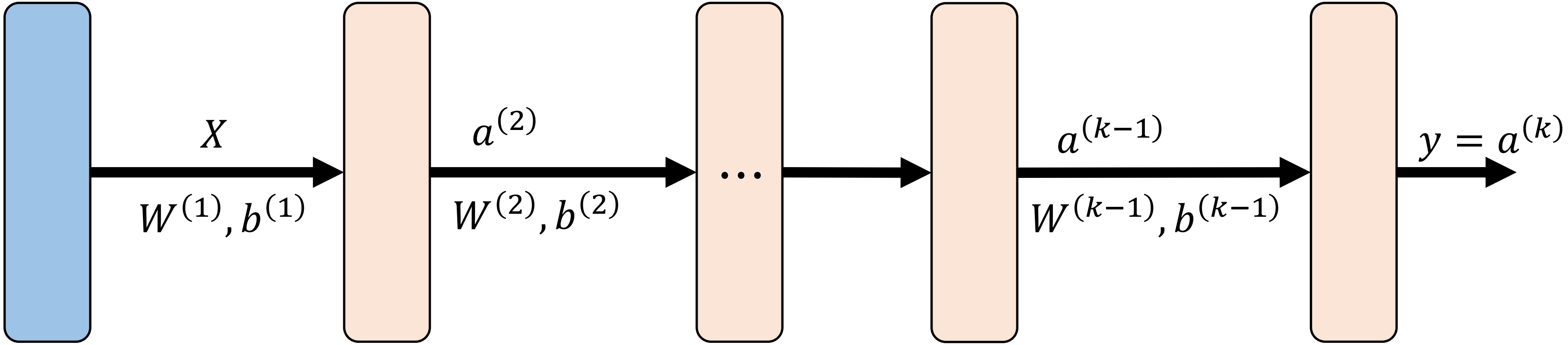
深度学习 第五讲

深层神经网络

王文中
安徽大学计算机学院

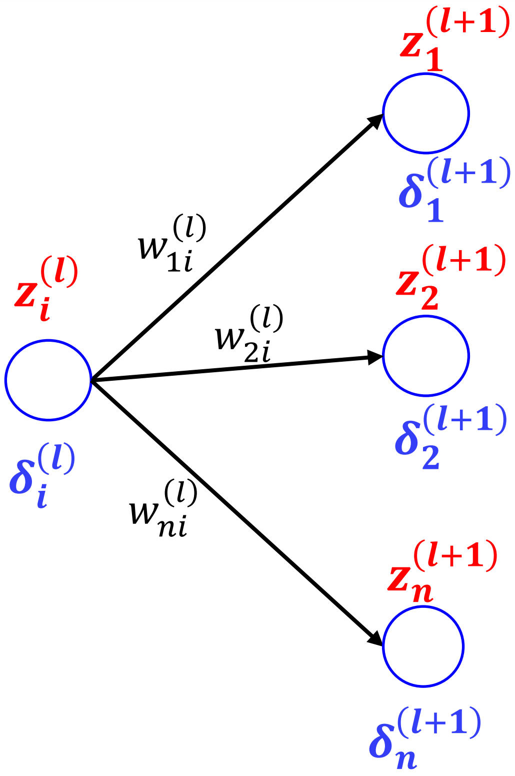
回顾：多层感知机



$$a^{(l+1)} = f_l\left(a^{(l)}; W^{(l)}, b^{(l)}\right) = f_l\left(W^{(l)} a^{(l)} + b^{(l)}\right)$$

$$y = h(X;\Theta) = f_k\left(f_{k-1}\left(\cdots f_2\left(X; W^{(1)}, b^{(1)}\right)\cdots; W^{(k-2)}, b^{(k-2)}\right); W^{(k-1)}, b^{(k-1)}\right)$$

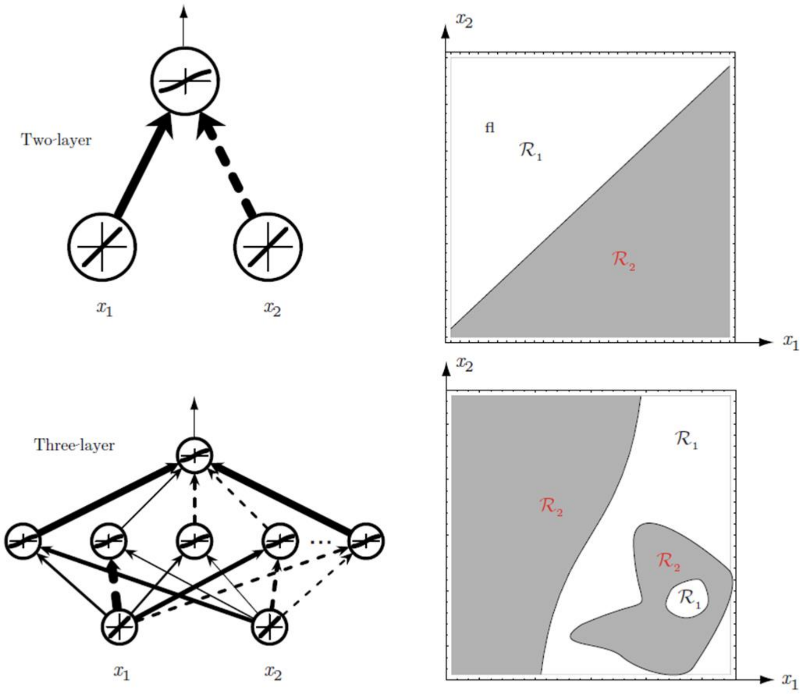
回顾：BP算法



$$\boxed{\delta_i^{(l)} = \left[\sum_{j=1}^{n} w_{ji}^{(l)} \delta_j^{(l+1)}\right] f'\left(z_i^{(l)}\right)}$$

$$\Delta W_{ji}^{(l)} = \delta_j^{(l+1)} a_i^{(l)}$$

回顾：通用逼近定理



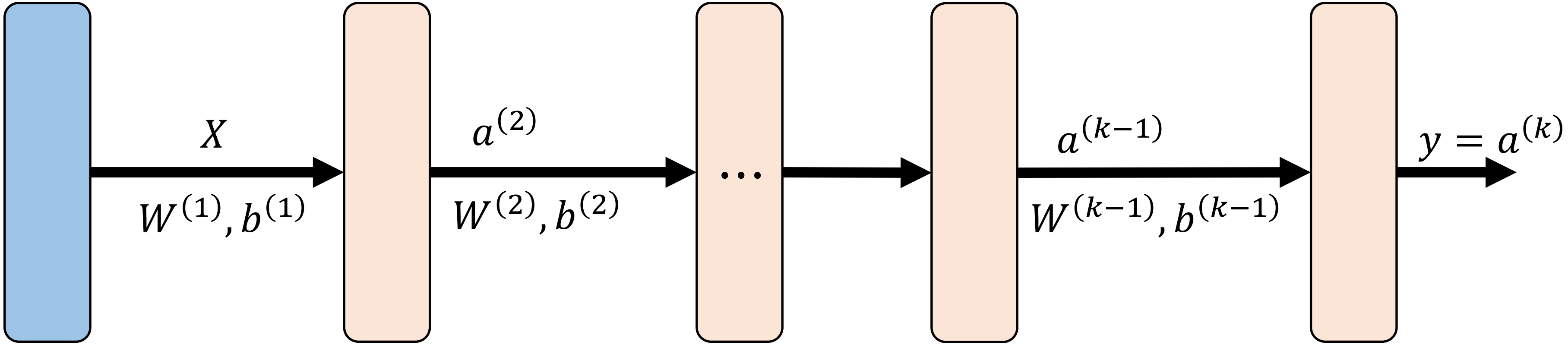
本次课内容

- 梯度消失与爆炸
- 偏差与方差
- 正则化
- 梯度下降法
- 超参数优化

深层神经网络

Deep Neural Networks

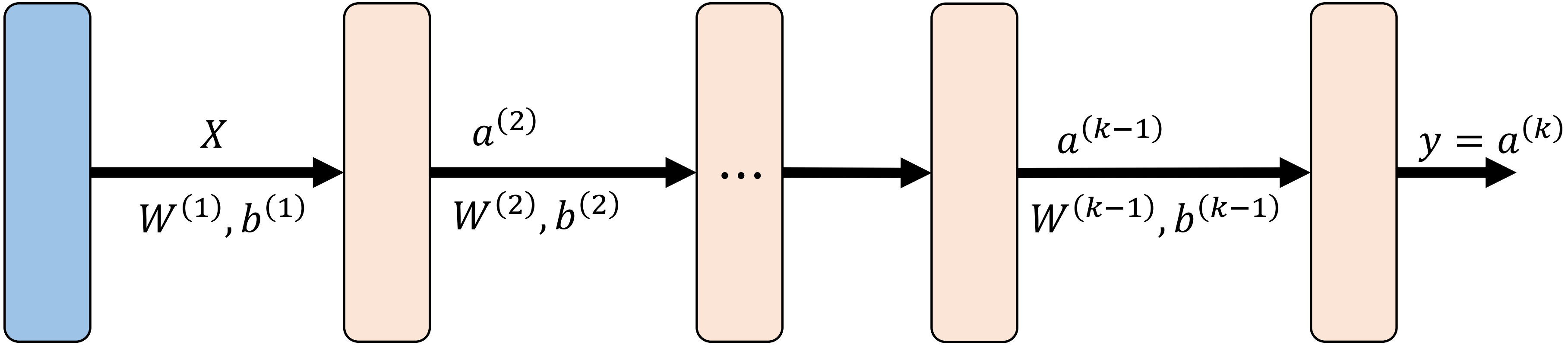
深层神经网络



$$y = h(X;\Theta) = f_k\left(f_{k-1}\left(\cdots f_2\left(X; W^{(1)}, b^{(1)}\right)\cdots; W^{(k-2)}, b^{(k-2)}\right); W^{(k-1)}, b^{(k-1)}\right)$$

$k > 2$ 多层函数复合；对输入特征X做多层次非线性变换，得到层次化的表达（特征/表示学习）。

全连接深层神经网络



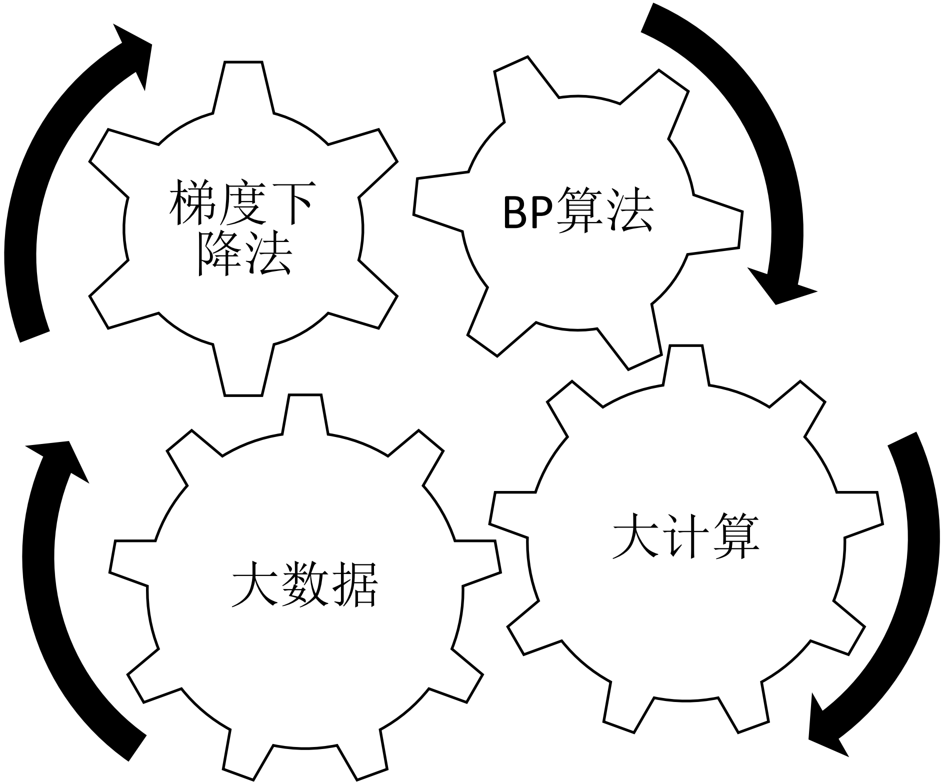
全连接神经网络：相邻两层神经元之间为全连接，连接（突触）总数为$N_l \times N_{l+1}$

K层神经网络的参数总数为：$N = \sum_{l=1}^{K-1} N_{l+1} \times (N_l + 1)$

一次前向运算的总浮点乘法运算量为N

训练深层神经网络

训练深层神经网络



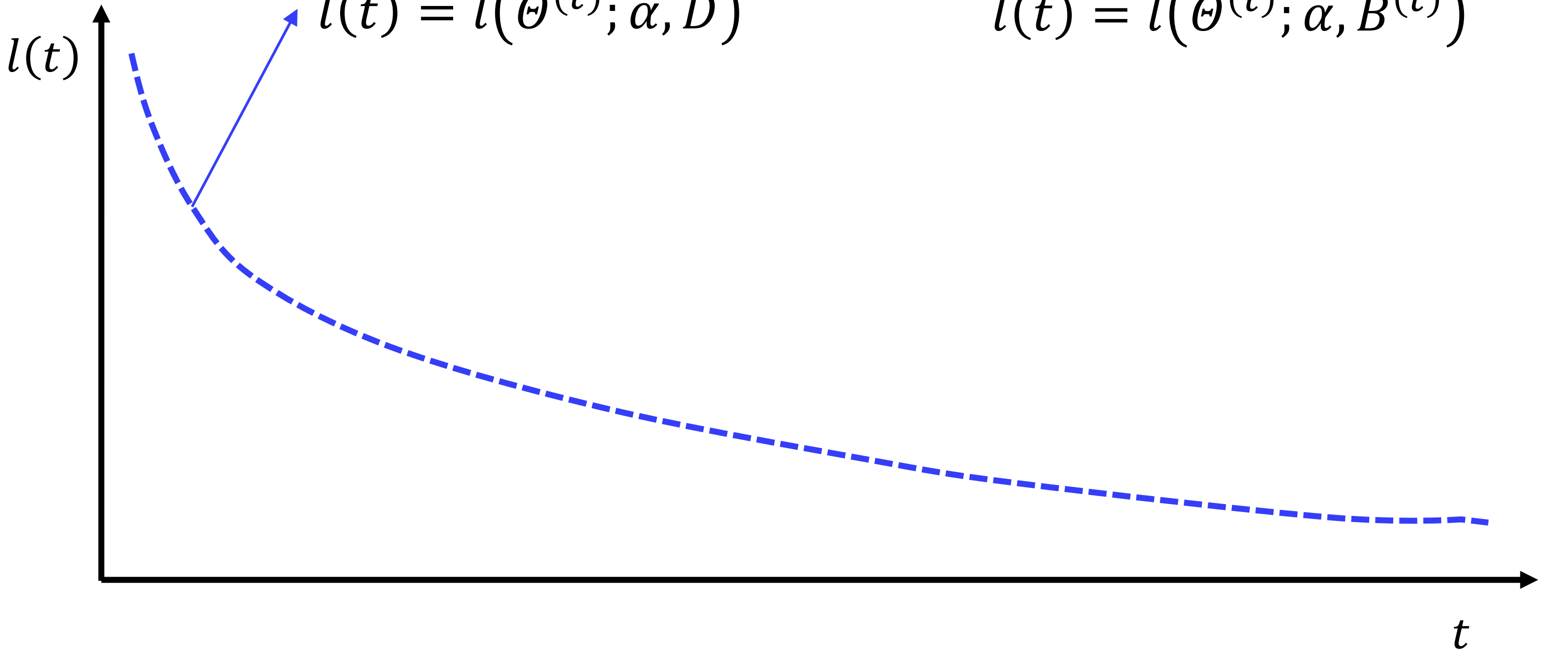
梯度下降法(Gradient Descend)

- Batch GD:
  - $l(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} l\left(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)}\right)$
  - $\boxed{\theta^{(t+1)} = \theta^{(t)} - \eta\nabla l\left(\theta^{(t)}\right) = \theta^{(t)} - \eta\frac{1}{n}\sum_{i=1}^{n} \nabla l\left(\theta^{(t)}; \boldsymbol{x}^{(i)}, y^{(i)}\right)}$
- Stochastic GD:
  - $\boxed{\theta^{(t+1)} = \theta^{(t)} - \eta\nabla l\left(\theta^{(t)}; \boldsymbol{x}^{(i)}, y^{(i)}\right)}$
- Mini-Batch GD:
  - $\boxed{\theta^{(t+1)} = \theta^{(t)} - \eta\nabla l\left(\theta^{(t)}; B^{(k)}\right)}$
  - $B^{(k)} = \left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\}_{i=(k-1)*batch\_size+1}^{k*batch\_size}$
- Epoch: 遍历整个样本集合$\mathcal{D}$，每一个Epoch随机排列$\mathcal{D}$中元素

1次迭代(iteration)：
前向+后向传播
权值更新

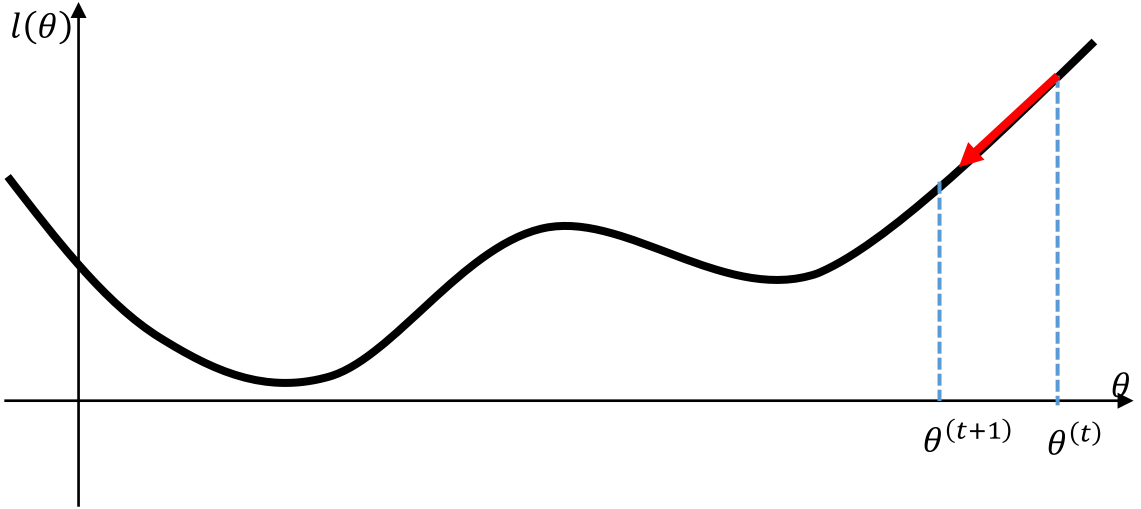
随机梯度下降法

$$l(t) = l\left(\Theta^{(t)}; \alpha, D\right) \qquad\qquad l(t) = l\left(\Theta^{(t)}; \alpha, B^{(t)}\right)$$



学习曲线

带动量的梯度下降法



$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla l\left(\theta^{(t)}\right) \Longrightarrow$$

Momentum:

$$m = \beta m + \eta \nabla l\left(\theta^{(t)}\right)$$

$$\theta^{(t+1)} = \theta^{(t)} - m$$

(通常$\beta = 0.9$)

$m = 0$

$t = 1$:

$m \leftarrow \beta m + \eta\delta = \eta\delta$

$\theta \leftarrow \theta - m = \theta - \eta\delta$

$t = 2$:

$m \leftarrow \beta m + \eta\delta = \eta(1+\beta)\delta$

$\theta \leftarrow \theta - m = \theta - \eta(1+\beta)\delta$

$t = 3$:

$$m \leftarrow \beta m + \eta\delta = \eta\frac{1-\beta^3}{1-\beta}\delta$$

$$\theta \leftarrow \theta - m = \theta - \eta\frac{1-\beta^3}{1-\beta}\delta$$

$\vdots$

$t = T$(T很大):

$$m \leftarrow \beta m + \eta\delta \approx \eta\frac{1}{1-\beta}\delta = \eta \times 10\delta$$

$$\theta \leftarrow \theta - m \approx \theta - \eta \times 10\delta$$

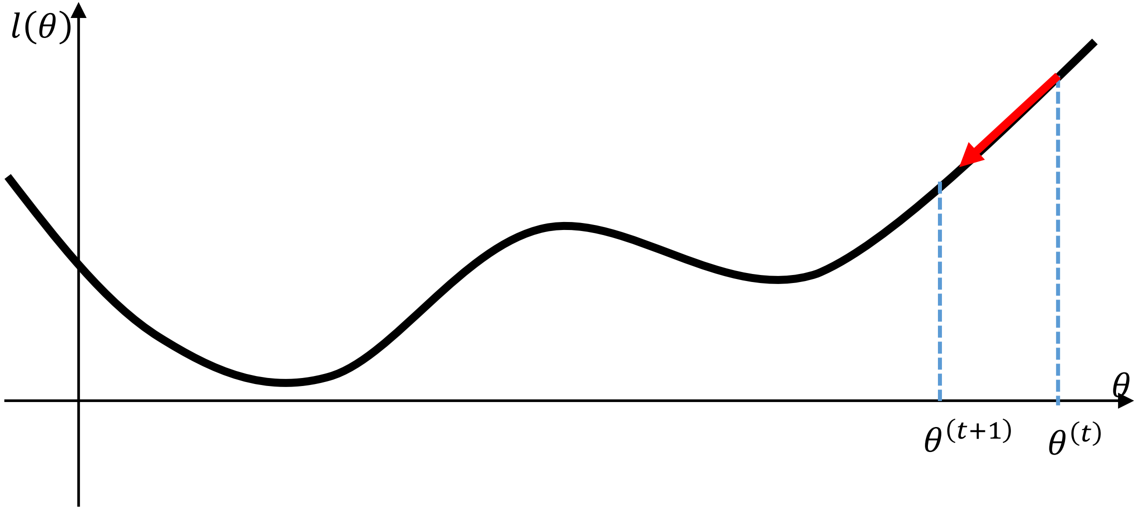
Why Momentum Really Works: https://distill.pub/2017/momentum/

梯度下降法(Gradient Descend)

- Nesterov Accelerated Gradient
  - Yurii Nesterov(1983): A Method for Unconstrained Convex Minimization Problem with the Rate of Convergence.
- AdaGrad
  - J. Duchi et al.(2011): Adaptive Subgradient Methods for Online Learning and Stochastic Optimizations.
- RMSProp
  - Tijmen Tieleman & Geof Hinton, Coursera course.
- Adam(Adaptive Momentum Estimation)
  - D. Kingma, J.Ba(2015): Adam: A Method for Stochastic Optimization.

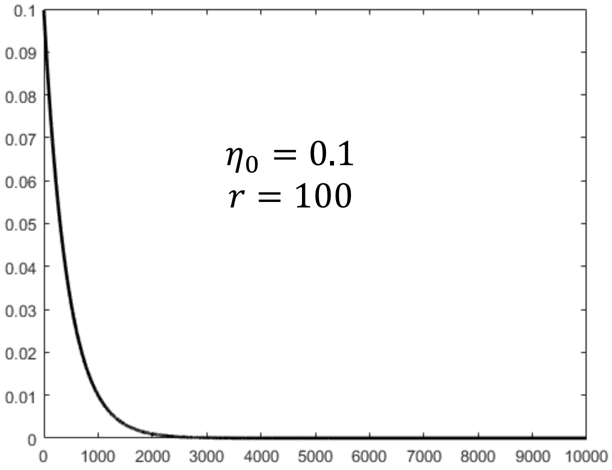
更详细的综述：http://ruder.io/optimizing-gradient-descent/

学习率η

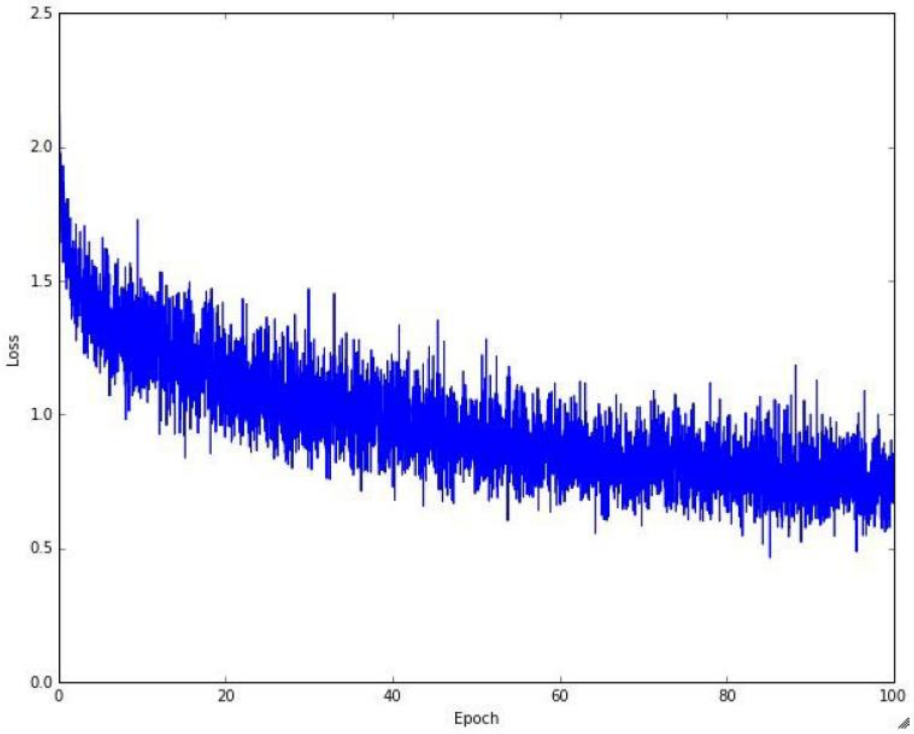


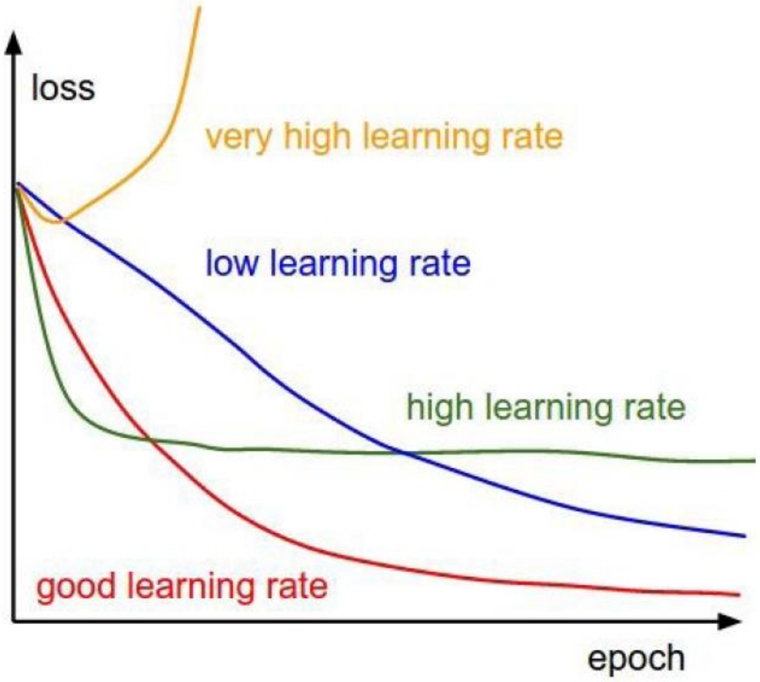
$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla l\left(\theta^{(t)}\right)$$

$$\eta(t) = \eta_0 10^{-\frac{t}{r}}$$



调节学习率，确保损失正常下降

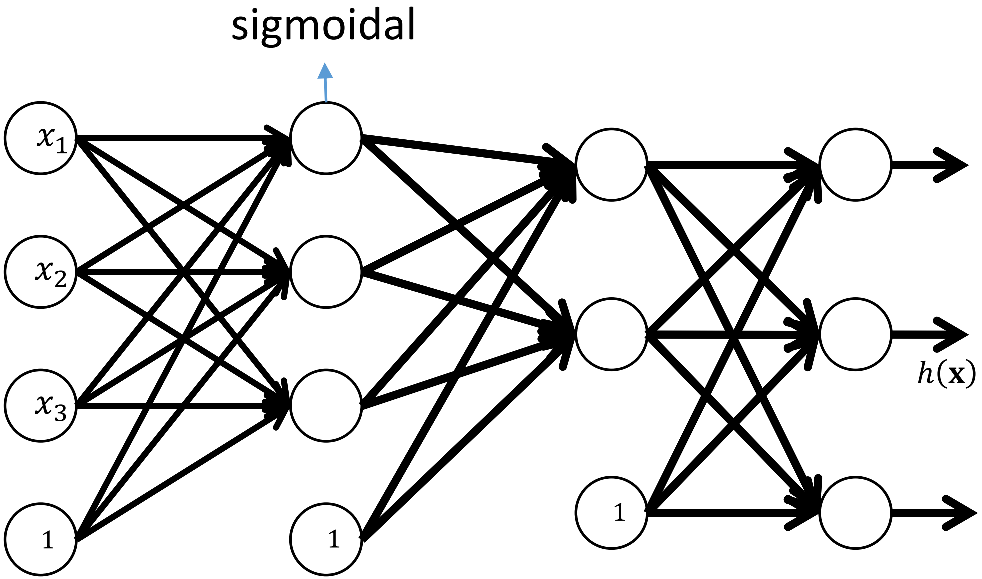




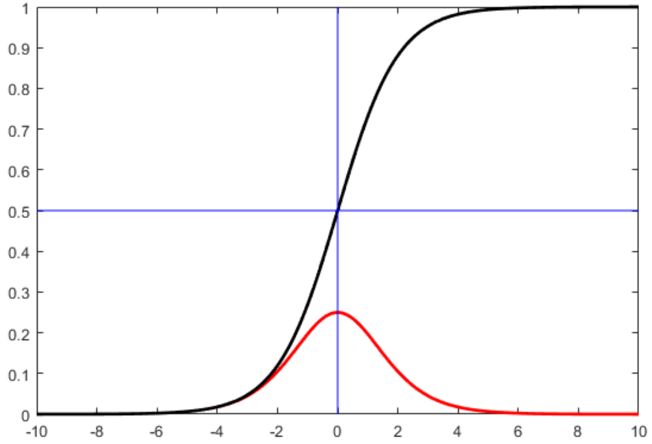
梯度消失与梯度爆炸

Vanishing and Exploding Gradients

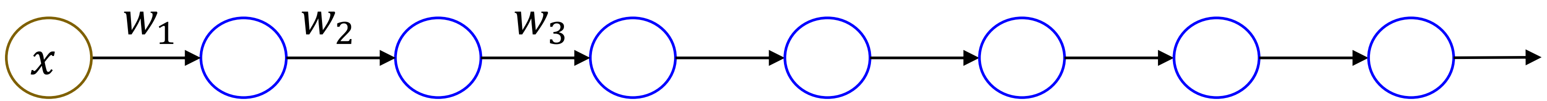
梯度消失与梯度爆炸问题



$$\delta_i^{(j)} = f'\left(z_i^{(j)}\right) \cdot \left[\sum_{k=1}^{N_{j+1}} w_{k,i}^{(j+1)} \delta_k^{(j+1)}\right]$$

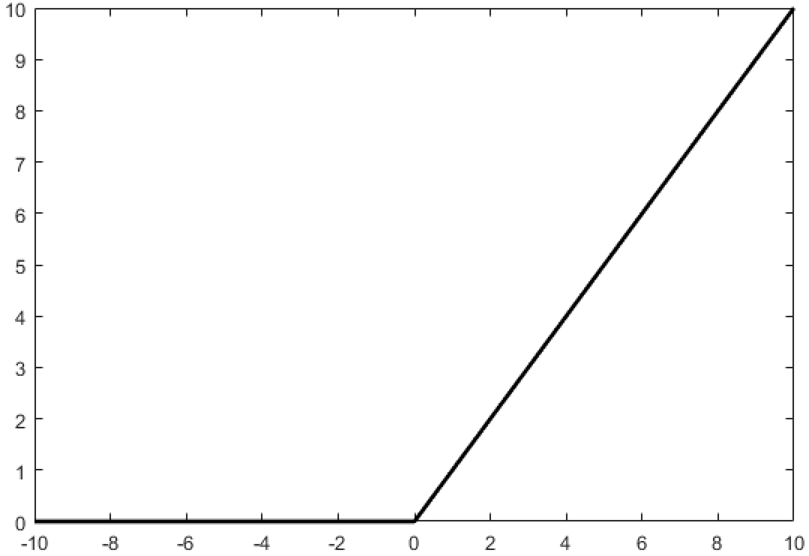


梯度消失与梯度爆炸问题

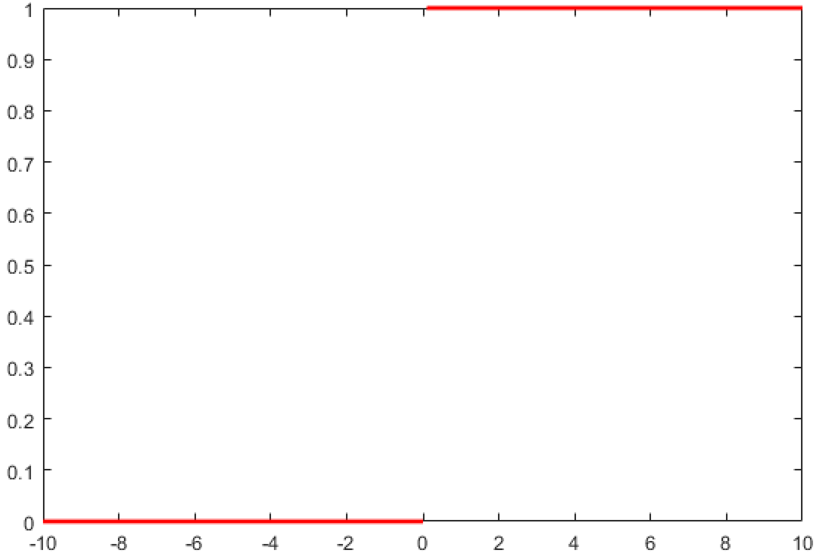


ReLU (Rectified Linear Unit)

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



$$ReLU'(x)$$



权值初始化

$$\begin{array}{lll} & Xavier & He \\ & w_{i,j}^{(k)} \sim N(0,\sigma), & w_{i,j}^{(k)} \sim U(-r,r) \\ Sigmoid: & \sigma = \sqrt{\dfrac{2}{N_{k-1}+N_k}}, & r = \sqrt{\dfrac{6}{N_{k-1}+N_k}} \\ Tanh: & \sigma = 4\sqrt{\dfrac{2}{N_{k-1}+N_k}}, & r = 4\sqrt{\dfrac{6}{N_{k-1}+N_k}} \\ ReLU: & \sigma = \sqrt{\dfrac{4}{N_{k-1}+N_k}}, & r = \sqrt{\dfrac{12}{N_{k-1}+N_k}} \end{array}$$

Xavier Glorot & Yoshua Bengio,2010, Understanding the Difficulty of Training Deep Feedforward Neural Networks
Kaiming He, et,al: 2015, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

梯度裁剪(Gradient Clipping)

$$W \leftarrow W - \alpha \nabla W$$

$$if\ |\nabla W| > \tau_{max}: \nabla W = \frac{\nabla W}{|\nabla W|} \times \tau_{max}$$

$$if\ \nabla W_i > +\nu: \nabla W_i = +\nu$$

$$if\ \nabla W_i < -\nu: \nabla W_i = -\nu$$

欠拟合与过拟合

Underfitting / Overfitting

训练误差与测试误差

$$\mathcal{H} = \{h(x;\theta)\}$$

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}, (x^{(i)}, y^{(i)}) \sim P(X, Y)\,;$$

$$\theta^* = argmin_{\theta} l(\theta; \mathcal{D}) = argmin_{\theta} \frac{1}{n}\sum_{i=1}^{n} l(\theta; x^{(i)}, y^{(i)})$$

$$h^*(x) = h(x; \theta^*)$$

训练误差与测试误差

$$l(h^*;x,y) \equiv l(\theta^*;x,y)$$

$$E_{train}(h^*) = \frac{1}{n}\sum_{i=1}^{n} l\left(h^*;x^{(i)},y^{(i)}\right)$$

$$E_{test}(h^*) = E_P l(h^*;x,y) = \iint l(h^*;x,y)P(x,y)dxdy$$

$$E_{train}(h^*) < E_{test}(h^*)$$

训练集/测试集

- 训练集$\mathcal{D}_{train}$用于训练模型参数，并计算训练误差:
  - $\theta^* = argmin_\theta \frac{1}{|\mathcal{D}_{train}|}\sum_{(x,y)\in\mathcal{D}_{train}} l(\theta; x, y)$
  - $E_{train}(h^*) = \frac{1}{|\mathcal{D}_{train}|}\sum_{(x,y)\in\mathcal{D}_{train}} l(h^*(x), y)$

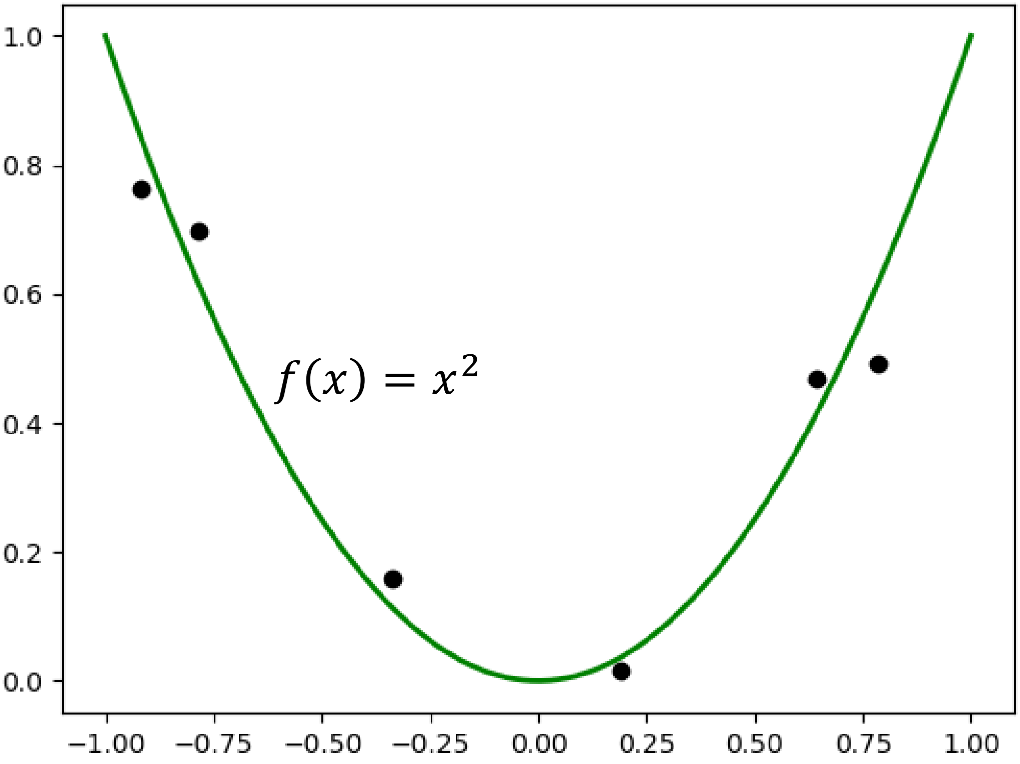
- 测试集$\mathcal{D}_{test}$<u>**仅用于**</u>评估模型的泛化性能（测试误差）：
  - $E_{test}(h^*) \approx \frac{1}{|\mathcal{D}_{test}|}\sum_{(x,y)\in\mathcal{D}_{test}} l(h^*(x), y)$

训练误差与测试误差

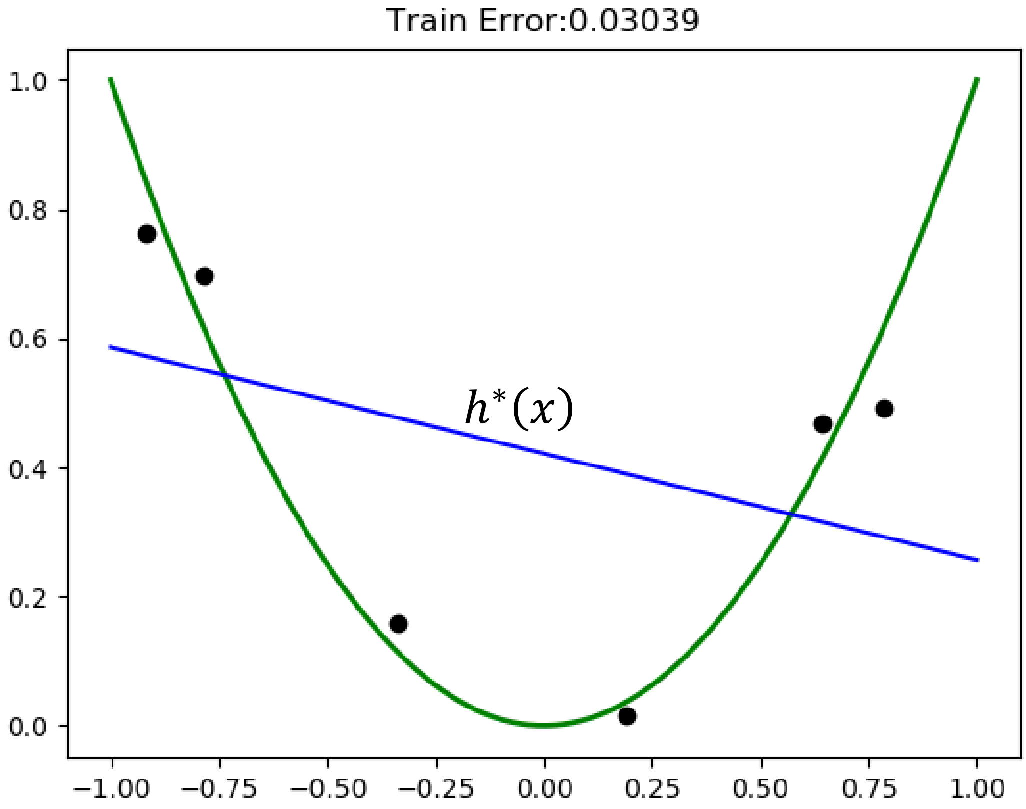
1. 给定训练样本$\mathcal{D}$,模型空间$\mathcal{H}$和算法$\mathcal{A}$，是否可以得到足够小的训练误差$E_{train}(h^*)$?

2. 训练误差$E_{train}(h^*)$是否可以近似表示测试误差$E_{test}(h^*)$?

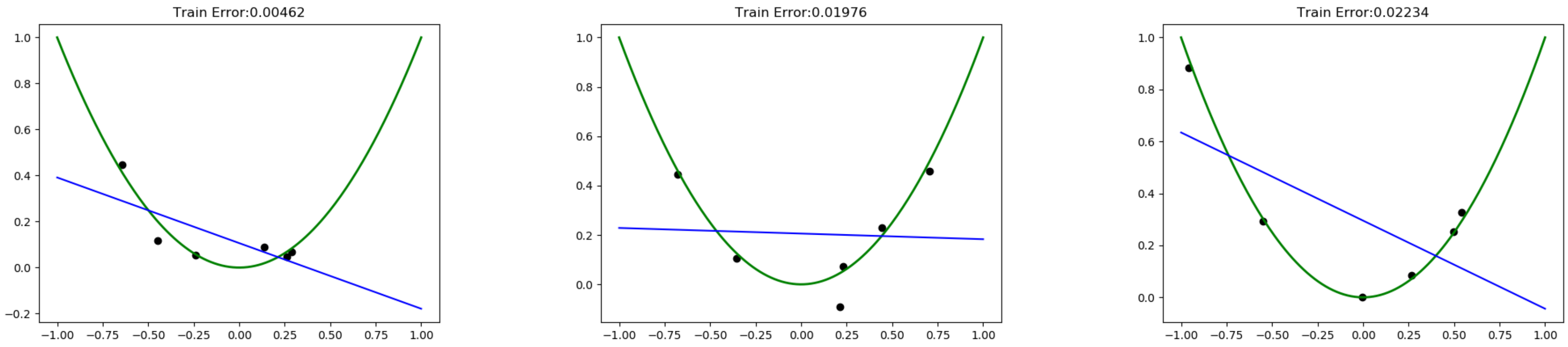
$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{6}, y_i = f(x_i) + \epsilon_i$$



$$\mathcal{H}_1 = \{h(x; w, b) = wx + b\}$$



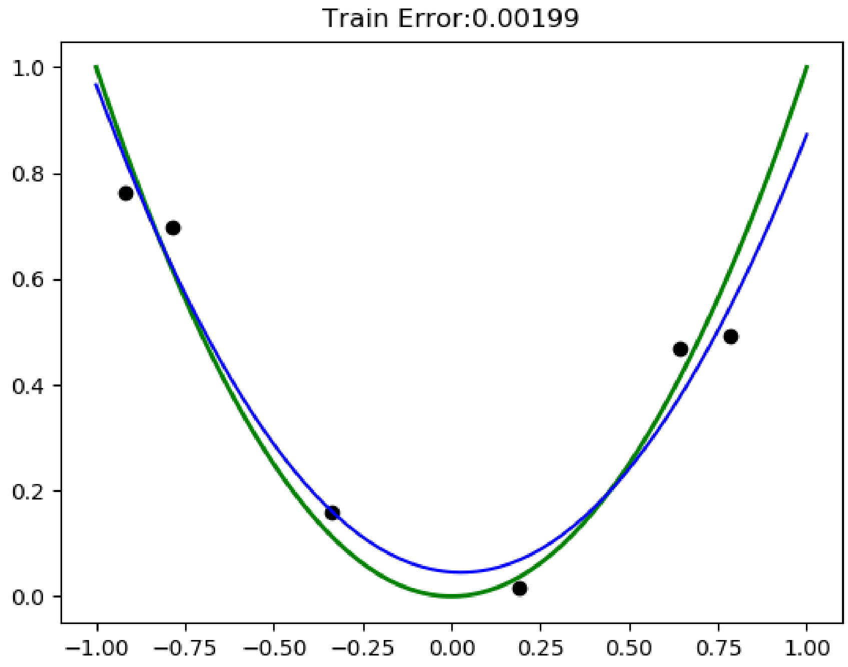
欠拟合(Underfitting)



欠拟合：模型空间$\mathcal{H}$相对于真实函数f太过于简单，无法很好地拟合训练样本。表现为训练误差比较大。

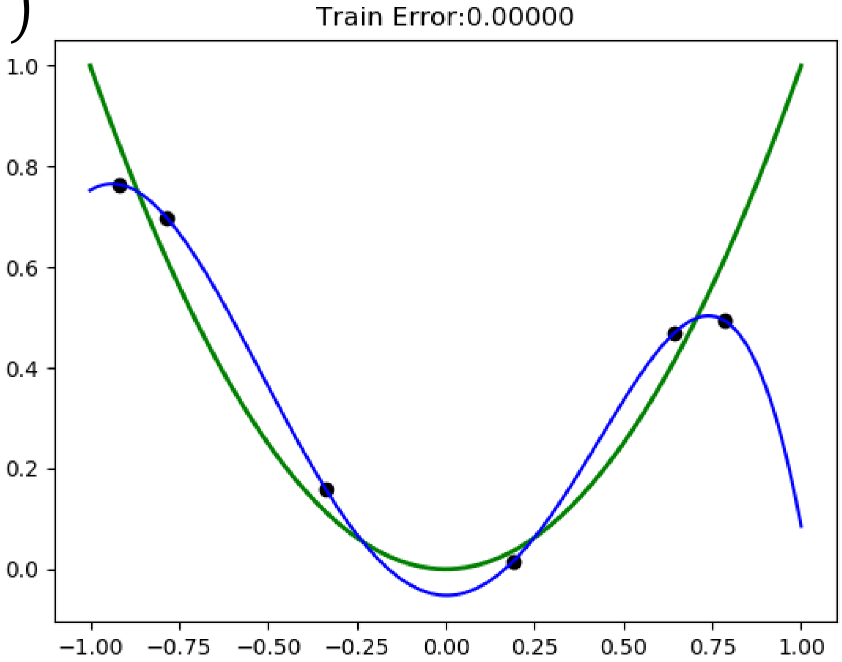
使用复杂的模型解决欠拟合问题

$$\mathcal{H}_2 = \{h(x;\theta) = w_2x^2 + w_1x + w_0\}$$

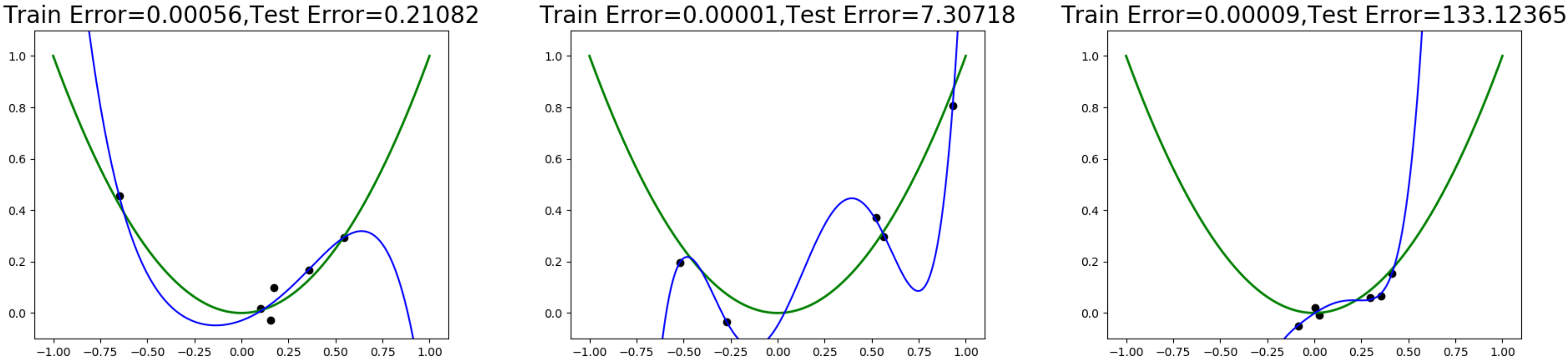


使用复杂的模型解决欠拟合问题

$$\mathcal{H}_5 = \left\{ h(x;\theta) = \sum_{j=0}^{5} w_j x^j \right\}$$



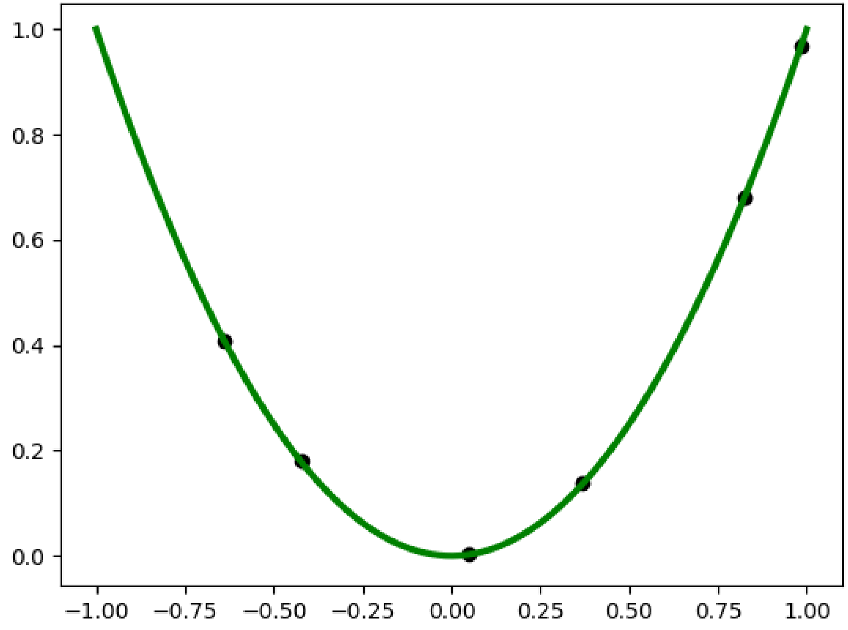
过拟合



过拟合：模型空间$\mathcal{H}$相对于训练数据$\mathcal{D}$过于复杂（拟合能力很强），以致于拟合了训练样本中的噪声规律，而不是数据中的真实规律f。

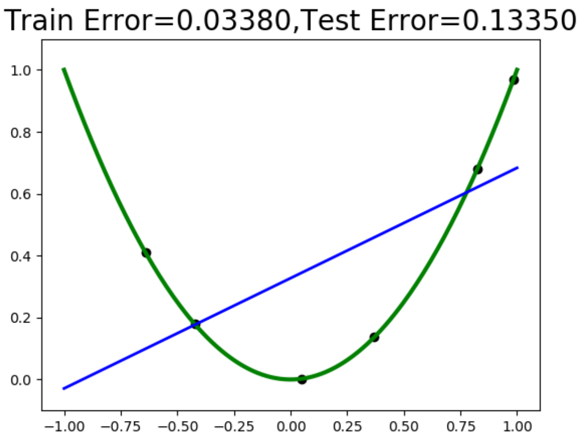
表现为训练误差很小，而测试误差很大（$E_{test} \gg E_{train}$）。

无噪声样本

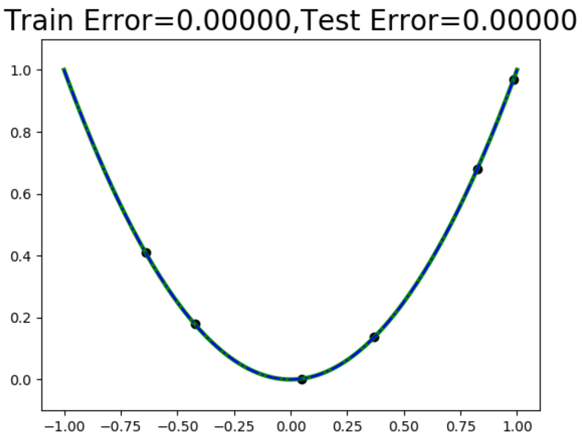


$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{6}, y_i = f(x_i) = x_i^2$$

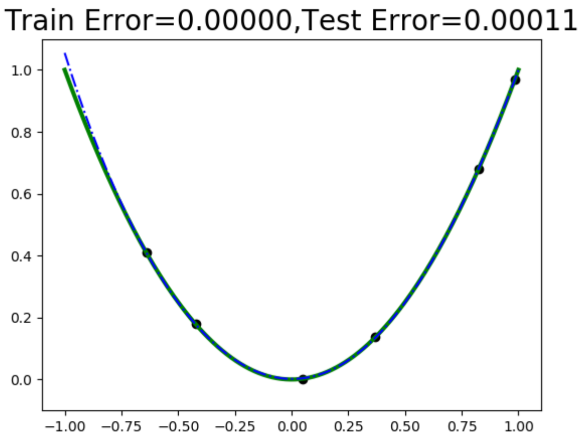
无噪声样本的拟合结果



$\mathcal{H}_1$

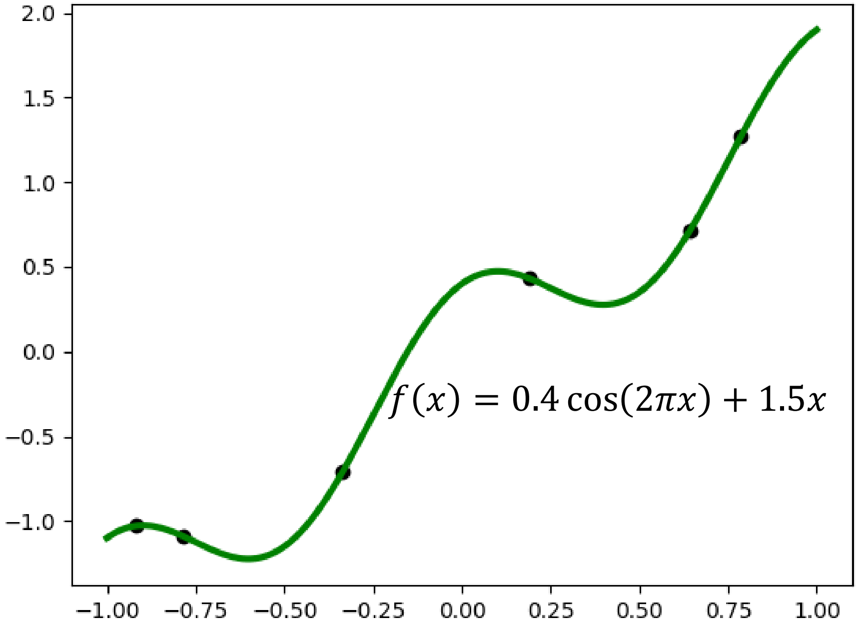


$\mathcal{H}_2$



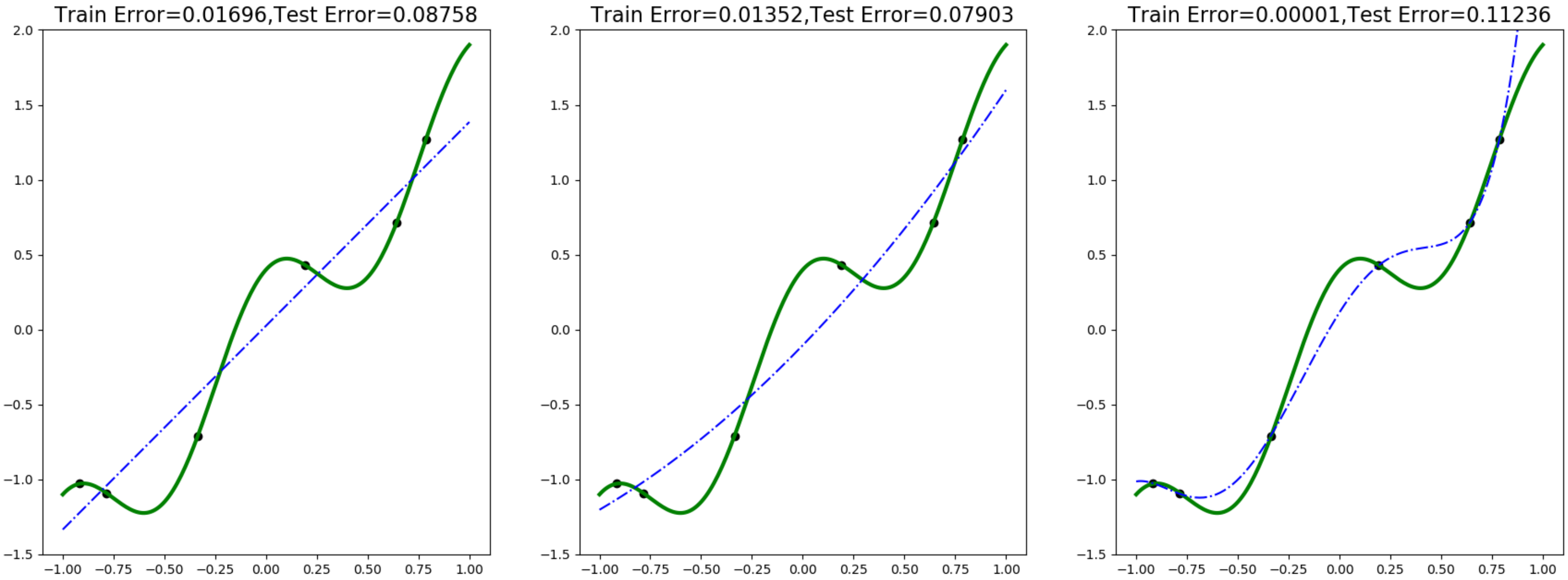
$\mathcal{H}_5$

无噪声样本



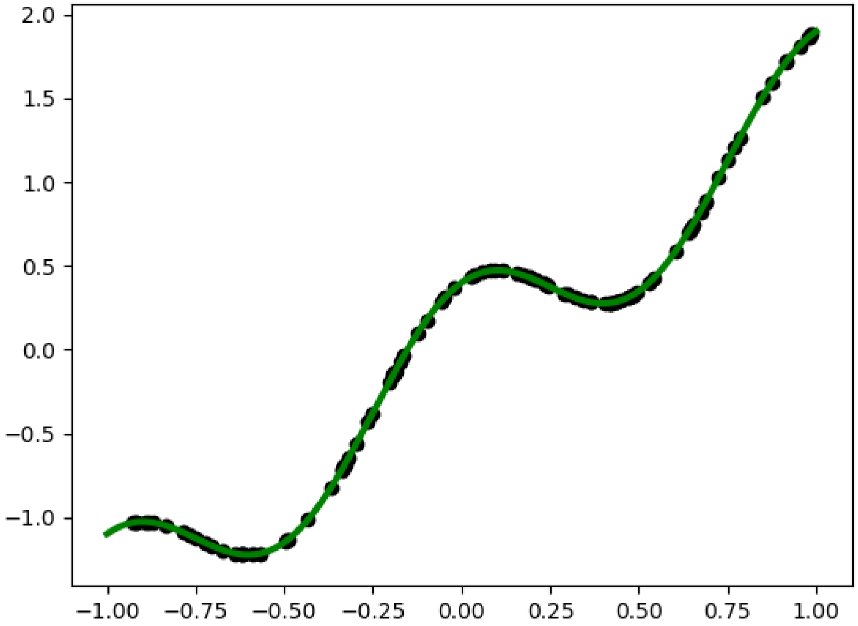
$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{6}, y_i = f(x_i) = x_i^2$$

无噪声样本的拟合结果

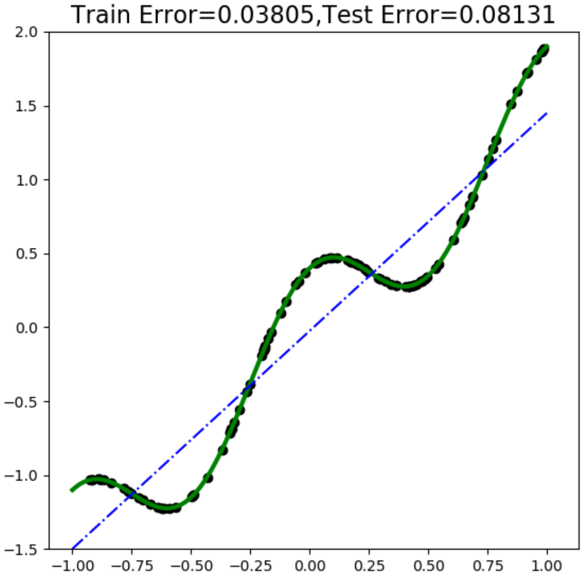


$\mathcal{H}_1$ $\mathcal{H}_2$ $\mathcal{H}_5$

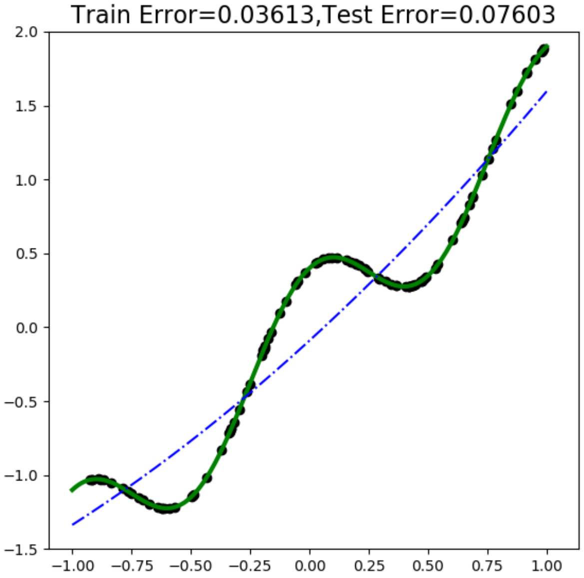
增加样本数量



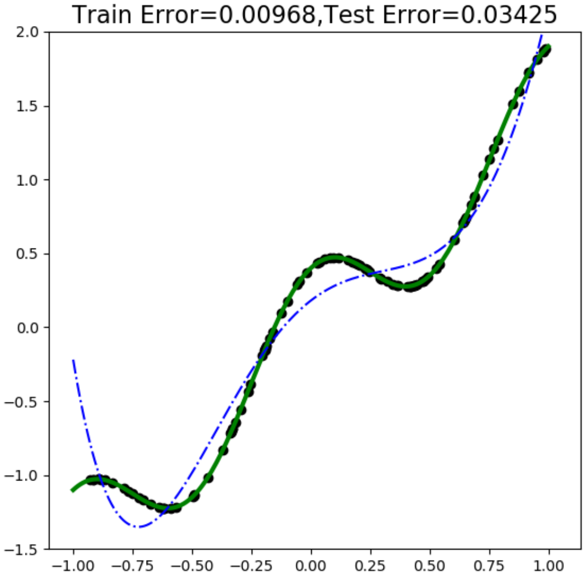
增加样本数量对抗过拟合



$\mathcal{H}_1$

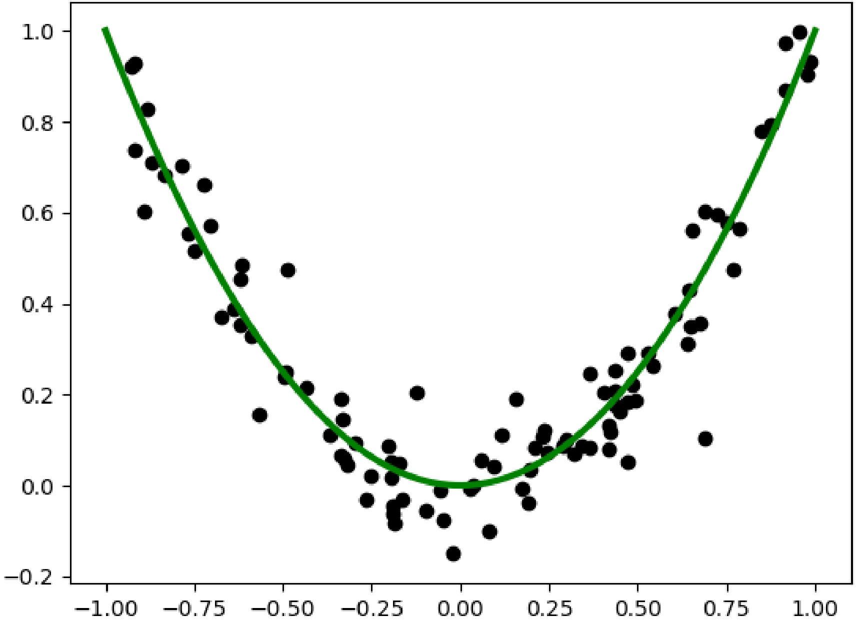


$\mathcal{H}_2$

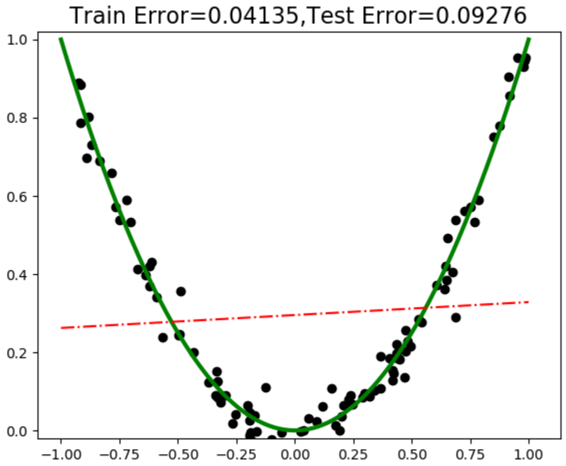


$\mathcal{H}_5$

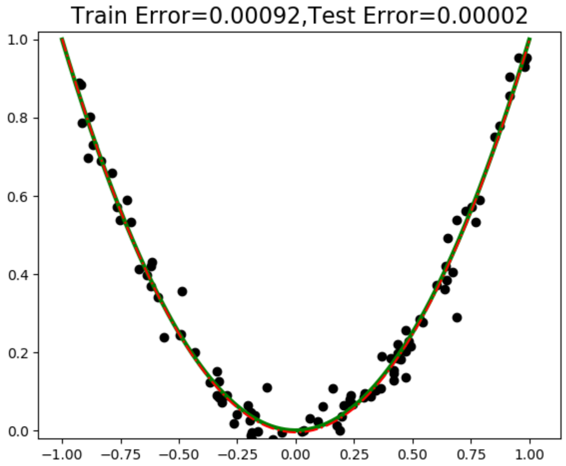
增加样本数量对抗过拟合



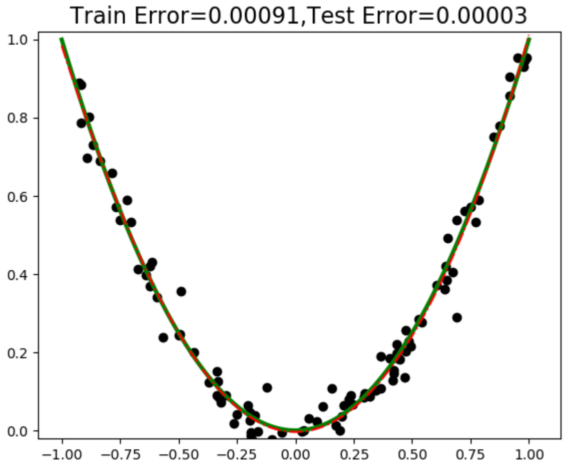
增加样本数量对抗过拟合



$\mathcal{H}_1$



$\mathcal{H}_2$



$\mathcal{H}_5$

过拟合与欠拟合

- 过拟合(高方差)
  - 表现：训练误差很小，测试误差很大
  - 原因：数据噪声很大、模型复杂度过高、样本太少
  - 对策：样本去噪、使用复杂度低的模型、增加样本量
- 欠拟合(高偏差)
  - 表现：训练误差很大
  - 原因：模型过于简单
  - 对策：使用复杂的模型

如何减少过拟合现象

- 1. 降低模型复杂度
  - 采用浅层模型
  - 正则化
- 2. 增加训练样本
  - 样本增广(augmentation)
- 3. 样本清洗
  - 减少样本中的错误和噪声

正则化

Regularization

正则化(Regularization)

$$l(W,b;\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} l\left(W,b;X^{(i)},y^{(i)}\right) \qquad W = \left\{W_{ij}^{(l)}\right\}, b = \left\{b_j^{(l)}\right\}$$

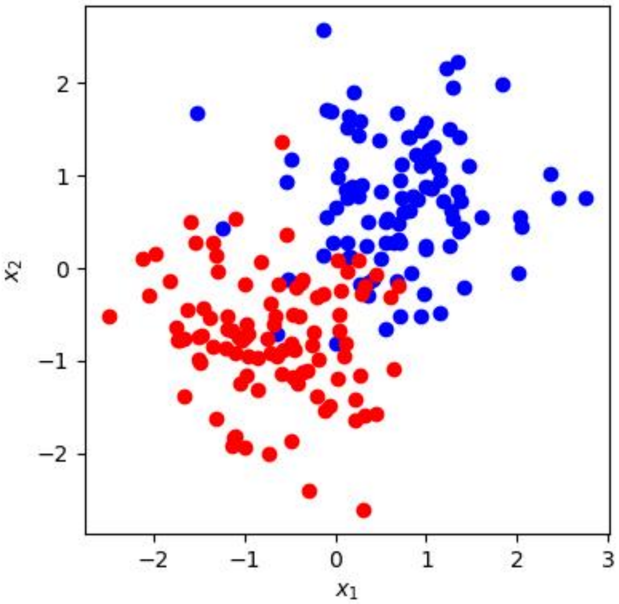
$$l_{reg}(W,b;\lambda,\mathcal{D}) = l(W,b;\mathcal{D}) + \lambda\,\Omega(W)$$

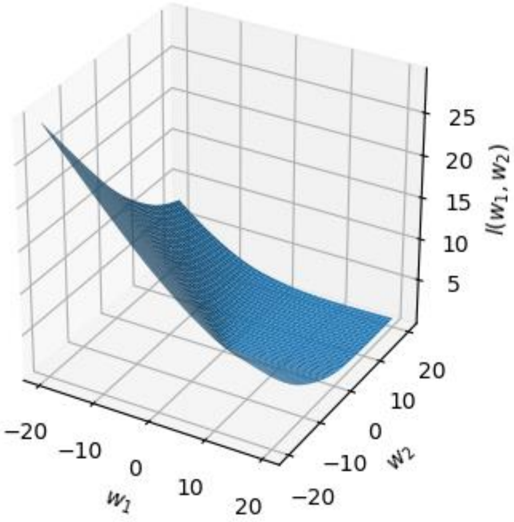
$\Omega(W)$:与训练样本无关的正则化项

$\lambda \geq 0$:正则化因子，控制正则化的强度

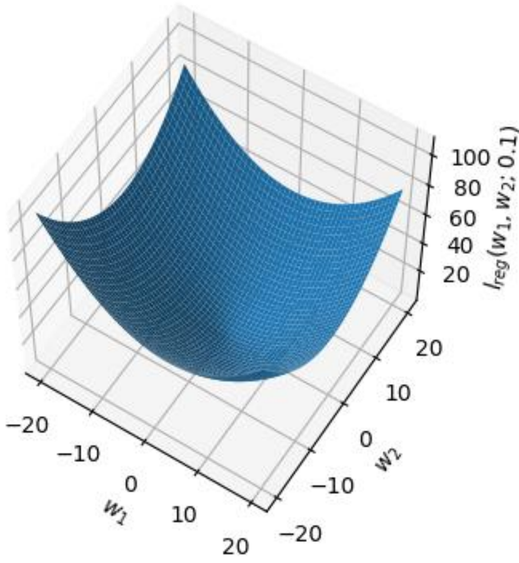
权值衰减(Weight Decay)正则化:$\Omega(W) = \sum_l \sum_i \sum_j \left[W_{ij}^{(l)}\right]^2$

权值衰减正则化



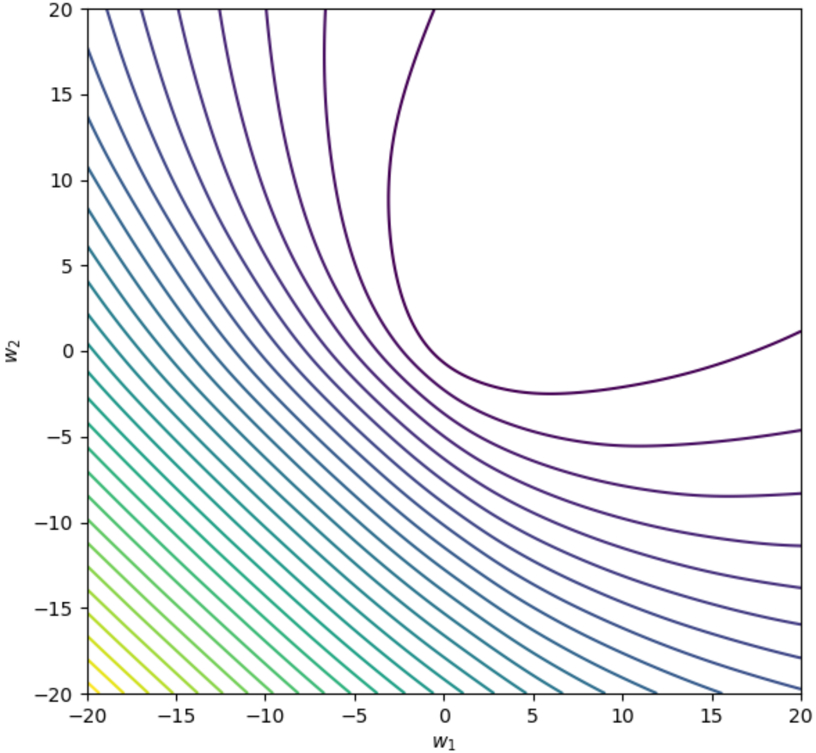


$l(w_1, w_2)$

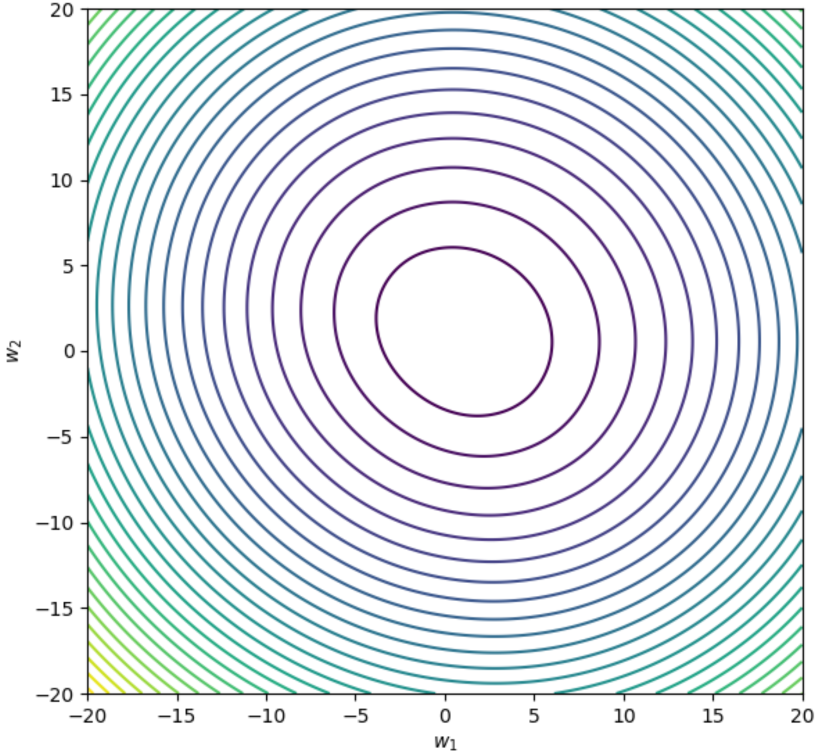


$l_{reg}(w_1, w_2; 0.1)$

权值衰减正则化

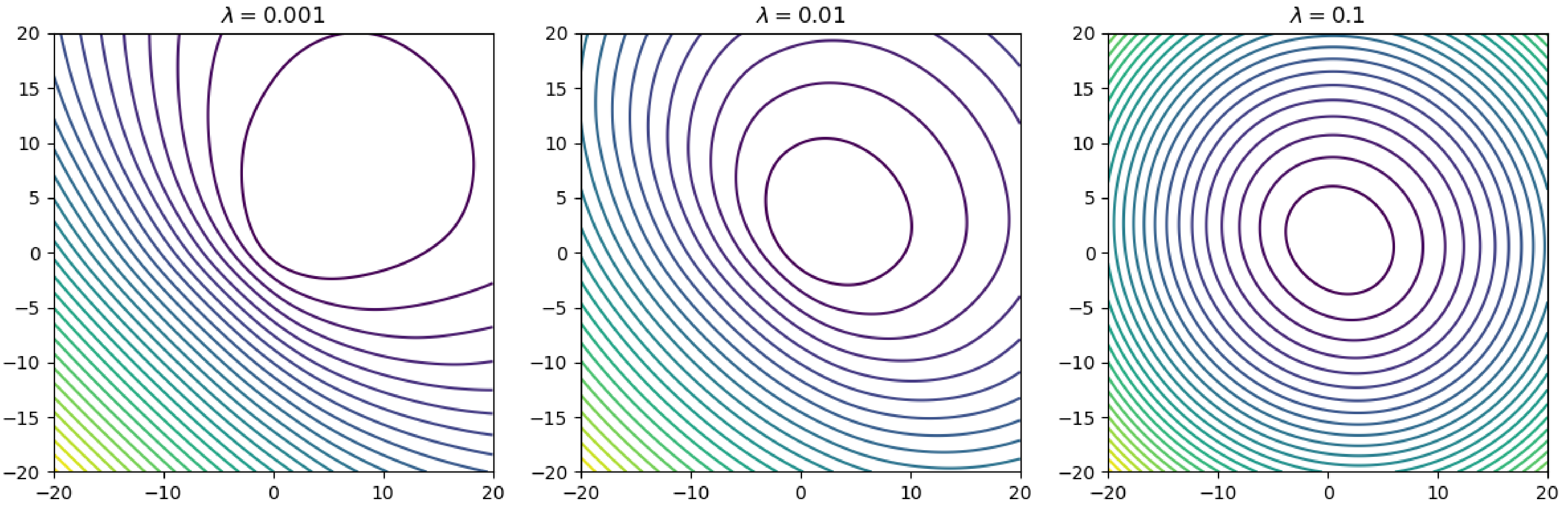


$l(w_1, w_2)$



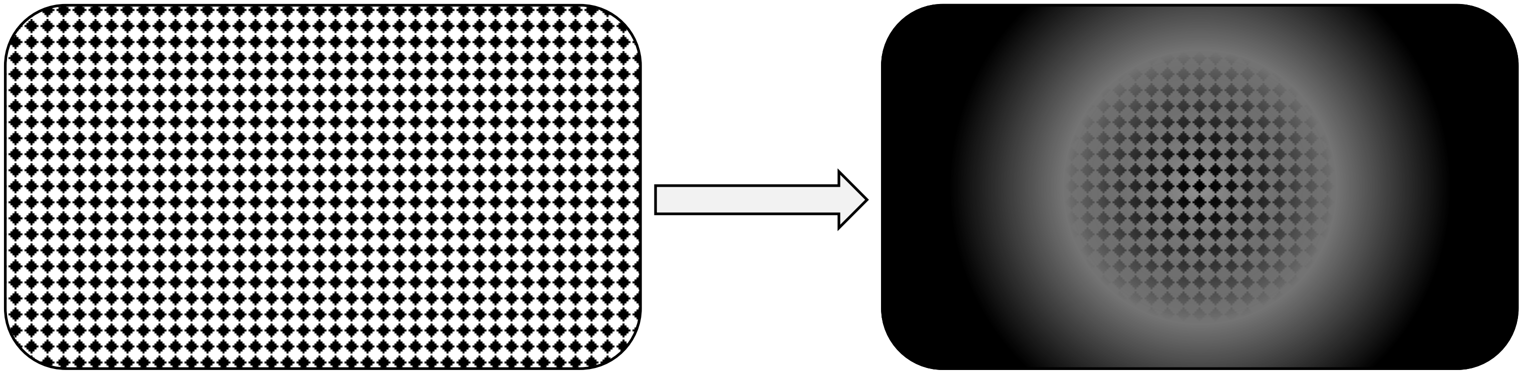
$l_{reg}(w_1, w_2; 0.1)$

权值衰减正则化



$\lambda = 0.001$
$\lambda = 0.01$
$\lambda = 0.1$
20
15
10
5
0
−5
−10
−15
−20
−20
−10
0
10
20

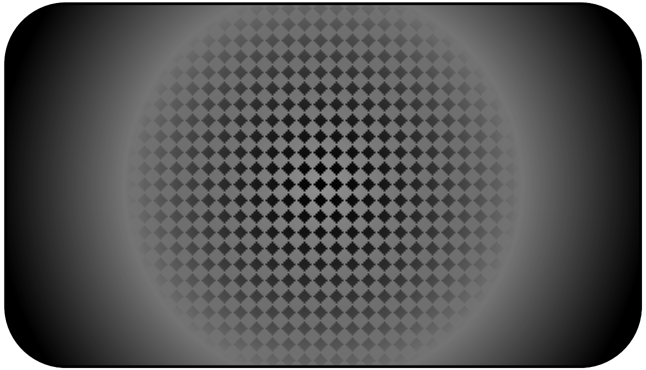
用正则化控制模型复杂度



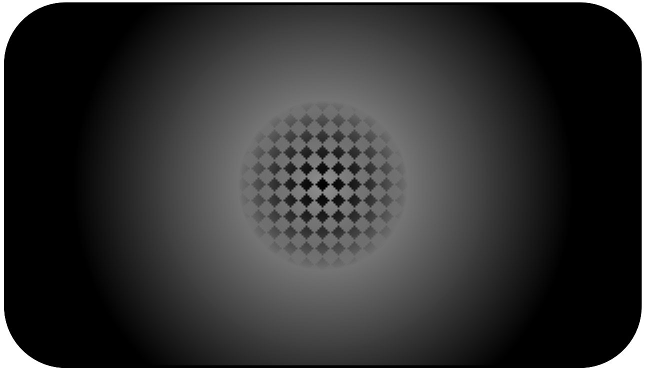
$\mathcal{H} = \{y = h(X;\theta)\}$

$\mathcal{H}_\lambda = \{y = h(X;\theta), \|\theta\|_2 < c_\lambda\}$

用正则化控制模型复杂度

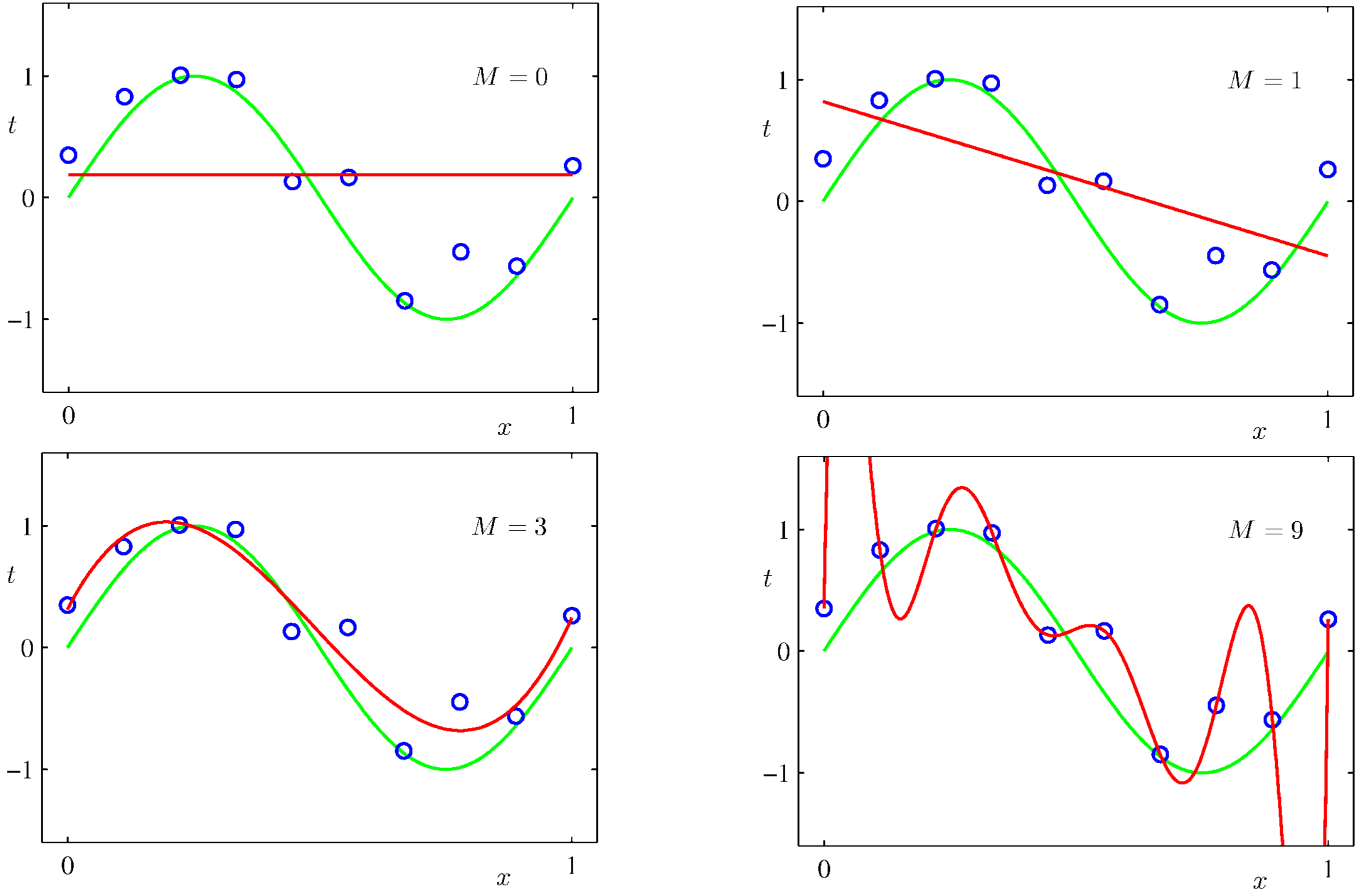


$\mathcal{H}_{\lambda_1}$



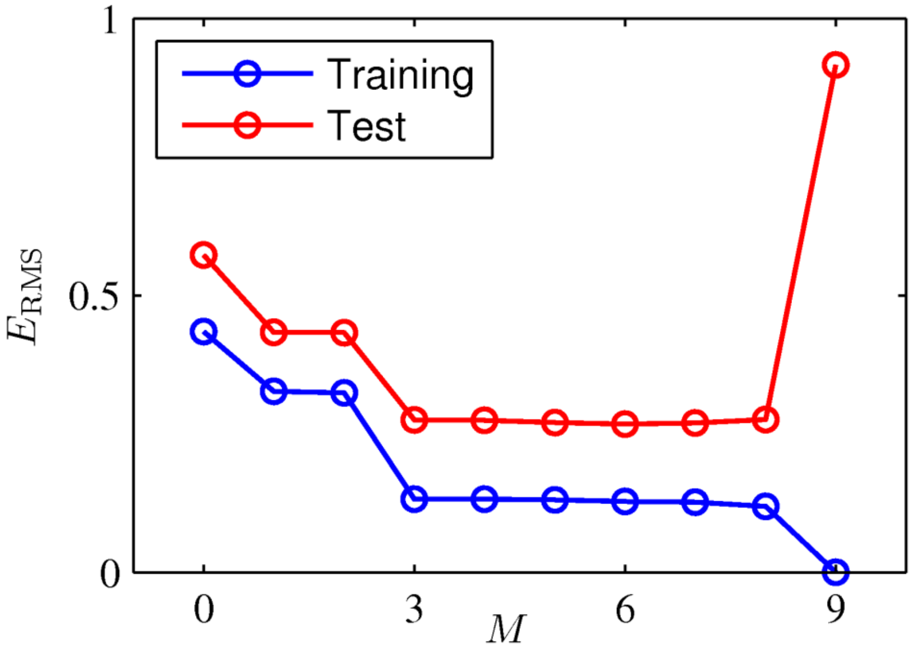
$\mathcal{H}_{\lambda_2}$

$\lambda_1 < \lambda_2$



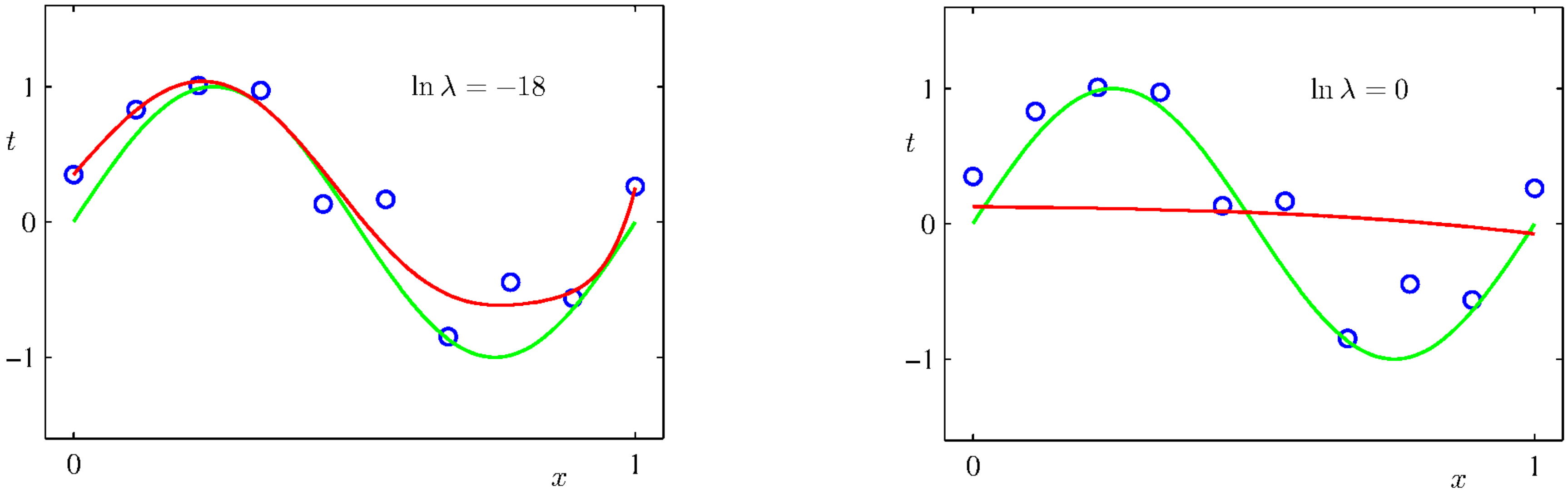
C.M.Bishop:PRML

$$h(x; W) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$

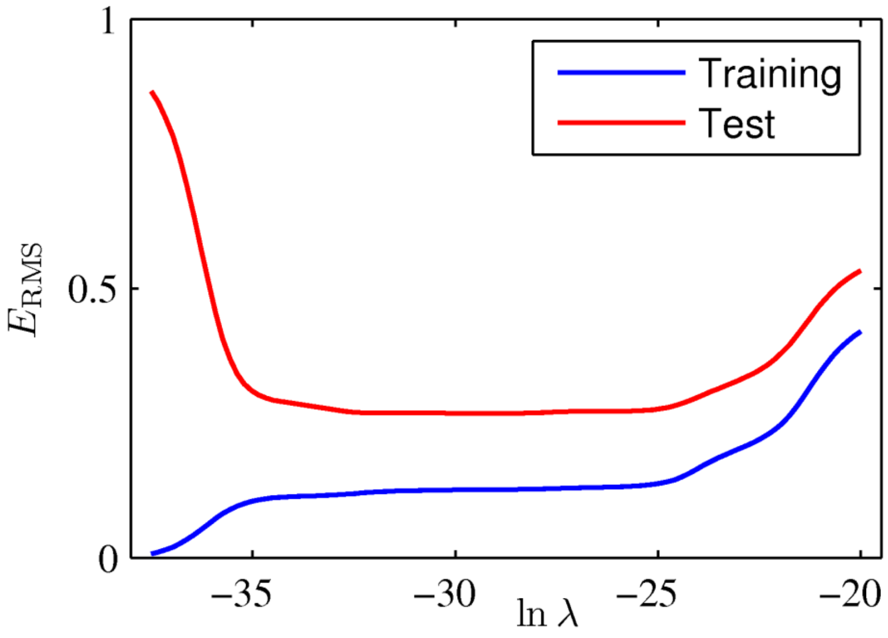


	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

C.M.Bishop:PRML



C.M.Bishop:PRML



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

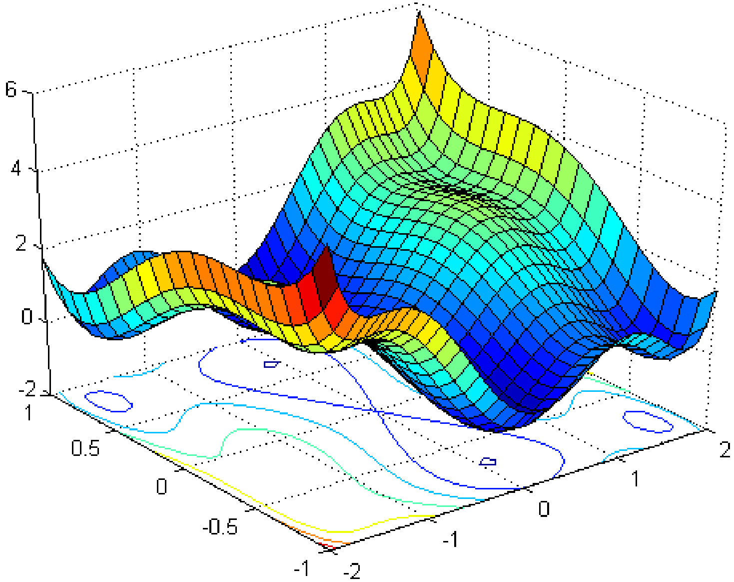
C.M.Bishop:PRML

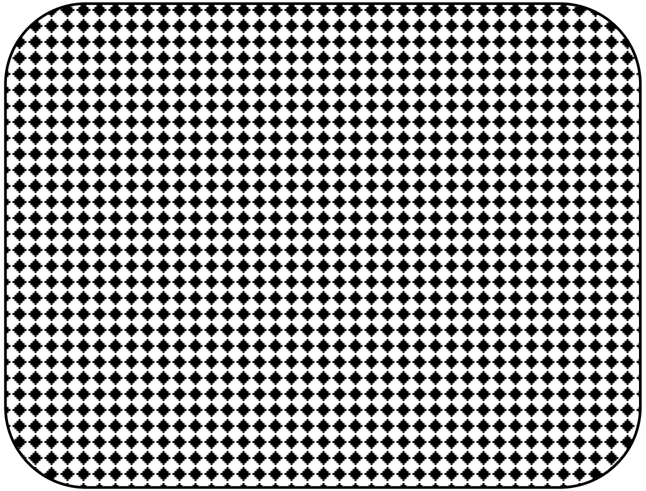
用正则化控制模型复杂度

控制损失函数的最小化过程

神经网络参数的优化过程：$\Theta^* = argmin_{\Theta} l(\Theta; \mathcal{D})$

本质是利用梯度下降法在参数空间中搜索最优参数的过程。

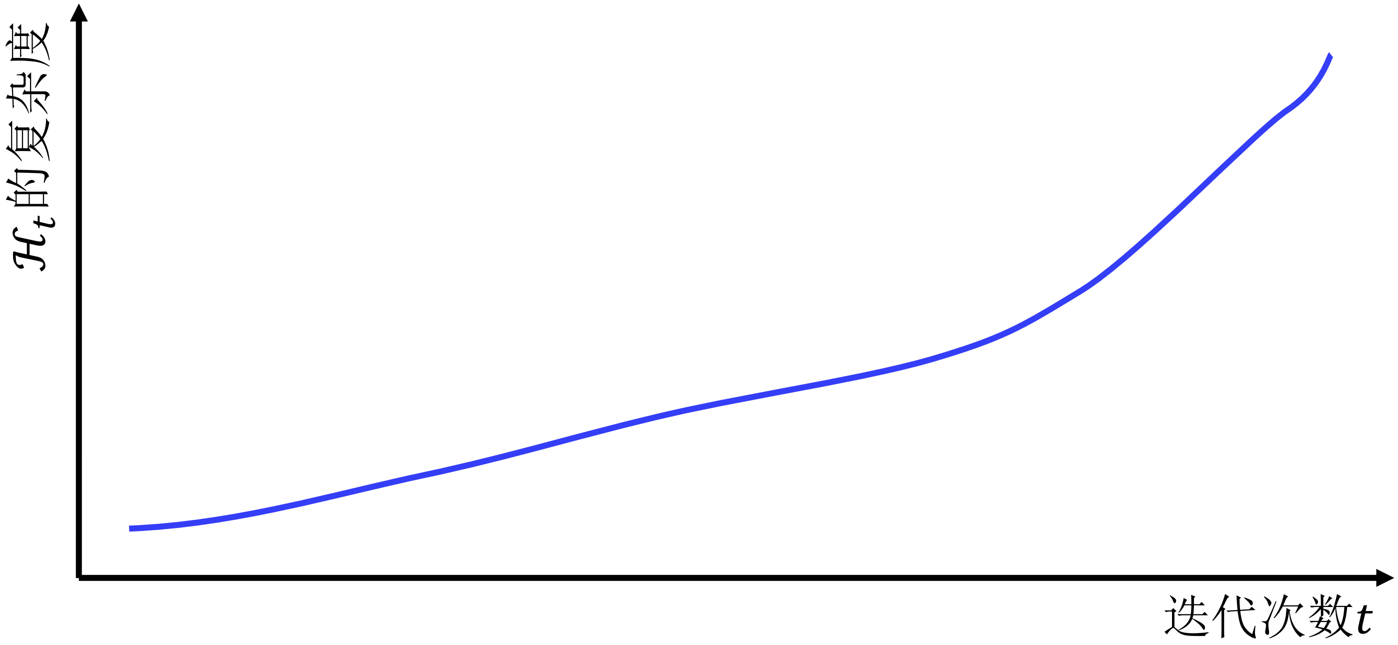




$\mathcal{H} = \{y = h(X; \theta)\}$

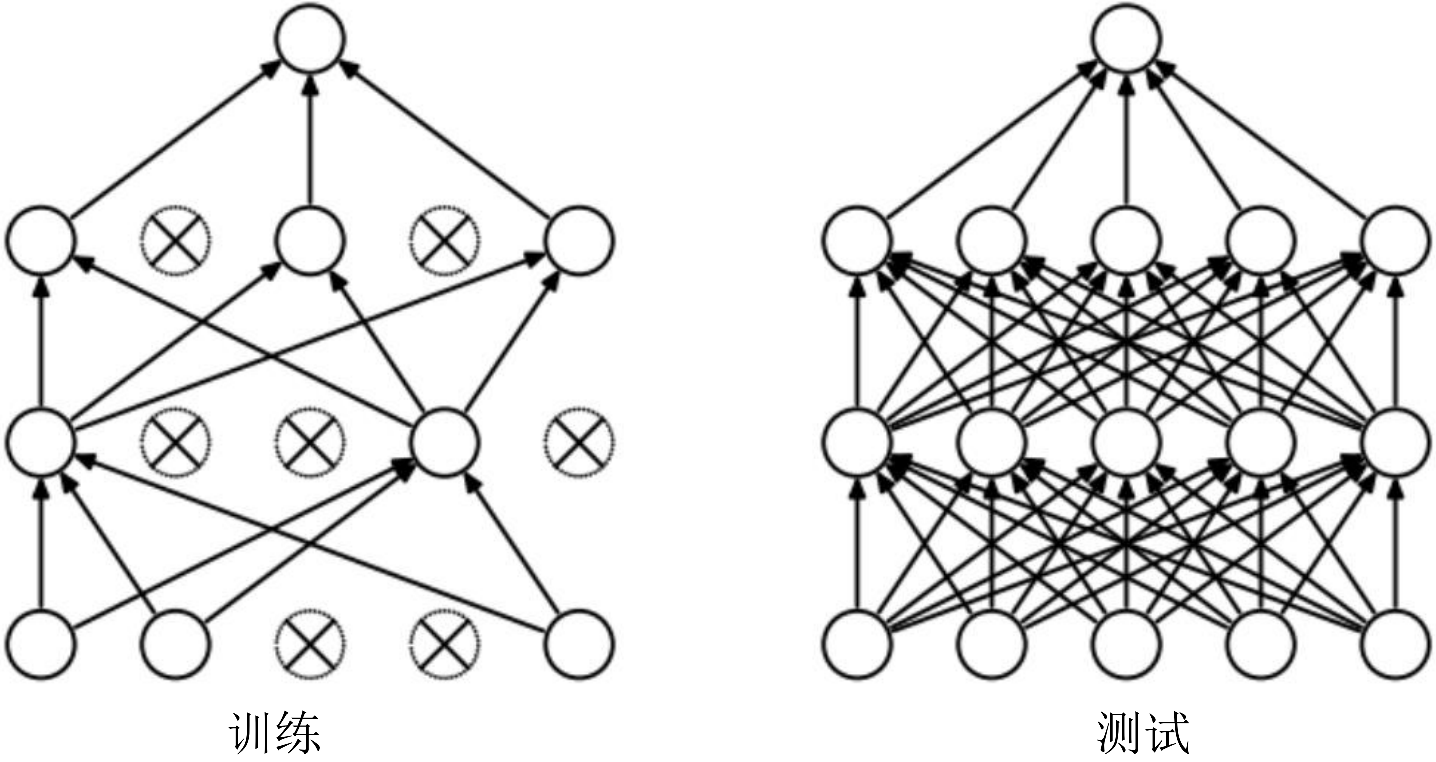
控制损失函数的最小化过程

第t次迭代搜索的模型空间记为$\mathcal{H}_t$



控制迭代次数，可以控制模型空间的搜索范围，从而控制模型的等效复杂度

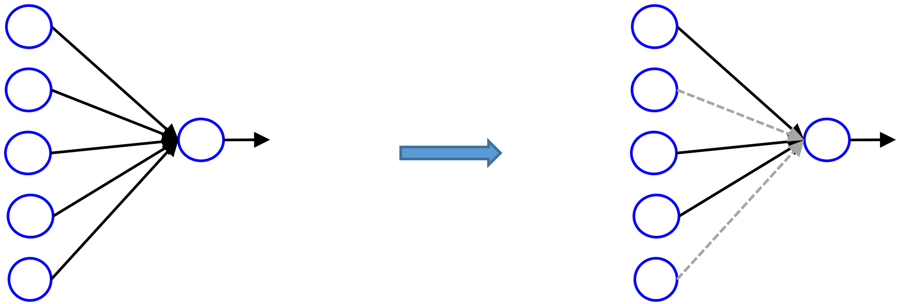
随机失活(Dropout)



Nitish Srivastava, Geoffrey Hinton, et al (2014): Dropout: a simple way to prevent neural networks from overfitting

Dropout

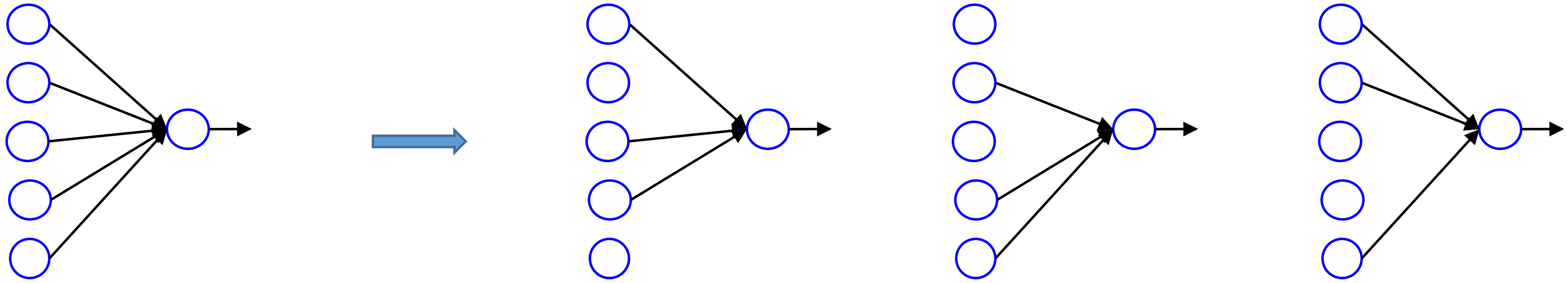
在训练阶段，按照一定的概率p把每一个神经元的输出随机变为0



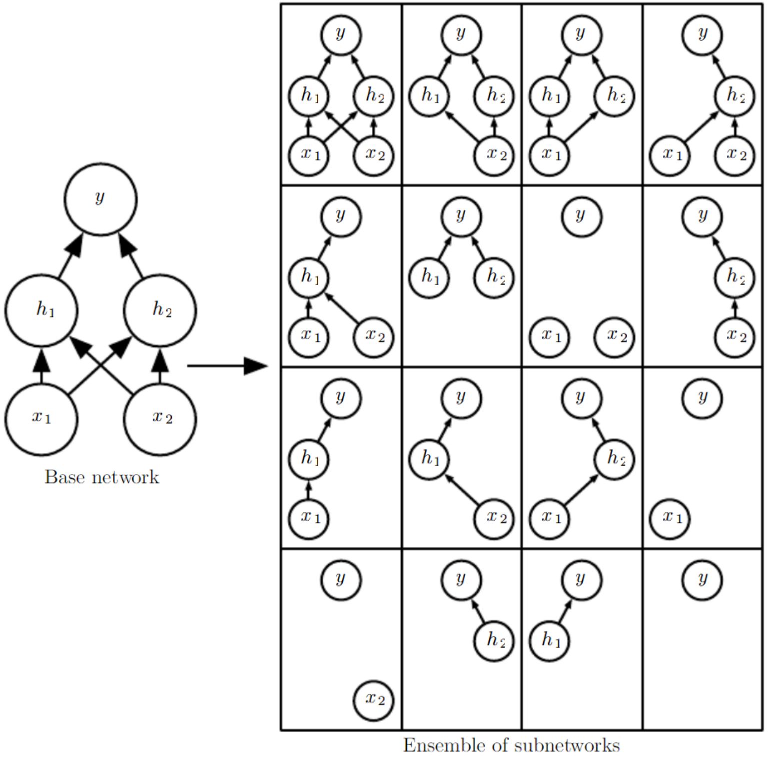
Dropout

在测试阶段，把每一个权重w修正为$(1-p)w$

Dropout



Dropout

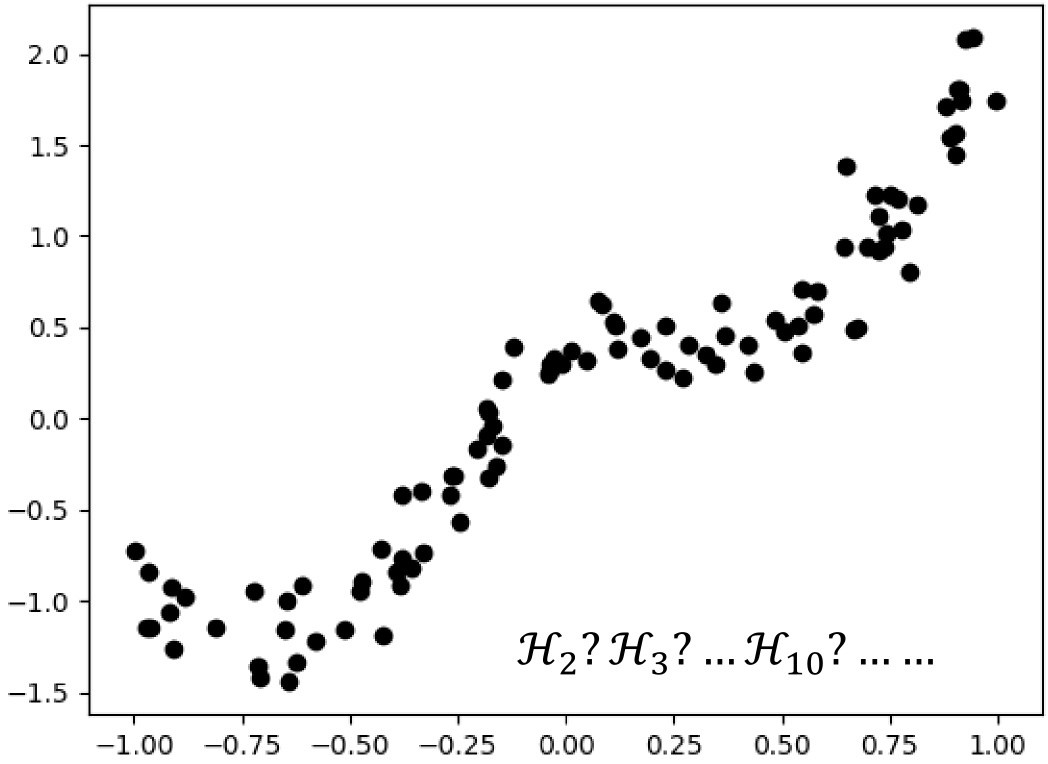


Ian Goodfellow, et al: Deep Learning

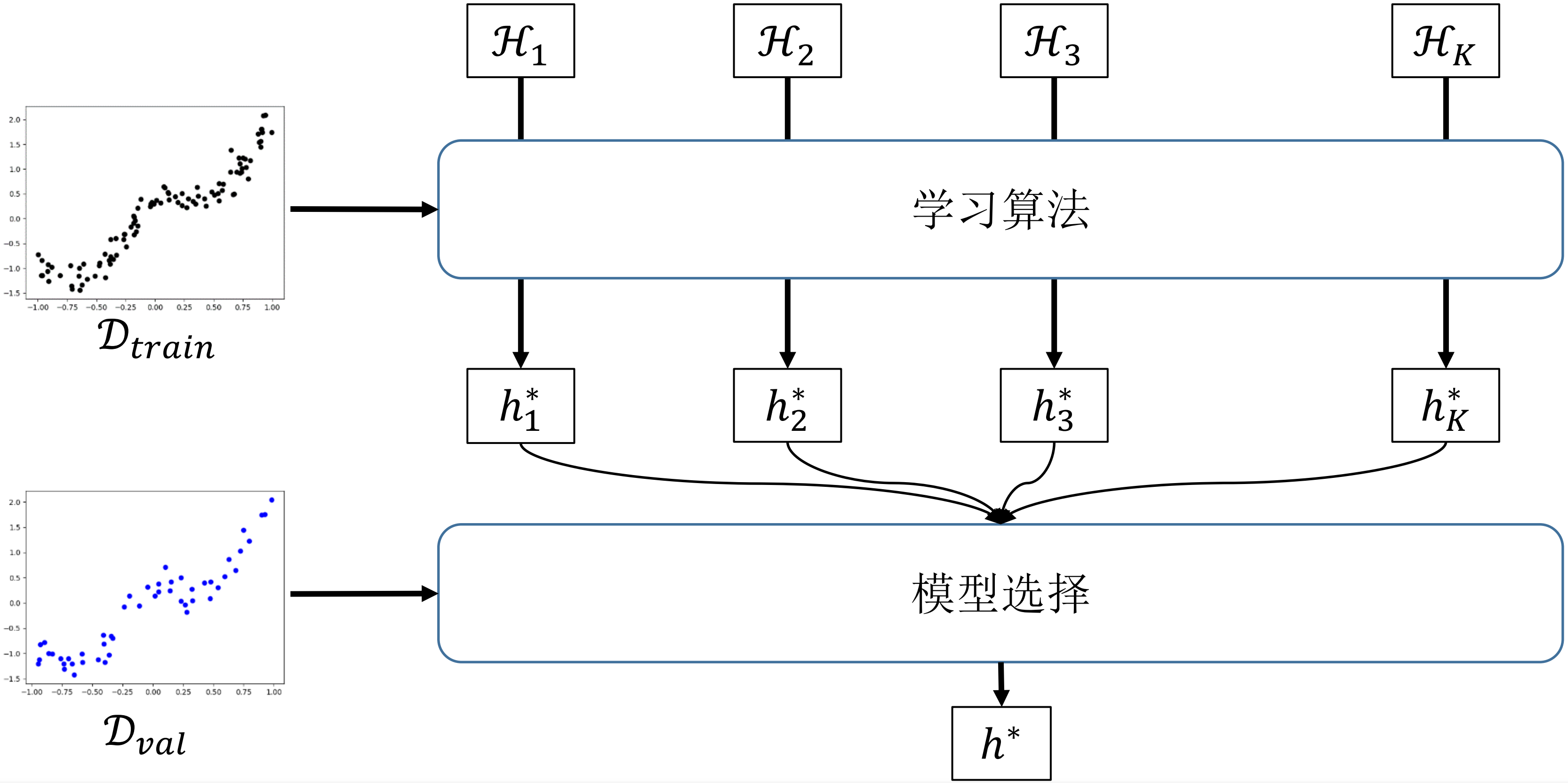
模型选择

Model Selection

如何选择合适的模型空间$\mathcal{H}$?



模型选择



$\mathcal{H}_1$
$\mathcal{H}_2$
$\mathcal{H}_3$
$\mathcal{H}_K$
学习算法
h_1^*
h_2^*
h_3^*
h_K^*
模型选择
h^*
$\mathcal{D}_{train}$
$\mathcal{D}_{val}$

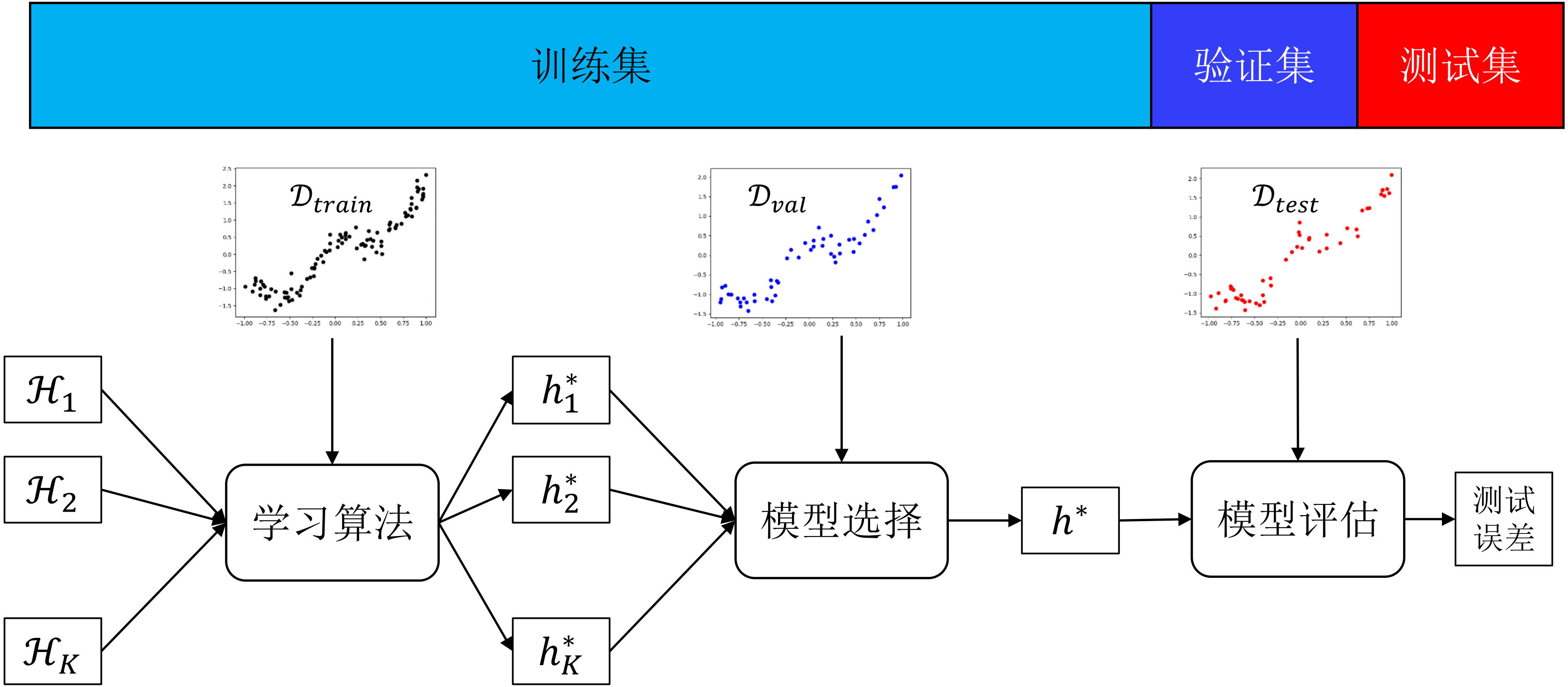
模型选择

- 用独立于训练集的数据集，验证集$\mathcal{D}_{val}$评估每一个模型h_j^*，得到每一个模型的验证集误差:

$$E_{val}(h_j^*) = \frac{1}{|D_{val}|} \sum_{(x,y)\in\mathcal{D}_{val}} l\left(h_j^*(x), y\right)$$

- 输出验证集误差最小的模型h^*: $h^* = argmin_j E_{val}(h_j^*)$

训练集、验证集、测试集

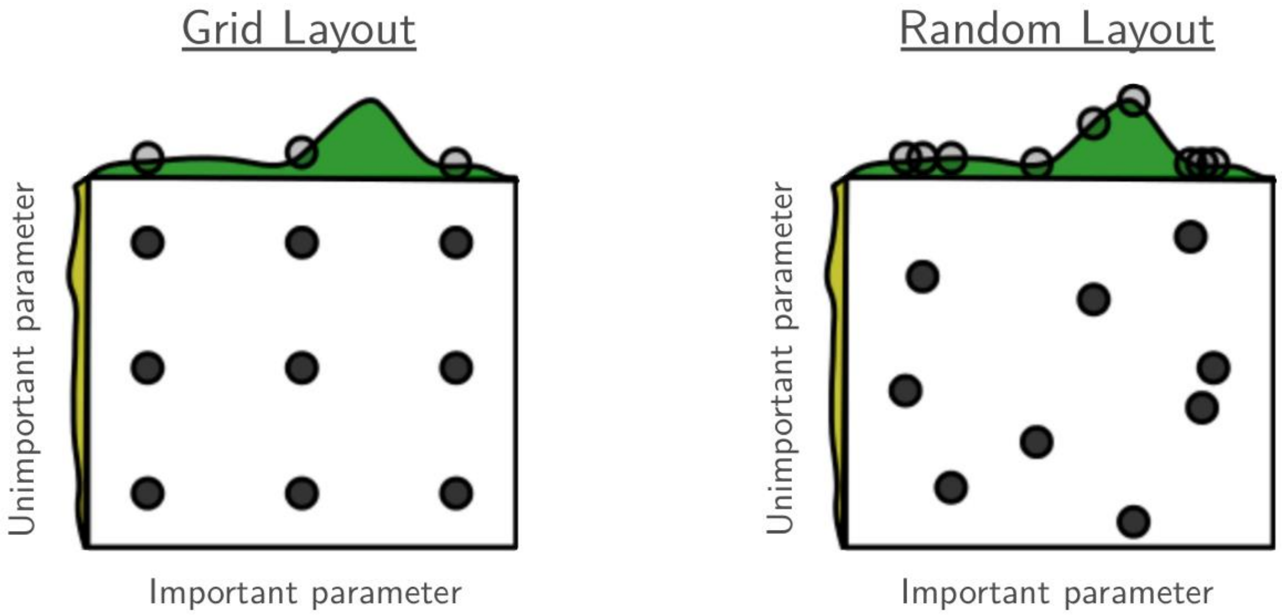


训练集
验证集
测试集
$\mathcal{D}_{train}$
$\mathcal{D}_{val}$
$\mathcal{D}_{test}$
$\mathcal{H}_1$
$\mathcal{H}_2$
$\mathcal{H}_K$
学习算法
h_1^*
h_2^*
h_K^*
模型选择
h^*
模型评估
测试误差

用验证集估计超参数(hyper-parameter)

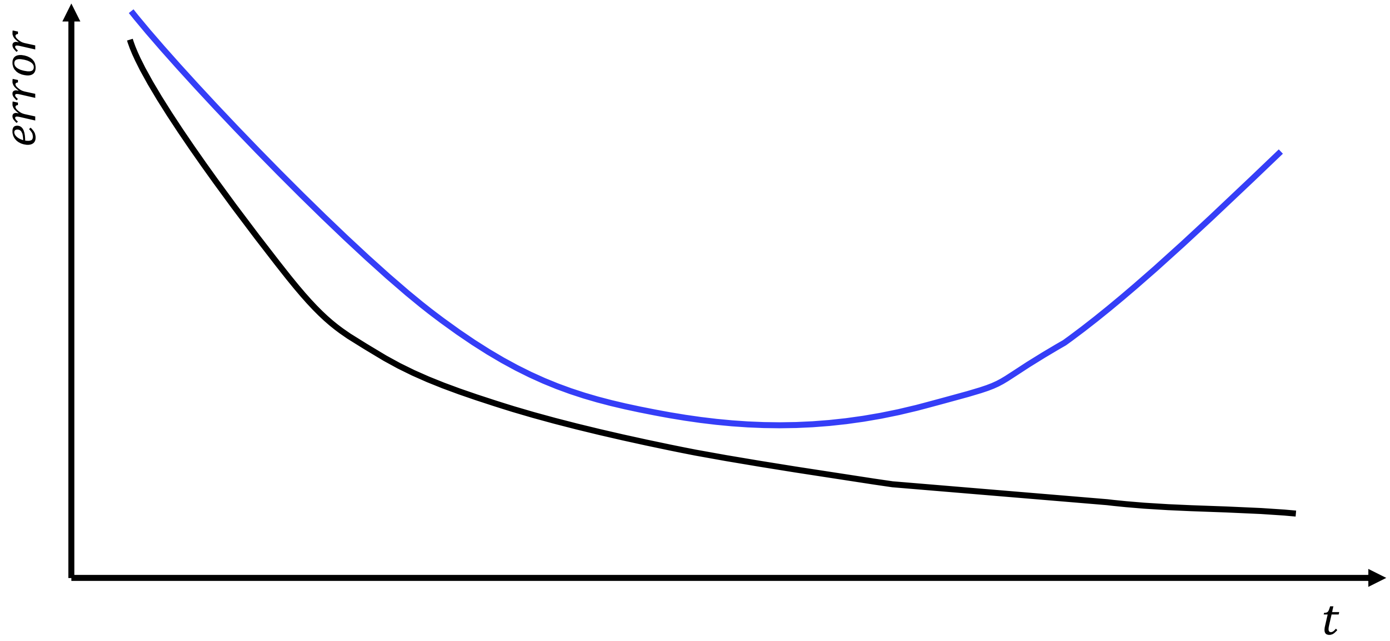
- 训练超参数:
  - 正则化因子λ:$l_{reg}(\theta;\mathcal{D}) = l(\theta;\mathcal{D}) + \lambda\Omega(\theta)$➔$\mathcal{H}_\lambda$
  - 失活（Dropout）概率p➔$\mathcal{H}_p$
  - 迭代次数t➔$\mathcal{H}_t$
  - 学习速率α
- 结构超参数:
  - 神经网络的层数
  - 每一层神经元数目
  - 每一层神经元的响应函数

调节超参数



Bergstra, James; Bengio, Yoshua (2012). "Random Search for Hyper-Parameter Optimization" . Journal of Machine Learning Research.

调节迭代次数：Early Stopping



总结

- 梯度下降法
  - 随机梯度下降
  - 带动量的梯度下降
  - 学习率调节
- 梯度消失与梯度爆炸问题
  - ReLU
  - 权值初始化
  - 梯度裁剪
- 过拟合与欠拟合
- 过拟合问题的解决方法
  - 正则化
  - EarlyStopping
  - 样本增广