2.2.1 $h_\theta(x_i) = \theta^T x_i$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i)-y_i)^2+\lambda\theta^T\theta = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x_i-y_i)^2+\lambda\theta^T\theta = \frac{1}{m}(X\theta-y)^T(X\theta-y)+\lambda\theta^T\theta = \frac{1}{m}\|X\theta-y\|_2^2+\lambda\|\theta\|_2^2$$

2.2.3 $\nabla_\theta J(\theta) = \frac{2}{m}X^T(X\theta-y)+2\lambda\theta$

2.3.1 一阶泰勒展开

$$J(\theta+\eta h) = J(\theta)+(\eta h)^T\nabla_\theta J(\theta)+\cdots$$

$h^T\nabla_\theta J(\theta)$ 只有 h 与 $\nabla_\theta J(\theta)$ 方向相同时取得最大值

2.3.2 $\theta^{t+1} = \theta^t - \eta\nabla_\theta J(\theta)|_{\theta=\theta^t}$

2.3.4 步长0.01时收敛最快，步长为0.5，0.1时会发散

2.4.1
$$X_k = \begin{pmatrix} x_{i_1 1} & x_{i_1 2} & \cdots & x_{i_1 d+1} \\ x_{i_2 1} & x_{i_2 2} & \cdots & x_{i_2 d+1} \\ \vdots & & & \\ x_{i_n 1} & x_{i_n 2} & \cdots & x_{i_n d+1} \end{pmatrix} \qquad y_k = \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \vdots \\ y_{i_n} \end{pmatrix}$$

$$J_{SGD}(\theta) = \frac{1}{n}\sum_{k=1}^{n}(h_\theta(x_{i_k})-y_{i_k})^2+\lambda\theta^T\theta = \frac{1}{n}(X_k\theta-y_k)^T(X_k\theta-y_k)+\lambda\theta^T\theta$$

$$\nabla_\theta J_{SGD}(\theta) = \frac{2}{n}X_k^T(X_k\theta-y)+2\lambda\theta$$

2.4.3 批增大，损失函数呈收敛趋势

2.4.5 发现在所取数值中，正则化系数越大越稀疏

3.2 发现distance对房价有较大负影响，latitude对房价有较大正影响，当前将时间运用特征工程加入了模型，且分为年月日，可能会与房屋年龄产生关联，同时也会引入过多时间波动，弱化其它指标作用。可以减少时间变量来重新估计模型。

把不同变量都进行特征规一化，但变量间方差不同，可能进行标准化会更好。

(最大最小) (正态)

4.1.4 $h_\theta(x) = P(y=1|x;\theta) = \frac{1}{1+e^{-\theta^T x}}$

$$J(\theta) = [y\log(1+e^{-\theta^T x})+(1-y)\log(\frac{1+e^{-\theta^T x}}{e^{-\theta^T x}})]$$

$$\nabla_\theta J(\theta) = y\frac{-xe^{-\theta^T x}}{1+e^{-\theta^T x}}-(1-y)\frac{-xe^{-\theta^T x}}{e^{-\theta^T x}}+(1-y)\frac{-xe^{-\theta^T}}{1+e^{-\theta^T x}} = -xe^{-\theta^T}h_\theta(x)+(1-y)x = -x(e^{-\theta^T}+1)h_\theta(x)+xh_\theta(x)+(1-y)x = (h_\theta(x)-y)x$$

4.1.5. 发现手动得出的结果与调用算法包得到的结果一样

4.2.2 选取了花萼长度与花瓣宽度

4.2.3 引入基函数 决策面变为曲线且拟合效果变好

分布式计算

Stochastic Gradient Descent (SGD)

大数据

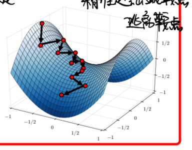
- In each iteration t ($\le T$): 随机采样 $d \times m \times T$
  - Randomly sample a minibatch of $m \ll n$ points $\{(x_i, y_i)\}_{i=1}^{m}$
  - Set $J^t(w^t) = \frac{1}{m}\sum_{i=1}^{m} \ell(w^t; x_i, y_i)$ 由设备决定内存
  - Compute gradient on minibatch:

$$\Delta^t = \nabla_w J^t(w^t)$$

  - Update parameters with learning rate η:

$$w^{t+1} = w^t - \eta \Delta^t$$

精准定会出现鞍点 逃离鞍点



- For linear regression: $w^{t+1} \leftarrow w^t - 2\eta X_m^T (X_m w^t - y)$

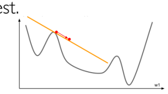
沿梯度下降

- Gradient shows direction that function varies fastest.

Same dimension with parameter w's dimension

$$g = \nabla_w J(w)$$ 梯度向量

$$g_j = \nabla_{w_j} J(w)$$



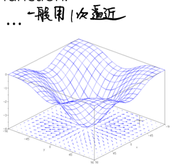
- First-order Taylor approximation of the objective function:

$$J(w) = J(w_0) + (w - w_0)^T g + \cdots$$ 一般用1次逼近

- Go along gradient for a step with a small rate η:

$$J(w - \eta g) \approx J(w) - \eta g^T g$$

  - Reach a point with smaller loss. $\boxed{-\eta g^T g \le 0}$



- Repeat this step, and we get the Gradient Descent (GD) algorithm.

- Compute the gradient and set it to zero

$$\nabla_w \hat{\epsilon}(w) = 2X^T(Xw - y) + 2\lambda w = 0$$

$$\Longrightarrow (X^T X + \lambda I) w = X^T y$$

$$\Longrightarrow w = (X^T X + \lambda I)^{-1} X^T y$$

有唯一解

X $n \times d$ ⇒ $d > n$ 不满秩

- Or using Gradient Descent (GD): Always invertible for $\lambda > 0$ $d \times d$-dim identity matrix
  - $w^{t+1} \leftarrow w^t - \eta \nabla_w \hat{\epsilon}(w)|_{w = w^t}$