西北大学 NORTHWEST UNIVERSITY

信息科学与技术学院 SCHOOL OF INFORMATION SCIENCE AND TECHONLOGY

大数据挖掘与统计学习

软件工程系
文化遗产数字化国家地方工程联合中心
可视化技术研究所
张海波
讲师/博士(后)

统计学习

- 统计学习的方法
  - 分类：
    - Supervised learning
    - Unsupervised learning
    - Semi-supervised learning
    - Reinforcement learning
  - 监督学习：
    - 训练数据 training data
    - 模型 model ------- 假设空间 hypothesis
    - 评价准则 evaluation criterion -------- 策略 strategy
    - 算法 algorithm

监督学习

- Instance， feature vector， feature space
- 输入实例x的特征向量：

$$x=(x^{(1)},x^{(2)},\cdots,x^{(i)},\cdots,x^{(n)})^{\mathrm{T}}$$

- $x^{(i)}$与x_i 不同,后者表示多个输入变量中的第i个

$$x_i=(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(n)})^{\mathrm{T}}$$

- 训练集：

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

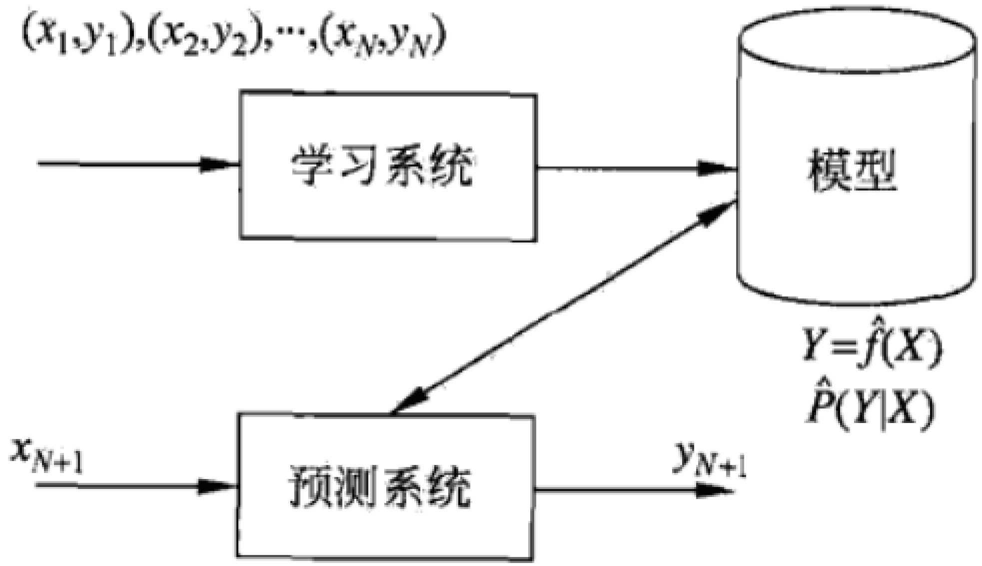
- 输入变量和输出变量：
  - 分类问题、回归问题、标注问题

监督学习

- 联合概率分布
  - 假设输入与输出的随机变量X和Y遵循联合概率分布P(X,Y)
  - P(X,Y)为分布函数或分布密度函数
  - 对于学习系统来说，联合概率分布是未知的，
  - 训练数据和测试数据被看作是依联合概率分布P(X,Y)独立同分布产生的。
- 假设空间
  - 监督学习目的是学习一个由输入到输出的映射，称为模型
  - 模式的集合就是假设空间（hypothesis space）
  - 概率模型:条件概率分布P(Y|X), 决策函数：Y=f(X)

监督学习

- 问题的形式化



$$y_{N+1} = \arg\max_{y_{N+1}} \hat{P}(y_{N+1} \mid x_{N+1})$$

$$y_{N+1} = \hat{f}(x_{N+1})$$

无监督学习

- 训练集：

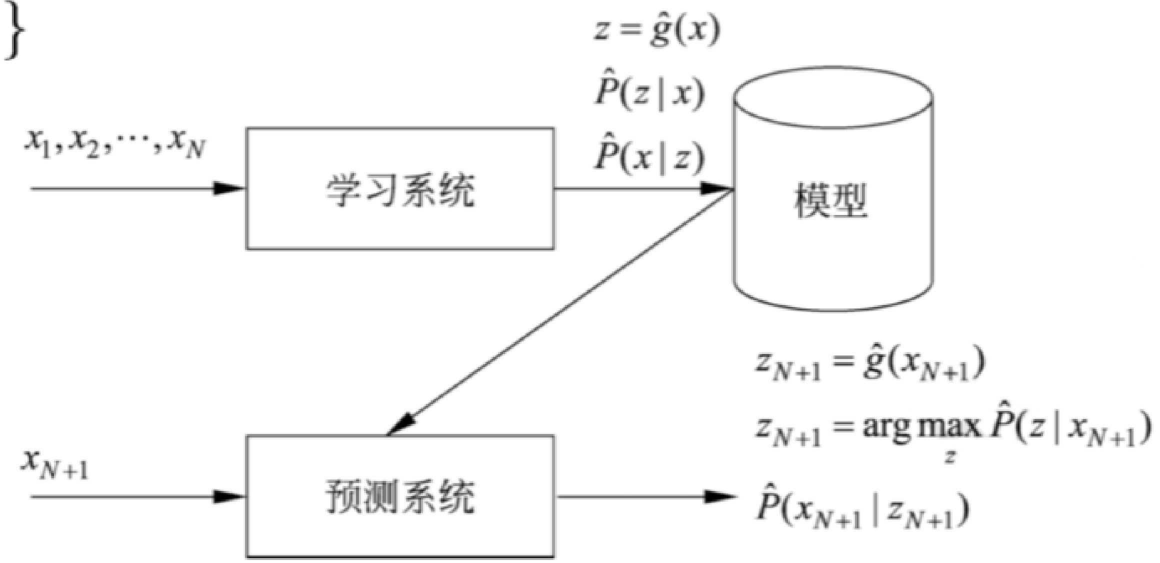
$$U=\{x_1,x_2,\cdots,x_N\}$$

- 模型函数：

$$z=g(x)$$

- 条件概率分布：

$$P(z|x)$$



统计学习三要素

方法=模型+策略+算法

- 模型：
  - 决策函数的集合： $\mathcal{F}=\{f \mid Y=f(X)\}$
  - 参数空间 $\mathcal{F}=\{f \mid Y=f_\theta(X), \theta \in \mathbf{R}^n\}$
  - 条件概率的集合： $\mathcal{F}=\{P \mid P(Y \mid X)\}$
  - 参数空间 $\mathcal{F}=\{P \mid P_\theta(Y \mid X), \theta \in \mathbf{R}^n\}$

统计学习三要素

- 策略
  - 损失函数：一次预测的好坏
  - 风险函数：平均意义下模型预测的好坏
  - 0-1损失函数 0-1 loss function

$$L(Y,f(X))=\begin{cases}1, & Y\neq f(X)\\ 0, & Y=f(X)\end{cases}$$

  - 平方损失函数 quadratic loss function $L(Y,f(X))=(Y-f(X))^2$
  - 绝对损失函数 absolute loss function $L(Y,f(X))=|Y-f(X)|$

统计学习三要素

- 策略
  - 对数损失函数 logarithmic loss function 或对数似然损失函数 loglikelihood loss function

$$L(Y,P(Y|X)) = -\log P(Y|X)$$

  - 损失函数的期望 $R_{\exp}(f) = E_P[L(Y,f(X))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y,f(x))P(x,y)\mathrm{d}x\mathrm{d}y$

  - 风险函数 risk function 期望损失 expected loss
  - 由P(x,y)可以直接求出P(x|y),但不知道，

$$T = \{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

  - 经验风险 empirical risk， 经验损失 empirical loss $R_{\mathrm{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i,f(x_i))$

统计学习三要素

- 策略：经验风险最小化与结构风险最小化
  - 经验风险最小化最优模型

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$$

  - 当样本容量很小时，经验风险最小化学习的效果未必很好，会产生“过拟合 over-fitting”
  - 结构风险最小化 structure risk minimization，为防止过拟合提出的策略，等价于正则化（regularization），加入正则化项regularizer，或罚项 penalty term：

$$R_{\mathrm{srm}}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

统计学习三要素

- 求最优模型就是求解最优化问题：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

统计学习三要素

- 算法:
  - 如果最优化问题有显式的解析式，算法比较简单
  - 但通常解析式不存在，就需要数值计算的方法

模型评估与模型选择

- 训练误差，训练数据集的平均损失 $$R_{\mathrm{emp}}(\hat{f})=\frac{1}{N}\sum_{i=1}^{N}L(y_i,\hat{f}(x_i))$$
- 测试误差，测试数据集的平均损失 $$e_{\mathrm{test}}=\frac{1}{N'}\sum_{i=1}^{N'}L(y_i,\hat{f}(x_i))$$
- 损失函数是0-1 损失时： $$e_{\mathrm{test}}=\frac{1}{N'}\sum_{i=1}^{N'}I(y_i\neq\hat{f}(x_i))$$
- 测试数据集的准确率： $$r_{\mathrm{test}}=\frac{1}{N'}\sum_{i=1}^{N'}I(y_i=\hat{f}(x_i))$$

模型评估与模型选择

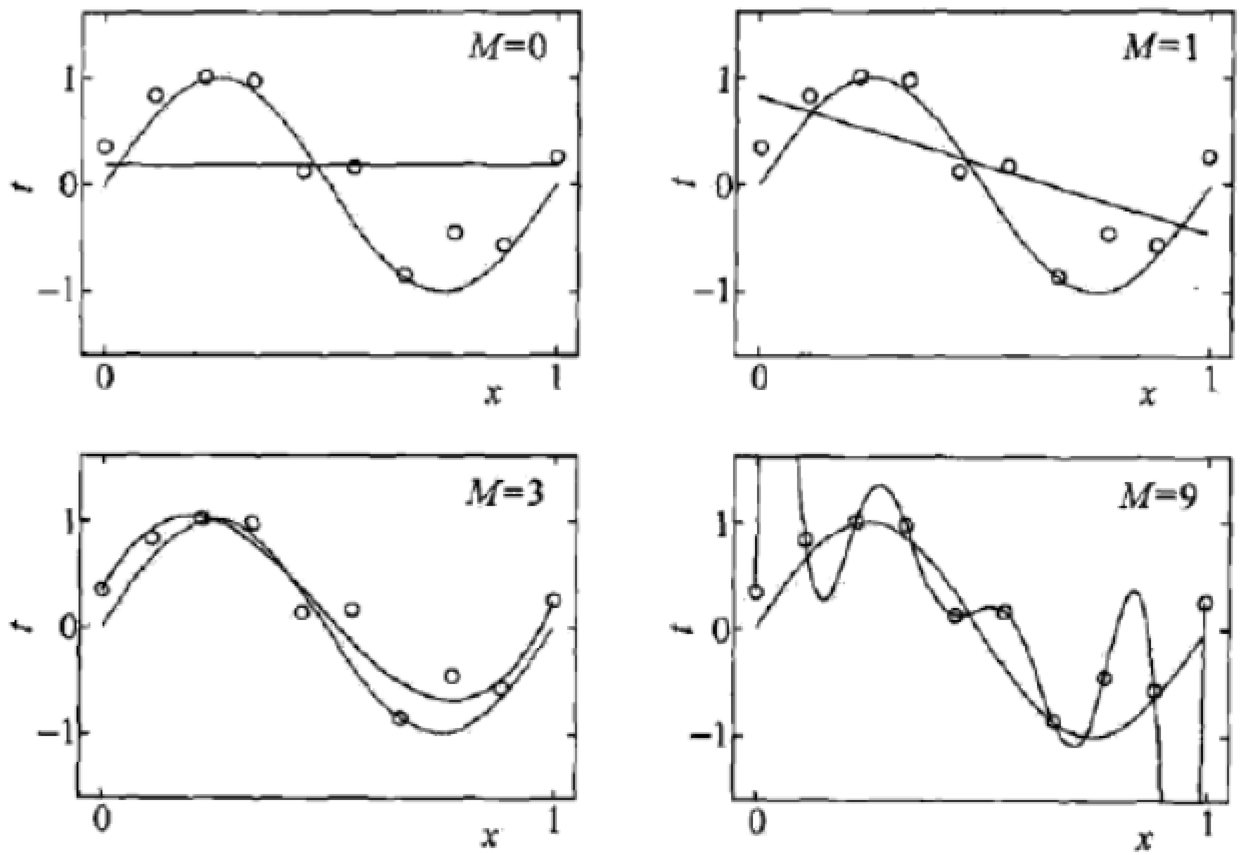
- 过拟合与模型选择
- 假设给定训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$

$$f_M(x,w)=w_0+w_1x+w_2x^2+\cdots+w_Mx^M=\sum_{j=0}^{M}w_jx^j$$

- 经验风险最小：

$$L(w)=\frac{1}{2}\sum_{i=1}^{N}(f(x_i,w)-y_i)^2 \qquad L(w)=\frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=0}^{M}w_jx_i^j-y_i\right)^2 \qquad w_j=\frac{\sum_{i=1}^{N}x_iy_i}{\sum_{i=1}^{N}x_i^{j+1}},\quad j=0,1,2,\cdots,M$$

模型评估与模型选择



正则化与交叉验证

- 正则化一般形式：

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

- 回归问题中：

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i; w) - y_i)^2 + \frac{\lambda}{2} \| w \|^2$$

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i; w) - y_i)^2 + \lambda \| w \|_1$$

正则化与交叉验证

- 交叉验证：
  - 训练集 training set： 用于训练模型
  - 验证集 validation set： 用于模型选择
  - 测试集 test set： 用于最终对学习方法的评估

  - 简单交叉验证
  - S折交叉验证
  - 留一交叉验证

- 交叉验证(Cross-validation)
  - 把数据集合划分成*s* 个子样本；
  - 使用*s* - 1 个子样本作为训练集，另一个作为测试样本—*s*-折交叉验证。
  - 适用于中等规模的数据。
- 留一测试(Leave One Out， *k* = *n*)
  - 适用于小规模数据。

泛化能力 generalization ability

- 泛化误差 generalization error $R_{exp}(\hat{f}) = E_P[L(Y, \hat{f}(X))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y, \hat{f}(x))P(x, y)\mathrm{d}x\mathrm{d}y$
- 泛化误差上界
  - 比较学习方法的泛化能力------比较泛化误差上界
  - 性质：样本容量增加，泛化误差趋于0，假设空间容量越大， 泛化误差越大
- 二分类问题 $X \in \mathbf{R}^n$， $Y \in \{-1, +1\}$
- 期望风险和经验风险 $R(f) = E[L(Y, f(X))]$ $\hat{R}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$

泛化能力 generalization ability

- 经验风险最小化函数： $f_N = \arg\min_{f \in \mathcal{F}} \hat{R}(f)$

- 泛化能力： $R(f_N) = E[L(Y, f_N(X))]$

- 定理：泛化误差上界，二分类问题，

  当假设空间是有限个函数的结合 $\mathcal{F} = \{f_1, f_2, \cdots, f_d\}$，

  对任意一个函数f，至少以概率1-δ，以下不等式成立：

  $$R(f) \leqslant \hat{R}(f) + \varepsilon(d, N, \delta)$$

  $$\varepsilon(d, N, \delta) = \sqrt{\frac{1}{2N}\left(\log d + \log\frac{1}{\delta}\right)}$$

生成模型与判别模型

- 监督学习的目的就是学习一个模型：
- 决策函数： $Y = f(X)$
- 条件概率分布： $P(Y \mid X)$
- 生成方法Generative approach 对应生成模型：generative model，

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)}$$

- 朴素贝叶斯法和隐马尔科夫模型

生成模型与判别模型

- 判别方法由数据直接学习决策函数f(X)或条件概率分布P(Y|X)作为预测的模型，即判别模型
- Discriminative approach对应discriminative model

$$Y = f(X)$$

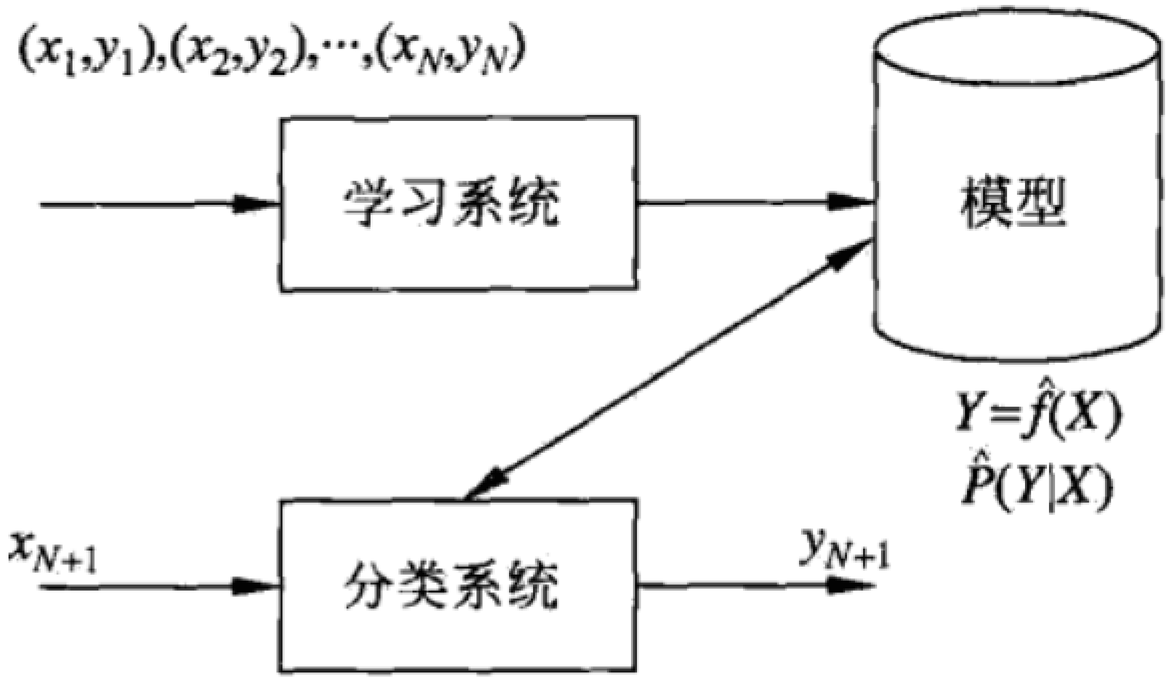
$$P(Y \mid X)$$

- K近邻法、感知机、决策树、logistic回归模型、最大熵模型、支持向量机、提升方法和条件随机场。

生成模型与判别模型

- 各自优缺点：
  - 生成方法：可还原出联合概率分布P(X,Y), 而判别方法不能。生成方法的收敛速度更快，当样本容量增加的时候，学到的模型可以更快地收敛于真实模型；当存在隐变量时，仍可以使用生成方法，而判别方法则不能用。
  - 判别方法：直接学习到条件概率或决策函数，直接进行预测，往往学习的准确率更高；由于直接学习Y=f(X)或P(Y|X),可对数据进行各种程度上的抽象、定义特征并使用特征，因此可以简化学习过程。

分类问题



分类问题

- 二分类评价指标
  - TP true positive 正到正
  - FN false negative 正到负
  - FP false positive 负到正
  - TN true negative 负到负

- 精确率 $$P = \frac{TP}{TP + FP}$$
- 召回率 $$R = \frac{TP}{TP + FN}$$
- F_1值 $$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R}$$

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

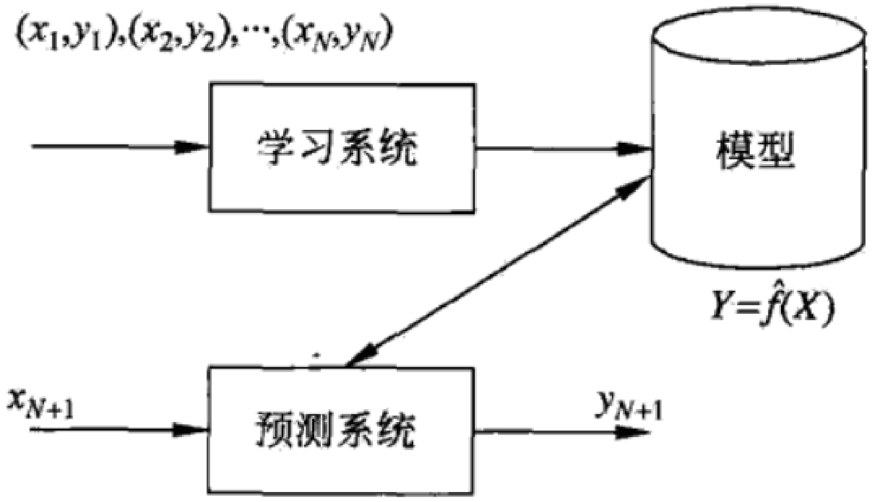
标注问题

- 标注：tagging， 结构预测：structure prediction
- 输入：观测序列， 输出：标记序列或状态序列
- 学习和标注两个过程
- 训练集： $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$
- 观测序列： $x_i=(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(n)})^{\mathrm{T}}$， $i=1,2,\cdots,N$
- 输出标记序列： $y_i=(y_i^{(1)},y_i^{(2)},\cdots,y_i^{(n)})^{\mathrm{T}}$
- 模型：条件概率分布 $P(Y^{(1)},Y^{(2)},\cdots,Y^{(n)}\mid X^{(1)},X^{(2)},\cdots,X^{(n)})$

回归问题

- 回归模型是表示从输入变量到输出变量之间映射的函数.回归问题的学习等价于函数拟合。
- 学习和预测两个阶段
- 训练集:

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$



回归问题

- 回归学习最常用的损失函数是平方损失函数，在此情况下，回归问题可以由 著名的最小二乘法(least squares)求解。

- 股价预测

4.4 最小二乘法

让我们应用上一节中得到的方程来推导最小二乘方程。假设我们得到矩阵$A \in \mathbb{R}^{m\times n}$（为了简单起见，我们假设$A$是满秩）和向量$b \in \mathbb{R}^{m}$，从而使$b \notin \mathcal{R}(A)$。在这种情况下，我们将无法找到向量$x \in \mathbb{R}^{n}$，由于$Ax=b$，因此我们想要找到一个向量$x$，使得$Ax$尽可能接近 b，用欧几里德范数的平方$\|Ax-b\|_2^2$来衡量。

使用公式$\|x\|^2 = x^T x$“，我们可以得到：

$$\begin{aligned}\|Ax-b\|_2^2 &= (Ax-b)^T(Ax-b)\\ &= x^T A^T Ax - 2b^T Ax + b^T b\end{aligned}$$

根据x的梯度，并利用上一节中推导的性质：

$$\begin{aligned}\nabla_x\left(x^T A^T Ax - 2b^T Ax + b^T b\right) &= \nabla_x x^T A^T Ax - \nabla_x 2b^T Ax + \nabla_x b^T b\\ &= 2A^T Ax - 2A^T b\end{aligned}$$

将最后一个表达式设置为零，然后解出x，得到了正规方程：

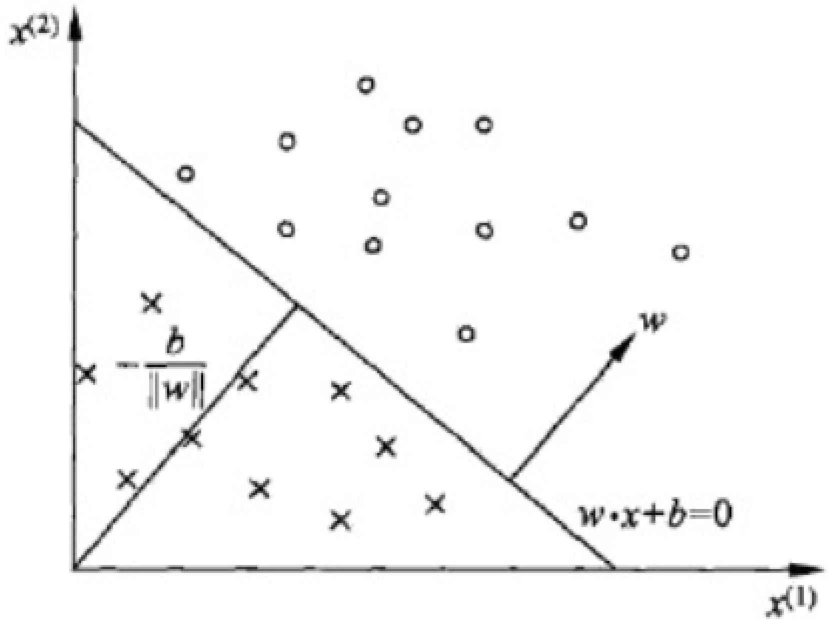
$$x = (A^T A)^{-1} A^T b$$

感知机(Perceptron)

- 输入为实例的特征向量，输出为实例的类别，取+1和-1；
- 感知机对应于输入空间中将实例划分为正负两类的分离超平面，属于判别模型；
- 导入基于误分类的损失函数；
- 利用梯度下降法对损失函数进行极小化；
- 感知机学习算法具有简单而易于实现的优点，分为原始形式和对偶形式；
- 1957年由Rosenblatt提出，是神经网络与支持向量机的基础。

感知机模型

- 感知机几何解释：
- 线性方程： $w \cdot x + b = 0$
- 对应于超平面S，w为法向量，b截距，分离正、负类：
- 分离超平面：



感知机学习策略

- 如何定义损失函数?
- 自然选择：误分类点的数目，但损失函数不是w,b 连续可导，不宜优化。
- 另一选择：误分类点到超平面的总距离：
- 距离： $$\frac{1}{\|w\|}|w \cdot x_0 + b|$$

误分类点： $$-y_i(w \cdot x_i + b) > 0$$

误分类点距离： $$-\frac{1}{\|w\|} y_i(w \cdot x_i + b)$$

总距离： $$-\frac{1}{\| w \|}\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

感知机学习策略

- 损失函数：

$$L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

- M为误分类点的数目

感知机学习算法

- 求解最优化问题：

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

- 随机梯度下降法，
- 首先任意选择一个超平面，w，b，然后不断极小化目标函数,损失函数L的梯度：

$$\nabla_w L(w,b) = -\sum_{x_i \in M} y_i x_i \qquad \nabla_b L(w,b) = -\sum_{x_i \in M} y_i$$

- 选取误分类点更新：

$$w \leftarrow w + \eta y_i x_i \qquad b \leftarrow b + \eta y_i$$

感知机学习算法

- 感知机学习算法的原始形式：

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，

其中 $x_i\in\mathcal{X}=\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$， $i=1,2,\cdots,N$

学习率 $\eta\ (0<\eta\leqslant 1)$；

输出： w,b；感知机模型 $f(x)=\mathrm{sign}(w\cdot x+b)$

（1）选取初值 w_0,b_0

（2）在训练集中选取数据 (x_i,y_i)

（3）如果 $y_i(w\cdot x_i+b)\leqslant 0$

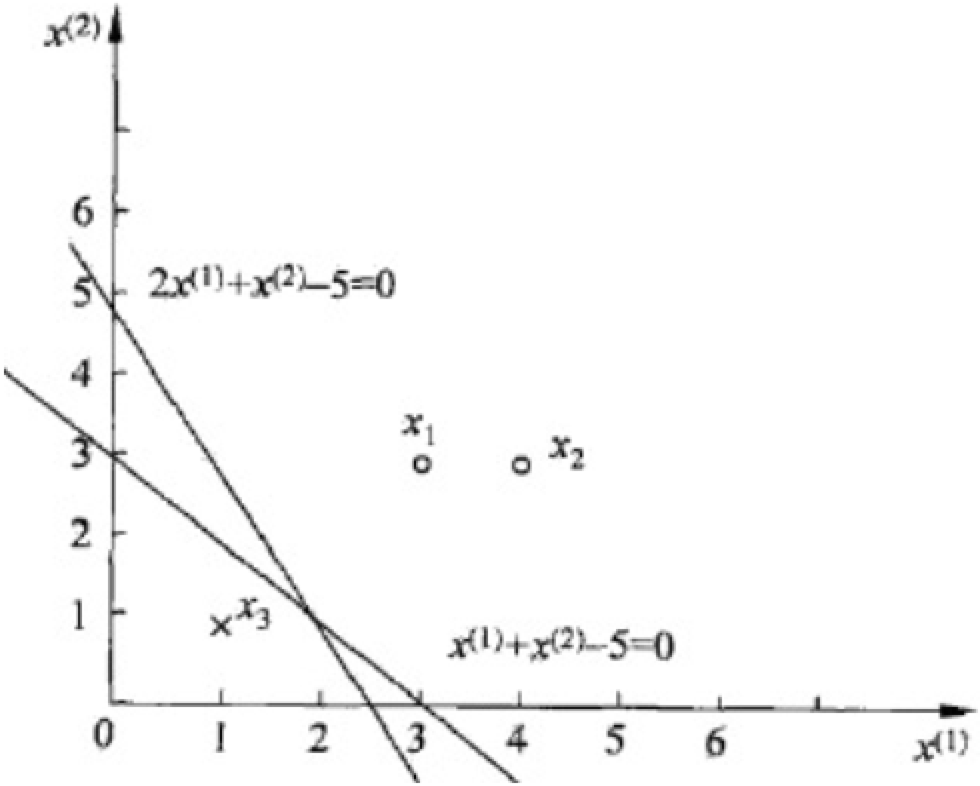
$$w\leftarrow w+\eta y_i x_i$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点

感知机学习算法

- 例：正例： $x_1=(3,3)^T$， $x_2=(4,3)^T$ 负例： $x_3=(1,1)^T$，



感知机学习算法

- 解：构建优化问题： $\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i(w \cdot x + b)$

- 求解：w，b， $\eta = 1$

  （1）取初值 $w_0 = 0$， $b_0 = 0$

  （2）对 $x_1 = (3,3)^T$， $y_1(w_0 \cdot x_1 + b_0) = 0$，未能被正确分类，更新 w, b

  $w_1 = w_0 + y_1 x_1 = (3,3)^T$， $b_1 = b_0 + y_1 = 1$

- 得线性模型： $w_1 \cdot x + b_1 = 3x^{(1)} + 3x^{(2)} + 1$

- （3）x_2，显然，$y_i(w_1 \cdot x_i + b_1) > 0$，被正确分类，

  对 $x_3 = (1,1)^T$， $y_3(w_1 \cdot x_3 + b_1) < 0$，被误分类，

  $w_2 = w_1 + y_3 x_3 = (2,2)^T$， $b_2 = b_1 + y_3 = 0$

感知机学习算法

- 得到线性模型：$w_2 \cdot x + b_2 = 2x^{(1)} + 2x^{(2)}$
- 如此继续下去：$w_7 = (1,1)^{\mathrm{T}}$， $b_7 = -3$

$$w_7 \cdot x + b_7 = x^{(1)} + x^{(2)} - 3$$

- 分离超平面： $x^{(1)} + x^{(2)} - 3 = 0$
- 感知机模型：$f(x) = \mathrm{sign}(x^{(1)} + x^{(2)} - 3)$

迭代次数	误分类点	w	b	$w \cdot x + b$
0		0	0	0
1	x_1	$(3,3)^{\mathrm{T}}$	1	$3x^{(1)} + 3x^{(2)} + 1$
2	x_3	$(2,2)^{\mathrm{T}}$	0	$2x^{(1)} + 2x^{(2)}$
3	x_3	$(1,1)^{\mathrm{T}}$	−1	$x^{(1)} + x^{(2)} - 1$
4	x_3	$(0,0)^{\mathrm{T}}$	−2	−2
5	x_1	$(3,3)^{\mathrm{T}}$	−1	$3x^{(1)} + 3x^{(2)} - 1$
6	x_3	$(2,2)^{\mathrm{T}}$	−2	$2x^{(1)} + 2x^{(2)} - 2$
7	x_3	$(1,1)^{\mathrm{T}}$	−3	$x^{(1)} + x^{(2)} - 3$
8	0	$(1,1)^{\mathrm{T}}$	−3	$x^{(1)} + x^{(2)} - 3$

梯度下降法

- 梯度下降（GD）是最小化风险函数、损失函数的一种常用方法。
- 在应用机器学习算法时，通常采用梯度下降法来对采用的算法进行训练。

- 梯度下降（GD）是最小化风险函数、损失函数的一种常用方法。
- 在应用机器学习算法时，通常采用梯度下降法来对采用的算法进行训练。

梯度下降法包含三种不同形式：

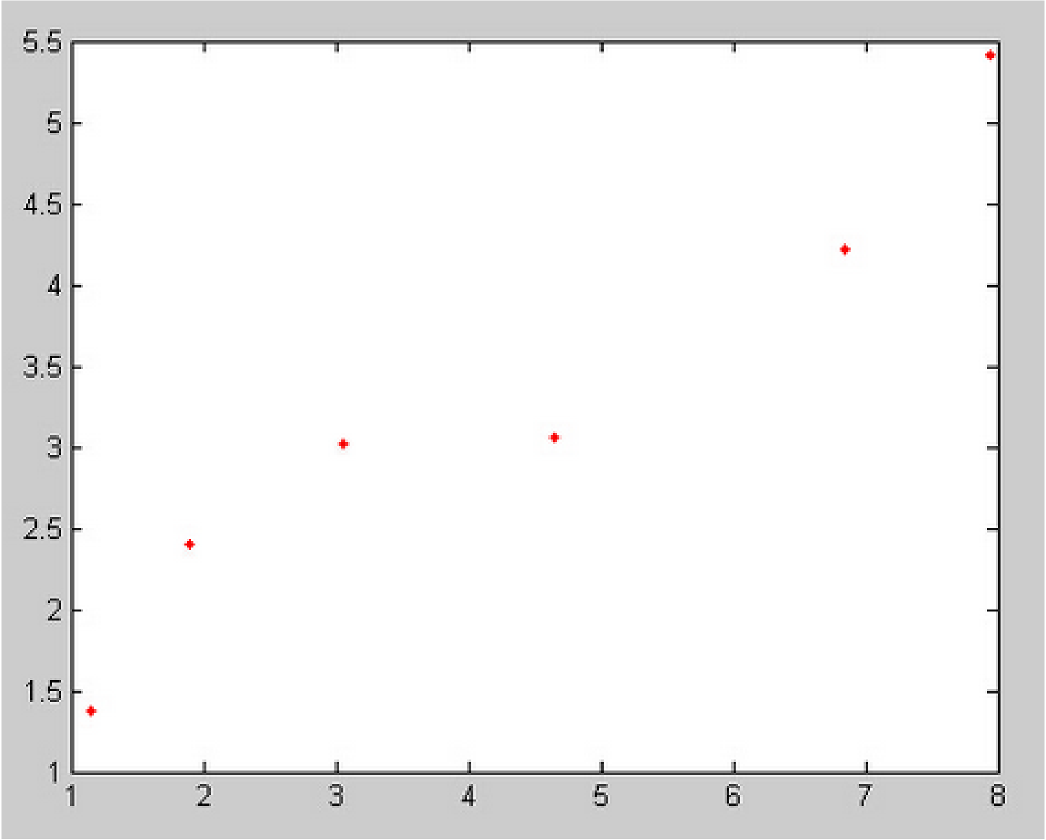
- 批量梯度下降BGD（Batch Gradient Descent ）
- 随机梯度下降SGD（Stochastic Gradient Descent ）
- 小批量梯度下降法MBGD (Mini-Batch Gradient Descent)

下文将以线性回归算法为例来对三种梯度下降法进行比较

2. 先导知识

- 一元线性回归(拟合曲线)
- 假设这里存在m=6组数据(x, y)

y	x
1.37	1.15
2.4	1.9
3.02	3.06
3.06	4.66
4.22	6.84
5.42	7.95



- 从图上可以看出，大致数据的大致走势是可以用线性模型y=kx+b来表示的，为此我们建立一维线性回归模型。
- 假设一维线性模型表达式如下：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$= \sum_{i=0}^{n} x_i \theta_i$$

$$= x\theta$$

其中：

- $h_\theta(x)$是假设函数，即要拟合的函数
- θ为待求解参数，即要迭代求解的值， θ求解出来了那最终要拟合的函数$h_\theta(x)$就确定了。
- n表示输入特征数，为方便计算，所有的样本都加入了$x_0=1$这个特征，所以维数为n+1维。

$$x = \begin{bmatrix} x_0 & x_1 & \cdots & x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad x_0 = 1$$

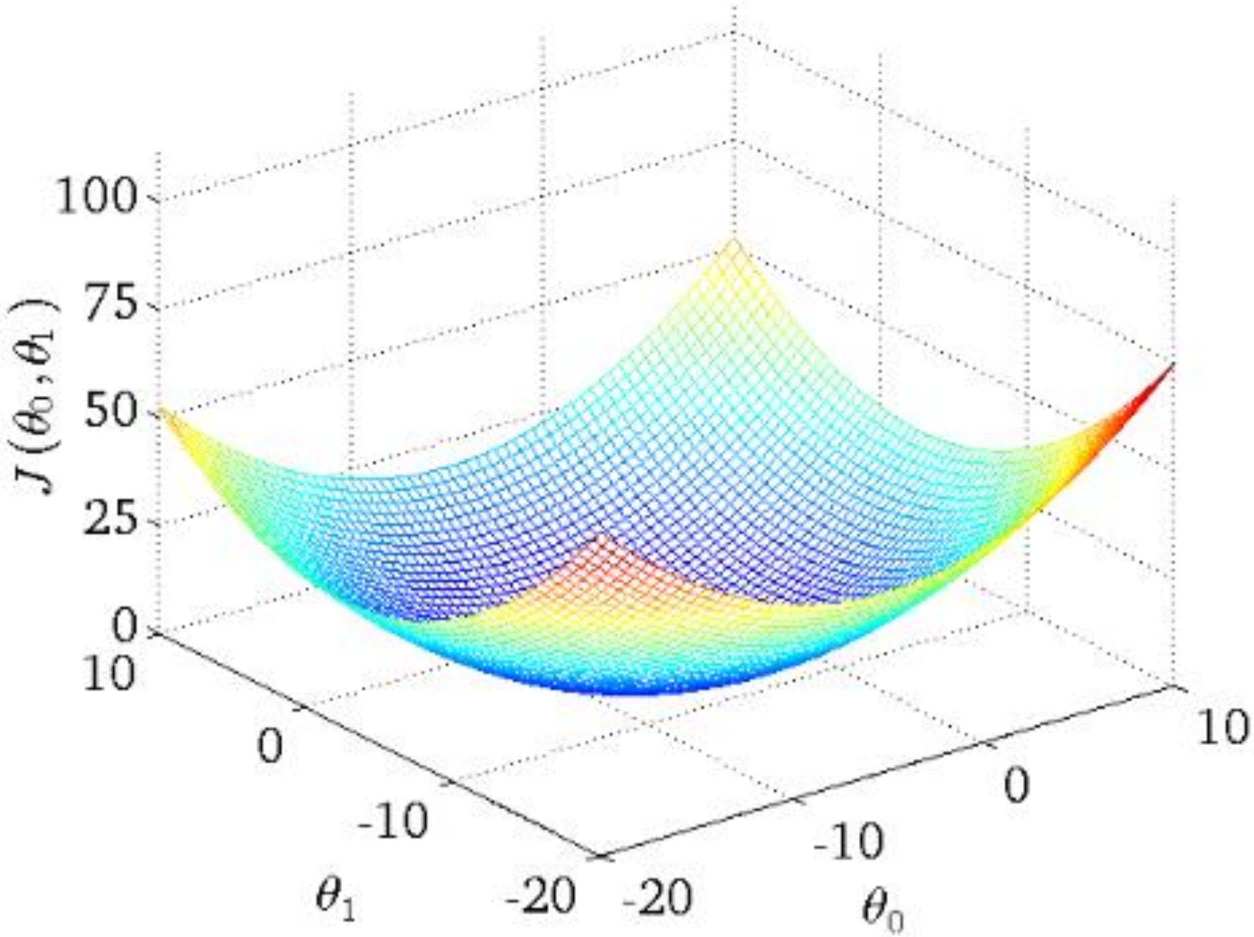
- 对应的损失/误差函数，即估计值与真实值之间的差距，这里用2-范数表示为：

$$J_{train}(\theta) = 1/(2m)\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

其中：

- m是训练集的样本个数
- 1/2是为了后面求导计算方便

- 一个二维参数（θ_0，θ_1）组对应能量函数（描述整个系统的优化程度，随着网络的变化而减小，最终网络稳定时能量达到最小）的可视化图



3. 批量梯度下降法BGD

- 更新算法的目的：误差函数尽可能小，即求解参数使误差函数尽可能小。
- 主要思想：
  - 首先，随机初始化参数；
  - 然后，不断反复的更新参数使得误差函数减小，直到满足要求时停止。

- 梯度下降算法，利用初始化的参数θ并且反复更新参数θ：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

- α代表学习率，表示每次向着函数J最陡峭的方向迈步的大小（步长）

（1）将J(θ)对 θ 求偏导，得到每个 θ 对应的的梯度

- 当m=1时，即只有一个样本数据（x, y），J对第j个参数 θ_j 的偏导数是：

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x)-y\right)^2 \\
&= 2\cdot\frac{1}{2}\left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x)-y\right) \\
&= \left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= \left(h_\theta(x)-y\right)x_j
\end{aligned}$$

- 对所有m个样本数据，上述损失函数的偏导（累和）为：

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

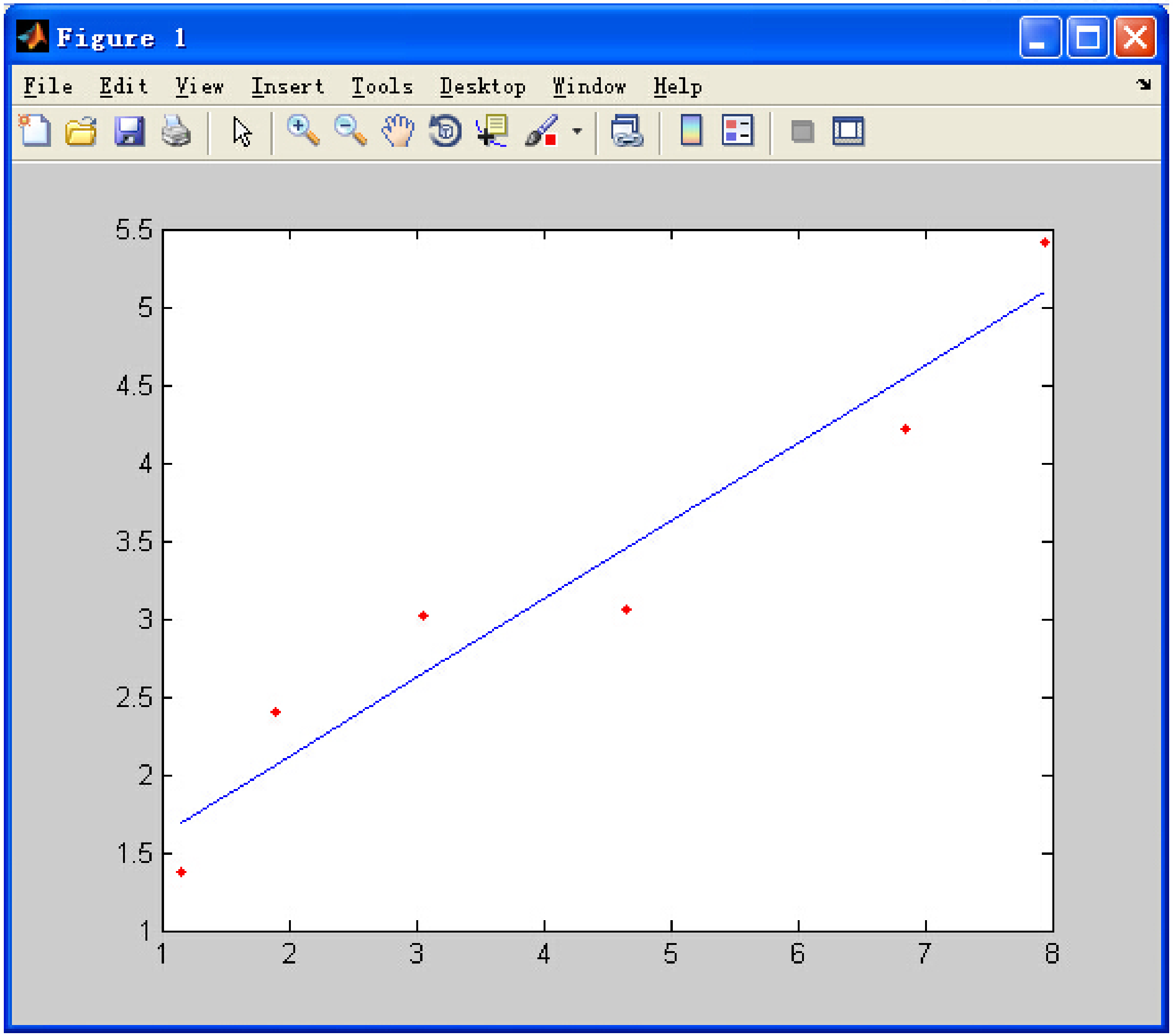
（2）由于是要最小化风险函数，所以按每个参数θ的梯度负方向，来更新每个θ $_j$(j=0，1，2，…，n)

$$\theta_j^{'} = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

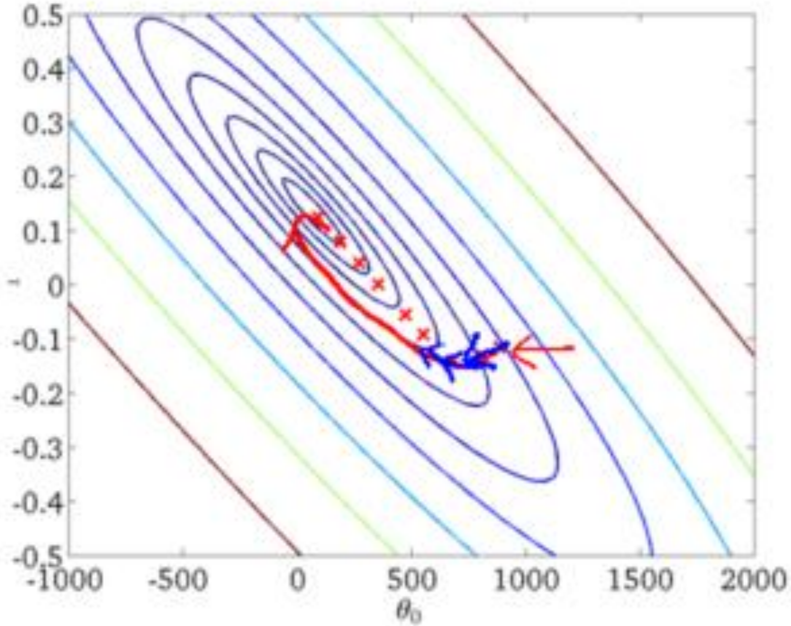
- 上例中，利用BGD求得

theta0=1.111094;

theta1=0.502401;



- 由更新公式可知，批量梯度下降得到的是一个全局最优解，每一次的参数更新都用到了所有的训练数据，如果训练数据非常多的话，执行效率较低。
- 批量梯度下降法的收敛图（迭代的次数相对较少）：



4. 随机梯度下降法SGD

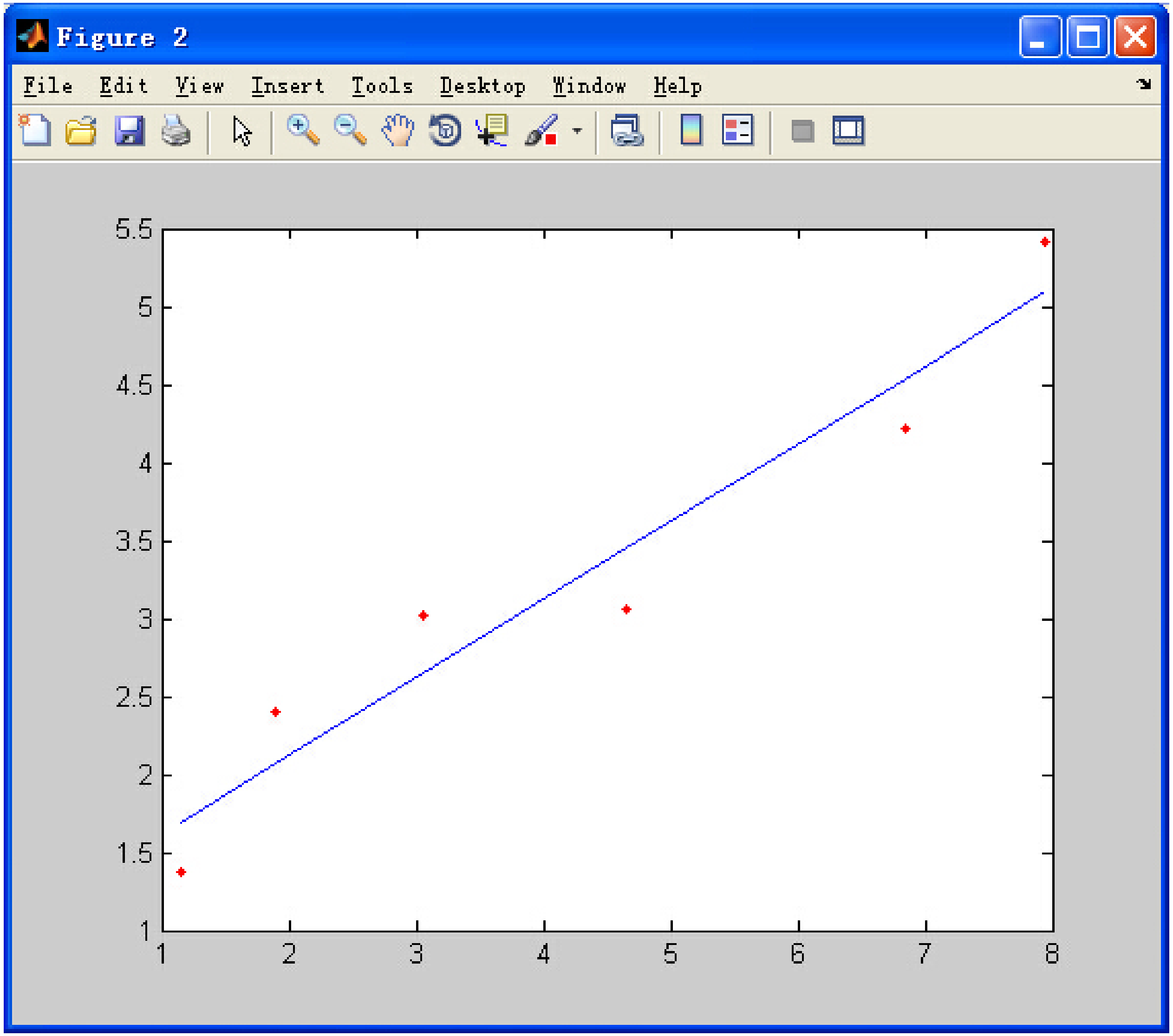
- 由于批梯度下降每更新一个参数的时候，要用到所有样本，所以训练速度会随着样本数量的增加而变得非常缓慢。
- 随机梯度下降正是为了解决这个办法而提出的。它是利用单个样本的损失函数对θ求偏导得到对应的梯度，来更新θ。

$$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i))x_j^i$$

- 上例中，利用SGD求得

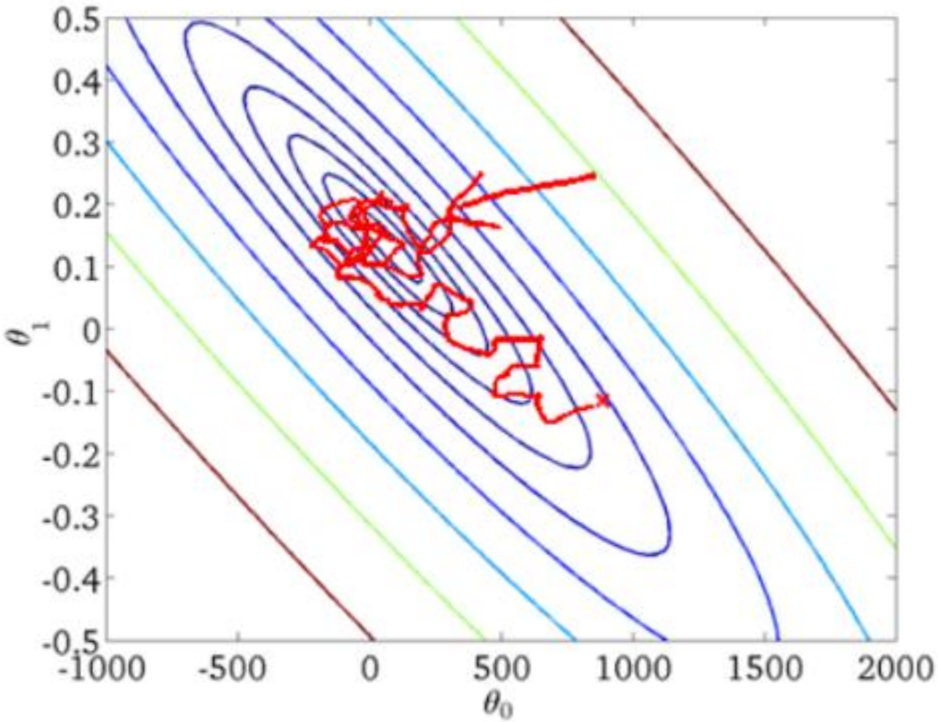
theta0=1.117690;

theta1=0.500151;



- 随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将参数迭代到最优解。
- 对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代10次的话就需要遍历训练样本10次。
- SGD的问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。

- 随机梯度下降收敛图（**SGD迭代的次数较多**，在解空间的搜索过程看起来很盲目。但是大体上是往着最优值方向移动。）



5. 小批量梯度下降法MBGD

- 为综合解决BGD的训练速度慢，以及SGD的准确性低的问题，提出MBGD
- 它是利用部分样本的损失函数对θ求偏导得到对应的梯度，来更新θ。

Repeat{

for i=1, 11, 21, 31, ... , 991{

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for every j=0, ... , n)

}

}

6. 总结

方法	优点	缺点
BGD	最小化所有训练样本的损失函数，使得最终求解的是全局的最优解	如果样本值很大的话，更新速度会很慢。
SGD	最小化每个样本的损失函数，大大加快更新速度，最终的结果在全局最优解附近。	训练数据的噪声较多，导致不是每次迭代得到的损失函数都向着全局最优方向。
MBGD	训练速度快，参数准确性高	不同的问题需要设置不同的小批量值。