Logistic regression as a "neural network"

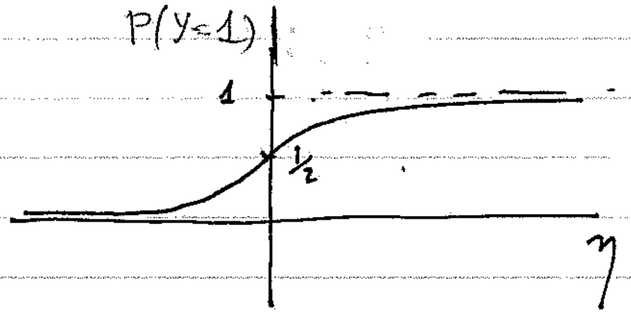
Reminder logreg: $\left(x_1^{(i)}, \dots x_p^{(i)}\right) \quad y^{(i)} \in \{0,1\} \quad i = 1, \dots, N$

Assume observable y is Bernoulli: $P(Y=y \mid \vec{x}) = P(Y=1)^y \left(1 - P(Y=1)\right)^{1-y}$
$= \mathrm{Ber}(y \mid P(Y=1))$

If we express probability in terms of 'log-odds':

$$\eta = \log \frac{P(Y=1)}{P(Y=0)} = \log \frac{P(Y=1)}{1-P(Y=1)} \Rightarrow P(Y=1) = \frac{1}{1+e^{-\eta}}$$

$$\mathrm{sigm}(\eta) = h(\eta)$$



Model: log odds is linear function of descriptors:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_p^{(1)} \\ \vdots & & & \\ 1 & x_1^{(N)} & \dots & x_p^{(N)} \end{bmatrix} \qquad \eta = \vec{x}^T \cdot \vec{\beta} \qquad \vec{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix} \qquad \vec{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

Infer $\vec{\beta}$ from data by maximising likelihood

$$P(\vec{y} \mid X, \vec{\beta}) = \prod_{i=1}^{N} \mathrm{Ber}\left(y^{(i)} \mid h(\vec{x}^{(i)T} \cdot \vec{\beta})\right) \qquad \boxed{\mathrm{Ber}(\ ) \equiv \text{Bernoulli}}$$

$$\boxed{L = \sum_{i=1}^{N} \left[y^{(i)} \log\left[h(\vec{x}^{(i)T} \cdot \vec{\beta})\right] + (1-y^{(i)}) \log\left(1 - h(\vec{x}^{(i)T} \cdot \vec{\beta})\right)\right]}$$

<u>Optimisation</u>: $\nabla_{\vec{\beta}} L \Big|_{\vec{\beta}^*_{\log}} = 0$ Normal equations $\boxed{X^T\left[\vec{y} - \vec{h}(X\vec{\beta}^*_{\log})\right] = \vec{0}}$

$$h_i(X\vec{\beta}^*_{\log}) = h\left(\vec{x}^{(i)T} \cdot \vec{\beta}^*_{\log}\right)$$

and the problem is <u>convex</u> with Hessian:

This is a concave function that can be maximised globally

$$H = \nabla_{\vec\beta}\left[\nabla_{\vec\beta} L\right] = -X^T\left[\mathrm{diag}(\vec h)\cdot\left[I - \mathrm{diag}(\vec h)\right]\right] X$$

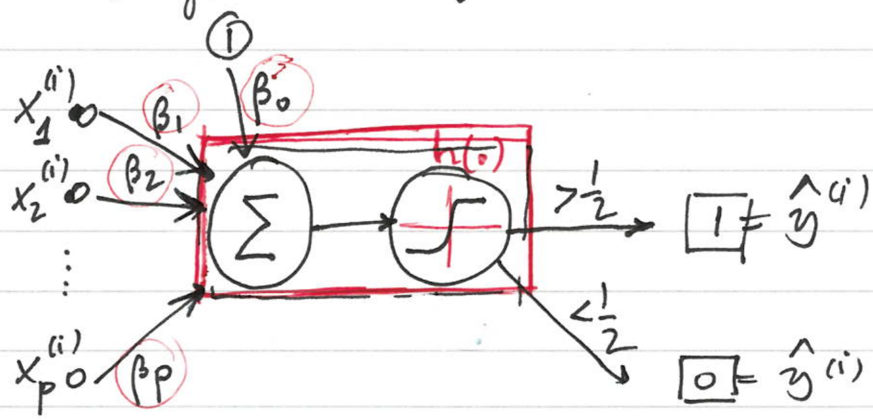
optimise using Newton or gradient methods, etc. to obtain $\vec\beta^*_{log}$

Classifier:

Given $\vec x^{in}$, compute

$$P(y \mid \vec x^{in}) = \frac{1}{1 + e^{-\vec x^{in} \cdot \vec\beta^*_{log}}} \begin{cases} > \frac{1}{2} & \hat y = 1 \\ < \frac{1}{2} & \hat y = 0 \end{cases}$$

Diagrammatically:



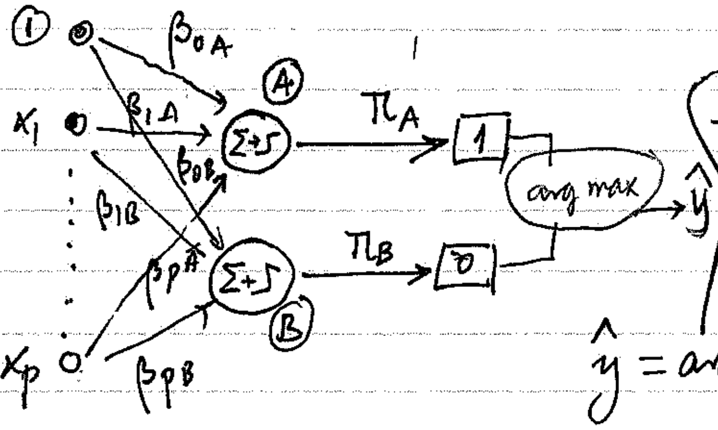
where $\vec\beta$ have to be optimised:

$$\max_{\vec\beta} L = \sum_{i=1}^{N}\left[y^{(i)} \log\left[h(\vec x^{(i)T} \cdot \vec\beta)\right] + (1 - y^{(i)}) \log\left(1 - h(\vec x^{(i)T} \cdot \vec\beta)\right)\right]$$

log likelihood is equal to minus cross-entropy between $\{y^{(i)}\}$ and $\{h(\vec x^{(i)T} \cdot \vec\beta)\}$

max of L is minimizing cross-entropy!

More generally:



$$\pi_A + \pi_B = 1$$

$$\begin{pmatrix} \pi_A \\ \pi_B \end{pmatrix} = \vec{\pi} = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_J \end{pmatrix}$$

$$\hat{y} = \underset{J}{\arg\max}\ \vec{\pi}$$

classes

In general for J classes

Infer $\underset{\approx}{\beta} = \begin{bmatrix} \vec{\beta}_A & \vec{\beta}_B \end{bmatrix}_{p \times J}$

J = 2 in this case

J is number of classes.

In this case $\vec{\beta}_A$ and $\vec{\beta}_B$ are related:

Original formulation

$$P(y=1) = \frac{e^{\vec{x}^T \cdot \vec{\beta}_A}}{e^{+x^T \cdot \beta_A} + 1}$$

$$1 - P(Y=1) = \frac{1}{e^{+x^T \beta_A} + 1}$$

$$\vec{\beta}_A' = \frac{\vec{\beta}_A}{2} = -\vec{\beta}_B'$$

Rewrite as

$$\underset{\boxed{A}}{P(Y=1)} = \frac{e^{\vec{x}^T \cdot \vec{\beta}_A'}}{e^{\vec{x}^T \cdot \vec{\beta}_A'} + e^{\vec{x}^T \cdot \vec{\beta}_B'}}$$

$$\underset{\boxed{B}}{P(Y=0)} = \frac{e^{\vec{x}^T \cdot \vec{\beta}_B'}}{e^{\vec{x}^T \cdot \vec{\beta}_A'} + e^{\vec{x}^T \vec{\beta}_B'}}$$

Rewritten in this form ✓