手写数字识别

1.2 神经网络

@tm9161

L3

神经网络

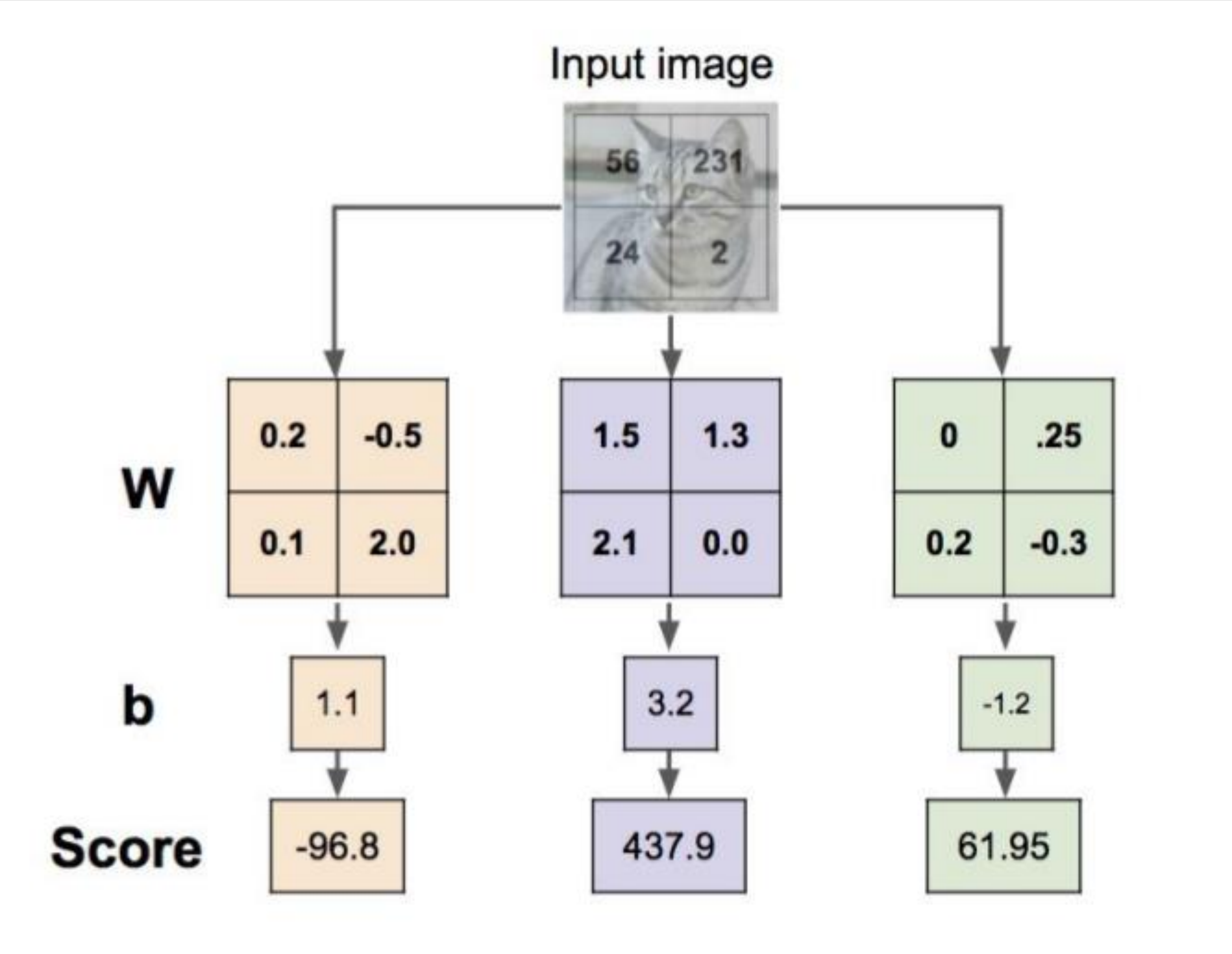
1.神经网络原理

2.神经网络的手写数字识别

3.算法问题与改进

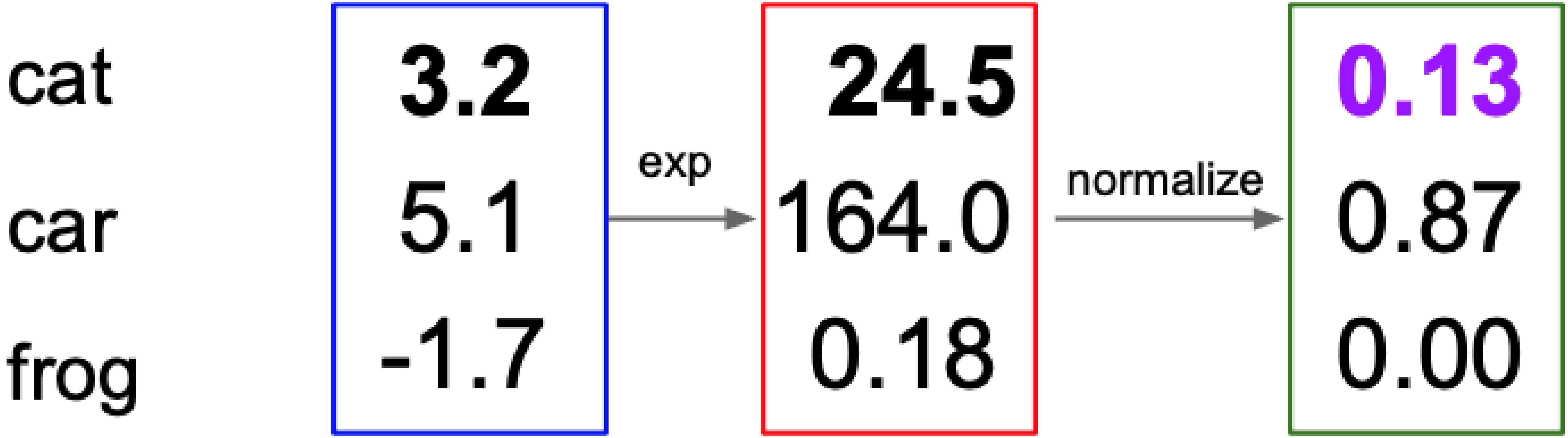
线性分类器 Linear classifier

由两部分组成一个是**评分函数**，是原始图像数据到类别分值的映射。另一个是**损失函数**，用来量化预测分类标签的与真实标签的一致性。**将通过更新评分函数的参数来最小化损失函数值。**



56X0.2+231X(-0.5)+24x0.1+2X2.0+1.1=-96.8

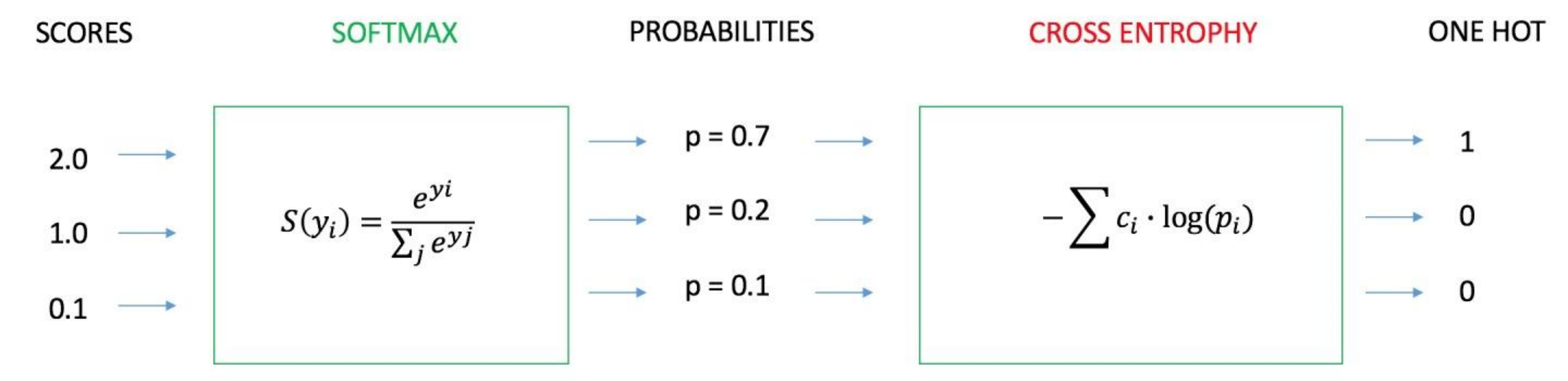
Softmax分类器



得分转化为概率

$$P(Y=k|X=x_i)=\frac{e^{s_k}}{\sum_j e^{s_j}}$$

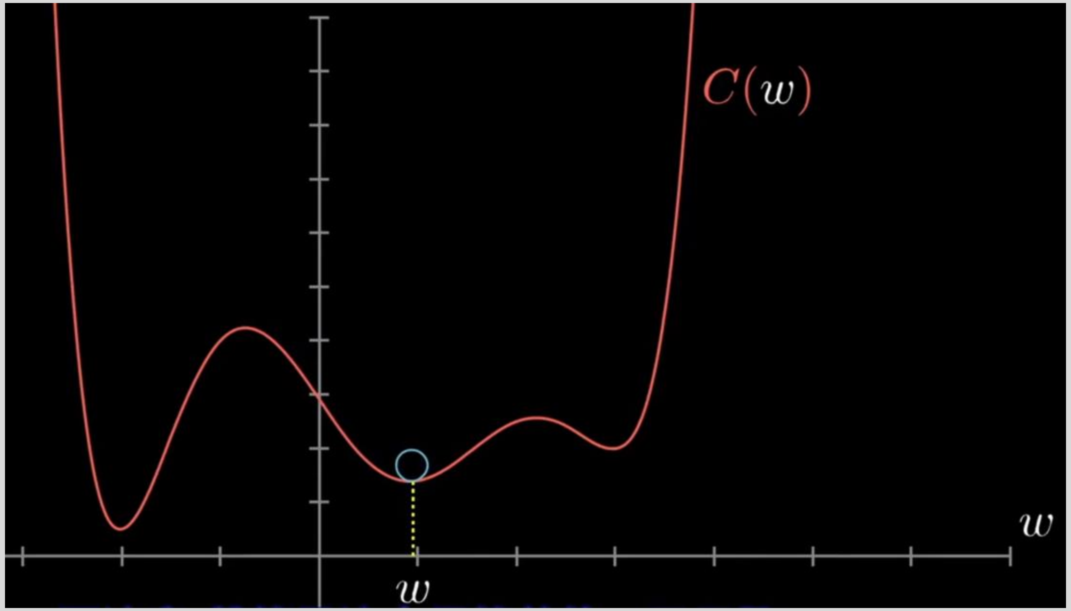
交叉熵损失 cross-entropy loss



Loss：-log（概率）与真实值比较

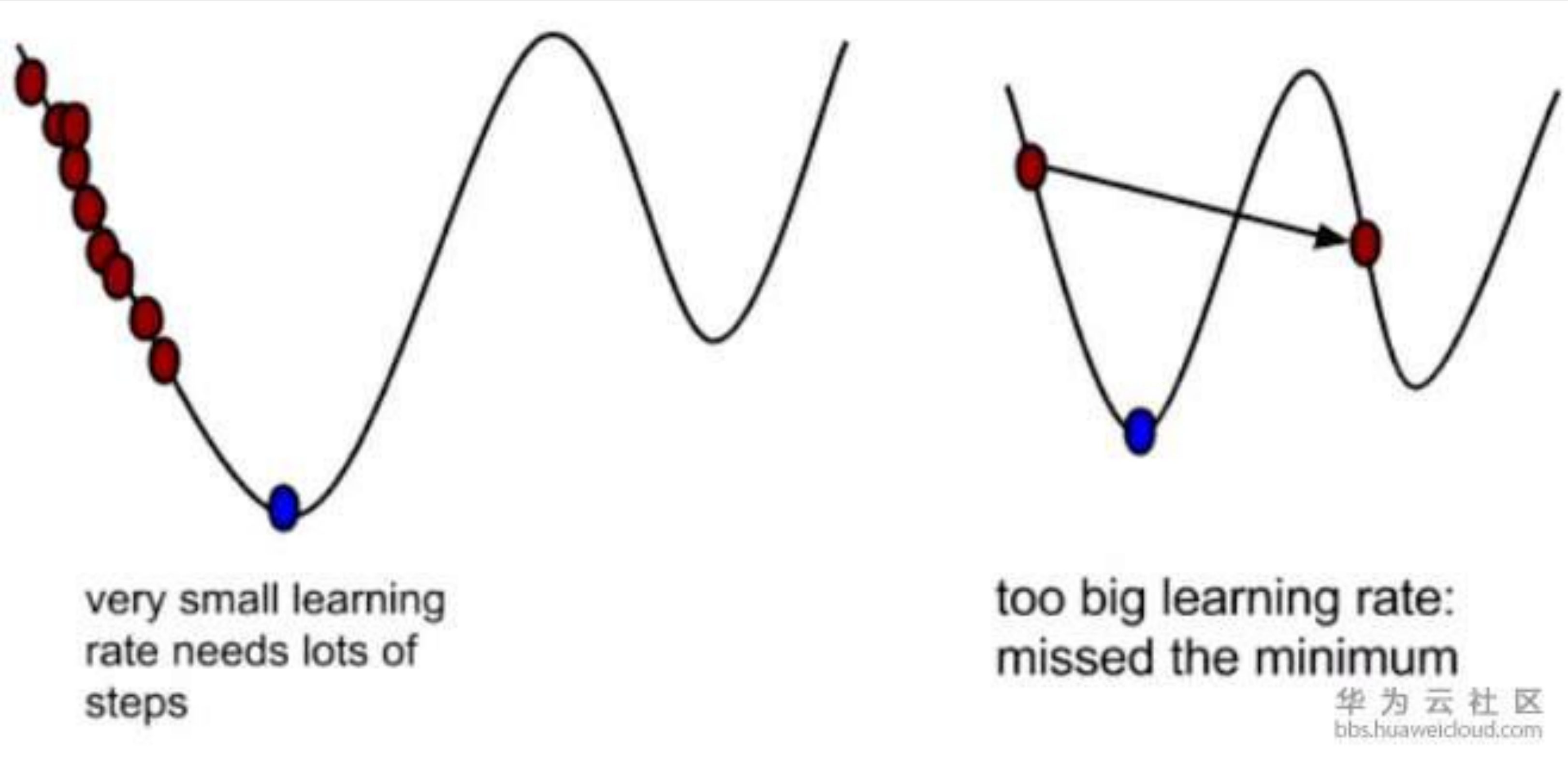
$$\text{sample 1 loss} = -(0 \times log0.3 + 0 \times log0.3 + 1 \times log0.4) = 0.91$$
$$\text{sample 2 loss} = -(0 \times log0.3 + 1 \times log0.4 + 0 \times log0.3) = 0.91$$
$$\text{sample 3 loss} = -(1 \times log0.1 + 0 \times log0.2 + 0 \times log0.7) = 2.30$$

梯度下降 Gradient descent



找到一个方向能降低损失函数的损失值。计算出最陡峭的方向，就是损失函数的梯度，向着梯度的负方向去更新，降低损失值。

学习率 Learning rate



梯度确定损失函数下降的方向。小步长下降稳定但进度慢，大步长进展快但是可能导致错过最优点，让损失值上升。

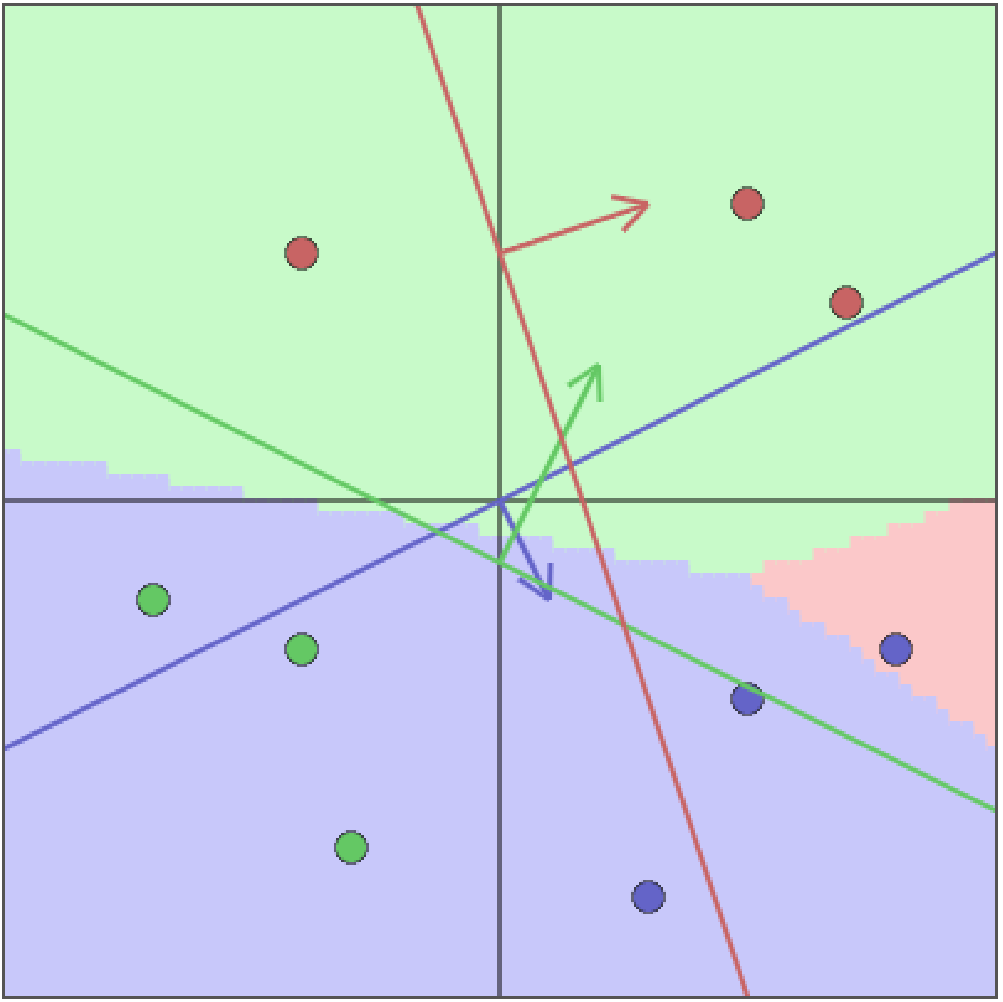
优化器 Optimization

```
In [12]: model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['acc'])
```

对梯度的一阶矩估计即梯度的均值和二阶矩估计即梯度的未中心化的方差，**进行综合考虑**，计算出**更新步长（学习率）**。

classifier computes scores as $W_{0,0}x_0 + W_{0,1}x_1 + b_0$ and the blue line shows the set of points (x_0, x_1) that give score of zero. The blue arrow draws the vector $(W_{0,0}, W_{0,1})$, which shows the direction of score increase and its length is proportional to how steep the increase is.

Note: you can drag the datapoints.



changes to control the parameters.

W[0,0]	W[0,1]	b[0]
1.00 *-0.16*	2.00 *0.16*	0.00 *0.15*
W[1,0]	**W[1,1]**	**b[1]**
2.00 *0.33*	-4.00 *-0.45*	0.50 *0.04*
W[2,0]	**W[2,1]**	**b[2]**
3.00 *0.13*	-1.00 *0.14*	-0.50 *-0.19*

Step size: 0.10000

Single parameter update

Start repeated update

a single example, L_i.

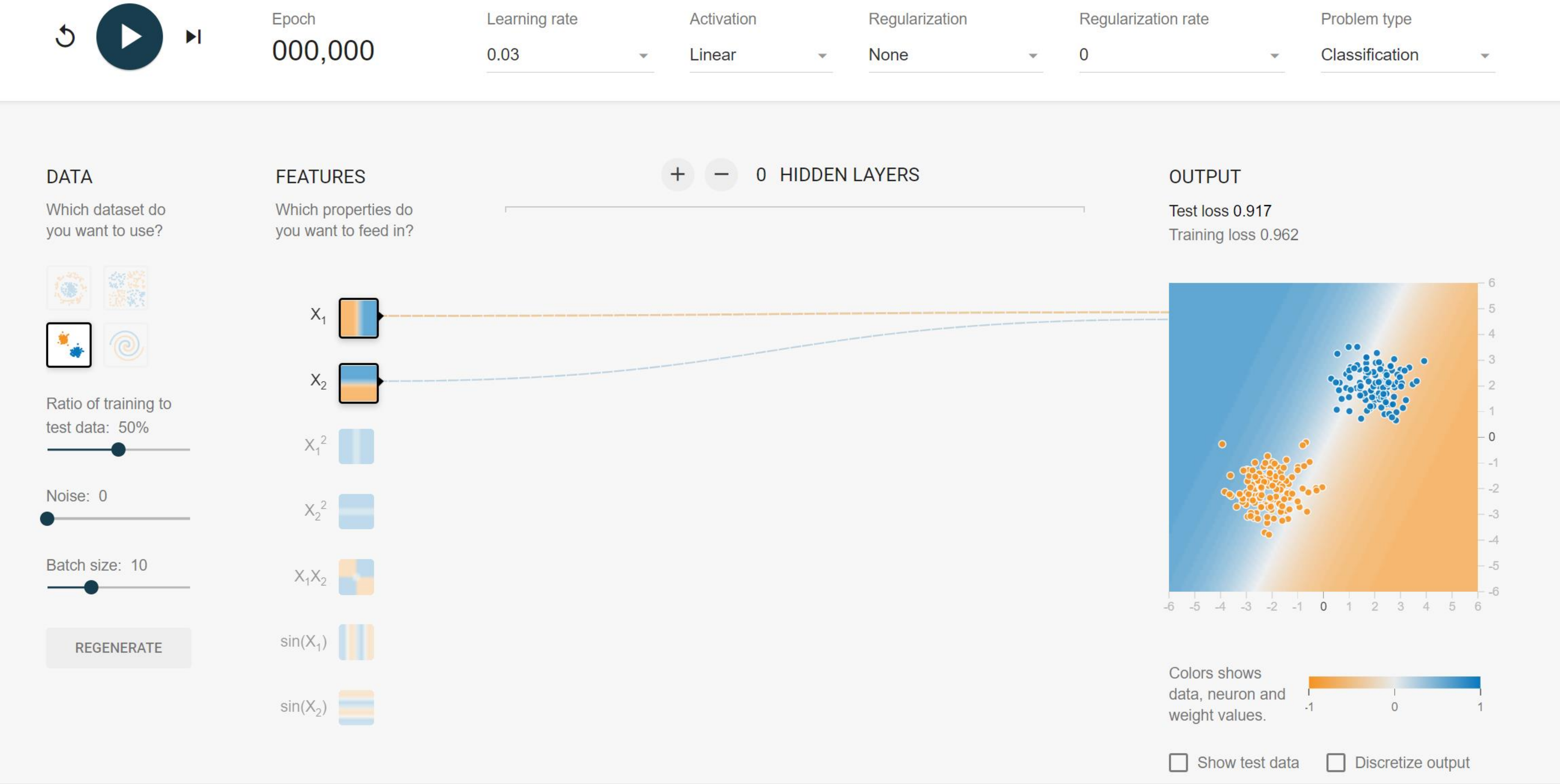
x[0]	x[1]	y	s[0]	s[1]	s[2]	L
0.50	0.40	0	1.30	-0.10	0.60	0.56
0.80	0.30	0	1.40	0.90	1.60	1.04
0.30	0.80	0	1.90	-2.10	-0.40	0.11
-0.40	0.30	1	0.20	-1.50	-2.00	1.96
-0.30	0.70	1	1.10	-2.90	-2.10	4.06
-0.70	0.20	1	-0.30	-1.70	-2.80	1.68
0.70	-0.40	2	-0.10	3.50	2.00	1.72
0.50	-0.60	2	-0.70	3.90	1.60	2.40
-0.40	-0.50	2	-1.40	1.70	-1.20	3.00
						mean:
						1.84

Total data loss: 1.84
Regularization loss: 3.50
Total loss: 5.34

L2 Regularization strength: 0.10000

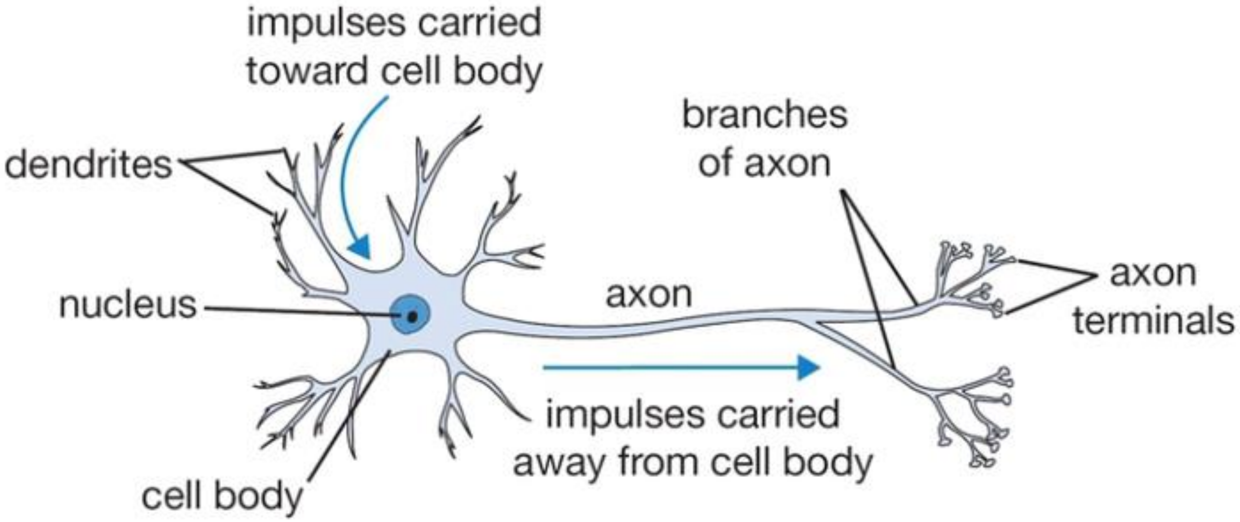
Multiclass SVM loss formulation:

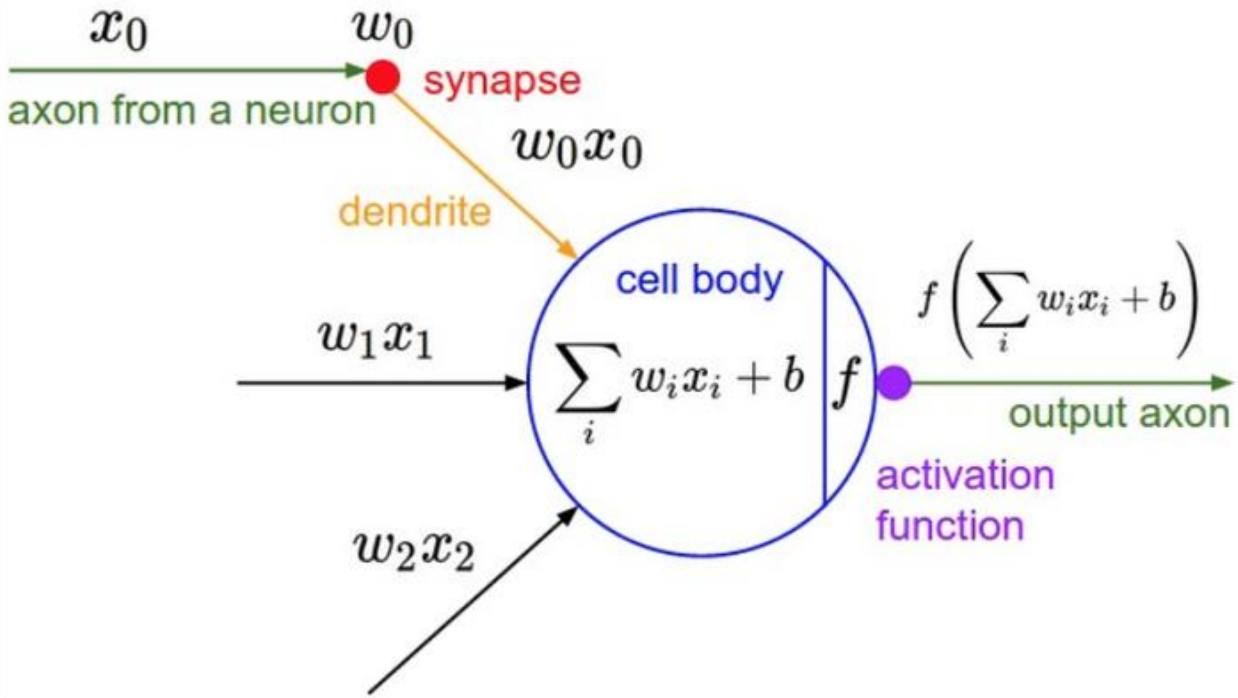
○ Weston Watkins 1999



http://playground.tensorflow.org

神经网络 Neural Network

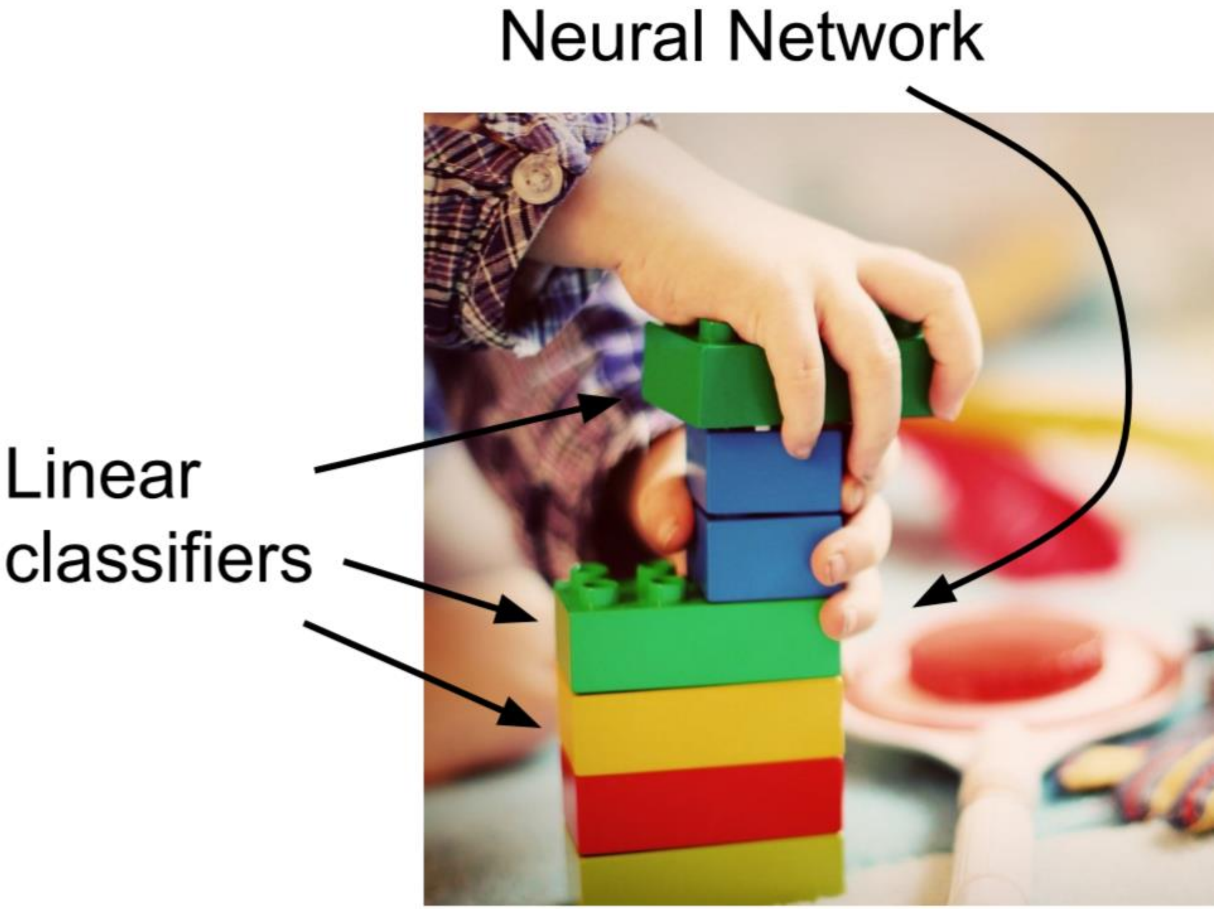




每个神经元都从它的树突获得**输入信号**，然后沿着它唯一的轴突产生**输出信号**。轴突在末端会逐渐分枝，通过突触和其他神经元的树突相连。

激活函数
Activation functions

树突将信号传递到细胞体，信号在细胞体中相加，如高于某个阈值，那么神经元将会激活，向其轴突输出信号。



Q: What if we try to build a neural network without one?

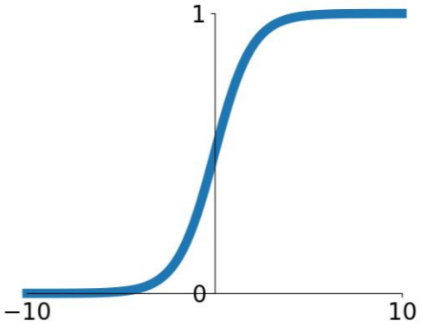
$$f = W_2 W_1 x \qquad W_3 = W_2 W_1 \in \mathbb{R}^{C \times H}, f = W_3 x$$

A: We end up with a linear classifier again!

Activation functions

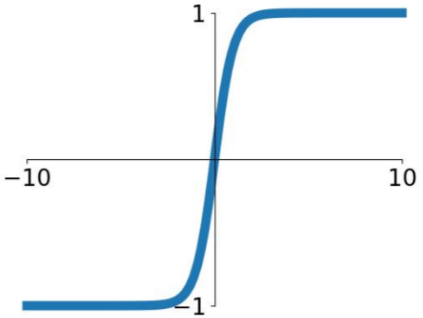
Sigmoid

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



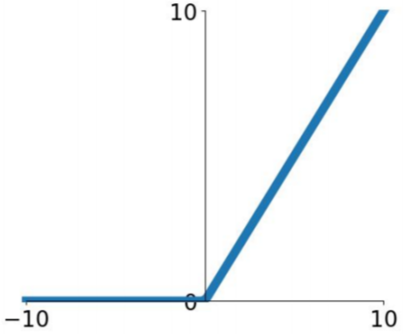
tanh

$$\tanh(x)$$



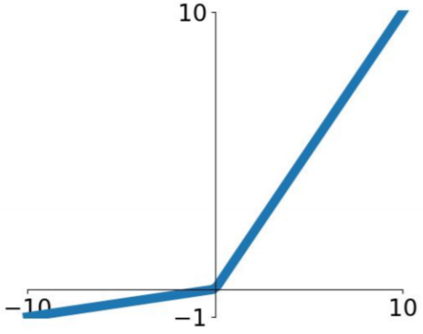
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

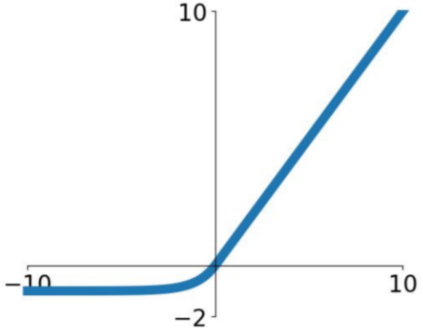


Maxout

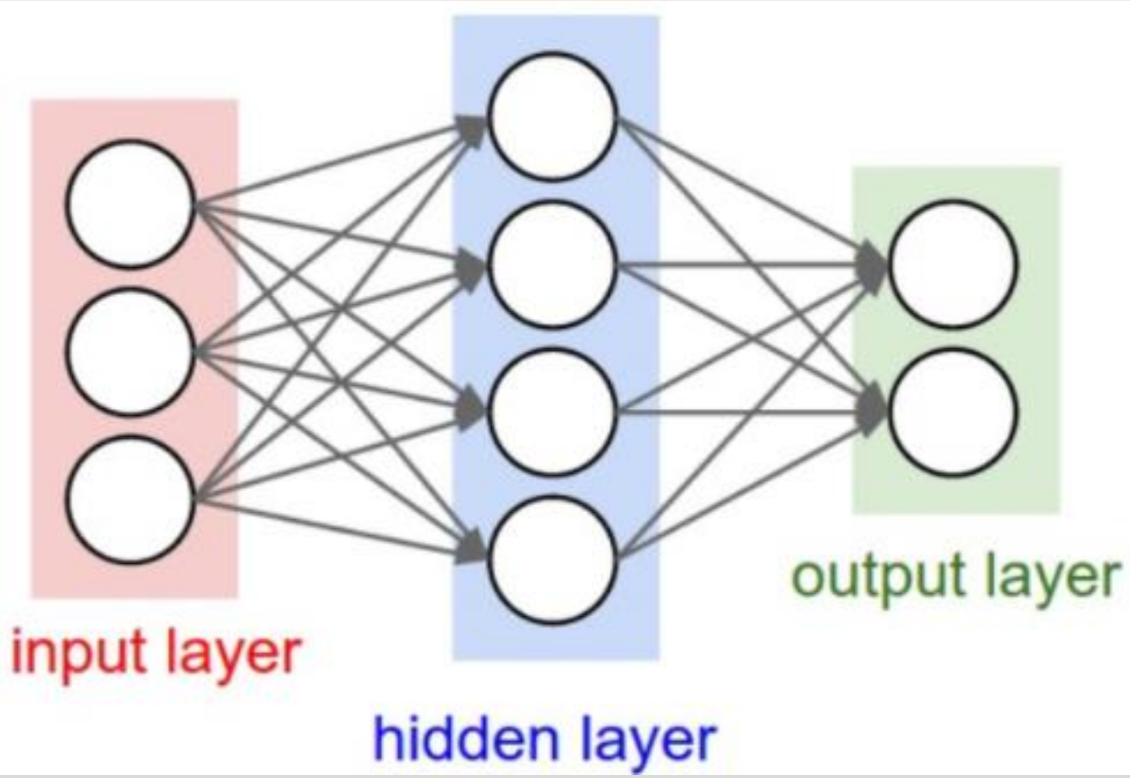
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

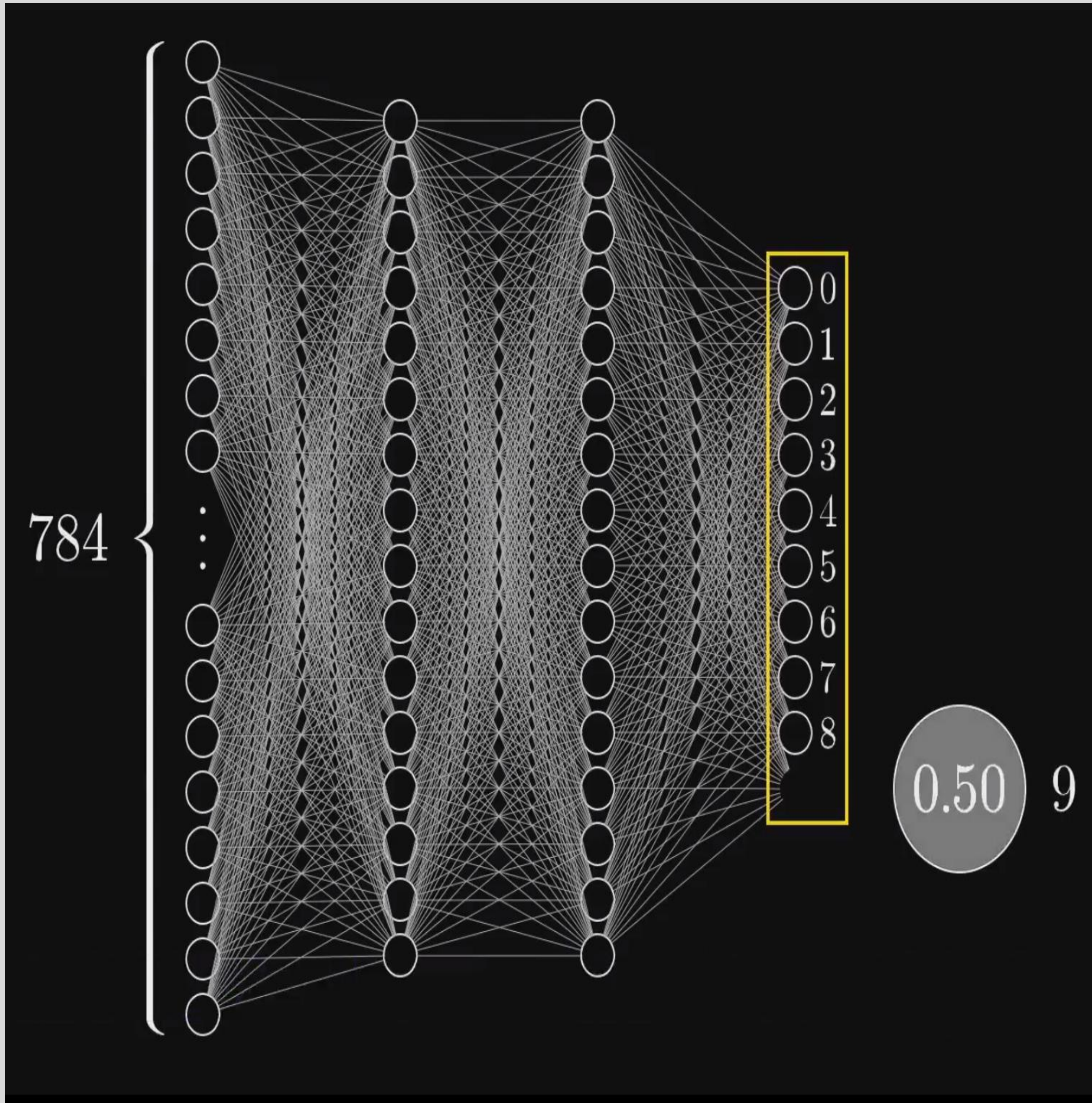
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



网络结构 Network structure





输入层　　隐藏层*n　　输出层

```
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(64, activation='sigmoid'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

前向传播 Forward propagation

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 64)	50240
dense_1 (Dense)	(None, 10)	650

Sigmoid

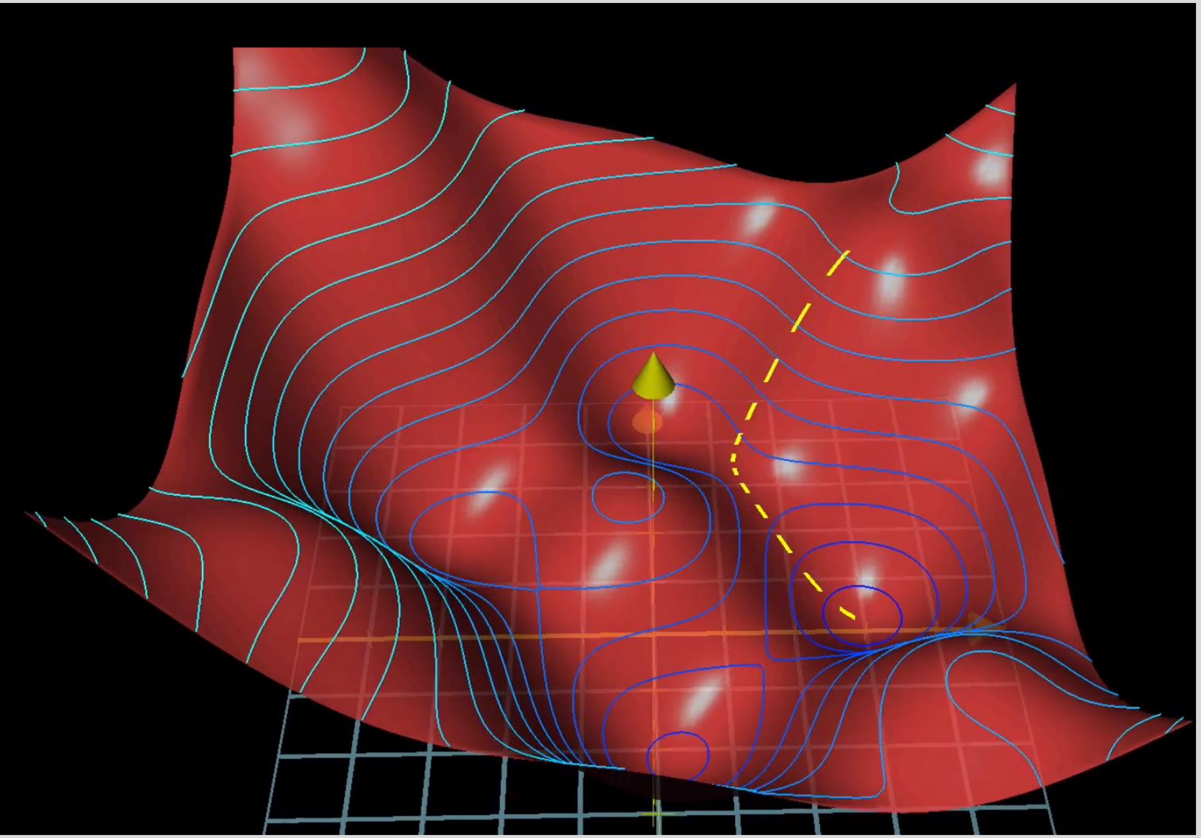
How positive is this? 有多正?

$$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \cdots + w_na_n \boxed{-10})$$

"bias"
偏置

隐藏层：神经元个数：64
隐藏层参数个数：784*64+64=50240

反向传播 Back propagation



1.计算损失函数（真实值与预测值）

2.计算权重对于整体误差的影响（导数）

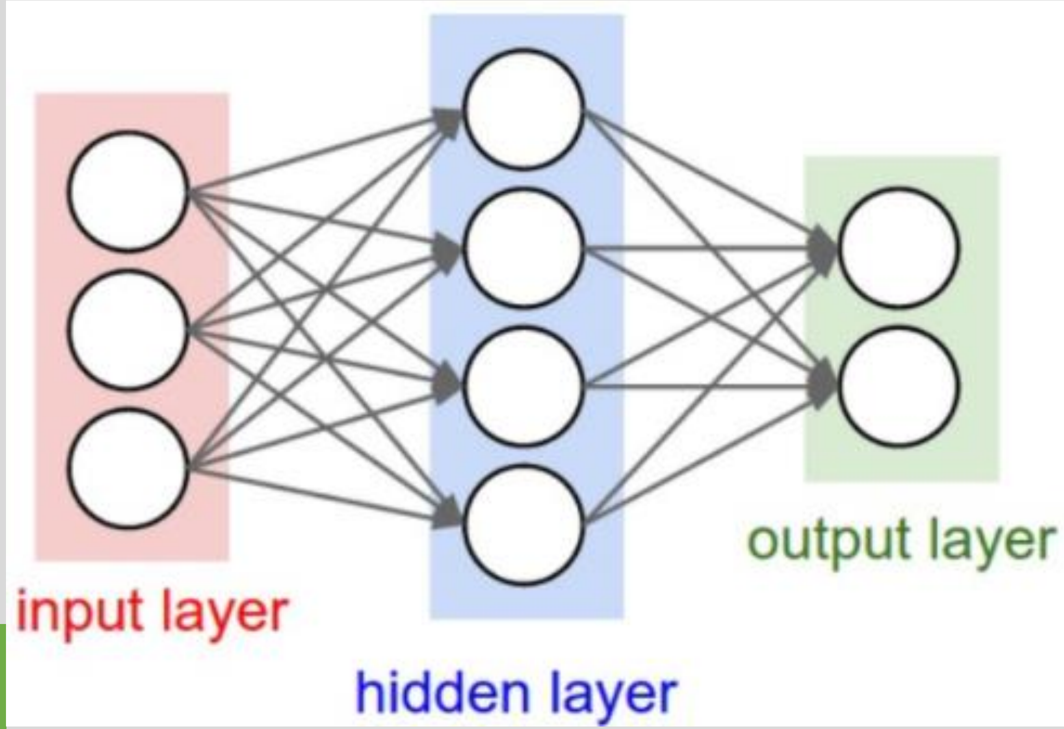
3.结合学习率更新权重

……

神经网络 Neural Network

训练集:
60000

测试集:
10000



1 数据采集

```
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()
```

```
train_images = train_images / 255
test_images = test_images/ 255
```

```
train_labels = np.array(pd.get_dummies(train_labels))
test_labels = np.array(pd.get_dummies(test_labels))
```

2 建立模型

```
model = tf.keras.Sequential()
```

```
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(64, activation='sigmoid'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                50240
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0
```

3 模型训练

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['acc'])
```

```
history = model.fit(train_images,train_labels,epochs = 10,validation_data=(test_images,test_labels))
```

```
Epoch 1/10
1875/1875 [==============================] - 3s 1ms/step - loss: 2.2646 - acc: 0.1985 - val_loss: 1.9990 - val_acc:
0.4633
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 1.8488 - acc: 0.5783 - val_loss: 1.3675 - val_acc:
0.6967
Epoch 3/10
1875/1875 [==============================] - 2s 982us/step - loss: 1.2523 - acc: 0.7143 - val_loss: 0.9392 - val_acc
0.7888
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.8921 - acc: 0.7958 - val_loss: 0.7163 - val_acc:
0.8283
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.6937 - acc: 0.8322 - val_loss: 0.5865 - val_acc:
0.8573
```

4 模型测试

```
model.evaluate(test_images, test_labels)
```

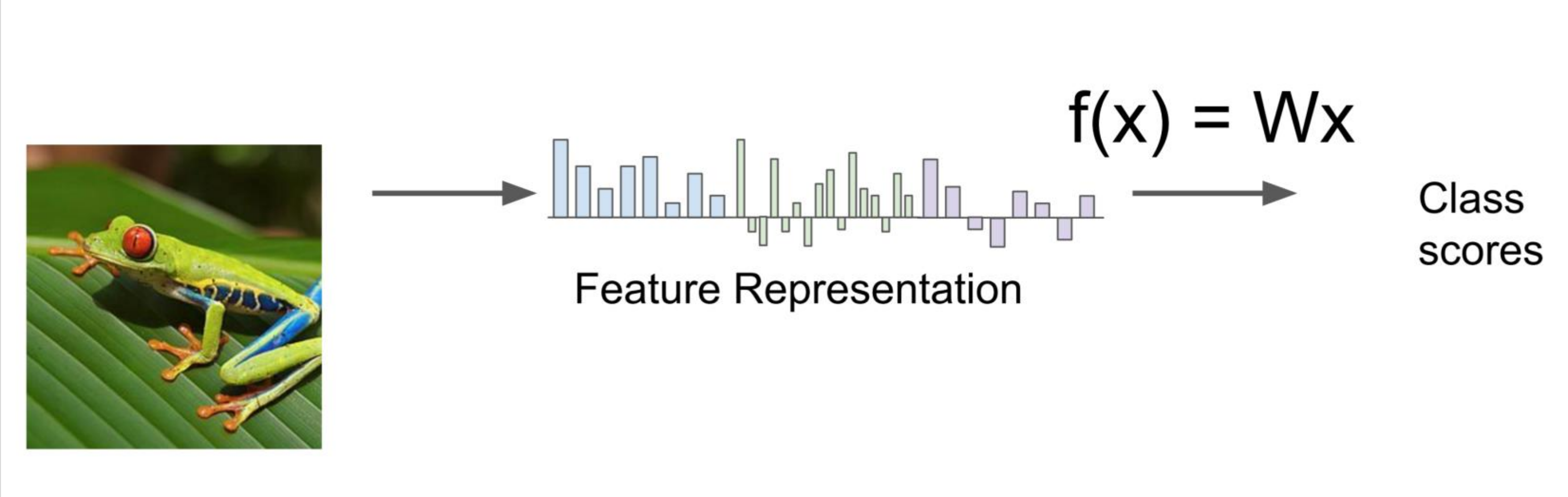
```
313/313 [==============================] - 0s 997us/step - loss: 0.0953 - acc: 0.9714
```

```
[0.09527470171451569, 0.9714000225067139]
```

思考

问题：

1.像素不能完全反映实际图像内容。



2.图片尺寸的增大、神经原的增加带来的训练参数的增多，训练时间的增加。

```
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
flatten_1 (Flatten)          (None, 784)               0
________________________________________________________________
dense_2 (Dense)              (None, 64)                50240
________________________________________________________________
dense_3 (Dense)              (None, 10)                650
================================================================
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0
```

参考资料：

1. CS231n: Convolutional Neural Networks for Visual Recognition
http://cs231n.stanford.edu/

2.【子豪兄】精讲CS231N斯坦福计算机视觉公开课
https://www.bilibili.com/video/BV1K7411W7So

3.CS231n课程笔记翻译
https://zhuanlan.zhihu.com/p/21930884

4. Tensorflow Playground
http://playground.tensorflow.org

5.损失函数 - 交叉熵损失函数
https://zhuanlan.zhihu.com/p/35709485

6.深度学习之神经网络的结构 Part 1 ver 2.0
https://www.bilibili.com/video/BV1bx411M7Zx