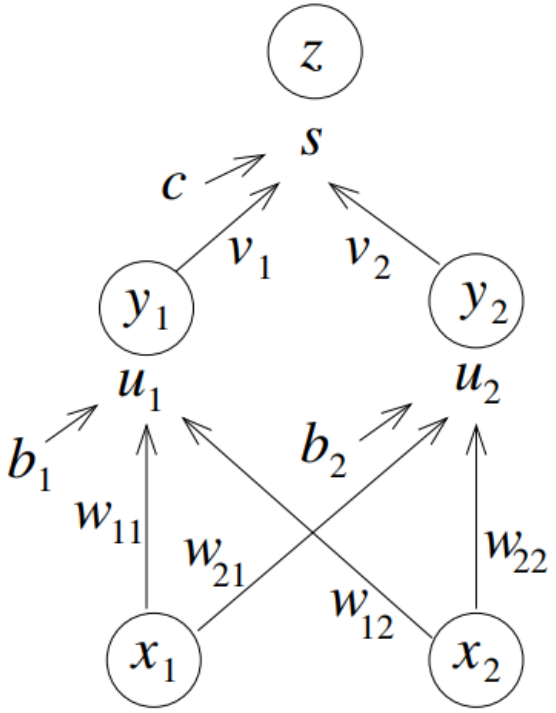
关键知识点

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

$$sigmoid' : z'(s) = z(1-z)$$
$$tanh' : z'(s) = (1-z^2)$$

For every parameter:

$$w_{new} = w_{old} + \Delta w$$



$$\begin{aligned} u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\ y_1 &= g(u_1) \\ s &= c + v_1 y_1 + v_2 y_2 \\ z &= g(s) \\ E &= \frac{1}{2}\sum (z-t)^2 \end{aligned}$$

Partial Derivatives

$$\begin{aligned} \frac{\partial E}{\partial z} &= z - t \\ \frac{dz}{ds} &= g'(s) = z(1-z) \\ \frac{\partial s}{\partial y_1} &= v_1 \\ \frac{dy_1}{du_1} &= y_1(1-y_1) \end{aligned}$$

Useful notation

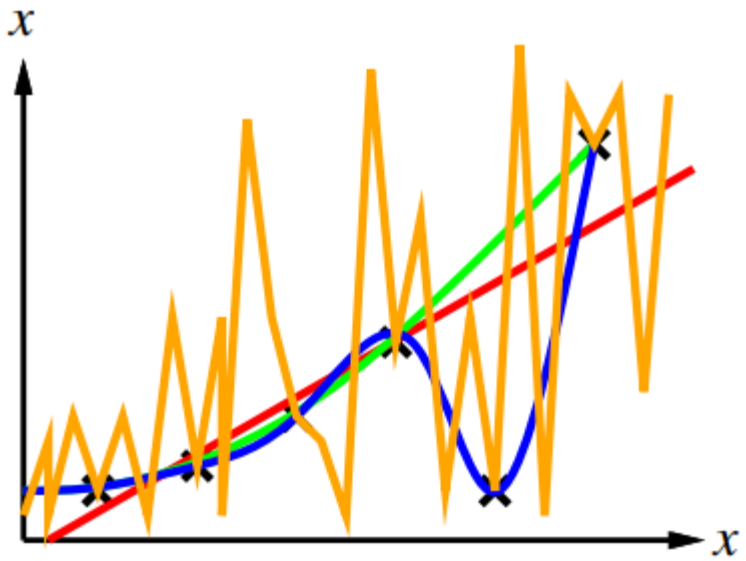
$$\delta_{out} = \frac{\partial E}{\partial s} \quad \delta_1 = \frac{\partial E}{\partial u_1} \quad \delta_2 = \frac{\partial E}{\partial u_2}$$

Then

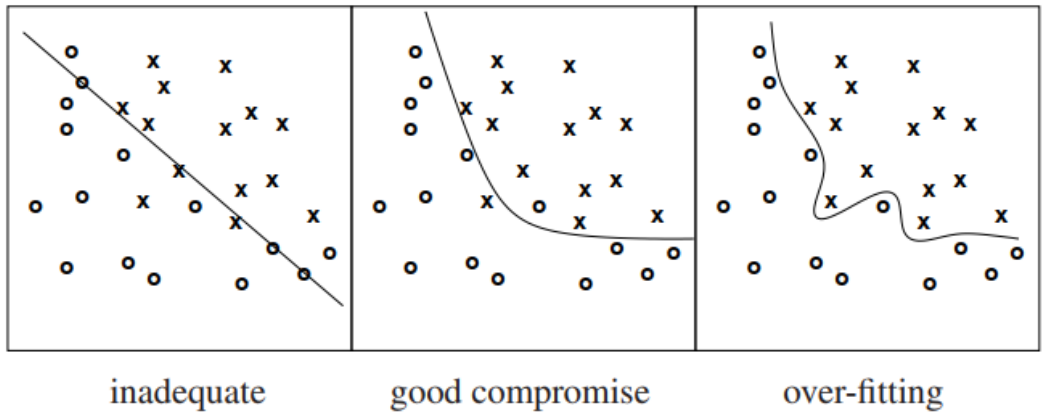
$$\begin{aligned} \delta_{out} &= (z-t)\, z\, (1-z) \\ \frac{\partial E}{\partial v_1} &= \delta_{out}\, y_1 \\ \delta_1 &= \delta_{out}\, v_1\, y_1\, (1-y_1) \\ \frac{\partial E}{\partial w_{11}} &= \delta_1\, x_1 \end{aligned}$$

Section2. Backpropagation and Variations

- Overfitting



Overfitting-过拟合是机器学习中常见的问题
我们在最后再总结避免overfitting的方法



Ockham Razor

The most likely hypothesis is the simplest one consistent with the data.

Section2. Backpropagation and Variations

- Supervised Learning – Chain Rule

Can we initialize all parameters to be zero? No!

this makes your model equivalent to a linear model.
When you set all weight to 0, the derivative with respect to loss function is the same for every w. Thus, all the weights have the same values in the subsequent iteration. This makes the hidden units symmetric and continues for all the n iterations you run. Thus setting weights to zero makes your network no better than a linear model.

Section2. Backpropagation and Variations

- Supervised Learning – Cross Entropy

For classification tasks, target t is either 0 or 1, so better to use

$$E = -t\log(z) - (1-t)\log(1-z)$$

注意这是0-1分布时的情况

如何理解 Entropy

事件A：巴西队进入了2018世界杯决赛圈。
事件B：中国队进入了2018世界杯决赛圈。

事件B的信息量比事件A的信息量要大。因为事件A发生的概率很大，事件B发生的概率很小。所以当越不可能的事件发生了，我们获取到的信息量就越大。越可能发生的事件发生了，我们获取到的信息量就越小。

信息量：时间发生概率的对数 $I(x_0) = -log(p(x_0))$

Entropy 熵：所有信息量的期望 $H(X) = -\sum_{i=1}^{n} p(x_i)log(p(x_i))$

Section2. Backpropagation and Variations

- Supervised Learning – Cross Entropy

$$D_{KL}(p||q) = \sum_{i=1}^{n} p(x_i)log(p(x_i)) - \sum_{i=1}^{n} p(x_i)log(q(x_i))$$
$$= -H(p(x)) + [-\sum_{i=1}^{n} p(x_i)log(q(x_i))]$$

KL散度表示两个分布p和q之间的差异，一般我们希望预测结果q和实际情况p之间的散度越小越好，也就是最小化公式右边的最后一项交叉熵

Section2. Backpropagation and Variations

- Supervised Learning – Cross Entropy. Example 1

假设有一个单分类问题

$$loss = -\sum_{i=1}^{n} y_i log(\hat{y}_i)$$

	猫	青蛙	老鼠
Label	0	1	0
Pred	0.3	0.6	0.1

$$\begin{aligned} loss &= -(0 \times log(0.3) + 1 \times log(0.6) + 0 \times log(0.1) \\ &= -log(0.6) \end{aligned}$$

一个batch里的loss为每个样本交叉熵的均值

$$loss = -\frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} y_{ji} log(\hat{y}_{ji})$$

Section2. Backpropagation and Variations

- Supervised Learning – Cross Entropy. Example 2

假设有一个多分类问题

	猫	青蛙	老鼠
Label	0	1	1
Pred	0.1	0.7	0.8

$$loss = -\sum_{i=1}^{n} y_i log(\hat{y}_i) \iff loss = -ylog(\hat{y}) - (1-y)log(1-\hat{y})$$ why

一个样本可能有多个输出标签，但每个标签满足0-1分分布

$$loss_{猫} = -0 \times log(0.1) - (1-0)log(1-0.1) = -log(0.9)$$
$$loss_{蛙} = -1 \times log(0.7) - (1-1)log(1-0.7) = -log(0.7)$$
$$loss_{鼠} = -1 \times log(0.8) - (1-1)log(1-0.8) = -log(0.8)$$

一个batch里的loss为每个样本的交叉熵的均值

$$loss = \sum_{j=1}^{m}\sum_{i=1}^{n} -y_{ji} log(\hat{y_{ji}}) - (1-y_{ji})log(1-\hat{y_{ji}})$$

所以一个样本的loss等于其所有可能标签的交叉熵之和

$$loss = loss_{猫} + loss_{蛙} + loss_{鼠}$$

要和单分类loss区分开来

Section2. Backpropagation and Variations

- Supervised Learning – Likelihood　　参数调整的核心思想

H is a class of hypotheses

$P(D|h)$ = probability of data D being generated under hypothesis $h \in H$.

$\log P(D|h)$ is called the likelihood.

ML Principle: Choose $h \in H$ which maximizes the likelihood,

i.e. maximizes $P(D|h)$　　[or, maximizes $\log P(D|h)$]

$$\log P(D|h) = \sum_{i=1}^{m} d_i \log h(x_i) + (1-d_i)\log(1-h(x_i))$$

$$h_{ML} = \text{argmax}_{h\in H} \sum_{i=1}^{m} d_i \log h(x_i) + (1-d_i)\log(1-h(x_i))$$

我们希望求得能使预测结果和实际样本差别最小的参数组合，这个过程就是最大似然，其中用于衡量差别大小标准的就是交叉熵

Section2. Backpropagation and Variations

- Conditional Probability – Bayes Rule

If we consider two random variables a and b, with $P(b) \neq 0$, then the conditional probability of a given b is

$$P(a|b) = \frac{P(a \wedge b)}{P(b)} \quad \Rightarrow \quad P(\text{Cause}|\text{Effect}) = \frac{P(\text{Effect}|\text{Cause})P(\text{Cause})}{P(\text{Effect})}$$

Alternative formulation: $P(a \wedge b) = P(a|b)P(b) = P(b|a)P(a)$

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
ca it	**.1 8**	**. 12**	**. 2**	**. 8**
¬ *ca it*	**. 16**	**. 64**	**.144**	**.5 6**

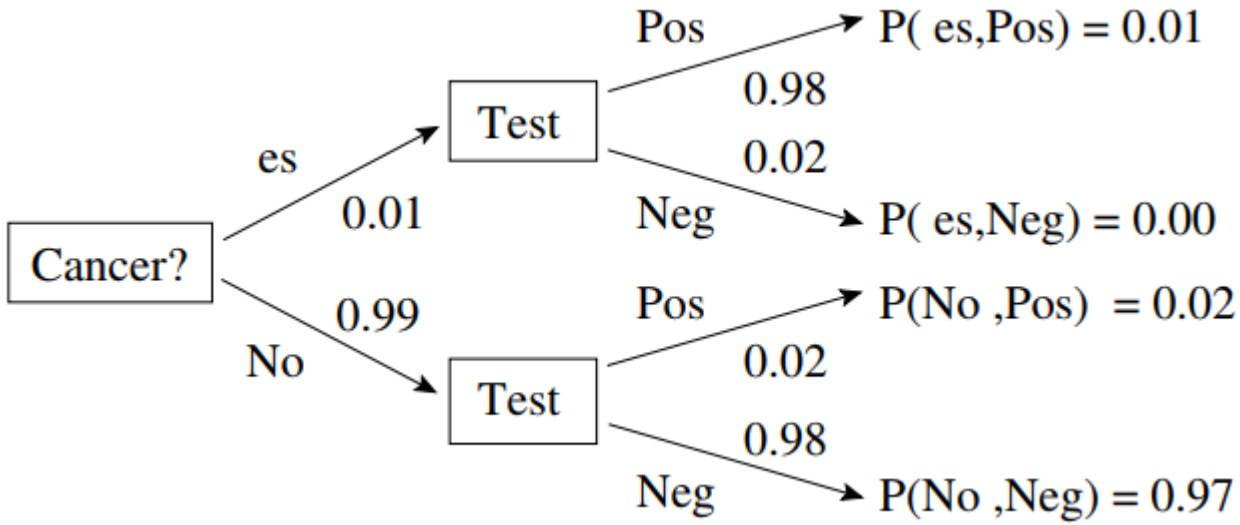
$$P(\neg\texttt{cavity}|\texttt{toothache}) = \frac{P(\neg\texttt{cavity} \wedge \texttt{toothache})}{P(\texttt{toothache})}$$

$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

Section2. Backpropagation and Variations

- Conditional Probability – Bayes Rule

Question: Suppose we have a 98% accurate test for a type of cancer which occurs in 1% of patients. If a patient tests positive, what is the probability that they have the cancer?



$$P(\texttt{cancer}\,|\,\texttt{positive}) = \frac{P(\texttt{positive}\,|\,\texttt{cancer})P(\texttt{cancer})}{P(\texttt{positive})}$$

$$= \frac{0.98 * 0.01}{0.98 * 0.01 + 0.2 * 0.99} = \frac{0.01}{0.01 + 0.02} = \frac{1}{3}$$

Section2. Backpropagation and Variations

- Conditional Probability – Bayes Rule. Exercise

One bag contains 2 red balls and 3 white balls. Another bag contains 3 red balls and 2 green balls. One of these bags is chosen at random, and two balls are drawn randomly from that bag, without replacement. Both of the balls turn out to be red. What is the probability that the first bag is the one that was chosen?

Let B = first bag is chosen, R = both balls are red. Then

$P(R \mid B) = (2/5)*(1/4) = 1/10$

$P(R \mid \neg B) = (3/5)*(2/4) = 3/10$

$P(R) = (1/2)*(1/10) + (1/2)*(3/10) = 1/5$

$P(B \mid R) = P(R|B)*P(B)/P(R) = (1/10)*(1/2)/(1/5) = 1/4$

Section2. Backpropagation and Variations

- Weight Decay

Assume that small weights are more likely to occur than large weights

为了避免参数在训练中变得过大，我们可以假设较小的参数值出现的概率较高

$$P(w) = \frac{1}{Z} e^{-\frac{\lambda}{2}\sum_j w_j^2}$$

把这项加入到整体cost上面，越大的参数带来的cost越高，这样系统就会尽量避免大参数的出现

$$E = \frac{1}{2}\sum_i (z_i - t_i)^2 + \frac{\lambda}{2}\sum_j w_j^2$$

Section2. Backpropagation and Variations

- Momentum

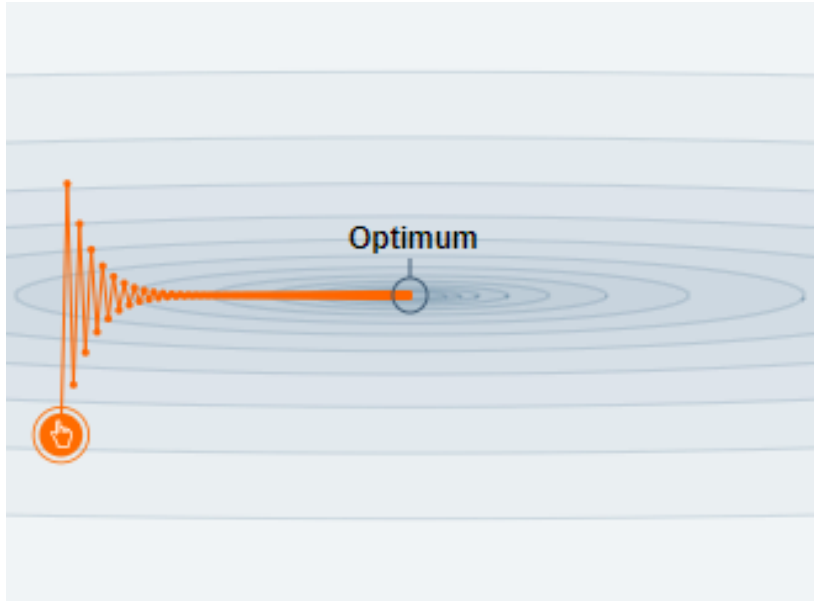
如果参数空间崎岖，训练过程可能出现频繁的侧向抖动 sideways oscillations，在某个区域徘徊而没有实质性进步，这时我们可以施加一个“惯性”，降低这种抖动的影响

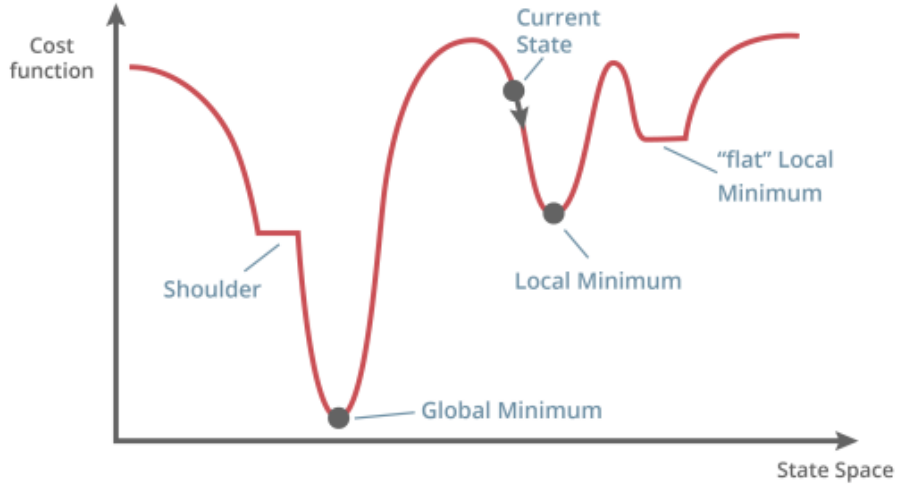
A running average of the differentials for each weight is maintained and used to update the weights

Momentum factor

$$\delta w \leftarrow \alpha \delta w + (1-\alpha)\frac{\partial E}{\partial w}$$

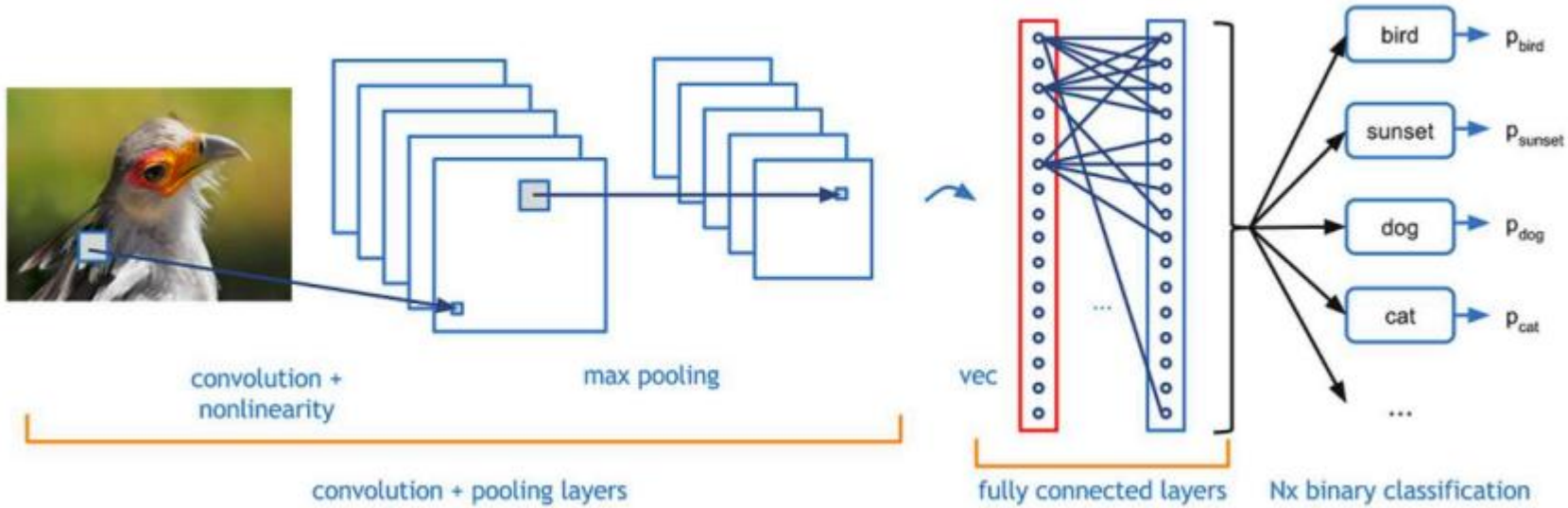
$$w \leftarrow w - \eta \delta w$$

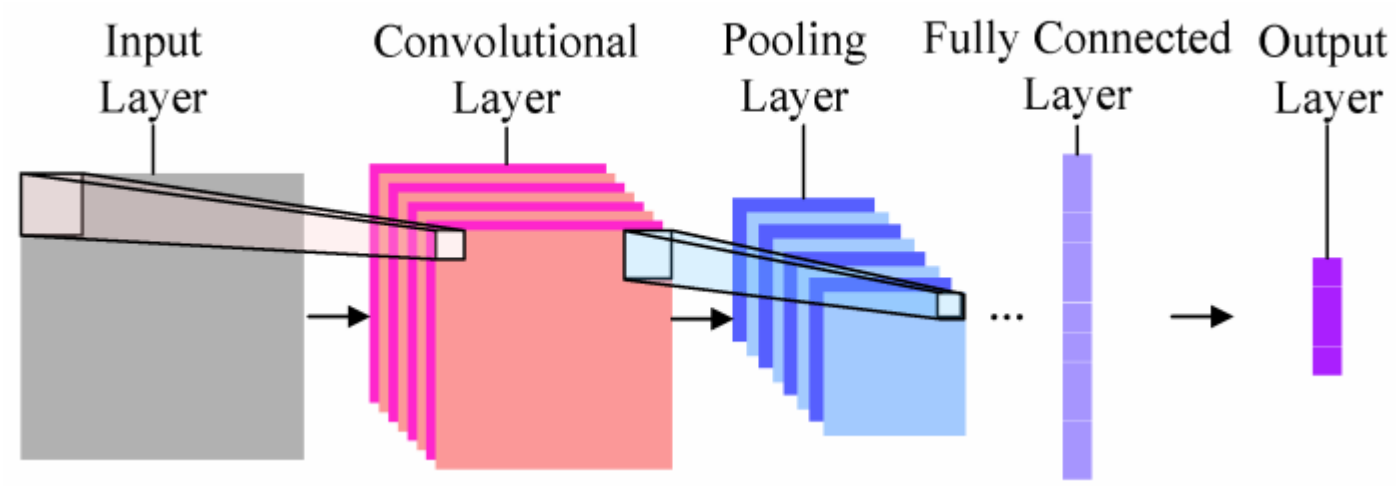




Section3. CNN and Image

- Introduction





Section3. CNN and Image

- Softmax

We assume the network's estimate of the probability of each class j is proportional to $\exp(z_j)$. Because the probabilites must add up to 1, we need to normalize by dividing by their sum:

$$\text{Prob}(i) = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)}$$

$$\log \text{Prob}(i) = z_i - \log \sum_{j=1}^{N} \exp(z_j)$$

lost function $-\log\text{Prob(i)}$

总共N个分类，z_j 表示模型对类别 j 判定的概率。
那么模型总共有N个输出（每个类别都有一个）
所有输出之和为1

为什么要用Soft-max

$$\frac{\partial L_i}{\partial f_{y_i}} = -\ln\left(\frac{e^{f_{y_i}}}{\sum_j e^j}\right)'$$

$$= -(1 - P_{f_{y_i}}) = P_{f_{y_i}} - 1$$

i 为正确的类别，其他类的导数就是$P_{f_{yj}}$自己

求导时只要将概率向量对应的正确结果的那一维减1即可

假设最后得到的某个训练样本的分类向量是[1, 5, 3],

$$\left[\frac{e^1}{e^1+e^3+e^5}, \frac{e^5}{e^1+e^3+e^5}, \frac{e^3}{e^1+e^3+e^5}\right] = [0.015, 0.866, 0.117]$$

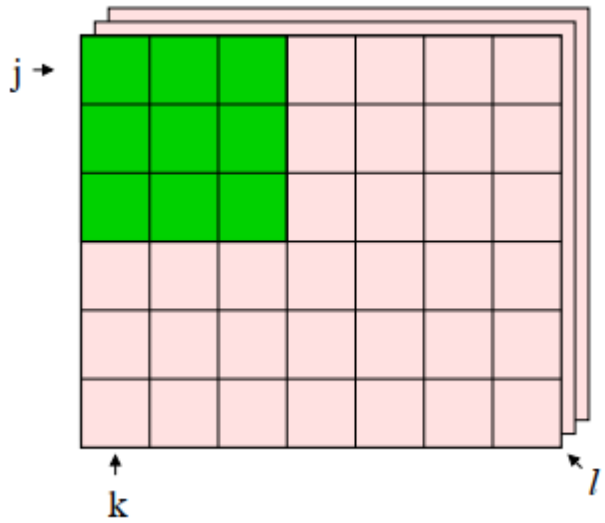
$$[0.015, 0.866 - 1, 0.117] = [0.015, -0.134, 0.117]$$

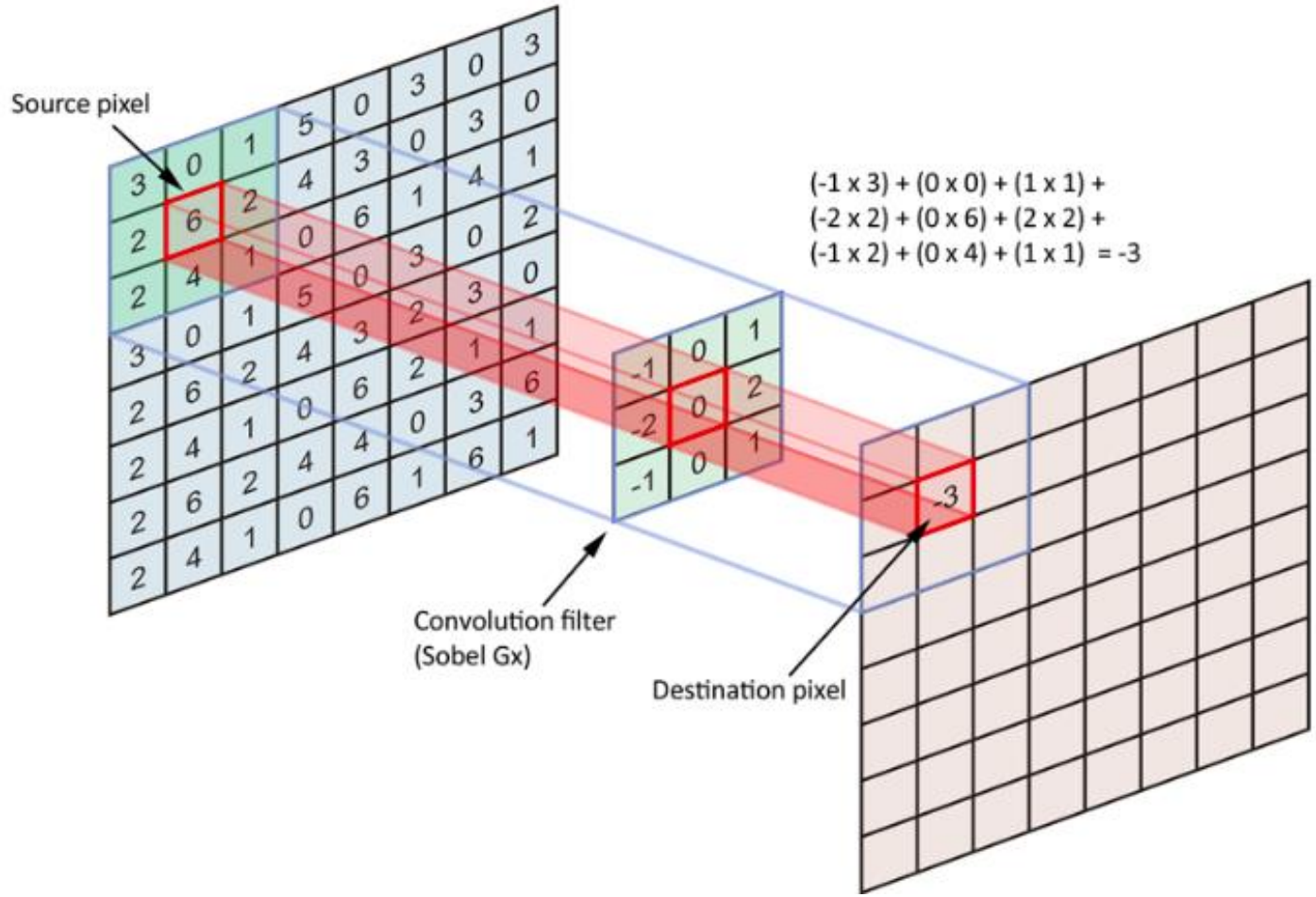
把这个结果代入到back propagation即可

Section3. CNN and Image

- Convolutional Neural Network

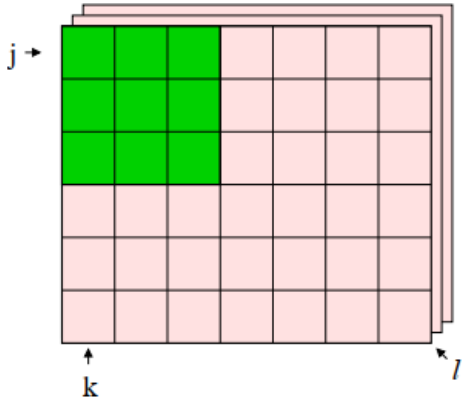
Convolution 卷积



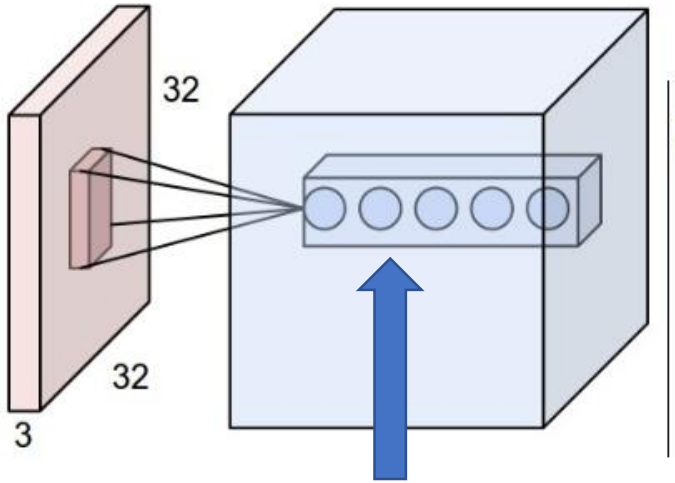


Section3. CNN and Image

- Convolutional Neural Network

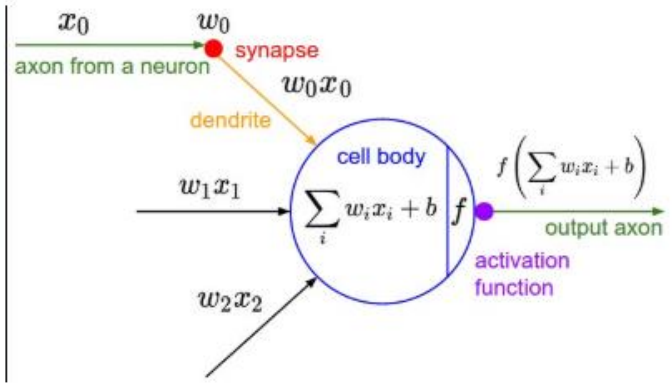


layer中的neuron个数取决于filter能在图像上移动的次数



filter的个数决定了layer的“深度”depth

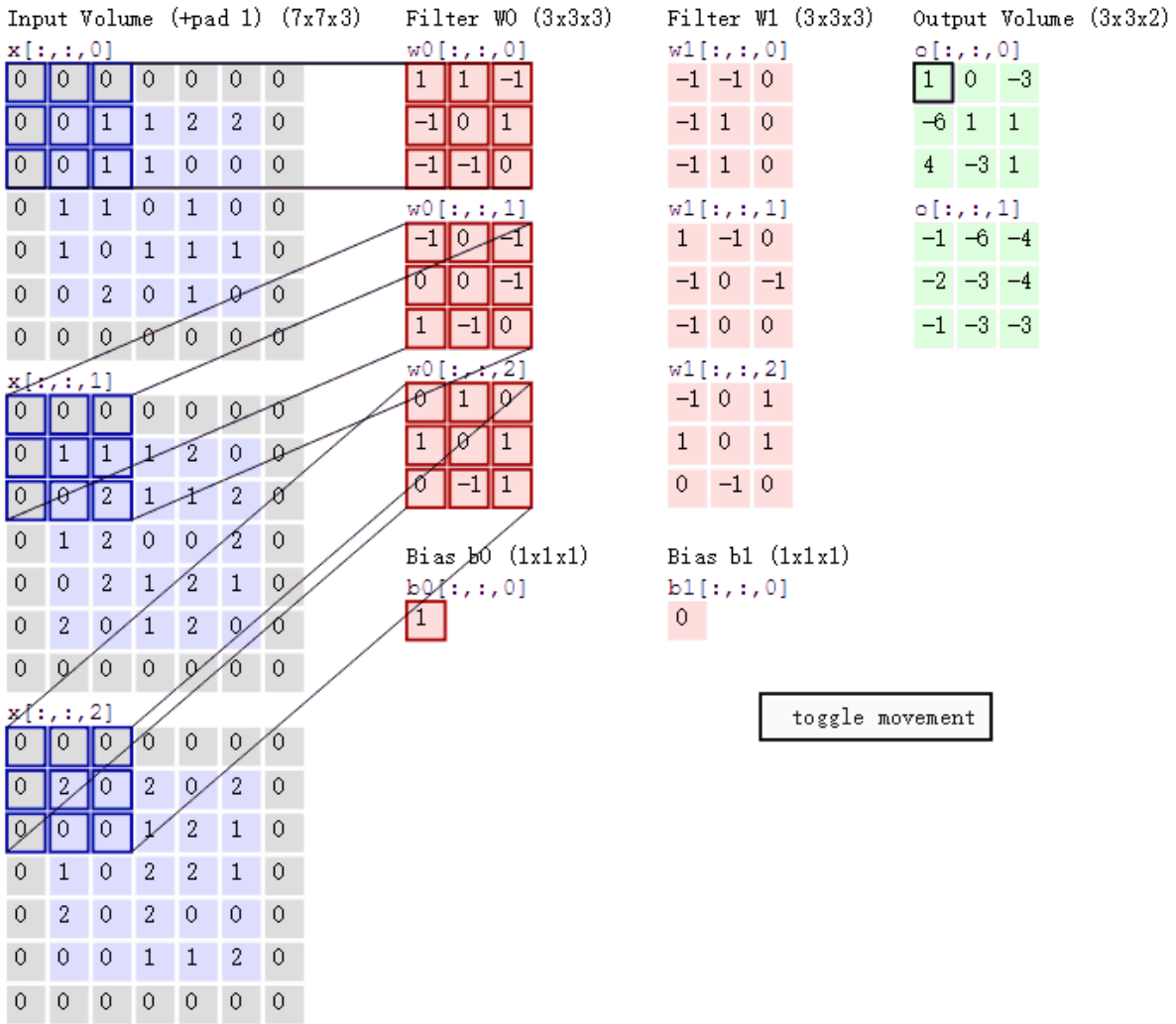
一个neuron就是filter一次计算得到的一个节点



一个filter的参数是固定的，所以同一个filter产生的neuron参数相同(weight sharing)

Section3. CNN and Image

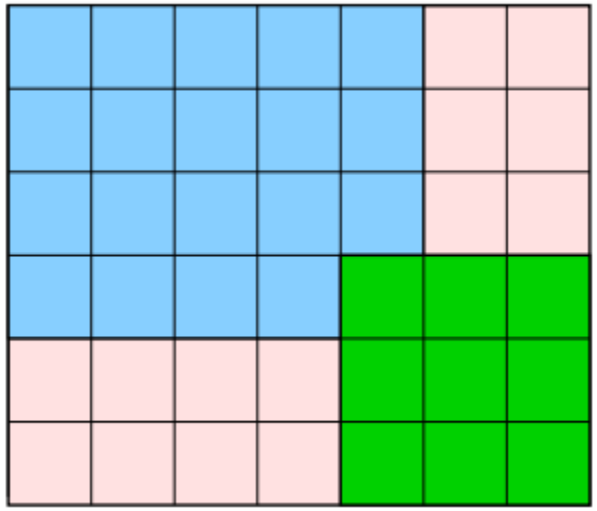
- Convolutional Neural Network

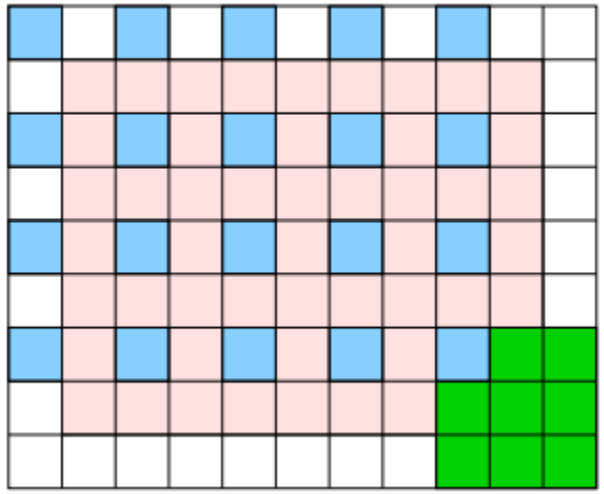


Section3. CNN and Image

- Zero Padding and Stride

If the original image size is $J \times K$ and the filter is size $M \times N$, the convolution layer will be $(J+1-M) \times (K+1-N)$





Zero-Padding: 在周围没有数据的地方补0

Stride: filter 移动的步长

When combined with zero padding of width P,

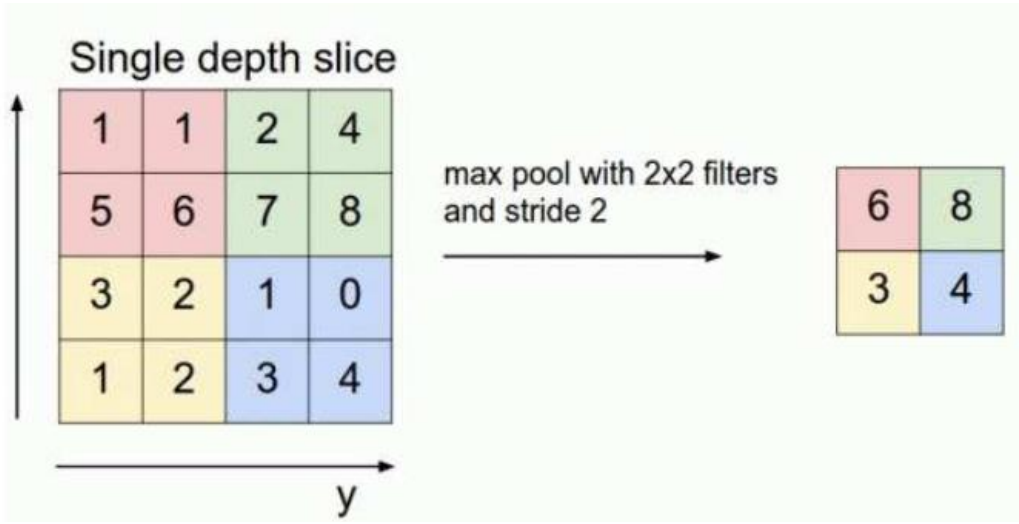
j takes on the values $0, s, 2s, \ldots, (J+2P-M)$

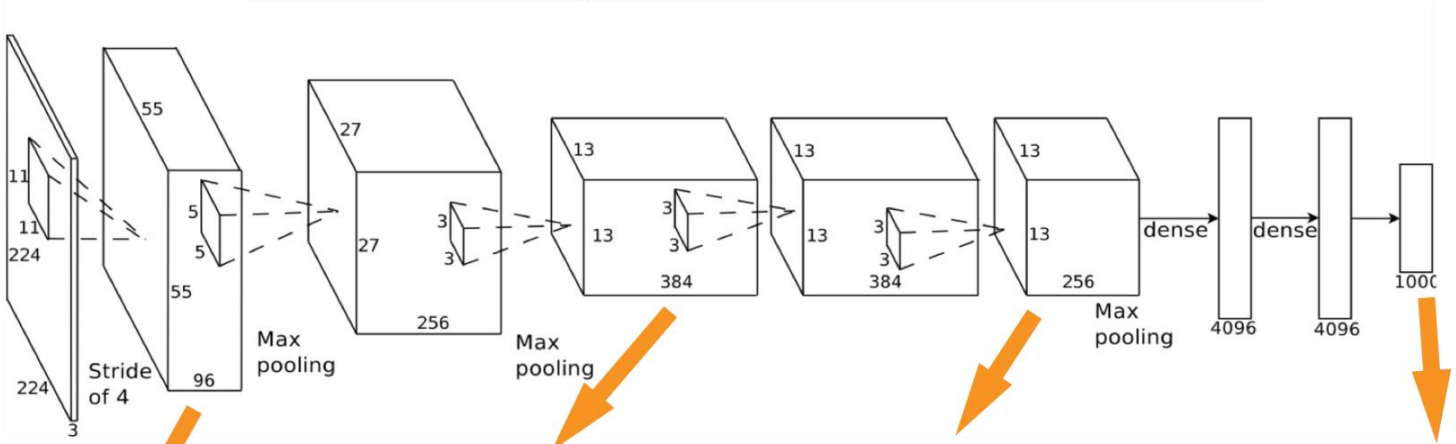
k takes on the values $0, s, 2s, \ldots, (K+2P-N)$

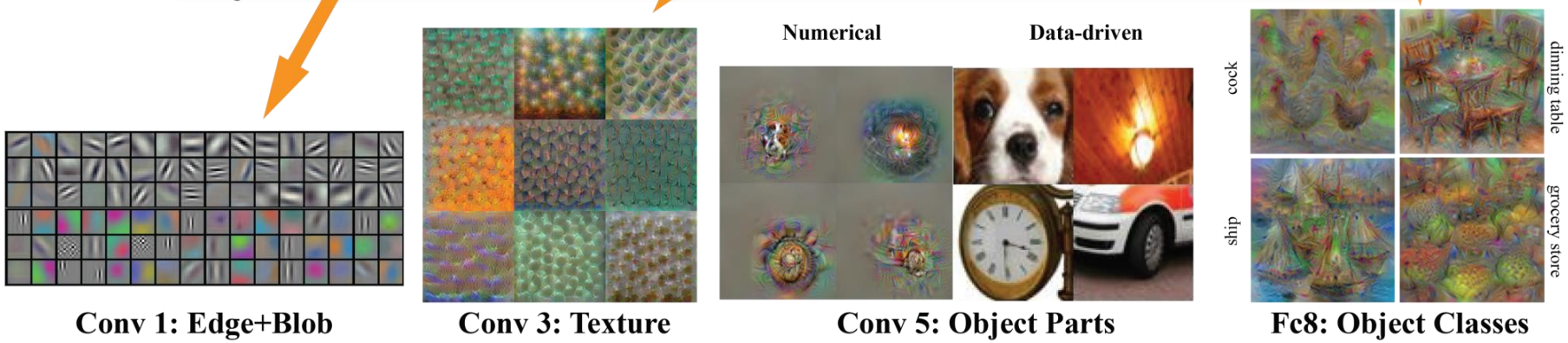
The next layer is $(1+(J+2P-M)/s)$ by $(1+(K+2P-N)/s)$

Section3. CNN and Image

- Max-Pooling







Section3. CNN and Image

- Convolutional Neural Network - Exercise: Softmax

Consider a classification task with three classes 1, 2, 3. Suppose a particular input is presented producing outputs z_1=1.0, z_2=2.0, z_3=3.0 and that the correct class for this input is Class 2. Compute the following, to two decimal places:

$$\text{Prob}(i) = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)}$$

$$\log \text{Prob}(i) = z_i - \log \sum_{j=1}^{N} \exp(z_j)$$

a. Prob(i), for i = 1, 2, 3

$$\text{Prob}(1) = e^1/(e^1 + e^2 + e^3) = 2.718/30.193 = 0.09$$
$$\text{Prob}(2) = e^2/(e^1 + e^2 + e^3) = 7.389/30.193 = 0.24$$
$$\text{Prob}(3) = e^3/(e^1 + e^2 + e^3) = 20.086/30.193 = 0.67$$

可以发现正确项的导数被push up，
错误项的被push down

b. d(log Prob(2))/dz_j, for j = 1, 2, 3

$$d(\log \text{Prob}(2))/dz_1 = d(z_2 - \log \Sigma_j \exp(z_j))/dz_1 = -\exp(z_1)/\Sigma_j \exp(z_j) = -0.09$$
$$d(\log \text{Prob}(2))/dz_2 = d(z_2 - \log \Sigma_j \exp(z_j))/dz_2 = 1 - \exp(z_2)/\Sigma_j \exp(z_j) = 1 - 0.24 = 0.76$$
$$d(\log \text{Prob}(2))/dz_3 = d(z_2 - \log \Sigma_j \exp(z_j))/dz_3 = -\exp(z_3)/\Sigma_j \exp(z_j) = -0.67$$

- Convolutional Neural Network – Exercise: CNN parameters

"The input to the neural network consists of an 84 × 84 × 4 image. The first hidden layer convolves 16 8 × 8 filters with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 32 4 × 4 filters with stride 2, again followed by a rectifier nonlinearity. The final hidden layer is fully-connected and consists of 256 rectifier units. The output layer is a fully-connected linear layer with a single output for each valid action. The number of valid actions varied between 4 and 18 on the games we considered."

You should assume the input images are gray-scale, there is no padding, and there are 18 valid actions (outputs).

a. weights per neuron in this layer (including bias)
b. neurons in this layer
c. connections into the neurons in this layer
d. independent parameters in this layer

- Convolutional Neural Network – Exercise: CNN parameters

First Convolutional Layer:

$J = K = 84, L = 4, M = N = 8, P = 0, s = 4$

weights per neuron: $1 + M \times N \times L = 1 + 8 \times 8 \times 4 = 257$
width and height of layer: $1+(J-M)/s = 1+(84-8)/4 = 20$
neurons in layer: $20 \times 20 \times 16 = 6400$
connections: $20 \times 20 \times 16 \times 257 = 1644800$
independent parameters: $16 \times 257 = 4112$

Second Convolutional Layer:

$J = K = 20, L = 16, M = N = 4, P = 0, s = 2$

weights per neuron: $1 + M \times N \times L = 1 + 4 \times 4 \times 16 = 257$
width and height of layer: $1+(J-M)/s = 1+(20-4)/2 = 9$
neurons in layer: $9 \times 9 \times 32 = 2592$
connections: $9 \times 9 \times 32 \times 257 = 666144$
independent parameters: $32 \times 257 = 8224$

- Convolutional Neural Network – Exercise: CNN parameters

Fully Connected Layer:

weights per neuron: $1 + 2592 = 2593$
neurons in layer: 256
connections: $256 \times 2593 = 663808$
independent parameters: 663808

Output Layer:

weights per neuron: $1 + 256 = 257$
neurons in layer: 18
connections: $18 \times 257 = 4626$
independent parameters: 4626

Section3. CNN and Image

- Weight Initialization

$$G_1 n_i^{\text{out}} \text{Var}[w^{(i)}] = 1$$

初始参数应该使各层乘积趋近1，否则就会造成梯度爆炸或者梯度消失

Exploding gradients & **Vanishing gradients**

Batch Normalization　　可以加速训练过程，提高准确度

$$\hat{x}_k^{(i)} = \frac{x_k^{(i)} - \text{Mean}[x_k^{(i)}]}{\sqrt{\text{Var}[x_k^{(i)}]}}$$

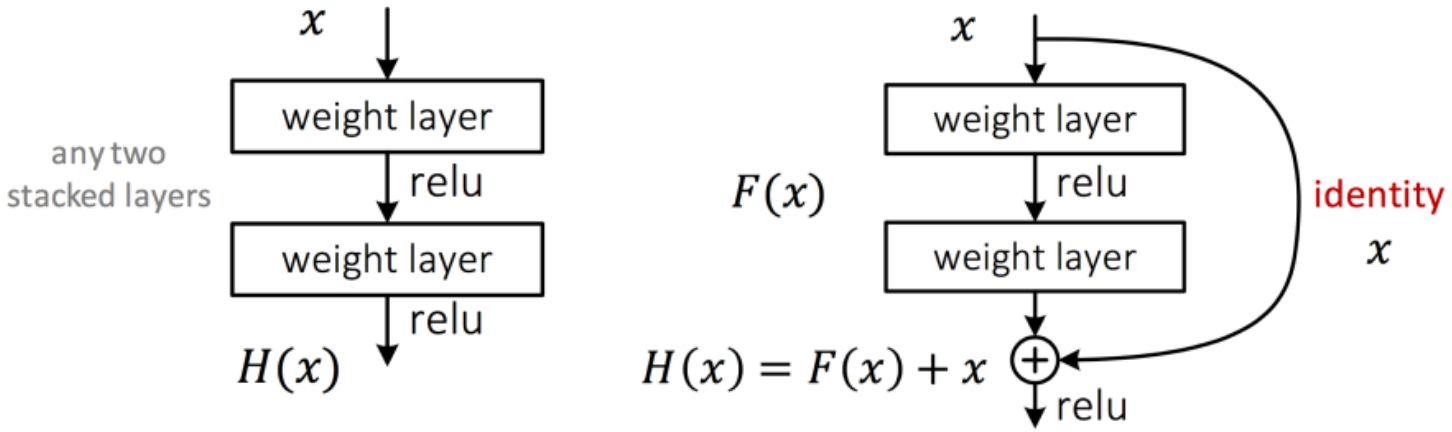
将输入数据标准化(Normalization)

$$y_k^{(i)} = \beta_k^{(i)} + \gamma_k^{(i)} \hat{x}_k^{(i)}$$

再通过训练参数re-scale

- Residual Network

Idea: Take any two consecutive stacked layers in a deep network and add a "skip" connection which bipasses these layers and is added to their output.

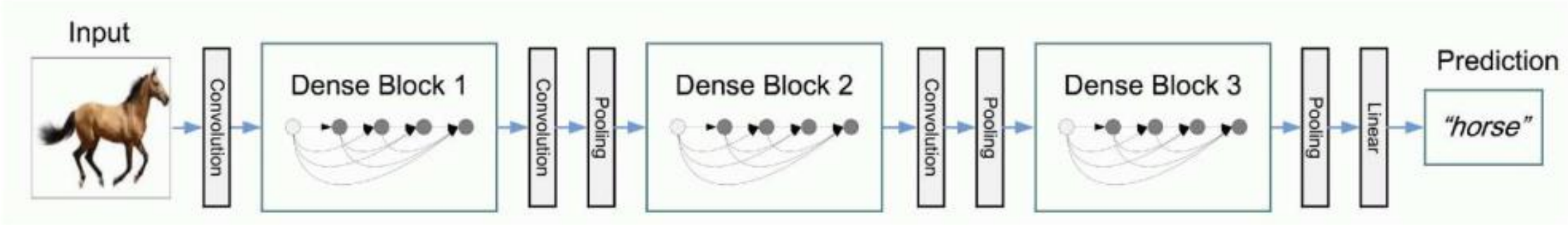


随着网络层数加深，前面的误差会一直积累越来越大，通过skip connection可以有效地把前面的信息引入后面的网络中进行修正

with more than 100 layers, need to apply ReLU before adding the residual instead of afterwards. This is called an identity skip connection.

Section3. CNN and Image

- Dense Network



网络由一系列的block组成，每个block里面的所有layer都和它之前的layer连接(skip connection)

Section3. CNN and Image

- Neural Style Transfer

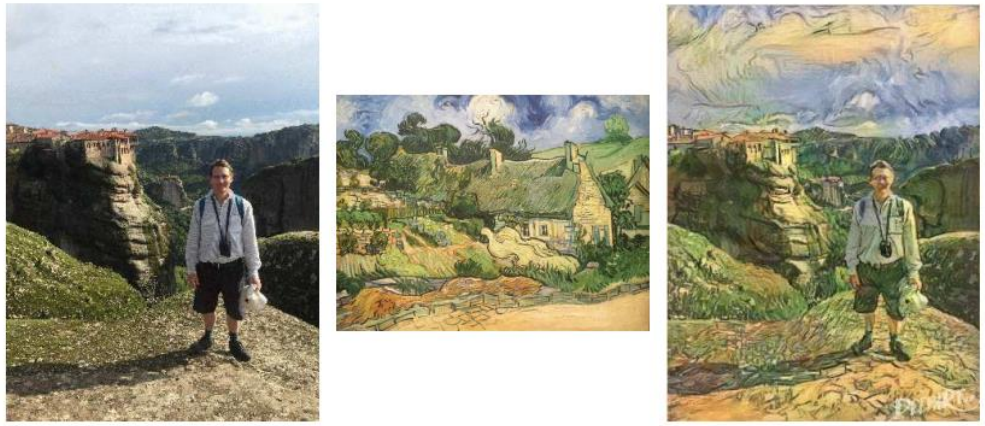
不需要背公式，知道概念和参数的意思即可

$$E_{\text{total}} = \alpha\, E_{\text{content}} + \beta\, E_{\text{style}}$$

$$= \frac{\alpha}{2}\sum_{i,k}||F_{ik}^{l}(x) - F_{ik}^{l}(x_c)||^2 + \frac{\beta}{4}\sum_{l=0}^{L}\frac{w_l}{N_l^2 M_l^2}\sum_{i,j}(G_{ij}^{l} - A_{ij}^{l})^2$$

where

x_c, x	=	content image, synthetic image
F_{ik}^{l}	=	i^{th} filter at position k in layer l
N_l, M_l	=	number of filters, and size of feature maps, in layer l
w_l	=	weighting factor for layer l
G_{ij}^{l}, A_{ij}^{l}	=	Gram matrices for style image, and synthetic image



content + style → new image

Section3. CNN and Image

- Overfitting - Solution　　可以从两个角度解决，数据+训练过程

数据

Data Augmentation
从已有的数据生成更多的新数据
AlexNet模型中，图片被分割处理为多个尺寸稍小的图片

Reduce features in Data
筛选数据以减少特征数量

训练

Early Stop
通过validation降低训练迭代次数

Dropout
每次迭代时屏蔽掉一部分神经元(一般选择50%概率），使其不参加计算和更新。
Testing时所有神经元都参加计算，但是以同样的概率不激活部分神经元

Reduce number of neurons
神经元过多也容易导致overfitting

Weight Decay
过大的参数会严重干扰结果

Questions