



# Predict the use of shared bicycles

---

商家煜  
郑铠奇



# Deep Learning Foundation

---



# Outline

---

- 1/ Introduction
  - 2/ Theory of BPNN
  - 3/ Demo of the project
  - 4/ Optimization
-



# Contents

---

1/ Introduction

2/ Theory of BPNN

3/ Demo of the project

4/ Optimization

---



### Back Propagation Neural Network

- Computing systems inspired by the biological neural networks that constitute animal brains

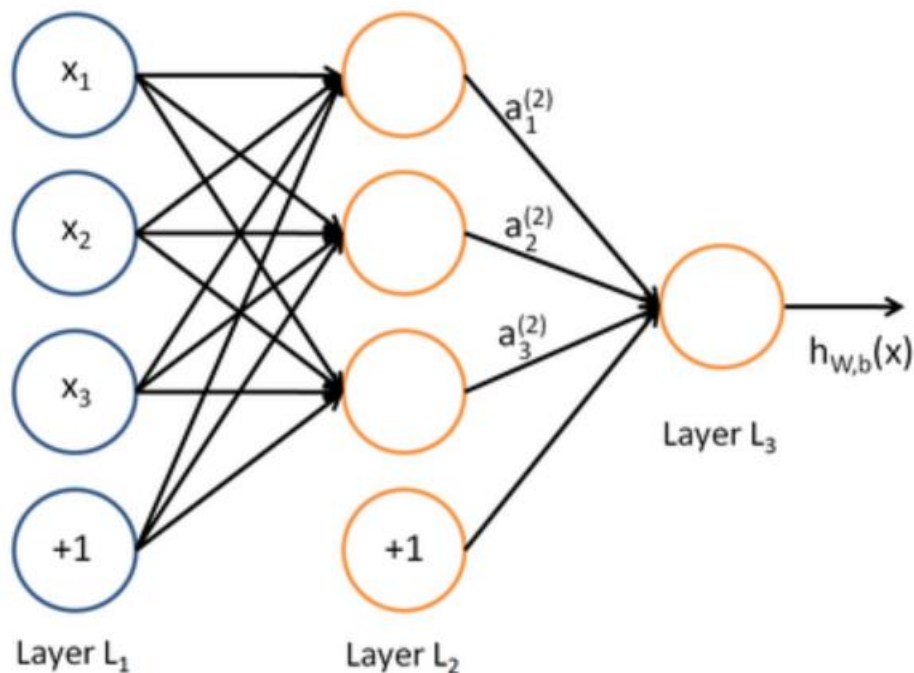


Fig.1 Figure of Three-layer Neural Network



# Convolutional Neural Network (CNN)

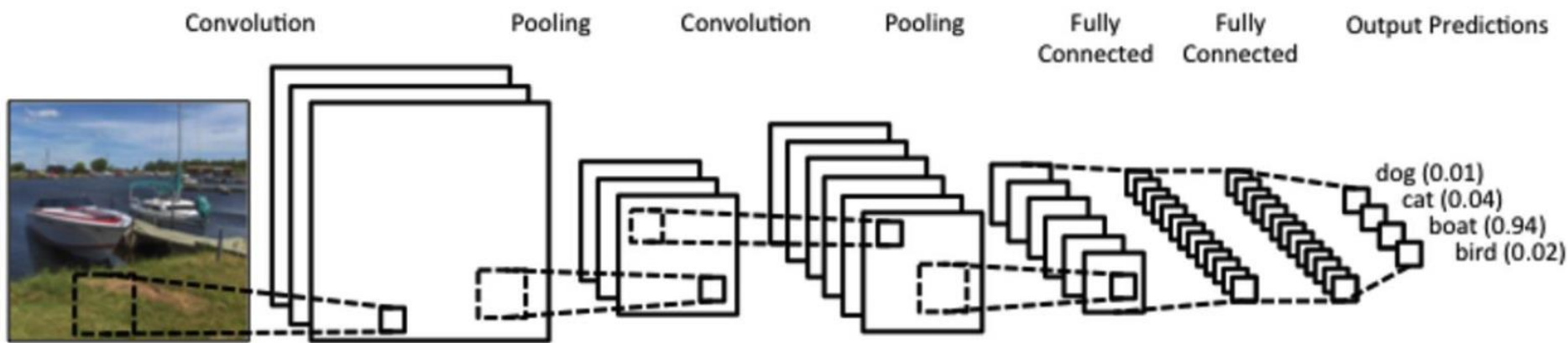


Fig.3 Convolutional neural network

Sensitive with image



## Recurrent Neural Network(RNN)

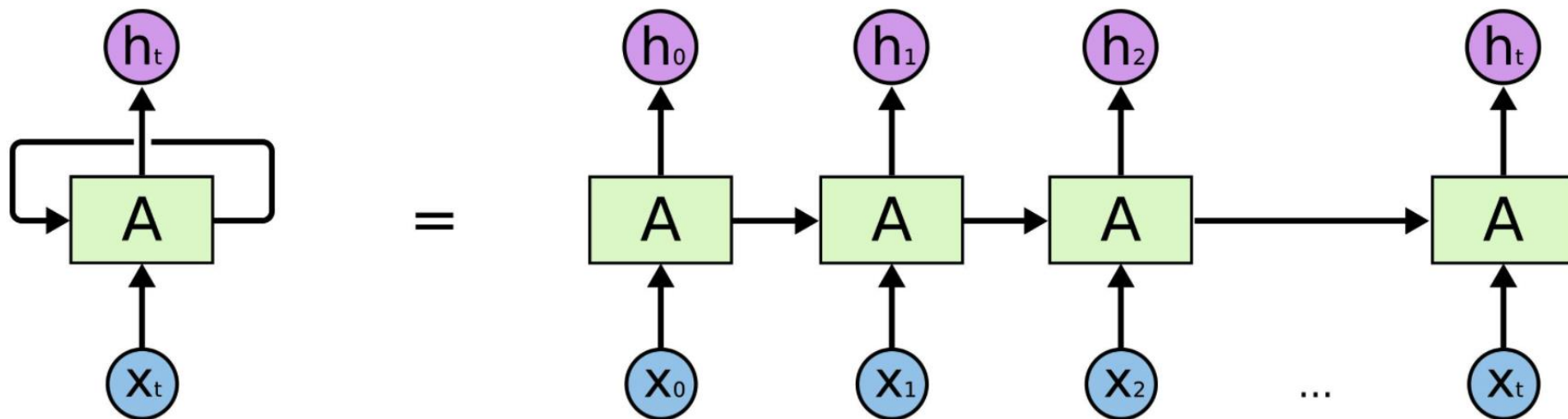


Fig.2 Recurrent Neural Network

Sensitive with sequential data



# Contents

---

1/ Introduction

**2/ Theory of BPNN**

3/ Experimental Result

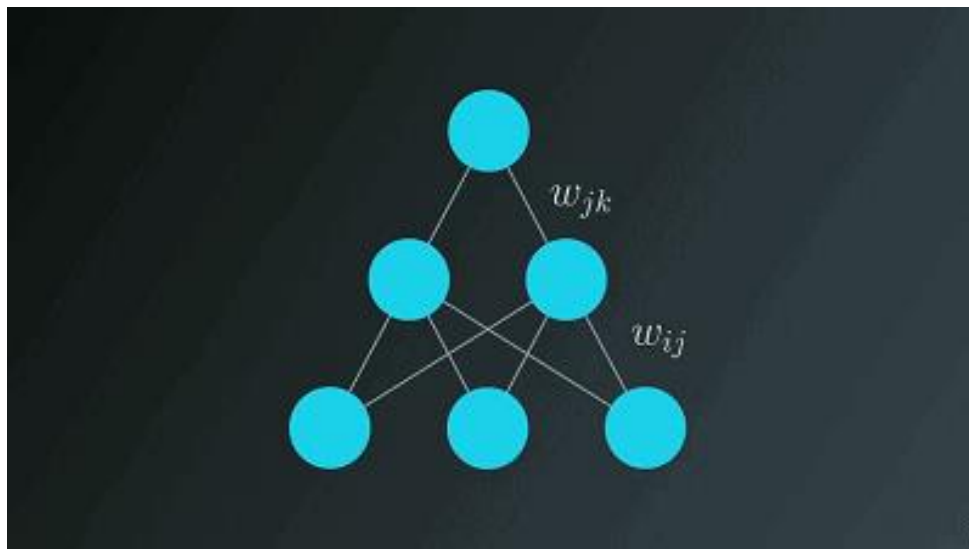
4/ Optimization

---



## Back propagation neural network

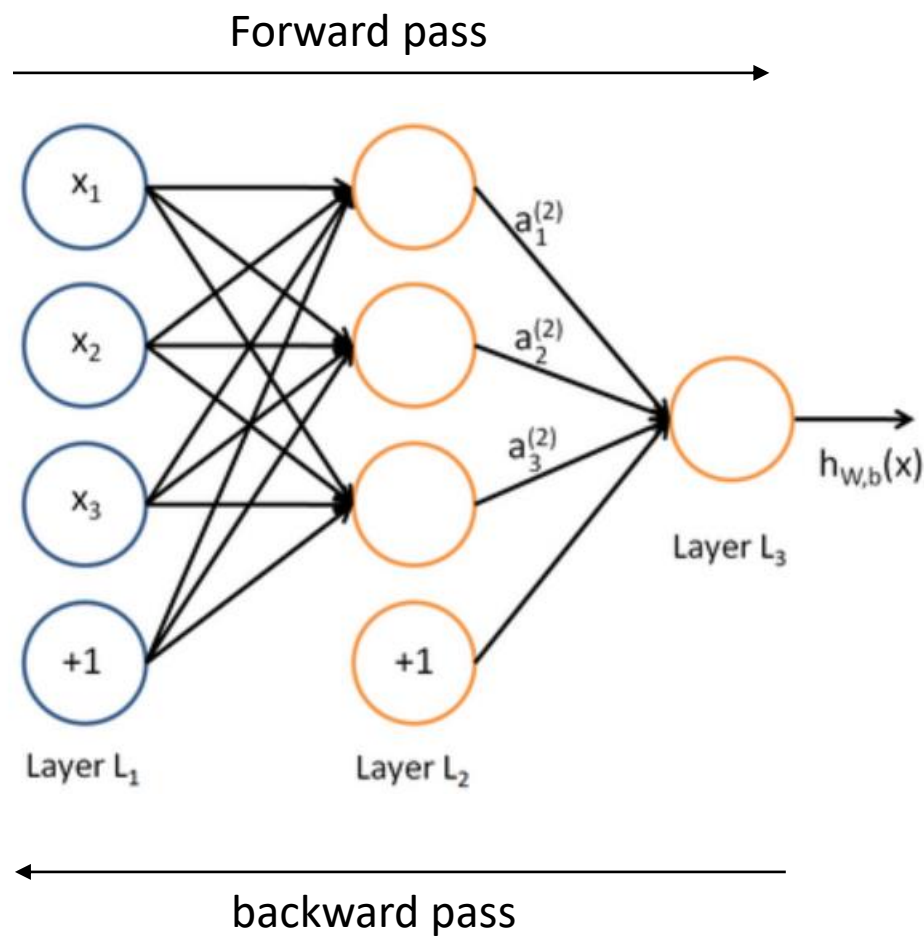
实验要求：三层神经网络（输入层，隐藏层，输出层）





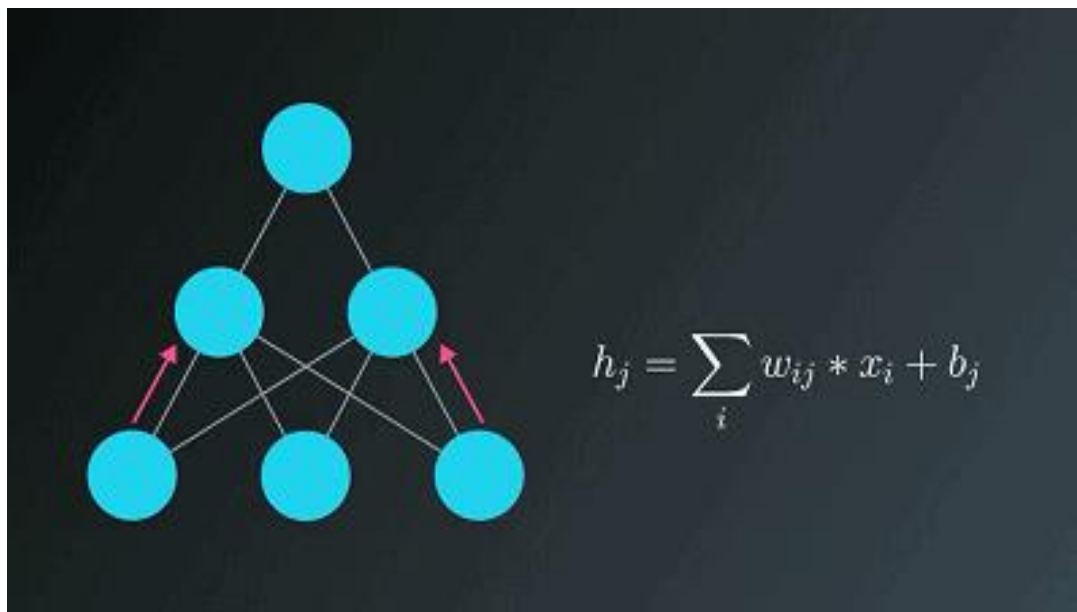


## Back propagation neural network





### Forward pass





### Forward pass

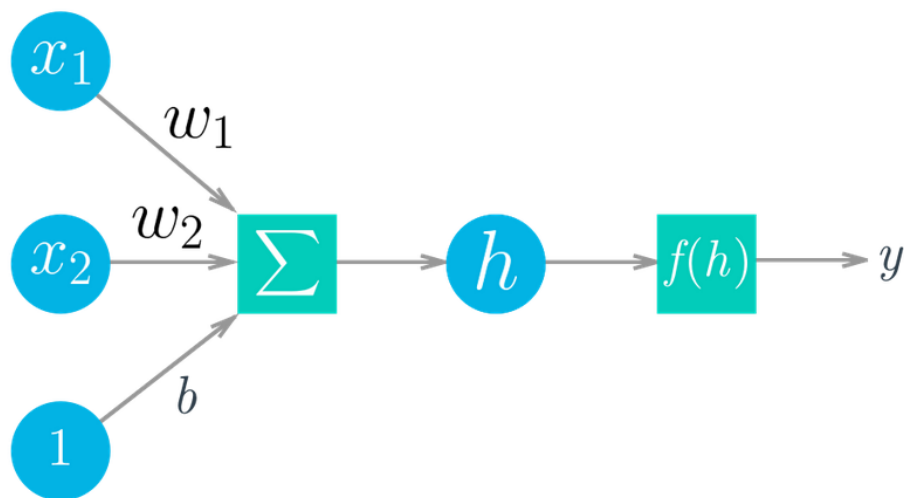


Fig.3 神经网络示意图

在这个架构中 $f(h)$ 称为激活函数，这个函数可以为很多不同的函数，例如如果让 $f(h) = h$ 。则网络的输出为：

$$y = \sum_i w_i x_i + b$$



### Forward pass

将数据从  $(-\infty, \infty)$  映射到  $(0,1)$

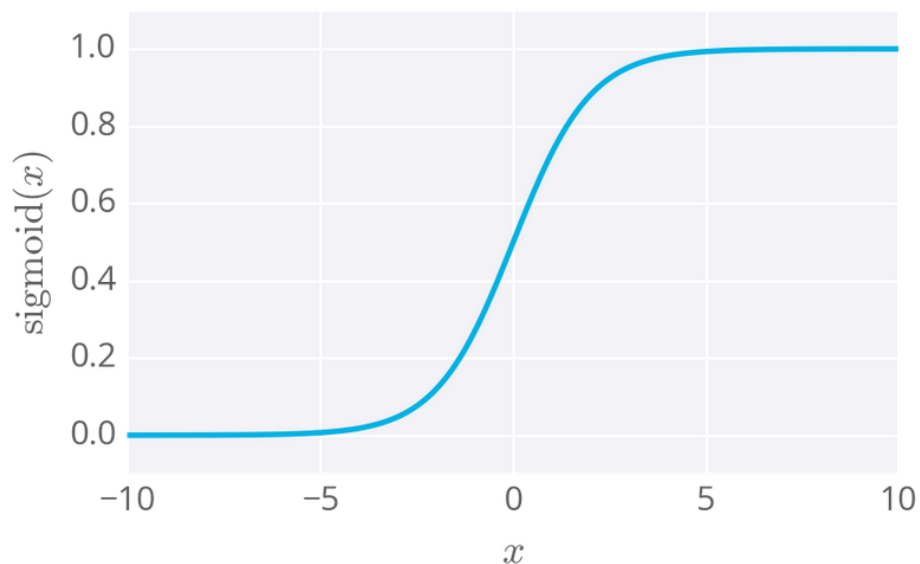


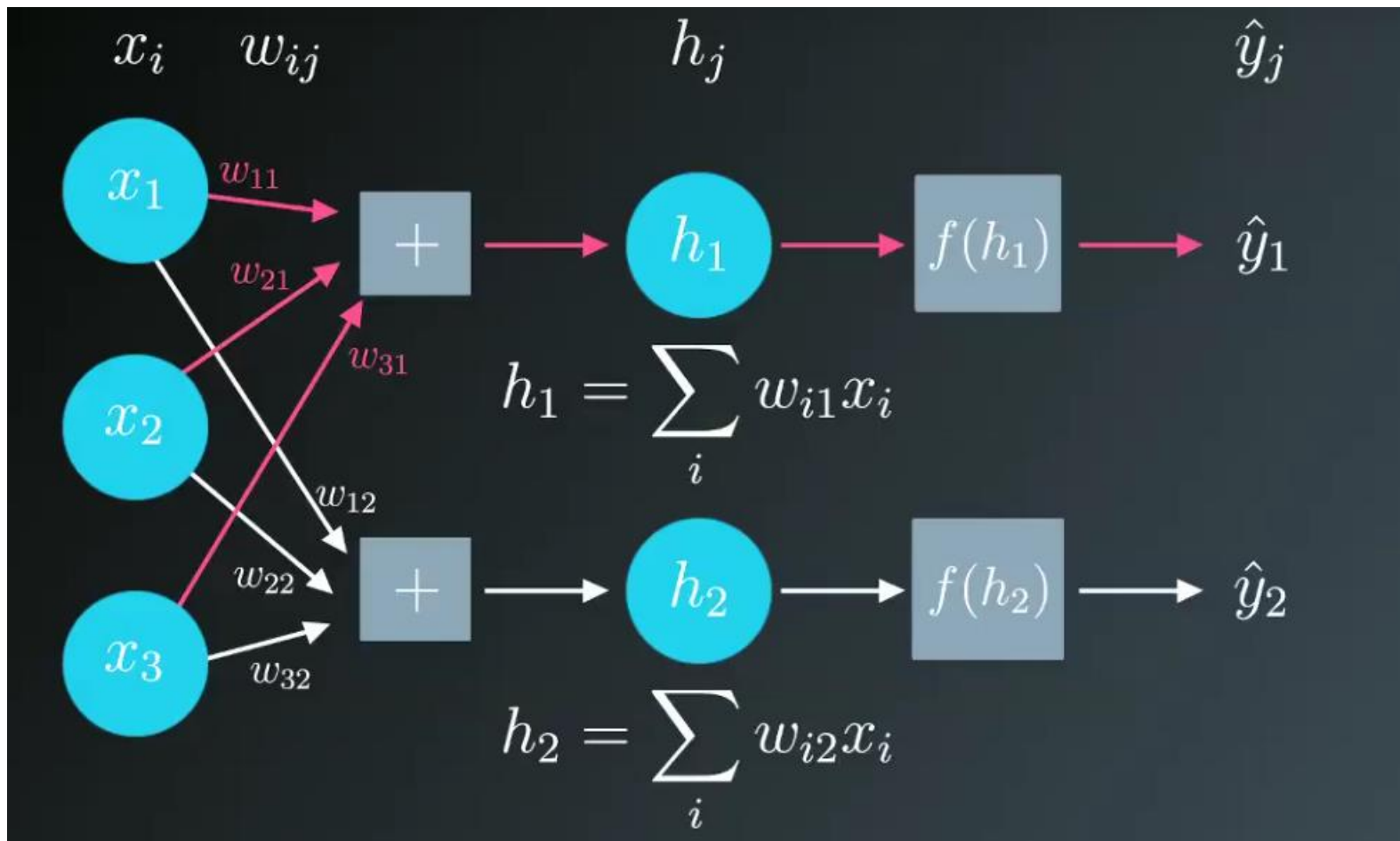
Fig.3 sigmoid函数

公式:

$$\text{sigmoid}(x) = 1 / (1 + e^{-x})$$



### Forward pass





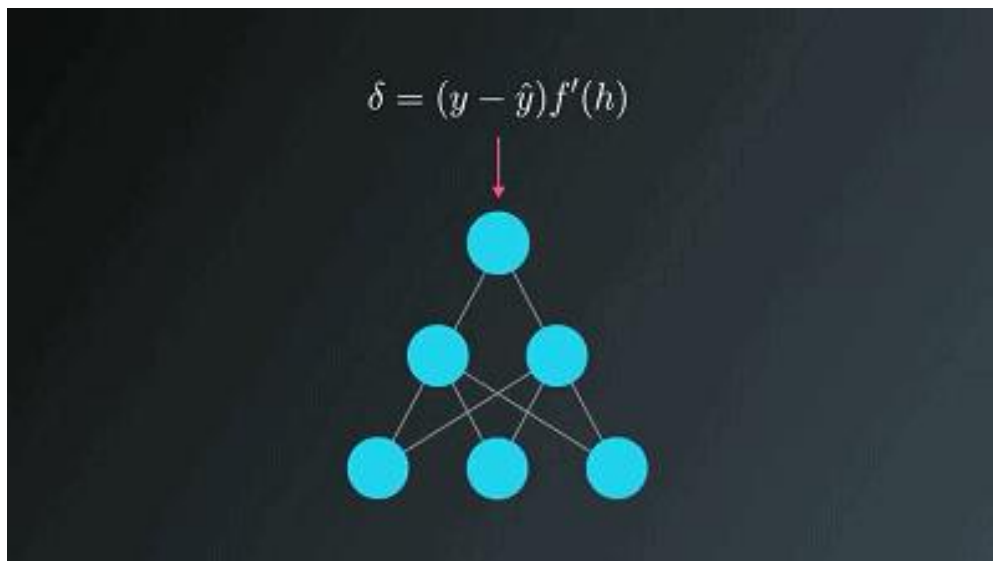
## Forward pass

Loss function

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$



## Backward pass





### Backward pass

Loss function:

$$J(W, b) = \frac{1}{m} \sum_{i=1}^m J(W, b; x^i, y^i) + \frac{\lambda}{2} \sum_{l=1}^{n^l-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} \left( W_{ji}^{(l)} \right)^2$$

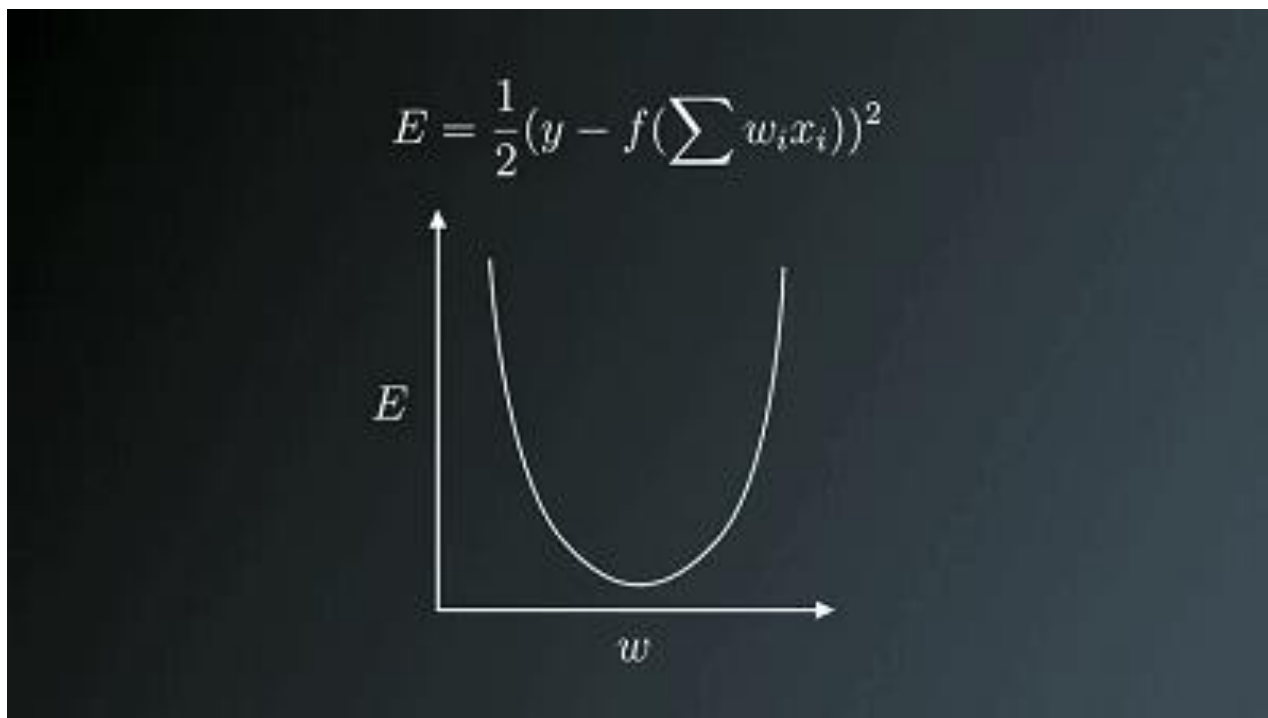
其中:

$$J(W, b; x, y) = \frac{1}{2} \|h_{wb}(x) - y\|^2$$





## Backward pass





### Backward pass

$$\begin{aligned}
 \frac{\partial J(W, b)}{\partial W_{ji}^{(l)}} &= \frac{1}{m} \sum_{t=1}^m \frac{\partial J(W, b; x^t, y^t)}{\partial W_{ji}^{(l)}} + \lambda W_{ji}^{(l)} \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \frac{\partial J(W, b; x^t, y^t)}{\partial Z_j^{l+1}} \times \frac{\partial Z_j^{l+1}}{\partial W_{ji}^{(l)}} \right) + \lambda W_{ji}^{(l)} \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \times \frac{\partial Z_j^{l+1}}{\partial W_{ji}^{(l)}} \right) + \lambda W_{ji}^{(l)} \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \times \frac{\partial \left( \sum_{k=1}^S (W_{jk}^l a_k^l) + b_i^l \right)}{\partial W_{ji}^{(l)}} \right) + \lambda W_{ji}^{(l)} \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \times a_i^l \right) + \lambda W_{ji}^{(l)}
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial J(W, b)}{\partial b_i^l} &= \frac{1}{m} \sum_{t=1}^m \frac{\partial J(W, b; x^t, y^t)}{\partial b_i^l} \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \frac{\partial J(W, b; x^t, y^t)}{\partial Z_j^{l+1}} \times \frac{\partial Z_j^{l+1}}{\partial b_i^l} \right) \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \times \frac{\partial Z_j^{l+1}}{\partial b_i^l} \right) \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \times \frac{\partial \left( \sum_{k=1}^S (W_{jk}^l a_k^l) + b_i^l \right)}{\partial b_i^l} \right) \\
 &= \frac{1}{m} \sum_{t=1}^m \left( \delta_j^{l+1} \right)
 \end{aligned}$$



## Backward pass

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i \propto -\frac{\partial E}{\partial w_i} \longrightarrow \text{THE GRADIENT}$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$



## Backward pass

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} (y - \hat{y})^2$$

$$= \frac{\partial}{\partial w_i} \frac{1}{2} (y - \hat{y}(w_i))^2$$



通过链式求导

$$\frac{\partial}{\partial z} p(q(z)) = \frac{\partial p}{\partial q} \frac{\partial q}{\partial z}$$



## Backward pass

$$\hat{y} = f(h) \text{ where } h = \sum_i w_i x_i$$

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial w_i}$$

$$= -(y - \hat{y}) f'(h) \frac{\partial}{\partial w_i} \sum w_i x_i$$



## Backward pass

$$\begin{aligned} & \frac{\partial}{\partial w_i} \sum_i w_i x_i \\ &= \frac{\partial}{\partial w_1} [w_1 x_1 + w_2 x_2 + \dots + w_n x_n] \\ &= x_1 + 0 + 0 + 0 + \dots \\ & \frac{\partial}{\partial w_i} \sum_i w_i x_i = x_i \end{aligned}$$



## Backward pass

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y})f'(h)x_i$$



$$\delta = (y - \hat{y})f'(h)$$

$$w_i = w_i + \eta\delta x_i$$

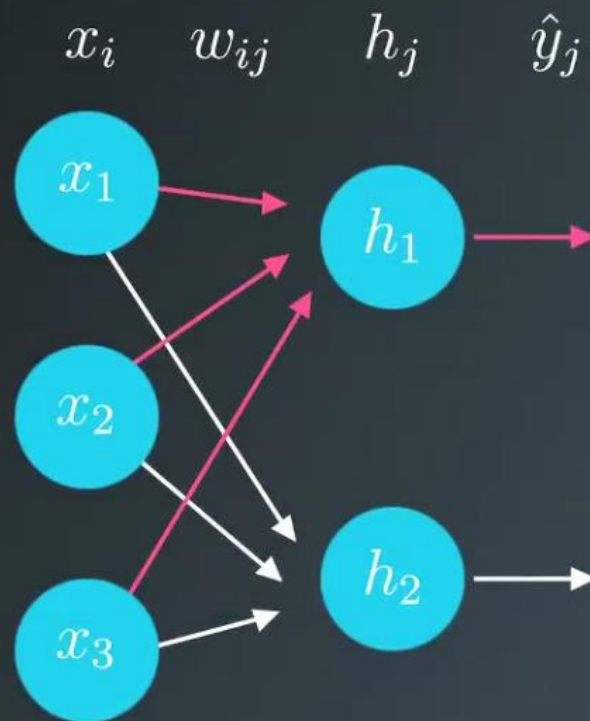


### Backward pass

IF MULTIPLE OUTPUT UNITS:

$$\delta_j = (y_j - \hat{y}_j) f'(h_j)$$

$$\Delta w_{ij} = \eta \delta_j x_i$$







## Backward pass

$h+1$ 层的误差为 $\delta_k^{h+1}$ ， $h$ 层节点 $j$ 的误差即为 $h+1$ 层误差乘以两层间的权重矩阵

$$\delta_j^h = \sum W_j \delta_k^{h+1} f'(h_j)$$

梯度下降与之前相同，只是用当前层的误差

$$\Delta w_{ij} = \eta \delta_j^h x_i$$



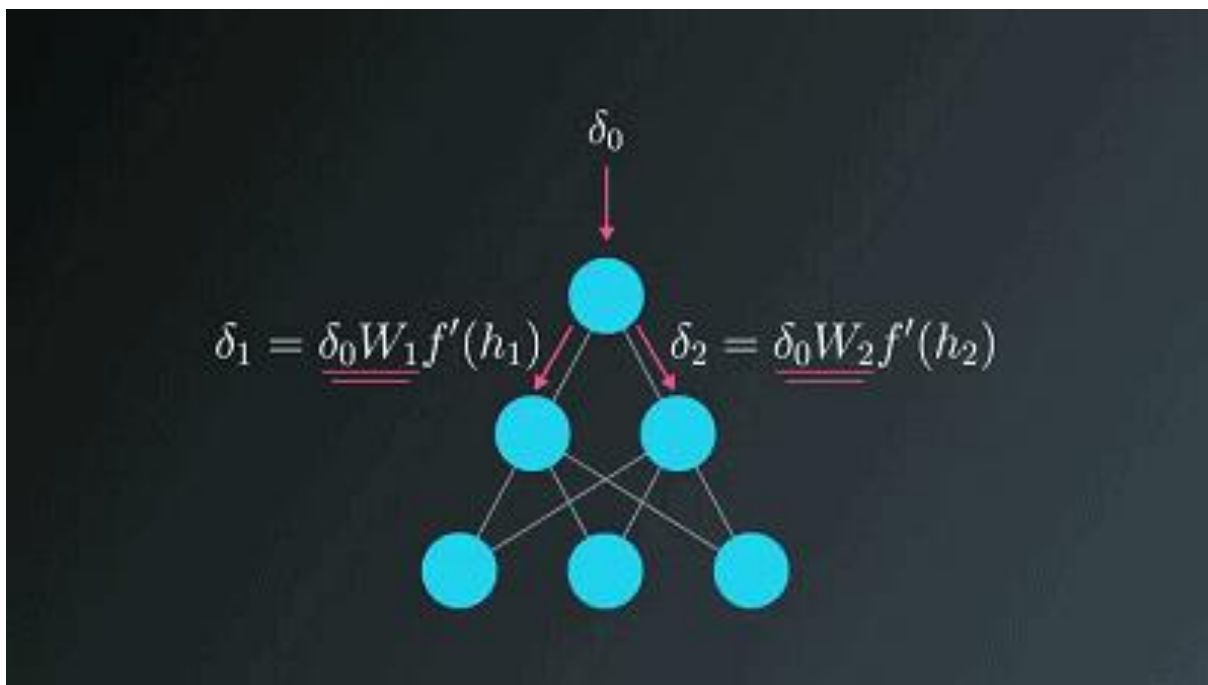
### Backward pass

当  $f(x) = \text{sigmoid}(x)$  时:

$$\begin{aligned}\frac{\partial E_d}{\partial \text{net}_j} &= \sum_{k \in \text{Downstream}(j)} \frac{\partial E_d}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial \text{net}_j} \\&= \sum_{k \in \text{Downstream}(j)} -\delta_k \frac{\partial \text{net}_k}{\partial \text{net}_j} \\&= \sum_{k \in \text{Downstream}(j)} -\delta_k \frac{\partial \text{net}_k}{\partial a_j} \frac{\partial a_j}{\partial \text{net}_j} \\&= \sum_{k \in \text{Downstream}(j)} -\delta_k w_{kj} \frac{\partial a_j}{\partial \text{net}_j} \\&= \sum_{k \in \text{Downstream}(j)} -\delta_k w_{kj} a_j (1 - a_j) \\&= -a_j (1 - a_j) \sum_{k \in \text{Downstream}(j)} \delta_k w_{kj}\end{aligned}$$



## Backward pass





# Contents

---

1/ Introduction

2/ Theory of BPNN

**3/ Dataset of the project**

4/ Optimization

---



## Background of the project

In this project, you'll build your first neural network and use it to predict daily bike rental ridership.

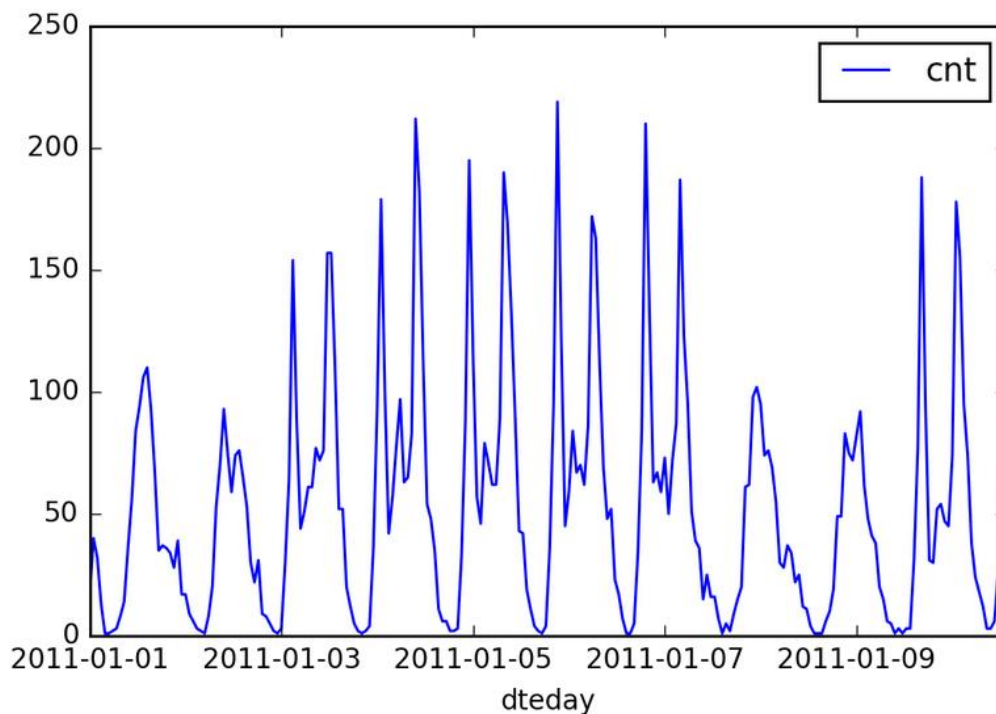


Fig. 4 Subset of the data



## Load and prepare the data

```
In [2]: data_path = 'Bike-Sharing-Dataset/hour.csv'

rides = pd.read_csv(data_path)
```

```
In [3]: rides.head()
```

Out[3]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday
0	1	2011-01-01	1	0	1	0	0	6
1	2	2011-01-01	1	0	1	1	0	6
2	3	2011-01-01	1	0	1	2	0	6
3	4	2011-01-01	1	0	1	3	0	6
4	5	2011-01-01	1	0	1	4	0	6

Fig. 8 show the dataset



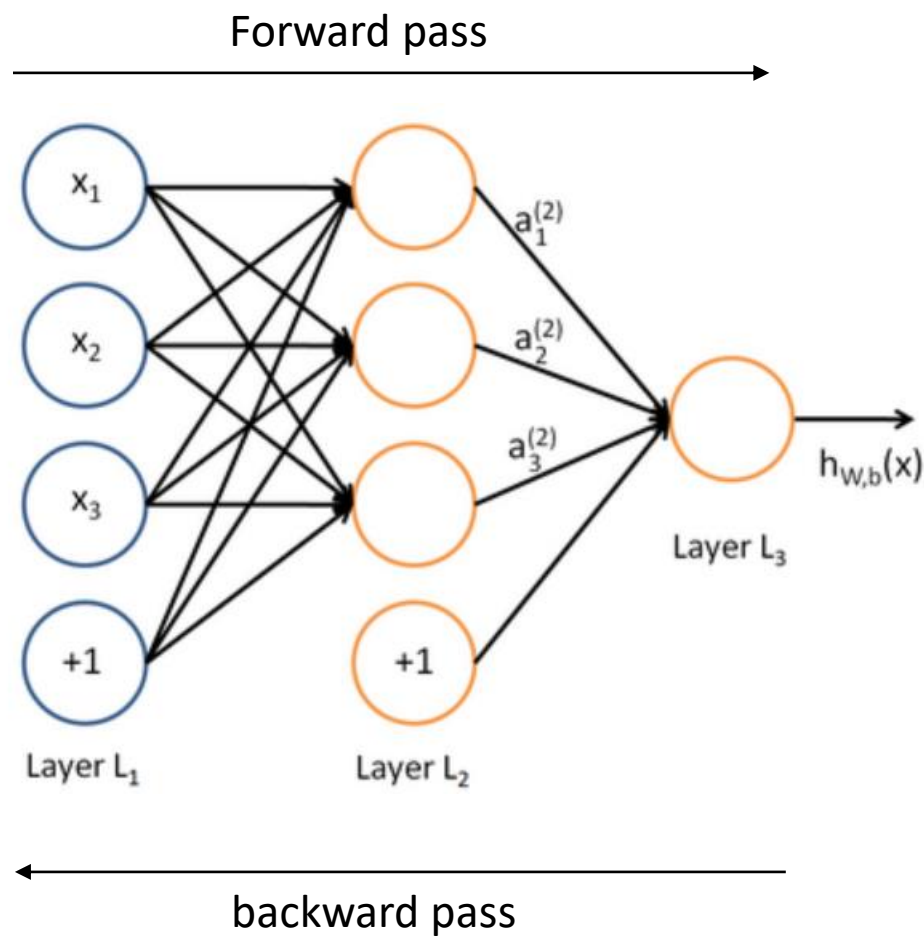
## Build the Network

### TODO list:

- 把每一层权重更新的初始步长设置为 0
  - 输入到隐藏层的权重更新是  $\Delta w_{ij} = 0$
  - 隐藏层到输出层的权重更新是  $\Delta W_j = 0$
- 对训练数据当中的每一个点
  - 让它正向通过网络, 计算输出  $\hat{y}$
  - 计算输出节点的误差梯度  $\delta^o = (y - \hat{y})f'(z)$  这里  $z = \sum_j W_j a_j$  是输出节点的输入。
  - 误差传播到隐藏层  $\delta_j^h = \delta^o W_j f'(h_j)$
  - 更新权重步长:
    - $\Delta W_j = \Delta W_j + \delta^o a_j$
    - $\Delta w_{ij} = \Delta w_{ij} + \delta_j^h a_i$
- 更新权重, 其中  $\eta$  是学习率,  $m$  是数据点的数量:
  - $W_j = W_j + \eta \Delta W_j / m$
  - $w_{ij} = w_{ij} + \eta \Delta w_{ij} / m$
- 重复这个过程  $e$  代。



## Training and Validating







# Contents

---

1/ Introduction

2/ Theory of BPNN

3/ Experimental Result

4/ **Optimization**

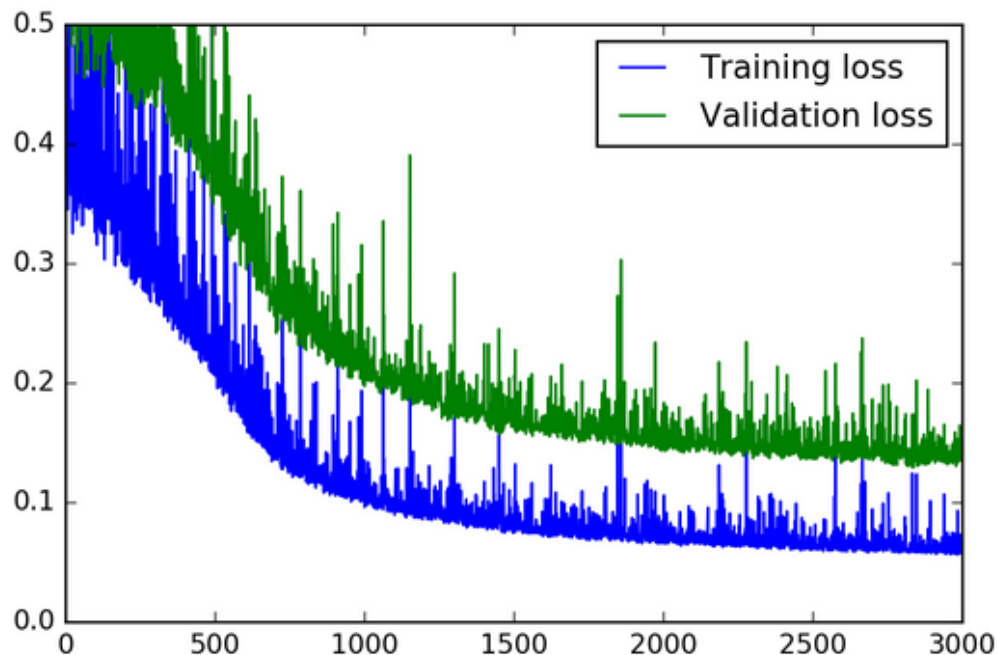
---



# Training and Validating

```
In [12]: plt.plot(losses['train'], label='Training loss')  
plt.plot(losses['validation'], label='Validation loss')  
plt.legend()  
plt.ylim(ymax=0.5)
```

Out[12]: (0.0, 0.5)



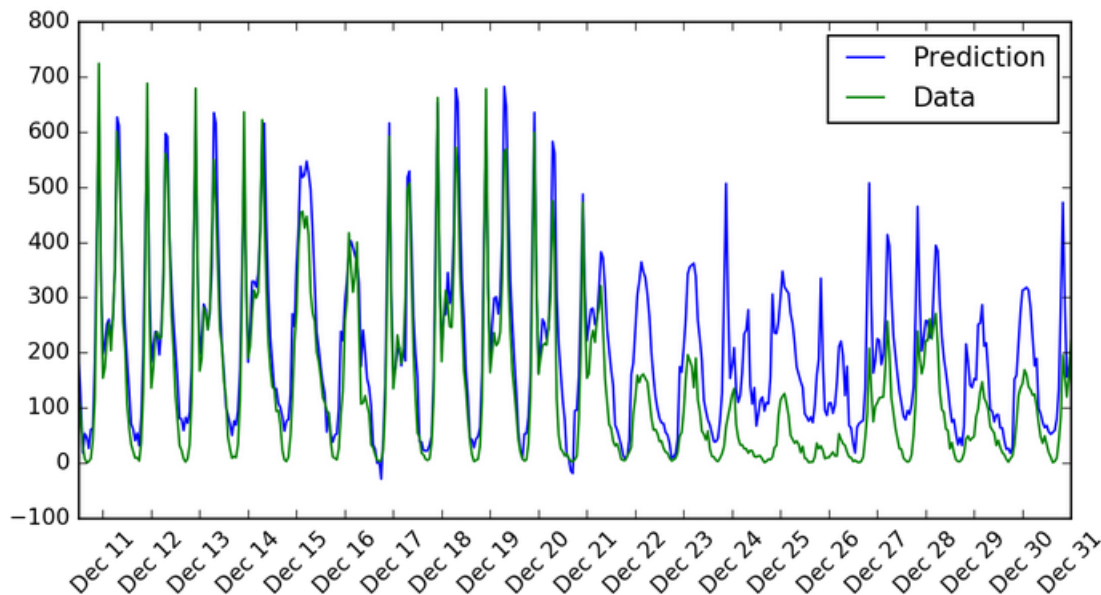


# Training and Validating

```
In [13]: fig, ax = plt.subplots(figsize=(8, 4))

mean, std = scaled_features['cnt']
predictions = network.run(test_features)*std + mean
ax.plot(predictions[0], label='Prediction')
ax.plot((test_targets['cnt']*std + mean).values, label='Data')
ax.set_xlim(right=len(predictions))
ax.legend()

dates = pd.to_datetime(rides.ix[test_data.index]['dteday'])
dates = dates.apply(lambda d: d.strftime('%b %d'))
ax.set_xticks(np.arange(len(dates))[12::24])
_ = ax.set_xticklabels(dates[12::24], rotation=45)
```



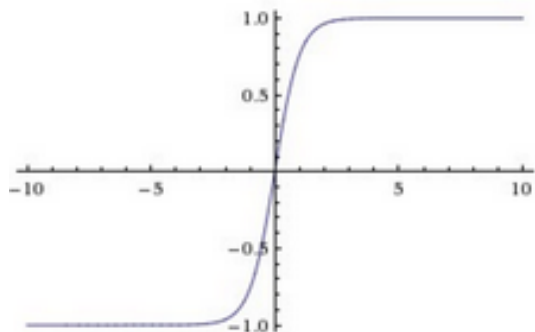


### L2正则化

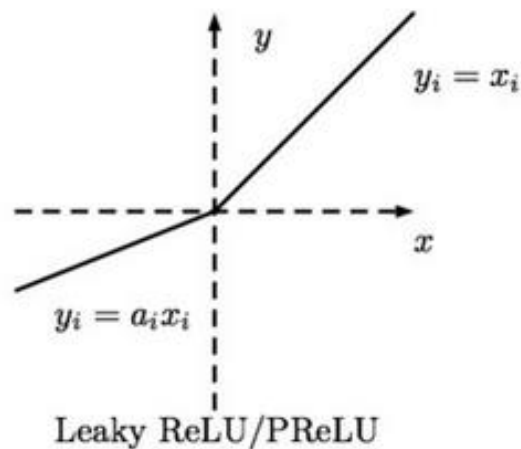
$$J(W, b) = \frac{1}{m} \sum_{i=1}^m J(W, b; x^i, y^i) + \frac{\lambda}{2} \sum_{l=1}^{n^l-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (W_{ji}^{(l)})^2$$

### 不同的激活函数

Tanh



Relu





数据预处理

多层深度神经网络

不同神经网络架构

Mini-batch



## 任务布置

- 必须实现三层神经网络（输入层，隐藏层，输出层）
- 必须在给出的优化建议中任意选择一项实现
- 自己划分验证集（报告里说明是怎么分的）调整参数
- 在测试集上预测，提交预测结果



## 思考题

- 尝试说明下其他激活函数的优缺点。
- 有什么方法可以实现传递过程中不激活所有节点？
- 梯度消失和梯度爆炸是什么？可以怎么解决？



## 注意事项

- 实验报告截止日期:
- **2017.12.06 晚 23:59:59 前**提交至 FTP 文件夹
- 提交文件:
  - 测试集结果: 15\*\*\*\*\*\_wangxiaoming.txt 每一行对应的是测试样例的标签。
  - 实验报告: 15\*\*\*\*\*\_wangxiaoming.pdf
  - 代码: 15\*\*\*\*\*\_wangxiaoming.zip 如果代码分成多个文件, 最好写份readme





# THANKS

---

