- Evaluation function: Maximize the conditional log-likelihood

l indexes the examples

$$W \leftarrow \arg\max_W \prod_l P(Y^l | X^l, W)$$

$$W = \langle w_0, w_1 \ldots w_n \rangle$$ Weight vector

- **Note that actually we are just computing P(Y|X)**
- **W is included in P(Y|X) just to show that the probability is computed using W**

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y^l = 1 | X^l, W) + (1 - Y^l) \ln P(Y^l = 0 | X^l, W) \\ &= \sum_l Y^l \ln \frac{P(Y^l = 1 | X^l, W)}{P(Y^l = 0 | X^l, W)} + \ln P(Y^l = 0 | X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

$$W \leftarrow \arg\max_{W} \sum_{l} \ln P(Y^l|X^l, W) - \frac{\lambda}{2}||W||^2$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_{l} X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \lambda w_i$$

- Weight update rule:

$$w_i \leftarrow w_i + \eta \sum_{l} X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \eta \lambda w_i$$

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

implies

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Classification Rule: Assign the label Y=0 if

$$1 < \frac{P(Y = 0|X)}{P(Y = 1|X)}$$

Take logs and simplify:

$$0 < w_0 + \sum_{i=1}^{n} w_i X_i$$

linear classification rule!

Y=0 if the RHS>0

The key problem is that you need to preprocess the data. After you get the data, you want, the problem can be much easier.

$$\theta := \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(\mathrm{x}^{(\mathrm{i})}) - y^{(i)} \right) x^{(\mathrm{i})}, \quad (j = 0 \ldots n)$$

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \ldots \\ x^{(\mathrm{m})} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \ldots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \ldots & x_n^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ x_0^{(\mathrm{m})} & x_1^{(\mathrm{m})} & \ldots & x_n^{(\mathrm{m})} \end{bmatrix}, \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \ldots \\ y^{(\mathrm{m})} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \ldots \\ \theta_n \end{bmatrix}$$

$$A = x \bullet \theta = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \ldots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \ldots & x_n^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ x_0^{(\mathrm{m})} & x_1^{(\mathrm{m})} & \ldots & x_n^{(\mathrm{m})} \end{bmatrix} \bullet \begin{bmatrix} \theta_0 \\ \theta_1 \\ \ldots \\ \theta_n \end{bmatrix} = \begin{bmatrix} \theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \ldots + \theta_n x_n^{(1)} \\ \theta_0 x_0^{(2)} + \theta_1 x_1^{(2)} + \ldots + \theta_n x_n^{(2)} \\ \ldots \\ \theta_0 x_0^{(\mathrm{m})} + \theta_1 x_1^{(\mathrm{m})} + \ldots + \theta_n x_n^{(\mathrm{m})} \end{bmatrix}$$

$$E = h_\theta(\mathrm{x}) - y = \begin{bmatrix} g\left(A^{(1)}\right) - y^{(1)} \\ g\left(A^{(2)}\right) - y^{(2)} \\ \\ g\left(A^{(\mathrm{m})}\right) - y^{(\mathrm{m})} \end{bmatrix} = \begin{bmatrix} e^{(1)} \\ e^{(2)} \\ \cdots \\ e^{(\mathrm{m})} \end{bmatrix} = g\left(A\right) - y$$

$$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{m} \left(h_\theta(\mathrm{x}^{(\mathrm{i})}) - y^{(i)} \right) x_0^{(\mathrm{i})}$$

$$= \theta_0 - \alpha \sum_{i=1}^{m} e^{(\mathrm{i})} x_0^{(\mathrm{i})}$$

$$= \theta_0 - \alpha \bullet \left(x_0^{(1)}, x_0^{(2)}, \dots, x_0^{(\mathrm{m})} \right) \bullet E$$

$$\theta_j := \theta_j - \alpha \bullet \left(x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(\mathrm{m})} \right) \bullet E$$

$$
\begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix} := \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix} - \alpha \bullet \begin{bmatrix} x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(m)} \\ x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(m)} \\ \\ x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(m)} \end{bmatrix} \bullet E
$$

$$
= \theta - \alpha \bullet x^T \bullet E
$$

In all

$$
\theta := \theta - \alpha \bullet \left(\frac{1}{m} \right) \bullet x^T \bullet \left(g\left(x \bullet \theta \right) - y \right)
$$

```
def sigmoid(inX):
    return 1.0/(1+exp(-inX))

def gradAscent(dataMatIn, classLabels):
    dataMatrix = mat(dataMatIn)               #convert to NumPy matrix
    labelMat = mat(classLabels).transpose() #convert to NumPy matrix
    m,n = shape(dataMatrix)
    alpha = 0.001
    maxCycles = 500
    weights = ones((n,1))
    for k in range(maxCycles):                #heavy on matrix operations
        h = sigmoid(dataMatrix*weights)      #matrix mult
        error = (labelMat - h)               #vector subtraction
        weights = weights + alpha * dataMatrix.transpose()* error #matrix mult
    return weights
```