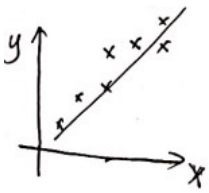
Linear Regression (线性回归)

1. Linear Regression



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

$$x_i \in \mathbb{R}^p, \quad y_i \in \mathbb{R} \quad i = 1, 2, \cdots, N$$

$$X = (x_1, x_2, \cdots, x_N)^T = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix}_{N\times p} \qquad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}_{N\times 1}$$

$$f(w) = w^T x \qquad w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix}$$

D: data(N samples)

(1) Algorithm: Least squares estimation (最小二乘估计)

Loss function： $L(w) = \sum_{i=1}^{N} ||W^T x_i - y_i||^2$

$$\begin{aligned}
L(w) &= \sum_{i=1}^{N} ||w^T x_i - y_i||^2 \\
&= \sum_{i=1}^{N} (w^T x_i - y_i)(w^T x_i - y_i) \\
&= \underbrace{(w^T x_1 - y_1, \cdots, w^T x_N - y_N)} \begin{pmatrix} w^T x_1 - y_1 \\ \vdots \\ w^T x_N - y_N \end{pmatrix} \\
&= (w^T x_1, \cdots, w^T x_N) - (y_1, \cdots, y_N)\ (XW - Y) \\
&= W^T (x_1, \cdots, x_N) - (y_1, \cdots, y_N) \cdot (XW - Y) \\
&= (w^T X^T - Y^T) \cdot (XW - Y) \\
&= W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y = W^T X^T X W - 2 W^T X^T Y + Y^T Y
\end{aligned}$$

$\hat{w} = argmin\, L(w)$

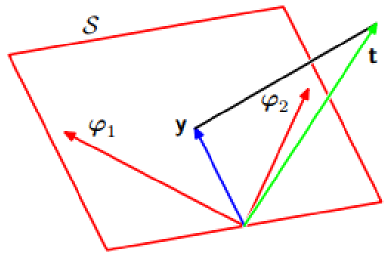
$$\frac{\partial L(w)}{\partial w} = 2X^T X W - 2X^T Y = 0$$

$\hat{w} = (X^T X)^{-1} X^T Y$ (X has to be a non-singular matrix, if X is a singular matrix, use SVD/QR decomposition)

(2) Geometric interpretation of LSE (source: PRML) (几何角度解释最小二乘估计)

The idea is to separate errors in every dimensions, thus in an N-dimensional space whose axes are the values of $t1, \ldots, tN$. The least-squares regression function is obtained by finding the orthogonal projection of the data vector t onto the subspace spanned by the basis functions $\varphi j(\mathbf{x})$ in which each basis function is viewed as a vector ϕj of length N with elements $\varphi j(\mathbf{x}n)$.

把误差分散到每个维度



(3) solving Loss function to obtain the minimum of loss function （求解损失函数的方法）

(Source: http://cs229.stanford.edu/notes/cs229-notes1.pdf)

① batch gradient descent: looks at every example in the entire training set on every step （批量梯度下降法）

$w_j := w_j - \alpha \frac{\partial L(w)}{\partial w_j}$ where α is the learning rate

Starts with some "initial guess" for w, and that repeatedly changes w to make L(w) smaller, until hopefully we converge to a value of w that minimizes L(w)

② Stochastic gradient descent: update the parameters according to the gradient of the error with respect to that single training example only. （随机梯度下降法）

```
Repeat {
    For I=1 to N,{
            w_j := w_j − α ∂L(w)/∂w_j     (for every j)
    }
}
```

2. Lasso/Ridge Regression （L1/L2 正则化）

In real life, X~ N*p matrix, if the N <<p, this will probably cause over fitting problem.

So there exist three solving method:

- Add more data (增加数据)
- Dimensionality reduction （降维）
- Regularization (正则化)

Regularization function's structure: argmin(L(w) + λ P(w))

L1 : P(w) = $||\mathbf{w}||_1$

L2: P (w) = $||\mathbf{w}||_2^2 = \boldsymbol{w^T w}$

L(w): loss function $L(w) = \sum_{i=1}^{N} ||W^T x_i - y_i||^2$

For L2 (from frequency prospective):

$$\text{Goal function: } J(w) = \sum_{i=1}^{N} ||w^T x_i - y_i||^2 + \lambda w^T w$$

$$= (w^T x_1 - y_1, \cdots, w^T x_N - y_N) \begin{pmatrix} w^T x_1 - y_1 \\ \vdots \\ w^T x_N - y_N \end{pmatrix} + \lambda w^T w$$

$$= (w^T X^T - Y)(Xw - Y) + \lambda w^T w$$

$$= w^T X^T X w - 2 w^T X^T Y + Y^T Y + \lambda w^T w$$

$$= w^T (X^T X + \lambda I) w - 2 w^T X^T Y + Y^T Y$$

$$\hat{w} = \text{argmin } J(w)$$

$$\frac{dJ(w)}{dw} = 2(X^T X + \lambda I) w - 2 X^T Y = 0$$

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

For L2 (from Bayesian perspective):

$$w \sim N(0, \sigma_0^2)$$

$$P(w|y) = \frac{P(y|w) \cdot P(w)}{P(y)} \qquad y = f(x) + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$$

$$\hat{w} = \underset{w}{\text{argmax}}\, P(w|y)$$

$$= \underset{w}{\text{argmax}}\, P(y|w) \cdot P(w) \qquad \text{where } P(y|w) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{(y - w^T x)^2}{-2\sigma^2}\right\}$$

$$P(w) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{\frac{||w||^2}{-2\sigma_0^2}\right\}$$

$$= \underset{w}{\text{argmax}}\, \frac{1}{\sqrt{2\pi}\sigma} \cdot \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\frac{(y - w^T x)^2}{2\sigma^2} - \frac{||w||^2}{2\sigma_0^2}\right\}$$

$$= \arg\max_{w} \log\left[P(y|w)\,P(w)\right]$$

$$= \arg\max_{w} \log\left(\frac{1}{\sqrt{2\pi}\sigma}\cdot\frac{1}{\sqrt{2\pi}\sigma_0}\right) + \log\exp\left\{-\frac{(y-w^Tx)^2}{2\sigma^2} - \frac{\|w\|^2}{2\sigma_0^2}\right\}$$

$$= \arg\min_{w} \left(\frac{(y-w^Tx)^2}{2\sigma^2} + \frac{\|w\|^2}{2\sigma_0^2}\right)$$

$$= \arg\min_{w} \left(\|y-w^Tx\|^2 + \frac{\sigma^2}{\sigma_0^2}\|w\|_2^2\right)$$

$$\boxed{\hat{w}_{MAP} = \arg\min \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\sigma^2}{\sigma_0^2}\|w\|_2^2}$$

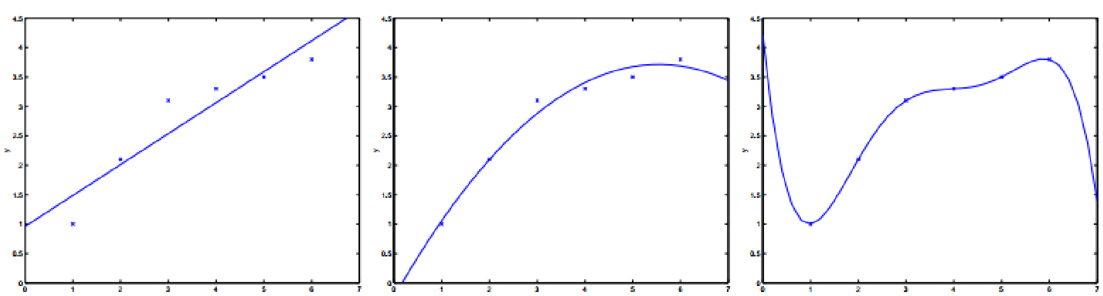
$$J(w) = \sum_{i=1}^{N} \|w^T x_i - y_i\|^2 + \lambda w^T w$$

Rugularized LSE $\Leftrightarrow$ MAP (noise is Gaussian distribution
prior is Gaussian too)

3. Locally Weighted Linear Regression

(Source: http://cs229.stanford.edu/notes/cs229-notes1.pdf)

This method is used to solving under fitting problem and it is also a non-parametric method.



Steps:

1. *Fit w to minimize* $L(w) = \sum_{i=1}^{N} \theta ||W^T x_i - y_i||^2$
2. output $W^T X$.