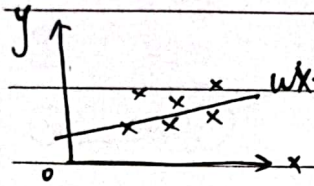
Date

线性回归部分.

1. 一元线性回归.



cost function: $J(w,b)=\sum_{i=1}^{m}(wx_i+b-y_i)^2$

其中 x_i 为输入, y_i 是 x_i 对应的 label

为更好地拟合, 需让 $J(w,b)$ 最小, 有

两种方法:

(1) Gradient Descent

重复执行{

$tmp := \alpha\frac{\partial J(w,b)}{\partial w}$ (:= 表示赋值操作)

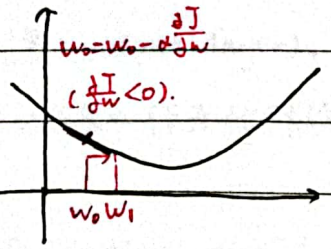
$tmp2 := \alpha\frac{\partial J(w,b)}{\partial b}$ ($\alpha>0$).

$w := w - tmp$

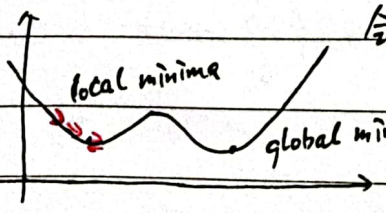
$b := b - tmp2$

}

(α 不能太大, 太大会出现不收敛于minima或发散的情况; α 太小会使学习速度过慢)



Gradient Descent 会出现极小值的问题



(2) 最小二乘法:

求 $\begin{cases}\frac{\partial J}{\partial w}=0\\ \frac{\partial J}{\partial b}=0\end{cases}$ 对应的 $w=w_0$, $b=b_0$.

即得 (w_0,b_0) 为 $J(w,b)$ 极小值点.

其中 $\frac{\partial J}{\partial w}=2\sum_{i=1}^{m}(wx_i+b-y_i)x_i=0$

$\frac{\partial J}{\partial b}=2\sum_{i=1}^{m}(wx_i+b-y_i)=0$

2. 多元线性回归.

$y=w^Tx+b=[w_1,w_2,\cdots,w_n]\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}+b$

$=\sum_{i=1}^{n}w_ix_i+b$.

需要使 $J(b)=\sum_{i=1}^{m}\left(\sum_{j=1}^{n}w_jx_j+b-y_i\right)^2$ 最小.

(1) Gradient Descent

类似地:

重复执行{

$tmp := \alpha\frac{\partial J(w_1,\cdots,w_n,b)}{\partial w_1}$

$tmp2 := \alpha\frac{\partial J(w_1,w_2,\cdots,w_n,b)}{\partial w_2}$

$\vdots$

$w_1 := w_1 - tmp$

$w_2 := w_2 - tmp2$

$\vdots$

}

(2) 最小二乘法:

$\begin{cases}\frac{\partial J}{\partial w_j}=0\ (j=1\sim n) & \frac{\partial J}{\partial w_j}=2\sum_{i=1}^{m}\left(\sum_{j=1}^{n}(w_jx_{ij}+b-y_i)x_{ij}\right)\\ \frac{\partial J}{\partial b}=0 & \frac{\partial J}{\partial b}=2\sum_{i=1}^{m}\left(\sum_{j=1}^{n}(w_jx_{ij}+b-y_i)\right)\end{cases}$

若用矩阵形式, 则有 自变量.

$D=\{(x^{(1)},y^{(1)}),\cdots,(x^{(m)},y^{(m)})\}$. $x^{(i)}$ 为向量.

$x^{(i)}=\begin{bmatrix}x_1^{(i)}\\x_2^{(i)}\\\vdots\\x_n^{(i)}\end{bmatrix}$ $y^{(i)}$ 为 label

and $y=\begin{bmatrix}y^{(1)}\\\vdots\\y^{(m)}\end{bmatrix}$. $w=\begin{bmatrix}w_1\\\vdots\\w_n\end{bmatrix}$

$X=\begin{bmatrix}-x^{(1)T}-\\-x^{(2)T}-\\\vdots\\-x^{(m)T}-\end{bmatrix}$

那么 $\left(y-Xw-b\begin{bmatrix}1\\\vdots\\1\end{bmatrix}\right)^T\left(y-Xw-b\begin{bmatrix}1\\\vdots\\1\end{bmatrix}\right)$ 为 $J(w_1,\cdots,w_n,b)$ 的矩阵表达形式

$$\frac{\partial J}{\partial w_i} = 2X^T(Xw + b\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} - y)$$

$$\frac{\partial J}{\partial b} = 2(Xw + b\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} - y)$$

或将 w, b 参数合并. $\hat{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \\ b \end{bmatrix}$

而 $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_m^{(i)} \\ 1 \end{bmatrix}$ $\quad X = \begin{bmatrix} - & x^{(1)} & - \\ - & x^{(2)} & - \\ & \vdots & \\ - & x^{(m)} & - \end{bmatrix}$

则 $J(w_1, \cdots, w_m, b) = (y - X\hat{w})^T(y - X\hat{w})$

$\frac{\partial J}{\partial \hat{w}} = 2X^T(X\hat{w} - y) = 0.$ （$\hat{w}$ 为列向量）

则 $X^TX\hat{w} = X^Ty$

(if $|X^TX| \neq 0$)

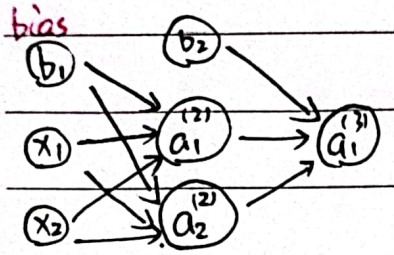
$\hat{w} = (X^TX)^{-1}X^Ty$ 参数的“最优解”

(else)

有解出多个 $\hat{w}$ 的情况

3. 逐步回归（先放着，暂无进展）

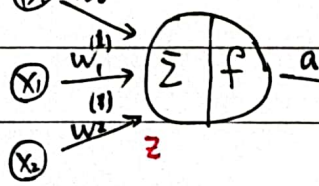
BP 神经网络.



第一层为输入层(input layer),

以输入的各属性与一个b值,乘上参数加和.

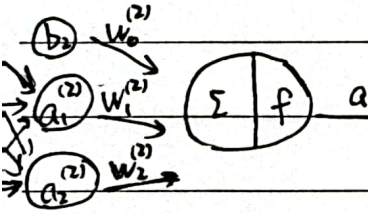
$f(z)=\frac{1}{1+e^{-z}}$ $z=\sum wx+b w_0$.



activation

然后套上 ~~activate~~ function, 计算出下一层的输出的值.

隐藏层(hidden layer). 类似上面计算方式.



不过 b_2 是 bias, 不是由前一层计算出来的

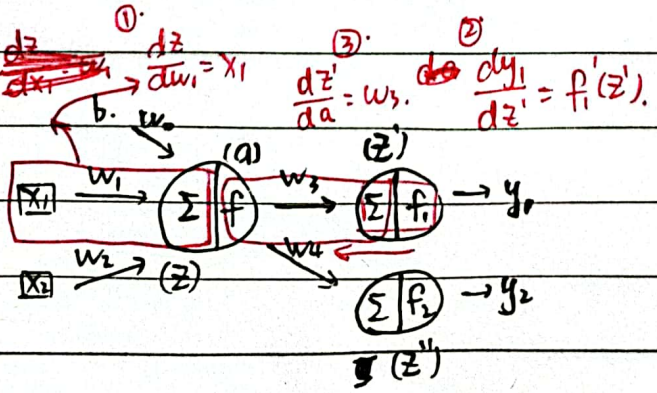
输出层(output layer). 类似上面计算方式.

输出的结果a作为估计值

BackPropagation Algorithm

用Gradient Descent来确定合适的w参数, b参数.

$\frac{\partial J}{\partial w}$ / $\frac{\partial J}{\partial b}$.



eg: $\frac{\partial J(w)}{\partial w_1}$

$$\frac{\partial J}{\partial w_1}=\frac{\partial J}{\partial z}\cdot\frac{\partial z}{\partial w_1}$$

① Forward pass: $z=w_1x_1+\cdots$

$$\Rightarrow \frac{\partial z}{\partial w_1}=x_1$$

② Backward pass:

$$\frac{\partial J}{\partial z}=\frac{\partial J}{\partial a}\cdot\frac{\partial a}{\partial z}=\frac{\partial J}{\partial a}\cdot f'(z)$$

$$\frac{\partial J}{\partial a}=\frac{\partial J}{\partial y_1}\cdot\frac{\partial y_1}{\partial a}+\frac{\partial J}{\partial y_2}\cdot\frac{\partial y_2}{\partial a}$$

$$=\frac{\partial J}{\partial y_1}\cdot\frac{\partial y_1}{\partial z'}\cdot\frac{\partial z'}{\partial a}+\frac{\partial J}{\partial y_2}\cdot\frac{\partial y_2}{\partial z''}\cdot\frac{\partial z''}{\partial a}$$

$$=\frac{\partial J}{\partial y_1}\cdot f_1'(z')\cdot w_3+\frac{\partial J}{\partial y_2}\cdot f_2'(z'')\cdot w_4$$

$J=\sum(\hat{y}-y)\Rightarrow\frac{\partial J}{\partial y_i}$ 也可算出.

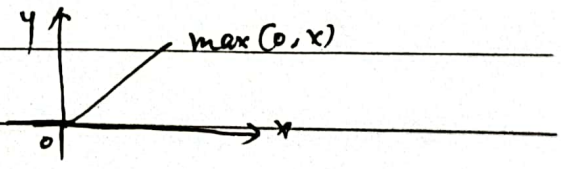
$$\therefore \frac{\partial J}{\partial w_1}=\left[\frac{\partial J}{\partial y_1}\cdot f_1'(z')\cdot w_3+\frac{\partial J}{\partial y_2}\cdot f_2'(z'')\cdot w_4\right]x_1$$

推导出 $\frac{\partial J}{\partial w_1}$ 的偏微分相乘表达式. 所以从近输入端至近输出端推导. 但是具体计算这些偏微分表达式时, 需要 Forward/Backward pass 来算出具体量.

当然, BP算法用软件即可.

关于activation function:

根据B站教程的建议, 用ReLU比较好



正则化缓解过拟合.(梯度下降法)

过拟合一般是w.b过大时产生的.会让曲线扭曲.

$$J(w,b)=\frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j\right]$$

当λ过大.则会使θj过小.造成欠拟合(underfitting)

反之.θj过大.过拟合(overfitting).