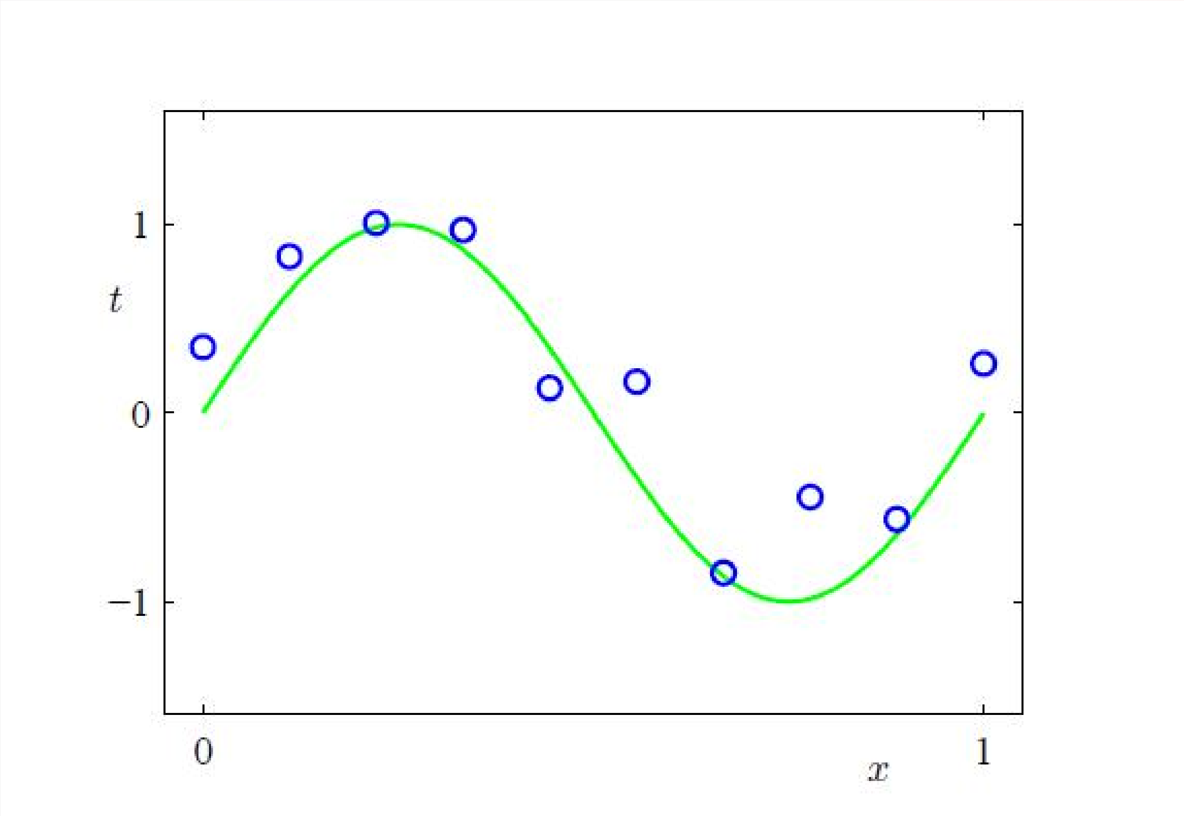
大数据分析与应用

第6章 回归方法

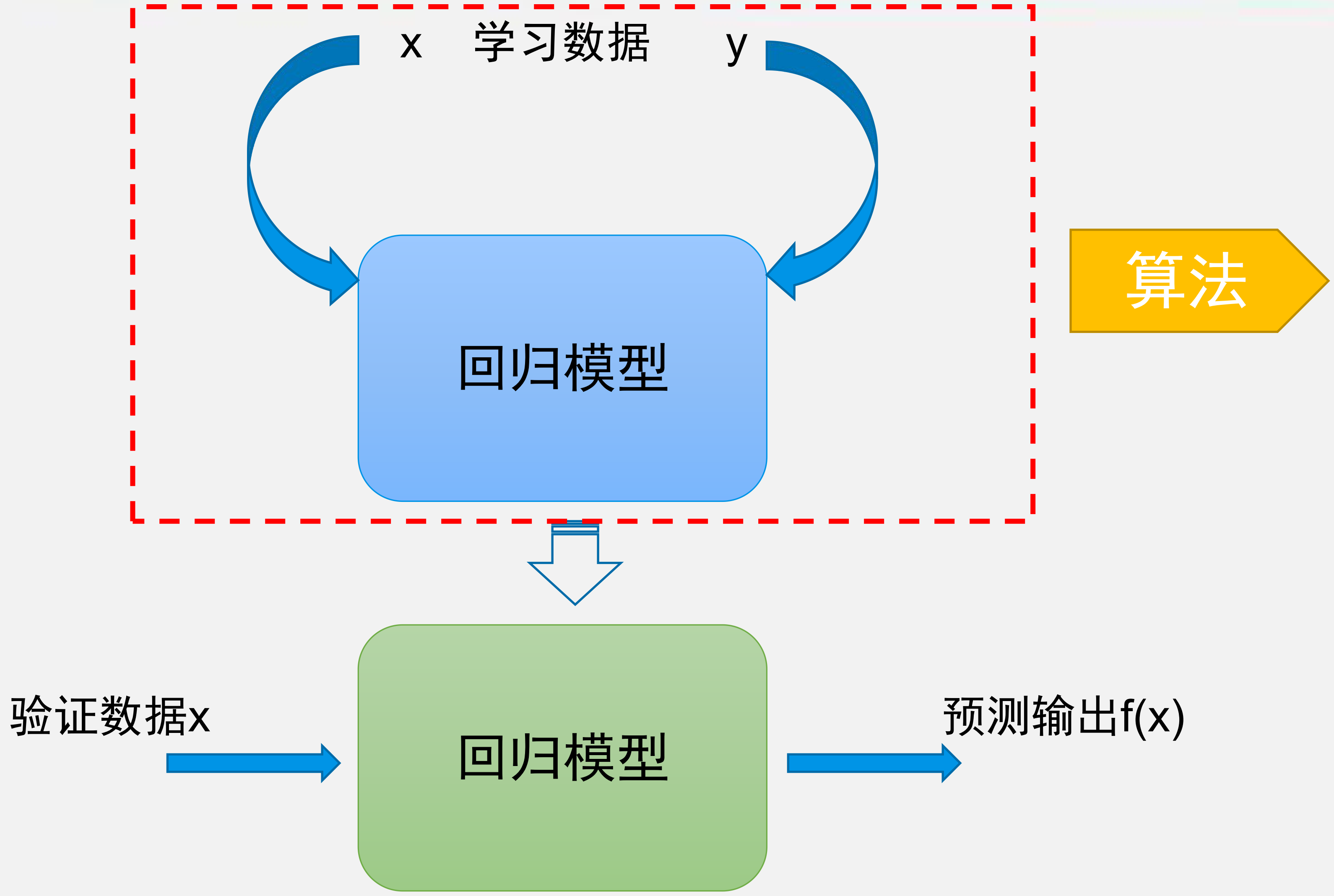


- 我们的目标是对于某些新的x值，预测t的值，而无需知道绿色曲线。

- 回归问题的应用场景：
  - 空气质量预测
  - 微博评论数预测
  - 优惠券使用间隔预测
  - 点击量预测

- 数量化的问题（输出可以相互之间比较大小的量值），一般首先尝试使用回归方法进行分析。

第6章 回归方法：一般步骤

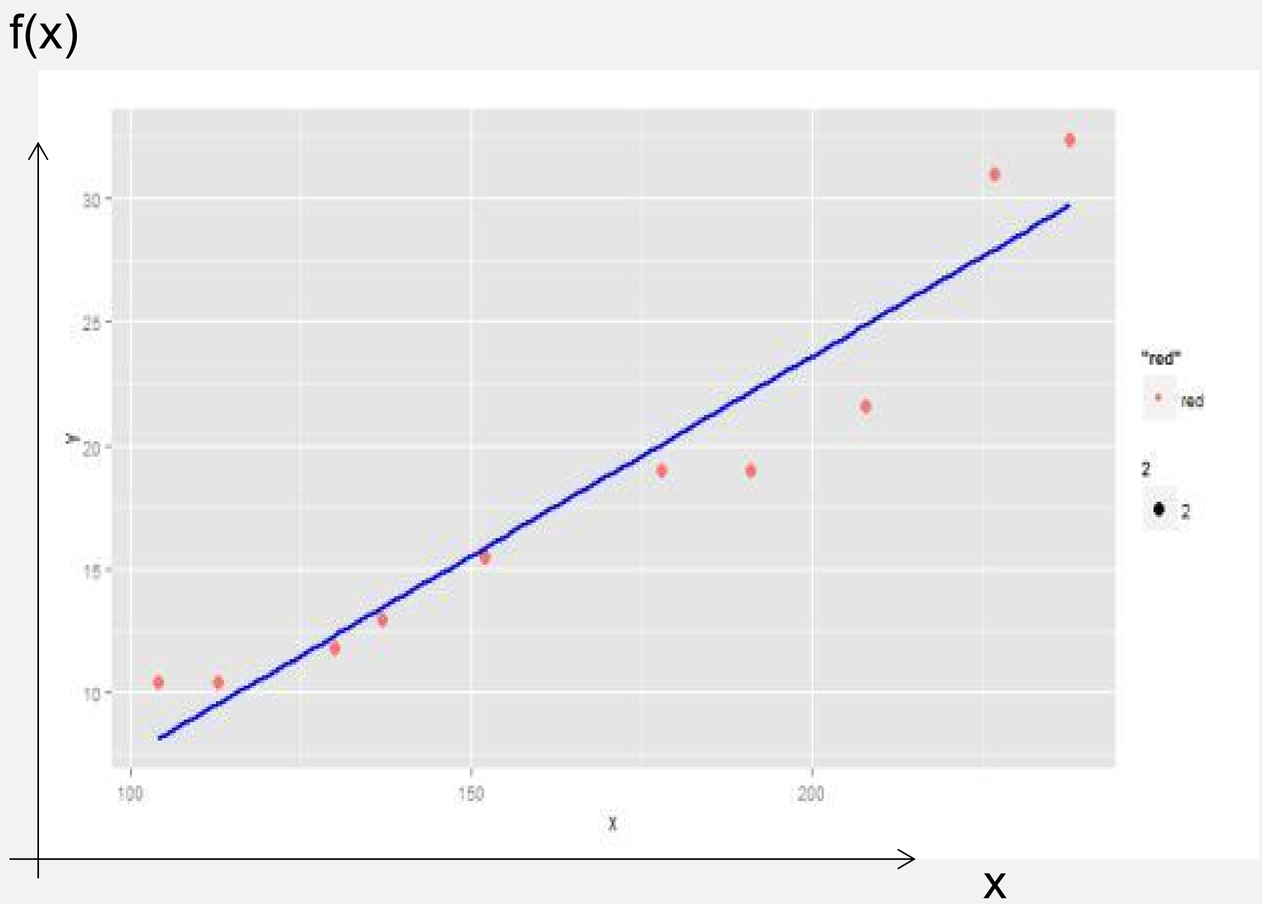


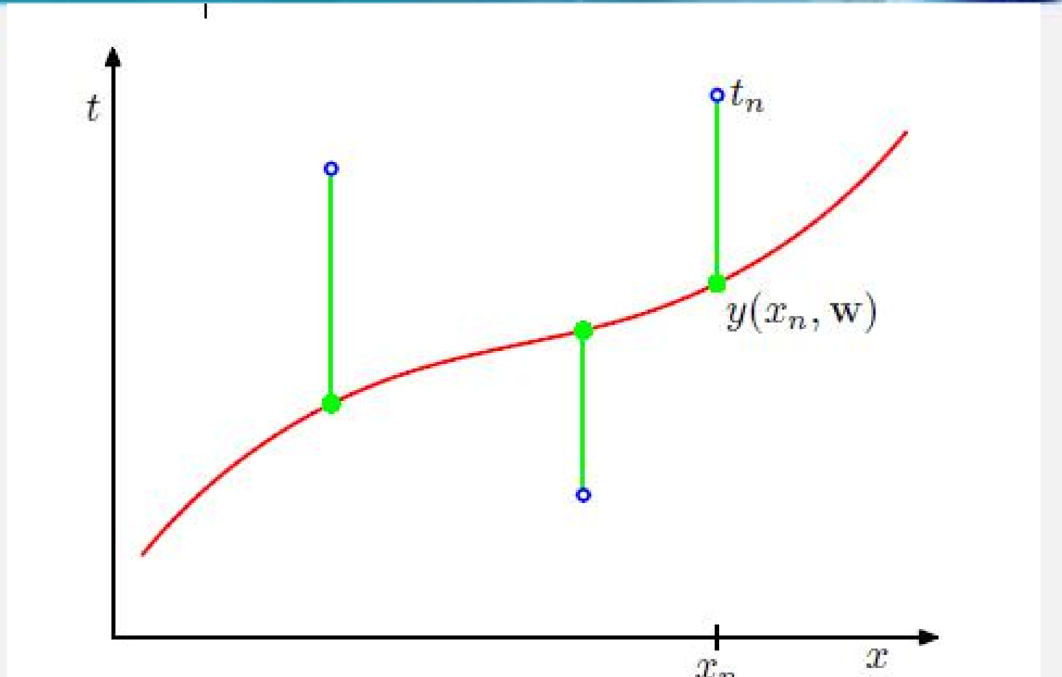
- 历史的提法：
  - 最小二乘方法
  - 极大似然方法
  - LMS算法（least mean square）
- 现代观点
  - 1构建损失函数
  - 2.构建解析解/计算数值解

第6章 回归方法:目录

- 6.1 最小二乘方法：给出均方误差损失函数下的解析解；
- 6.2 LMS算法：给出均方误差损失函数下的数值解；
- 6.3 极大似然方法：损失函数为极大似然函数；
- 6.4 正则化：如何避免过拟合；
- 6.5 时间序列问题：一类常见且具有特殊性质的数据；
- 6.6 神经网络：复杂模型下的数值解；

6.1最小二乘方法：损失函数





问题1：怎样定义预测的准确：误差函数（损失函数）

6.1最小二乘方法：模型表达

- 问题2：函数的形式

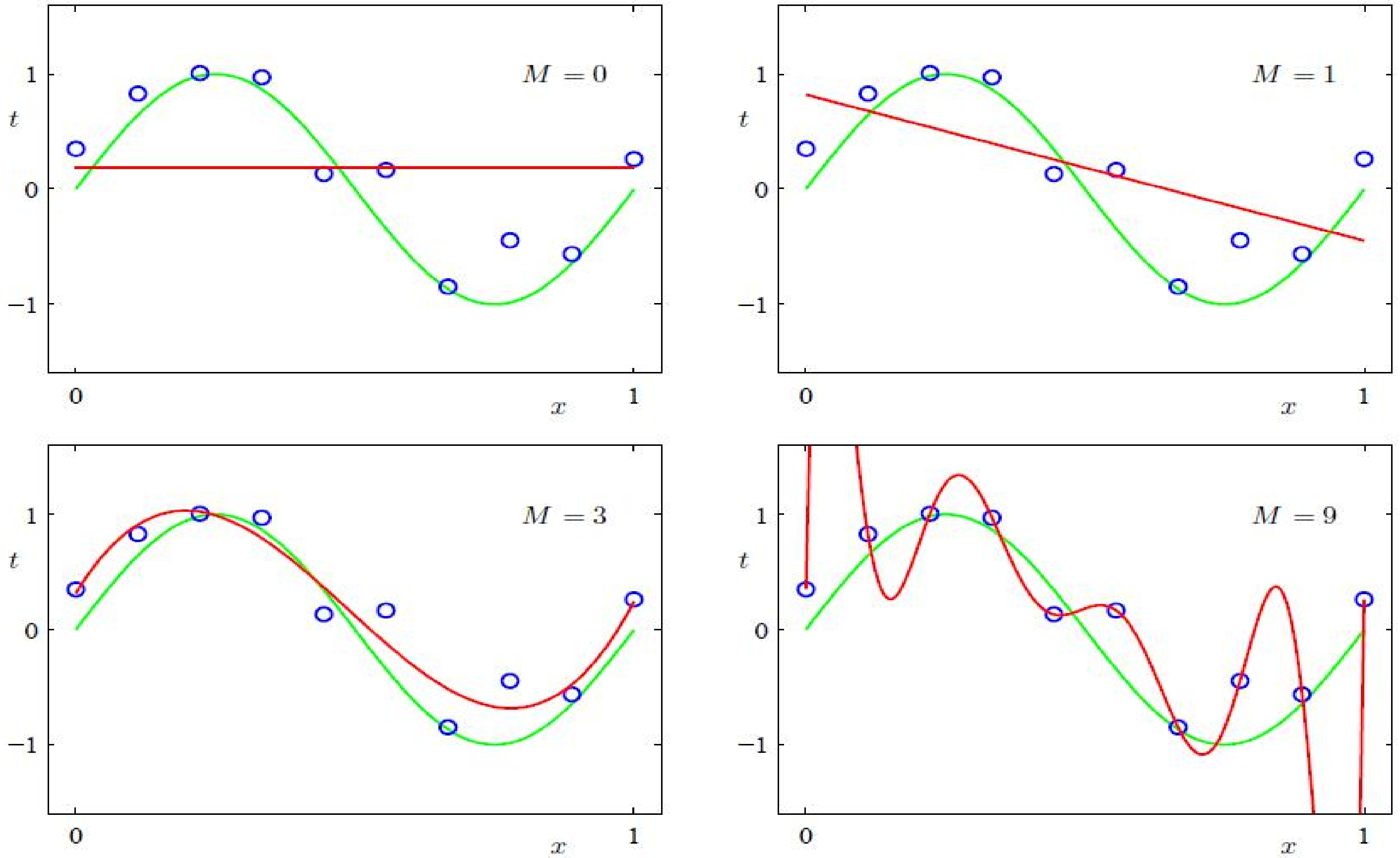
✓最基本的选择：线性模型

✓线性模型下具体的选择：

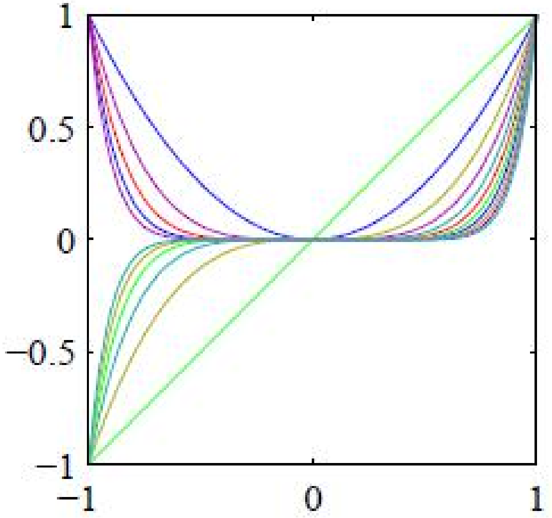
- 问题2.1选什么样的基函数
- 问题2.2基函数的阶数

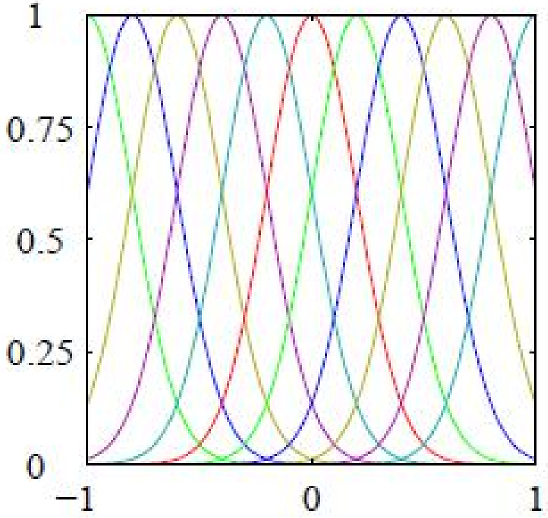
●以上选择构成了多项式模型

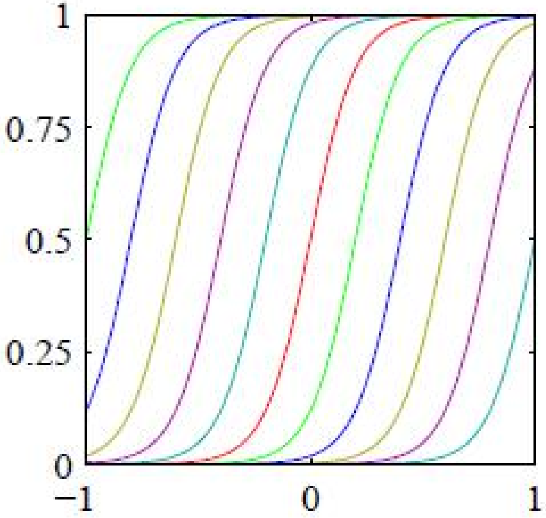
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



6.1最小二乘方法：模型表达







$$\phi_j(x) = x^j$$

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

通用性强

特征空间局部
性强

开关性质

其他基函数：
无线信号处理，傅里叶基函数
当应用中的输入值位于正规的网格中时（图像处理），应用小波基函数最合适

6.1最小二乘方法：解析解的获得

- 由于非线性可以通过基函数表达，等价于特征变化换，所以接下来只讨论线性模型问题

$$(w^*, b^*) = \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} \left(f(x_i) - y_i\right)^2 = \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} (y_i - w x_i - b)^2 .$$

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w \sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m} (y_i - b) x_i\right) ,$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m} (y_i - w x_i)\right) ,$$

令偏导数等于0

$$w = \frac{\sum_{i=1}^{m} y_i (x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2} ,$$

$$b = \frac{1}{m} \sum_{i=1}^{m} (y_i - w x_i) ,$$

6.1最小二乘方法：向量形式的解析解

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b,$$

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix}$$

$$\hat{\boldsymbol{w}}^* = \underset{\hat{\boldsymbol{w}}}{\arg\min} \, (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})$$

$$\frac{\partial E_{\hat{\boldsymbol{w}}}}{\partial \hat{\boldsymbol{w}}} = 2\, \mathbf{X}^{\mathrm{T}} (\mathbf{X}\hat{\boldsymbol{w}} - \boldsymbol{y})$$

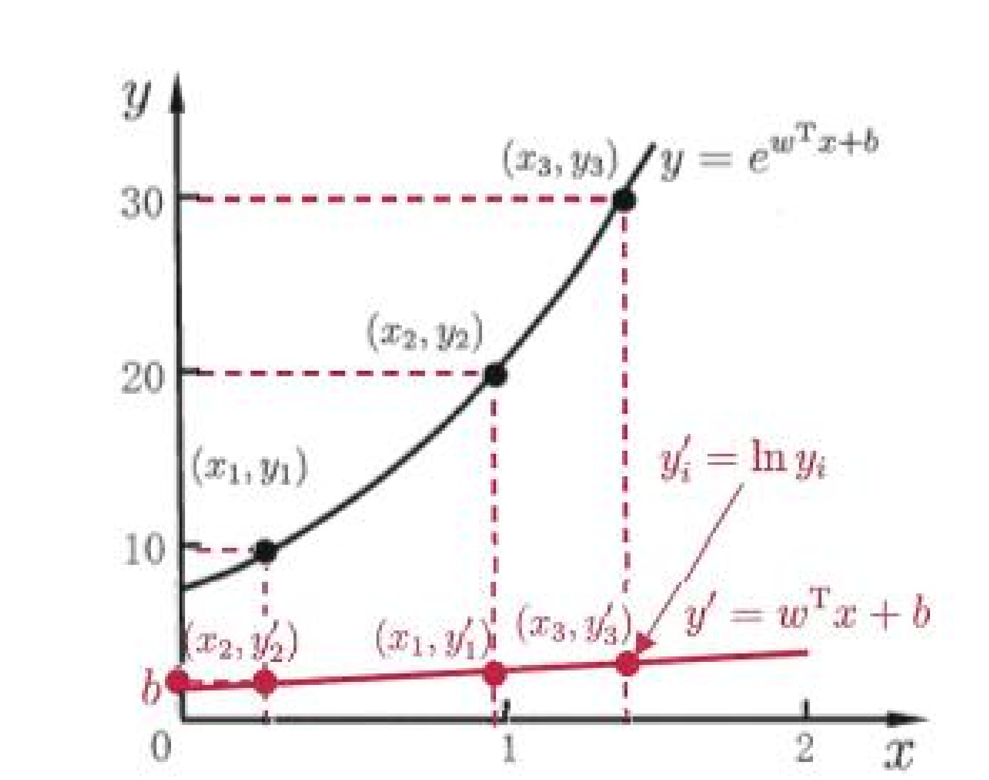
$$\hat{\boldsymbol{w}}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \mathbf{X}^{\mathrm{T}} \boldsymbol{y} ,$$

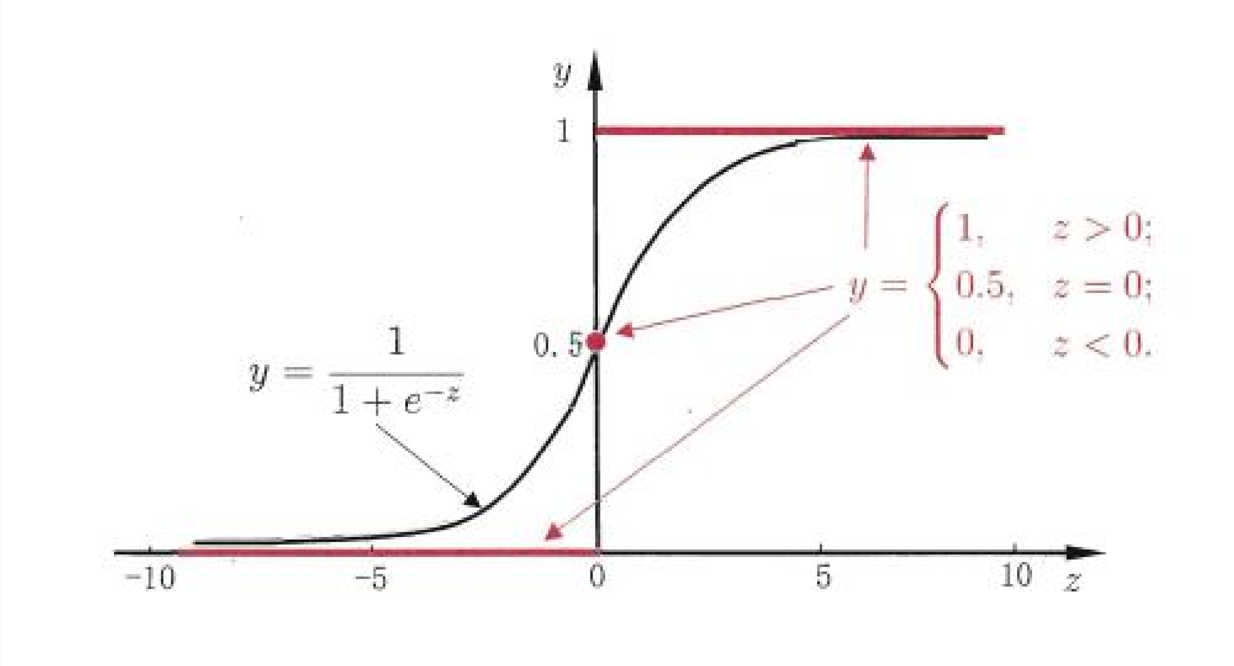
$$f(\hat{\boldsymbol{x}}_i) = \hat{\boldsymbol{x}}_i^{\mathrm{T}} (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \mathbf{X}^{\mathrm{T}} \boldsymbol{y} .$$

6.* 再看logistics regression / SGD

$$\ln y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \ .$$

$$\ln \frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \ .$$





$$y = \frac{1}{1+e^{-z}} \ .$$

6.* 再看logistics regression / SGD

$$y = \frac{1}{1 + e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}} .$$

$$\ln \frac{p(y=1 \mid \boldsymbol{x})}{p(y=0 \mid \boldsymbol{x})} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b .$$

$$p(y=1 \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}} ,$$

$$p(y=0 \mid \boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}} .$$

$$L(w) = \sum y_i \ln P(y_i = 1 \mid x_i) + (1-y_i) \ln P(y_i = 0 \mid x_i)$$

- 偏导数形势复杂，难以得到解析解
- 数据点逐个到达

$$\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}^t - \left(\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}^{\mathrm{T}}}\right)^{-1} \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

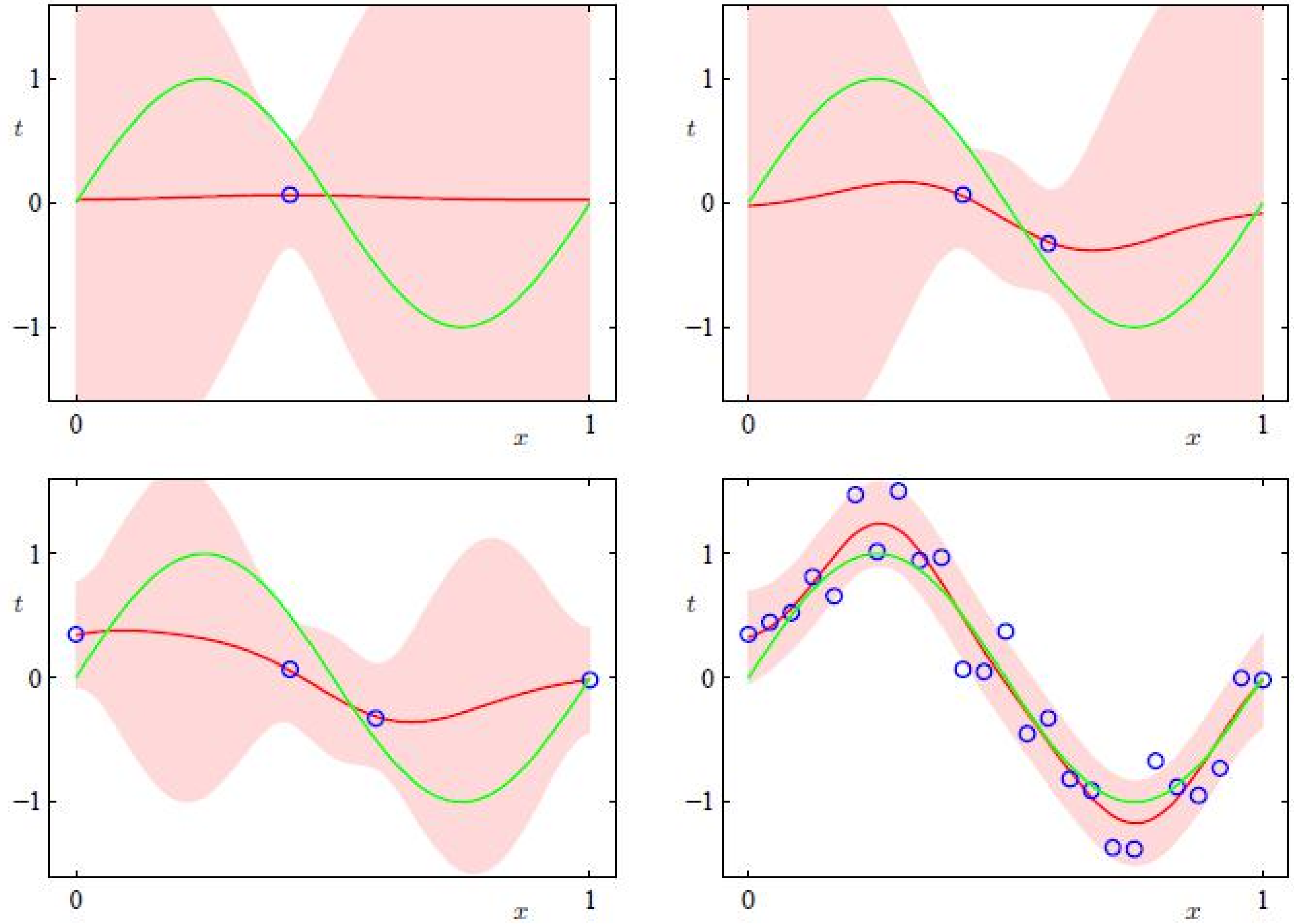
$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\sum_{i=1}^{m} \hat{\boldsymbol{x}}_i (y_i - p_1(\hat{\boldsymbol{x}}_i; \boldsymbol{\beta})) ,$$

$$\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^{\mathrm{T}}} = \sum_{i=1}^{m} \hat{\boldsymbol{x}}_i \hat{\boldsymbol{x}}_i^{\mathrm{T}} p_1(\hat{\boldsymbol{x}}_i; \boldsymbol{\beta})(1 - p_1(\hat{\boldsymbol{x}}_i; \boldsymbol{\beta})) .$$

6.2 LMS（最小均方算法,SGD的旧有提法）

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$$

$$w^{(\tau+1)} = w^{(\tau)} + \eta(t_n - w^{(\tau)T}\phi_n)\phi_n$$



- 优点：
- 数据可以是逐个到达的
- 模型可以是变化的

6.3 极大似然损失函数

- 假设y的观测由真实的系统输出+高斯噪声得到 $y = w^T x + e$
- y的后验概率： $p(y|x;w) = \prod (2\pi\sigma^2)^{-1/2} \exp[(y_i - w^T x_i)^2/\sigma^2]$
- 极小化损失函数等价于极大化后验概率：

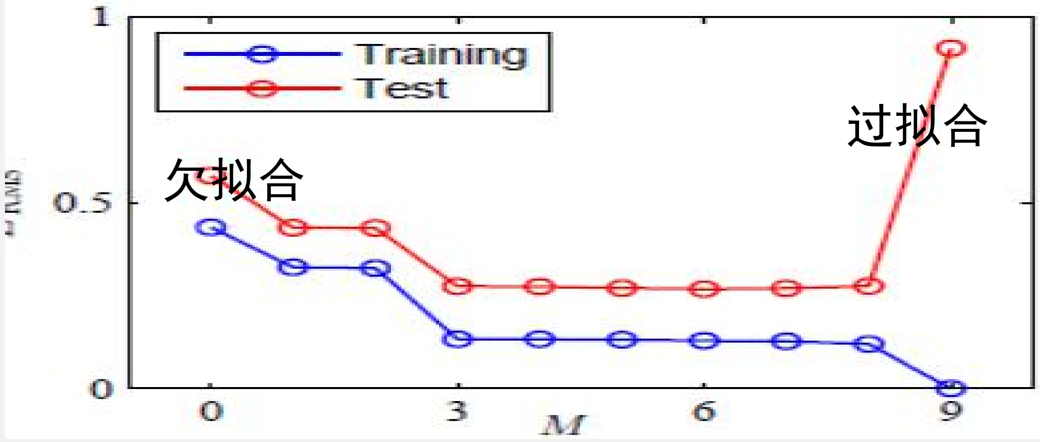
$$\ln p(y|x,w) = \sum \ln \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-(y_i - w^T x_i)^2/\sigma^2]$$

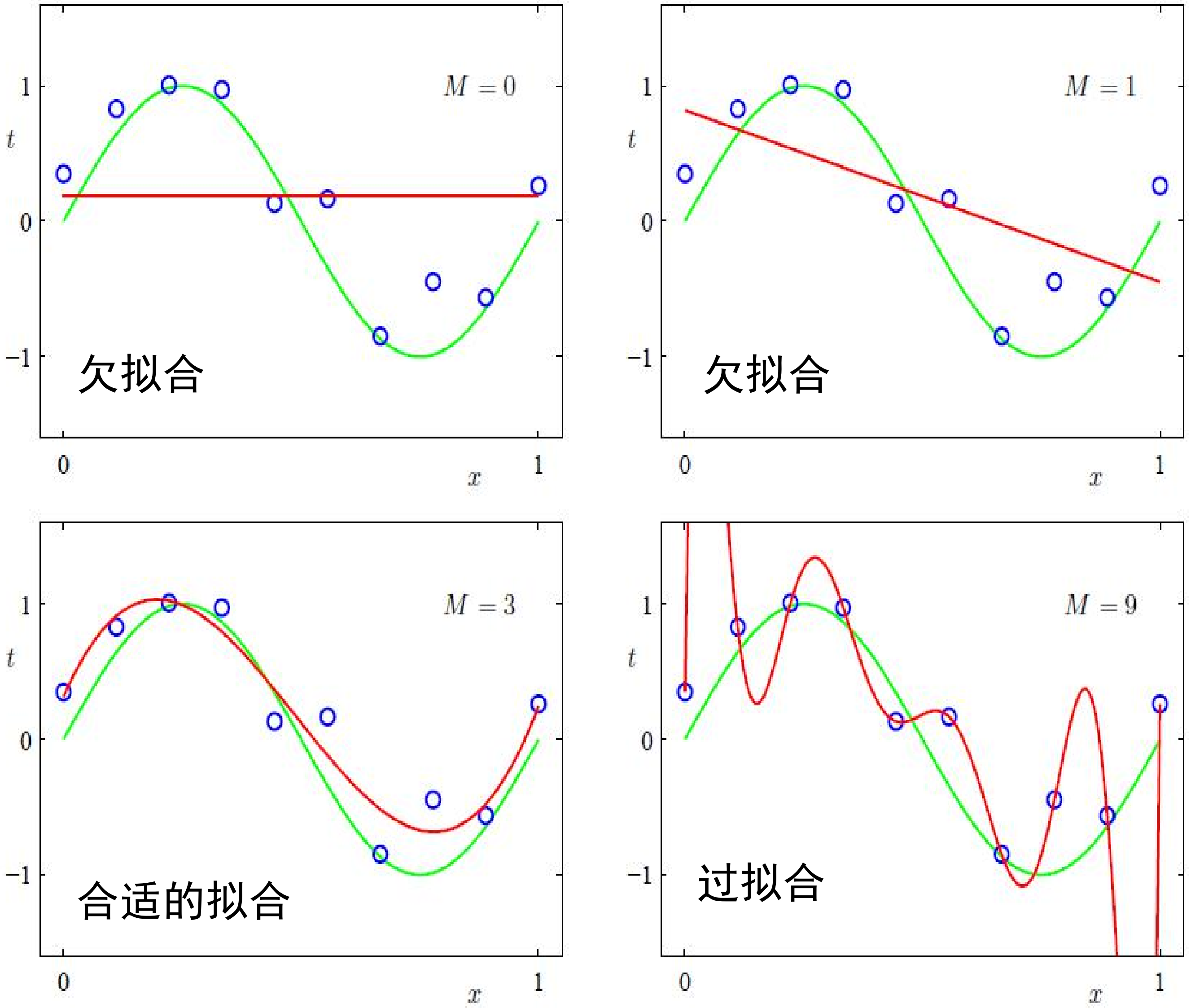
$$= \sum \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \sum (y_i - w^T x_i)^2/\sigma^2$$

结论：最小二乘方法是极大似然方法在高斯分布不确定性假设下的特例；

6.4 正则化：模型阶数的问题

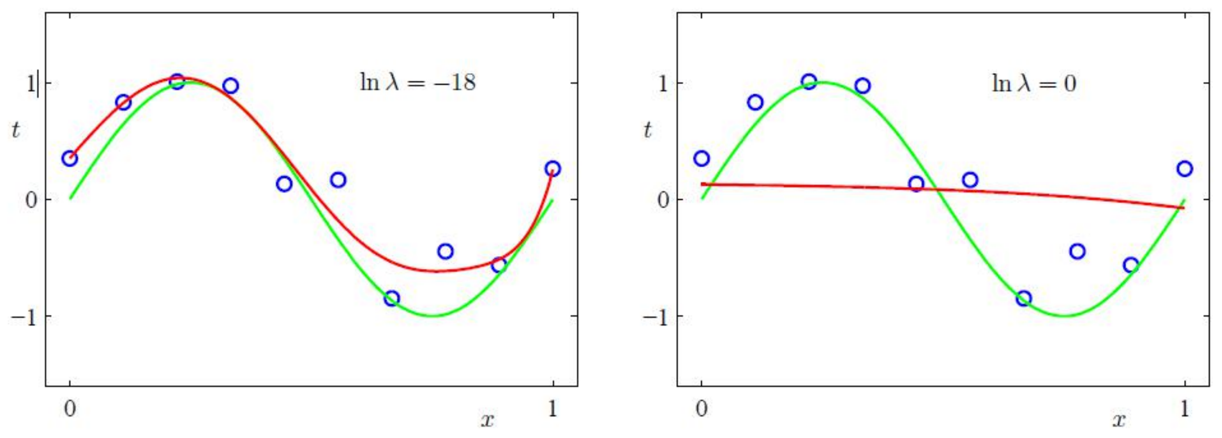
	$M=0$	$M=1$	$M=3$	$M=9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43





6.4 正则化：模型阶数的问题

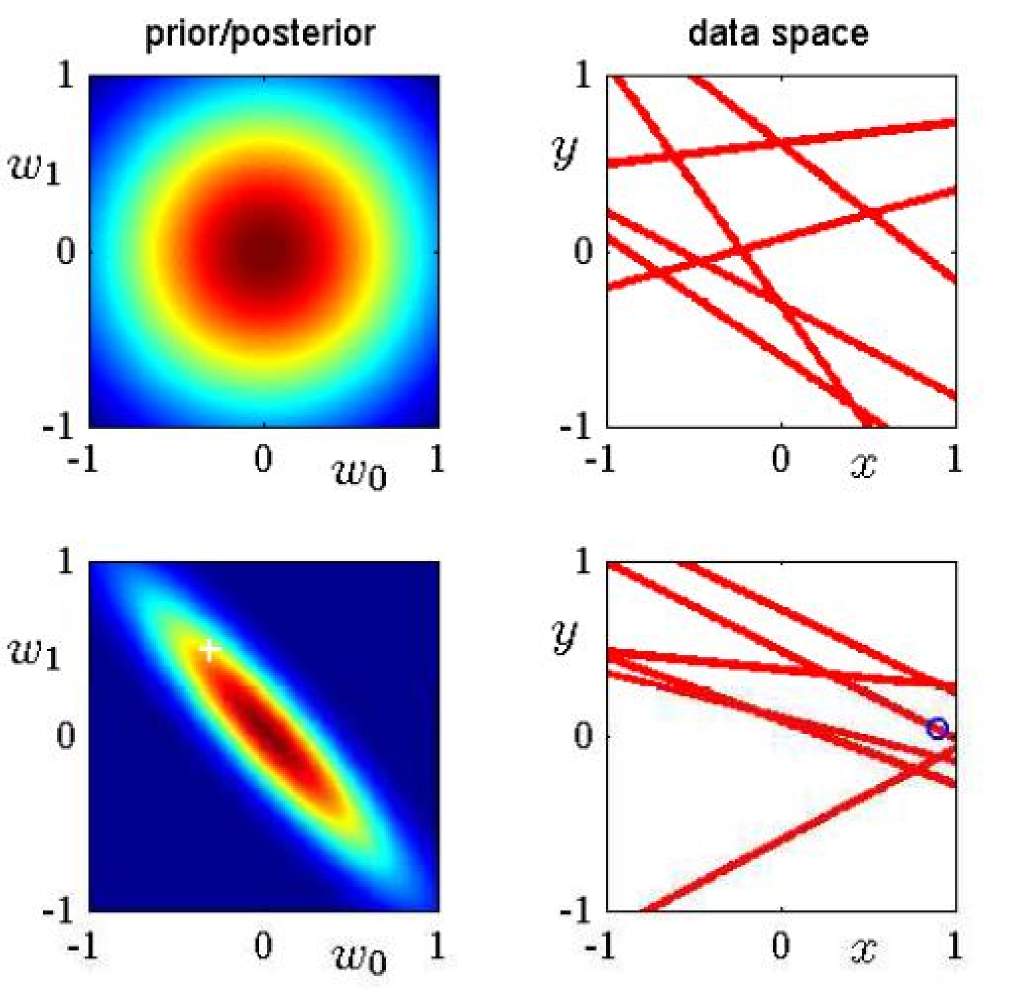
$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

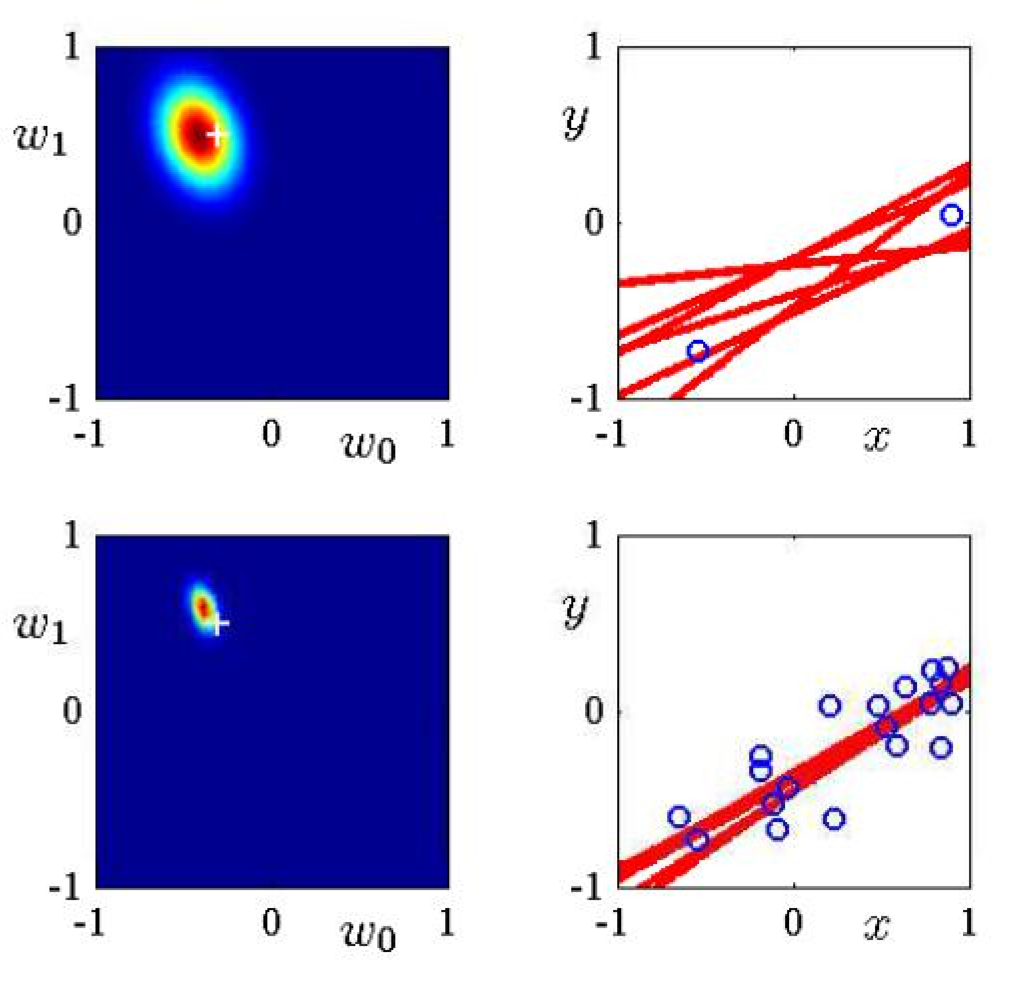


选择合适的参数平衡过拟合与欠拟合

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

6.4 正则化：贝叶斯解释





6.5 时间序列预测

■应用：

空气质量预测

微博评论数预测

优惠券使用间隔预测

点击量预测

■应用条件：

1. 如果数据在时间上具有比较大的相关性，可以使用历史值预测未来值
2. 只有历史数据可以使用，变量受其他因素的影响不明或者难以厘清

➢注意：空间相关性也可以套用同样的方法

6.5 时间序列预测

●时间序列方法概述：

- 平稳性：序列统计量是否随时间变化；
- 1. 平稳时间序列分析方法：
- 1）数据归一化
- 2）考察同一数据列前后数值之间的关系
- 2. 非平稳时间序列分析方法

□数据平稳化——取差值，差值对数

平稳化：服务于特征数据归一化

6.5 时间序列预测

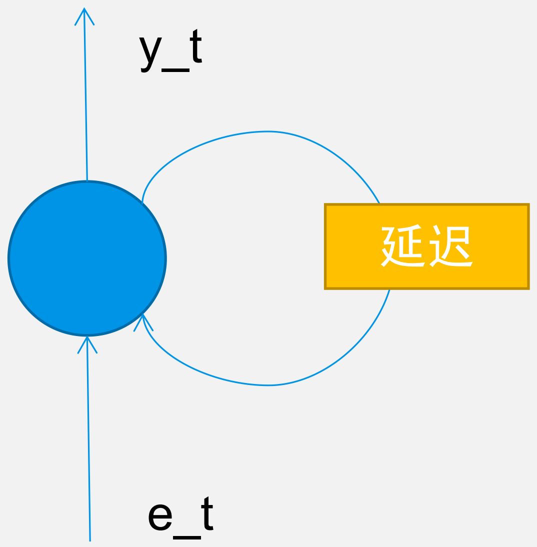
- AR模型 AR（n）

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_n y_{t-n} + e_t$$

- ARMA模型 ARMA(n,k)

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_n y_{t-n} + c_1 e_{t-1} + \ldots + c_k e_{t-k} + e_t$$

- ARIMA模型 ARIMA(n,k,d)



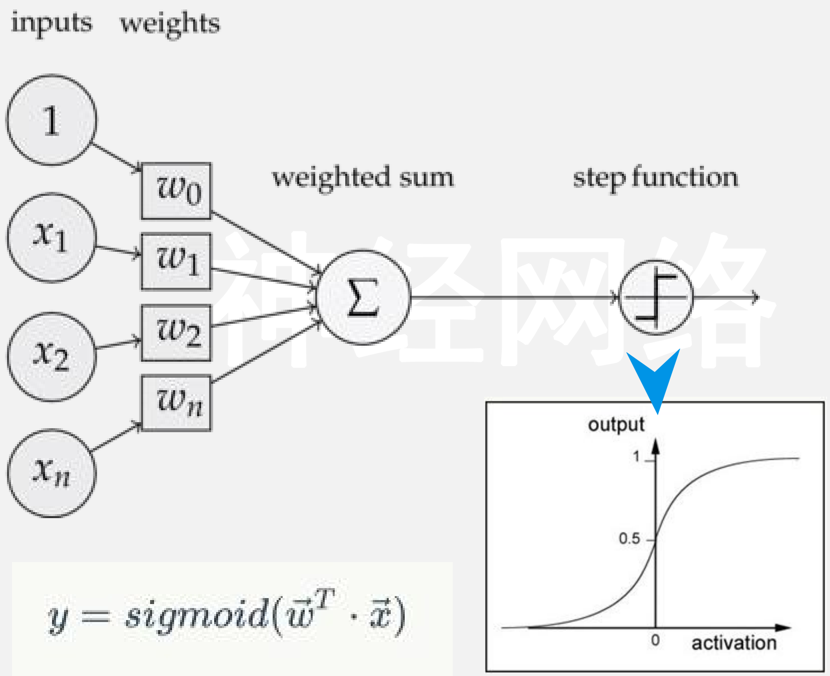
6.5 时间序列预测

- 目标：尽可能多的把有规律的信息描述在模型（数学公式）中
- 如何判断序列的噪声性质

- 白噪声：序列各时间点的值是否具有相关性；
- 高斯平稳白噪声：常见的描述噪声的概念，
  - 高斯：描述值的分布；
  - 平稳：描述统计量的变化与否；
  - 白：描述前后相关性；

6.6 神经网络：模型基本结构

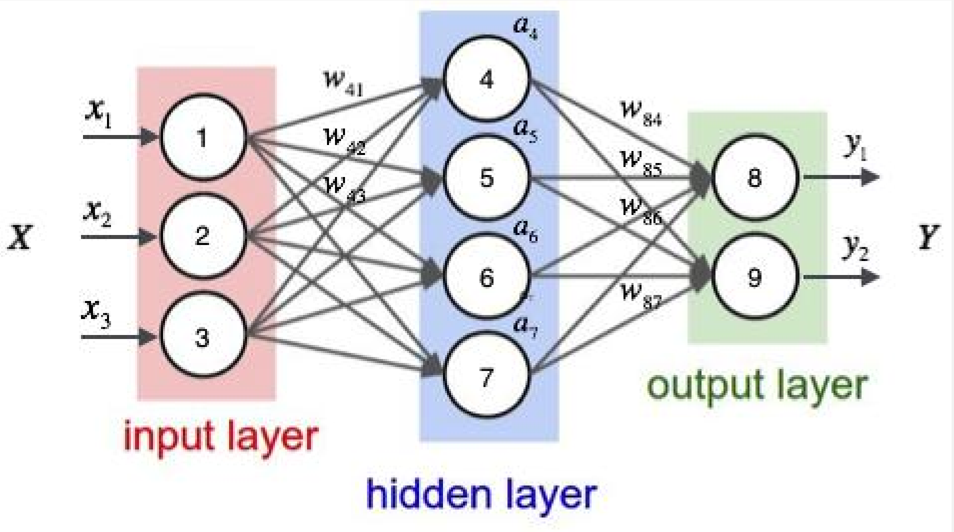
单个神经元



$$y = sigmoid(\vec{w}^T \cdot \vec{x})$$

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

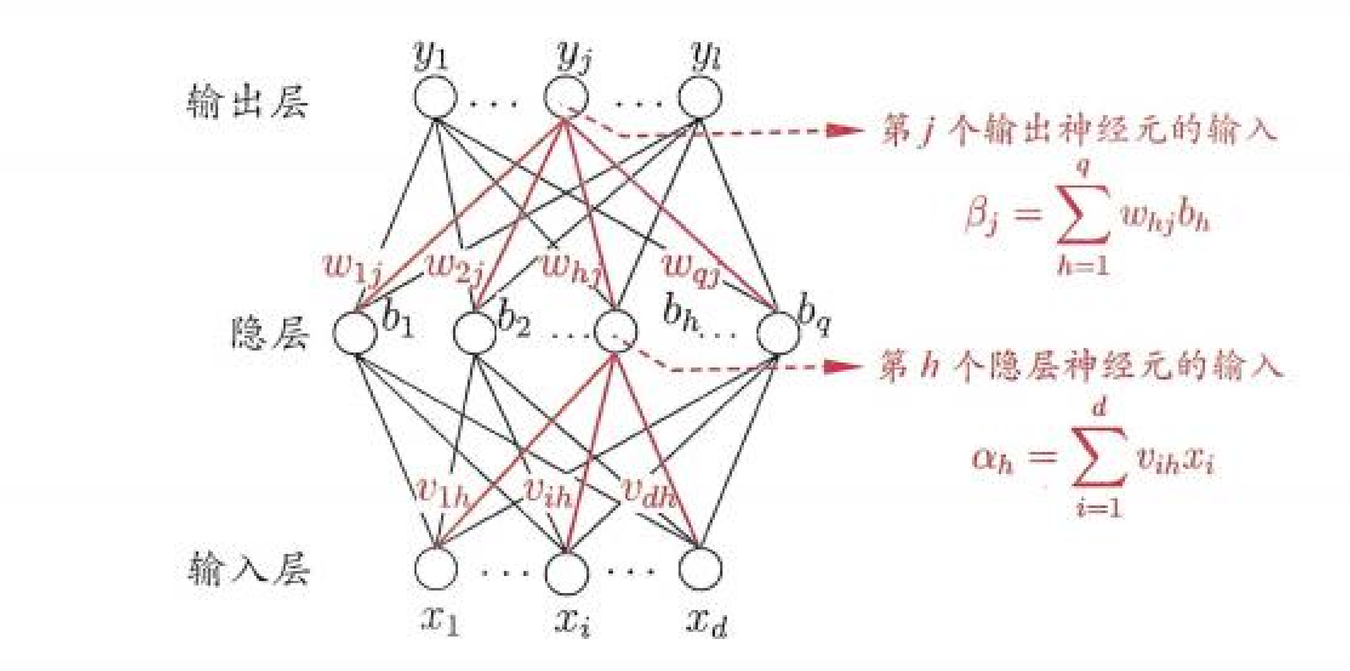
构成神经网络



$$\begin{aligned} a_4 &= sigmoid(\vec{w}^T \cdot \vec{x}) \\ &= sigmoid(w_{41}x_1 + w_{42}x_2 + w_{43}x_3 + w_{4b}) \end{aligned}$$

$$\begin{aligned} y_1 &= sigmoid(\vec{w}^T \cdot \vec{a}) \\ &= sigmoid(w_{84}a_4 + w_{85}a_5 + w_{86}a_6 + w_{87}a_7 + w_{8b}) \end{aligned}$$

6.6 神经网络：同样的损失函数+数值解



输出层参数的学习（以sigmoid为例）

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} .$$

$$(y_j^k - \hat{y}_j^k) .$$

$$f'(x) = f(x)(1 - f(x)) ,$$

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h .$$

设计中间变量

$$\begin{aligned} g_j &= -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \\ &= -(\hat{y}_j^k - y_j^k) f'(\beta_j - \theta_j) \\ &= \hat{y}_j^k (1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k) . \end{aligned}$$

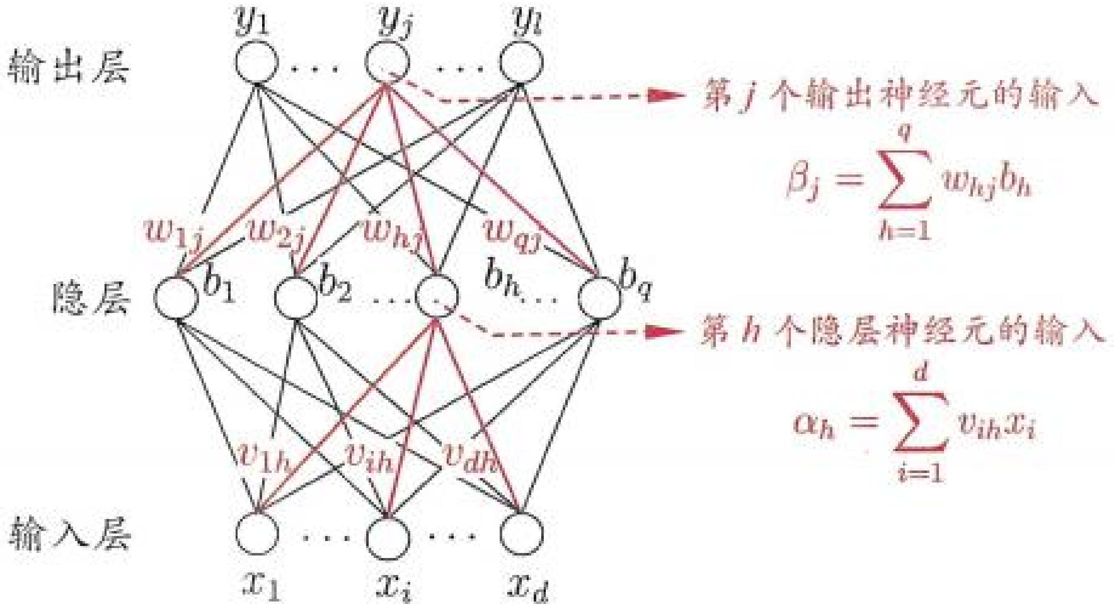
LMS——反向传播算法

$$E_k = \frac{1}{2} \sum_{j=1}^{l} (\hat{y}_j^k - y_j^k)^2$$

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} .$$

$$\Delta w_{hj} = \eta g_j b_h .$$

6.6 神经网络：同样的损失函数+数值解



隐藏层参数的学习（以sigmoid为例）

$$\Delta v_{ih} = \eta e_h x_i ,$$

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot b_h(1-b_h) \\ &= \sum_{j=1}^{l} w_{hj} g_j \ b_h(1-b_h) \\ &= b_h(1-b_h) \sum_{j=1}^{l} w_{hj} g_j . \end{aligned}$$

6.6 神经网络学习算法的构成

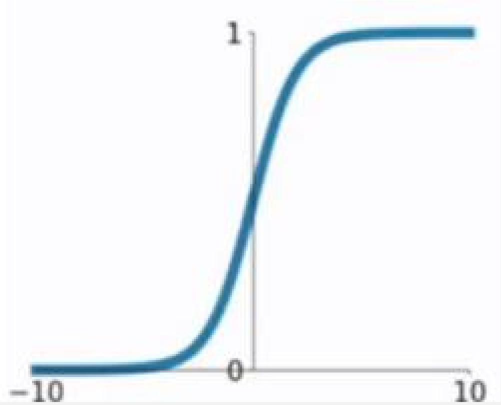
1. 激活函数
2. 网络结构
3. 损失函数
4. 数据的使用
5. 参数估计算法

```
1.  (train_images, train_labels), (test_images, test_labels) = load_mnist_data('./mnist.npz')
2.  model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(width, height, 1)))
3.  model.add(layers.MaxPool2D((2, 2)))
4.  model.add(layers.Dropout(0.3))
5.  model.add(layers.Flatten())
6.  model.add(layers.Dense(64, activation='relu'))
7.  model.add(layers.Dense(result_num, activation='softmax'))
8.  model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
9.  model.fit(train_images, train_labels, epochs=5, batch_size=64)
10. test_loss, test_acc = model.evaluate(test_images, test_labels)
```

6.6.1 激活函数的选择

Sigmod

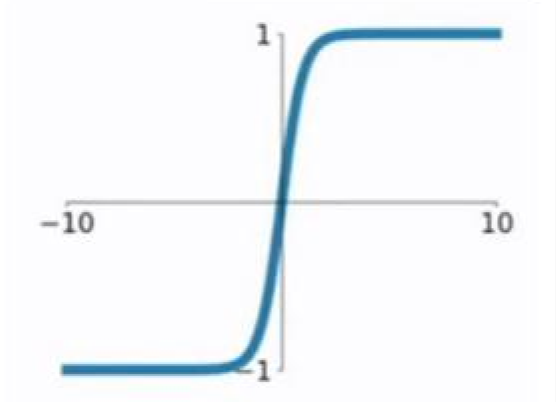
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



1) 当输入稍微远离了坐标原点，函数的梯度就变得很小了，几乎为零。
2) 函数输出不是以0为中心的，这样会使权重更新效率降低。
3) sigmod函数要进行指数运算，这个对于计算机来说是比较慢的。

tanh

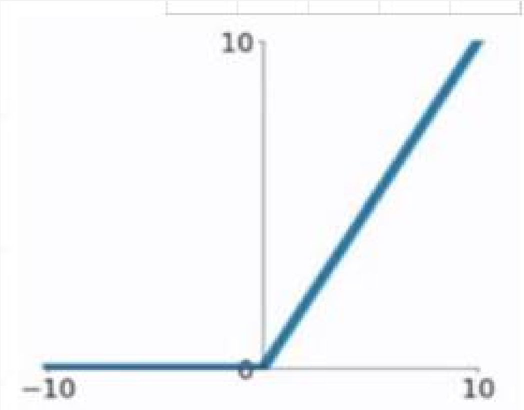
$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



1) tanh函数和sigmod函数的曲线是比较相近
2) 两个函数在输入很大或是很小的时候，输出都几乎平滑，梯度很小，不利于权重更新
3) 整个函数是以0为中心的

ReLU

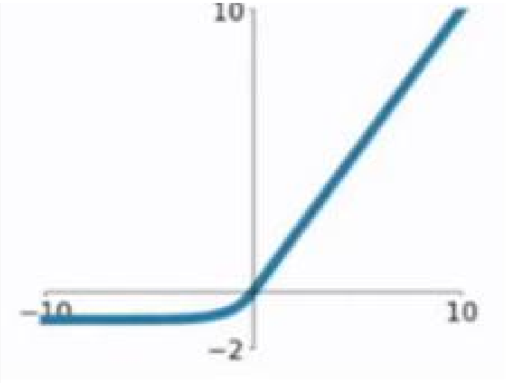
$$f(x) = max(0, x)$$



1) 在输入为正数的时候，不存在梯度饱和问题。
2) 计算速度要快很多。
3) 当输入是负数的时候，ReLU是完全不被激活的
4)LU函数也不是以0为中心的函数。

ELU

$$f(x) = \begin{cases} x & , x > 0 \\ \alpha(e^x - 1) & , x \le 0 \end{cases}$$



ELU函数是针对ReLU函数的一个改进型，相比于ReLU函数，在输入为负数的情况下，是有一定的输出的，但还是有梯度饱和和指数运算的问题。

6.6.2 网络结构的选择

- 1. 输入层、输出层节点数的选择；
- 2. 隐藏层结构的选择；

- 全连接网络
- CNN
- RNN

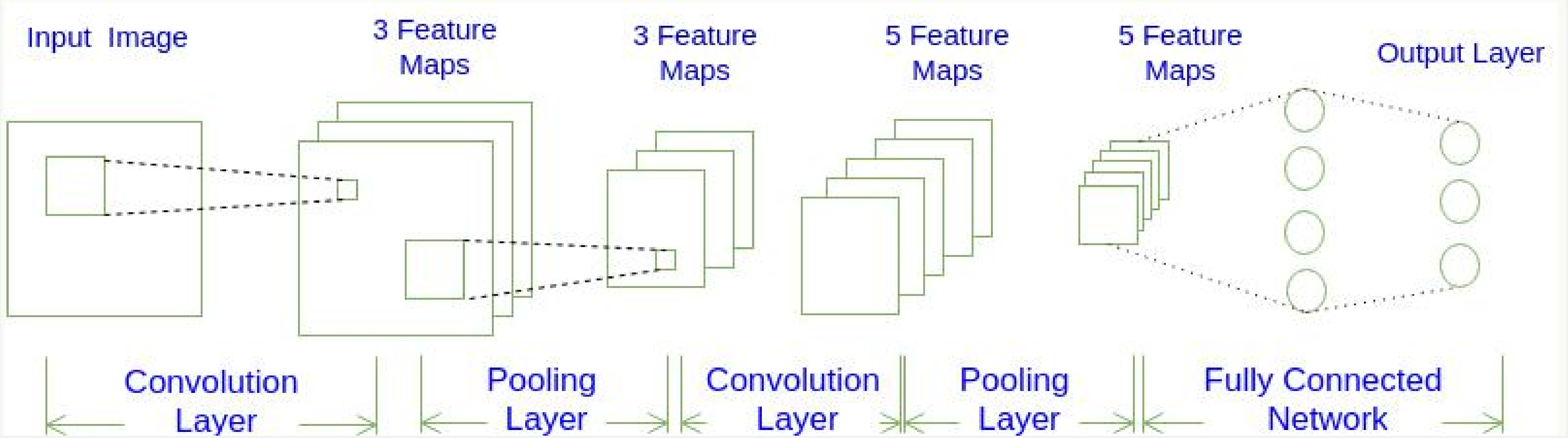
全连接神经网络不太适合图像识别任务:

1）参数数量太多：一个输入1000*1000像素的图片，输入层有100万节点。假设第一隐藏层有100个节点，那么仅这一层就有(1000*1000+1)*100=1亿参数；

2) 没有利用像素之间的位置信息：每个像素和其周围像素的联系是比较紧密的，具有局部性。如果一个神经元和上一层所有神经元相连，那么就相当于对于一个像素来说，把图像的所有像素都等同看待完成每个连接权重的学习之后，最终可能会发现，有大量的权重，它们的值都是很小的

3) 网络层数限制：网络层数越多其表达能力越强，但是通过梯度下降方法训练深度全连接神经网络很困难，因为全连接神经网络的梯度很难传递超过3层。因此，我们不可能得到一个很深的全连接神经网络，也就限制了它的能力。

6.6.2 网络结构：卷积神经网络CNN

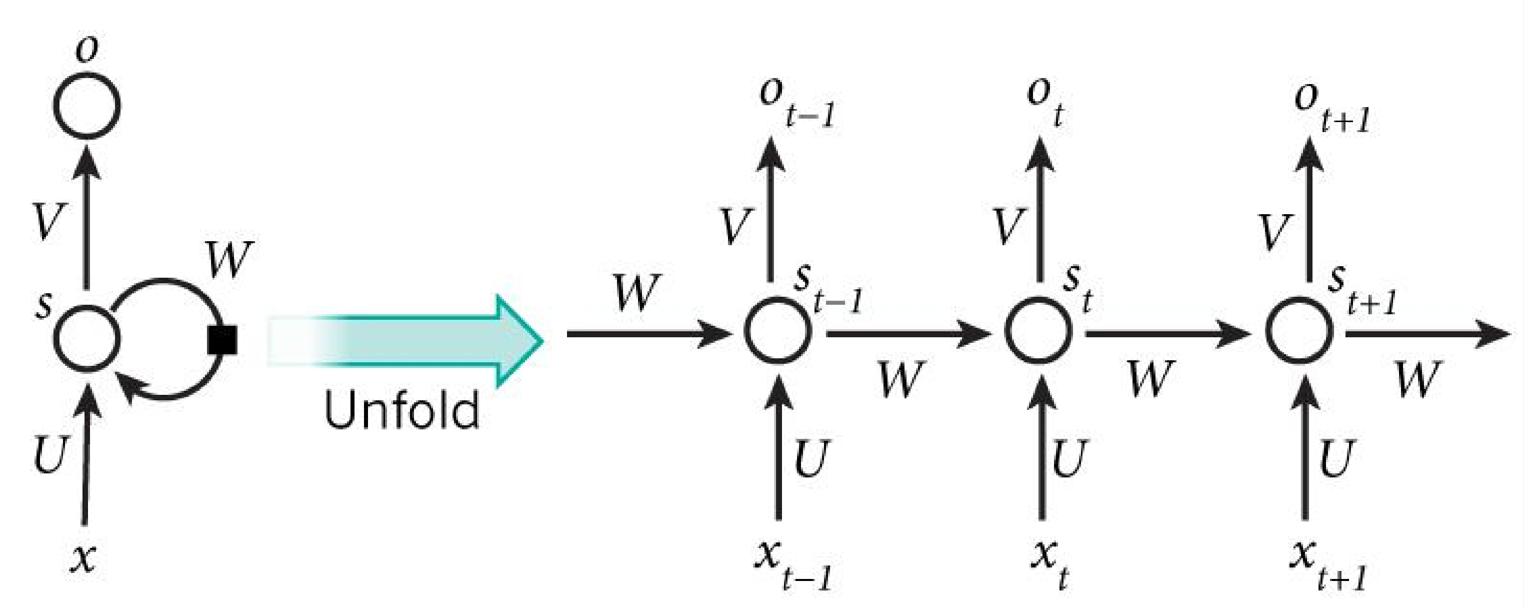


{{-1,-1,0},
{-1,0,1},
{0,1,1}}

{{-1,-1,-1},
{-1,8,-1},
{-1,-1,-1}}

6.6.2 网络结构：循环神经网络RNN

- 我昨天上学迟到了，老师批评了_____。



$$
\begin{aligned}
o_t &= g(Vs_t) \\
s_t &= f(Ux_t + Ws_{t-1})
\end{aligned}
$$

$$
\begin{aligned}
o_t &= g(Vs_t) \\
&= Vf(Ux_t + Ws_{t-1}) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2})) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3}))) \\
&= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))
\end{aligned}
$$

6.6.3 损失函数的选择

- Mean Square Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$$

$\mathcal{L}_1$	L_1 loss	$\|\mathbf{y}-\mathbf{o}\|_1$
$\mathcal{L}_2$	L_2 loss	$\|\mathbf{y}-\mathbf{o}\|_2^2$
$\mathcal{L}_1 \circ \sigma$	expectation loss	$\|\mathbf{y}-\sigma(\mathbf{o})\|_1$
$\mathcal{L}_2 \circ \sigma$	regularised expectation loss[1]	$\|\mathbf{y}-\sigma(\mathbf{o})\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$\max_j \lvert\sigma(\mathbf{o})^{(j)} - \mathbf{y}^{(j)}\rvert$
hinge	hinge [13] (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})$
hinge^2	squared hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^2$
hinge^3	cubed hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^3$
log	log (cross entropy) loss	$-\sum_j \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}$
$\log^2$	squared log loss	$-\sum_j [\mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2^2 + \|\mathbf{y}\|_2^2 - \sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}$
D_{CS}	Cauchy-Schwarz Divergence [3]	$-\log \frac{\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2 \|\mathbf{y}\|_2}$

- Categorical Crossentropy

$$CrossEntropy = -\sum (t_i \log y_i) + (1-t_i)\log(1-y_i)$$

6.6.4 数据的使用形式

■数据使用的两个极端：

- 1.batch批量算法
- 2.online在线算法（随机算法）

■深度学习的算法介于两者之间：小批量（随机）算法

- 使用多于1个的，但非全部的样本数据，进行多次特征空间中的梯度计算
- 大的batch梯度计算准确；
- 小的batch不能利用多核特性，但有一定泛化效果；

6.6.5 SGD算法——可看做LMS的推广

Require: 学习率ϵ_k

Require: 初始参数 $\boldsymbol{\theta}$

while 停止准则未满足**do**

从训练集中采包含m个样本$\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$ 的小批量，其中 $\boldsymbol{x}^{(i)}$ 对应目标为 $\boldsymbol{y}^{(i)}$。

计算梯度估计：$\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)})$

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}}$

end while

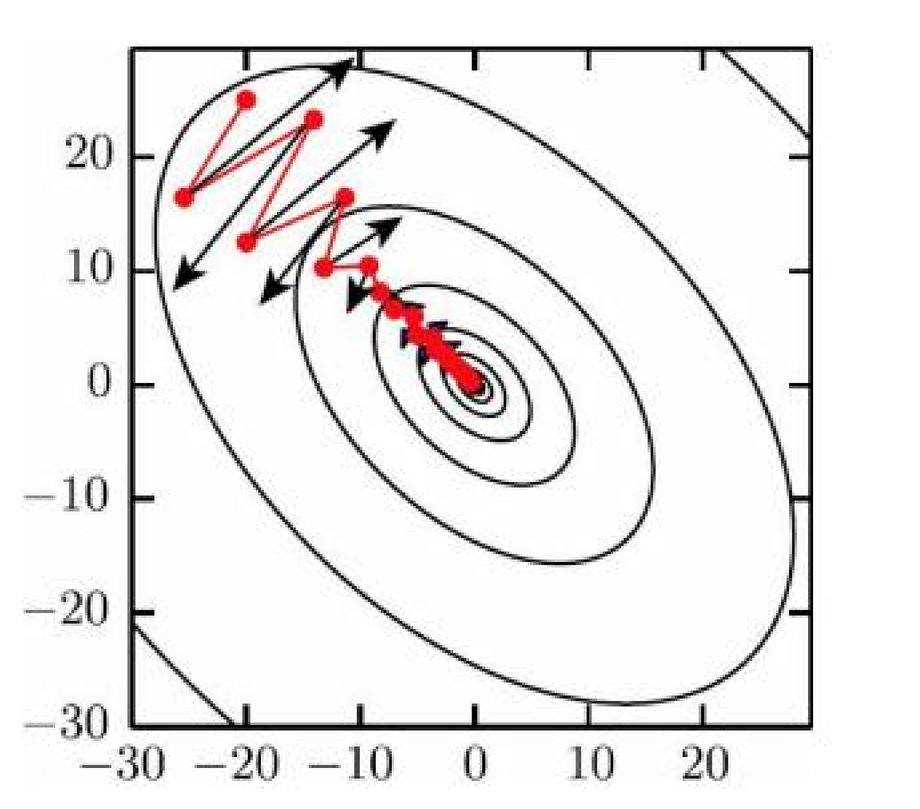
6.6.5 优化算法——梯度的修正

$$v \leftarrow \alpha v - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$$

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta} + \alpha \boldsymbol{v}), \boldsymbol{y}^{(i)}) \right]$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$$



6.6.5 优化算法——自适应学习速率

AdaGrad

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

累积平方梯度：$\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$

计算更新：$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta+\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$（逐元素地应用除和求平方根）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

Rmsprop

计算梯度：$\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

累积平方梯度：$\boldsymbol{r} \leftarrow \rho\boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$

计算参数更新：$\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta+\boldsymbol{r}}} \odot \boldsymbol{g}$（$\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$逐元素应用）

应用更新：$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

回归方法（不交）

- 任务1：自己写代码实现一元线性最小二乘回归；
- 任务2：利用api实现一元线性最小二乘回归和k阶多项式最小二乘回归，并在训练集上计算均方误差，画出k阶曲线。
- 任务3：将数据的10%作为验证集，即采用10倍交叉验证的方法，验证1阶-8阶模型的均方误差。

神经网络实验（不交）

本实验基于手写数字识别数据集（MNIST）：

- 内容1：通过安装运行样例代码所需要的包，成果运行样例代码，获得测试集正确率。
- 内容2：取消13,14行的注释，观察增加了一个卷积层和一个极大池化层后的测试集正确率。
- 内容3：自制一张符合模型输入要求的手写数字图片。（大小、颜色，可参考代码中的模型输入，或MNIST数据集）
- 内容4：使用model.predict()方法，对自制的图片进行手写数字识别。（可参考keras帮助文档）