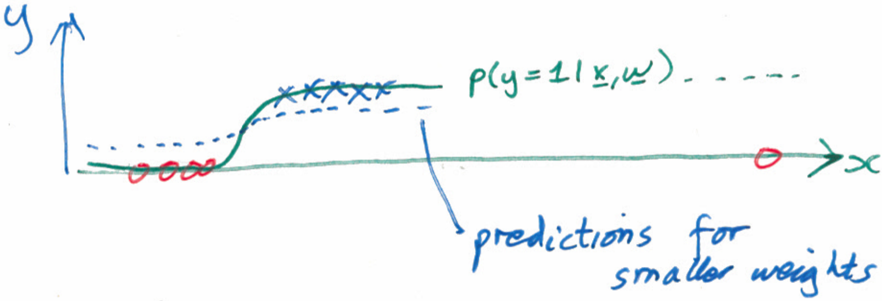
Robust Logistic Regression



Each example has binary variable

$$m^{(n)} \in \{0, 1\}$$

latent/hidden variable

I will assume $p(m) = \begin{cases} 1-\varepsilon & m=1 \\ \varepsilon & m=0 \end{cases}$

For example $\varepsilon = 0.01$

Model for labels

$$p(y=1 \mid \underline{x}, \underline{w}, m) = \begin{cases} \sigma(\underline{w}^T \underline{x}) & m=1 \\ \frac{1}{2} & m=0 \end{cases}$$

Need Likelihood of $\underline{w}, \varepsilon$

$$P(y=1 \mid \underline{x}, \underline{w}, \varepsilon) = \sum_{m \in \{0,1\}} P(y=1, m \mid \underline{x}, \underline{w}, \varepsilon)$$

(Sum Rule)

$$= \sum_{m \in \{0,1\}} P(y=1 \mid \underline{x}, \underline{w}, \cancel{\varepsilon}, m)\, P(m \mid \cancel{\underline{x}}, \cancel{\underline{w}}, \varepsilon)$$

(Product Rule) For this model

$$= (1-\varepsilon)\,\sigma(\underline{w}^T \underline{x}) + \varepsilon \frac{1}{2}$$

$$\frac{\partial \log P(y^{(n)} \mid \underline{x}^{(n)}, \underline{w})}{\partial \underline{w}} = \frac{1}{1 + \frac{1}{2}\left(\frac{\varepsilon}{1-\varepsilon}\right)\frac{1}{\sigma_n}} \nabla_{\underline{w}} \log P(y^{(n)} \mid \text{logistic reg.})$$

Prob. of being right.

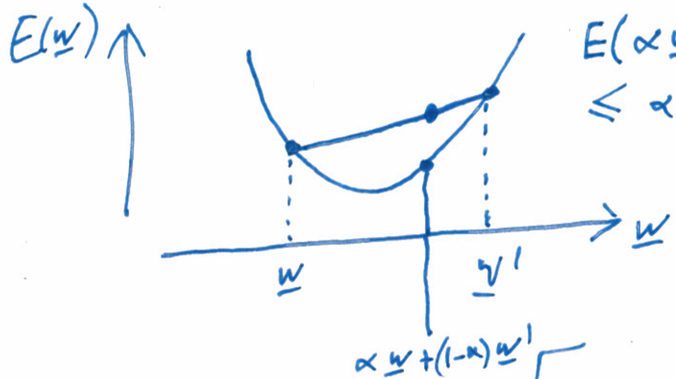
How to fit ε

Set it by hand?

Grid of settings in $\varepsilon \in [0, \frac{1}{2}]$

Maybe on a log scale $0.1, 0.01, 0.001, \ldots$?

For fixed ε the cost function is convex



$$E(\alpha \underline{w} + (1-\alpha)\underline{w}') \leqslant \alpha E(\underline{w}) + (1-\alpha) E(\underline{w}')$$

Not an Expectation!

$0 \leqslant \alpha \leqslant 1$

Could jointly fit $\theta = \begin{bmatrix} \underline{w} \\ \varepsilon \end{bmatrix}$, using $\nabla_\theta E$

But not convex.

Also ε are constrained, $\varepsilon \in [0,1]$

Trick: reparameterize model

$$\varepsilon = \sigma(b)$$

$-\infty < b < \infty$

logistic sigmoid.

$$b = \text{logit}(\varepsilon) = \log\left(\frac{\varepsilon}{1-\varepsilon}\right)$$

Derive $\frac{\partial E}{\partial b}$ and optimize b

Neural Networks

Linear regression with basis f^n's

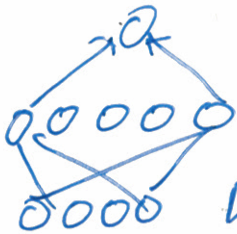
$$\phi_k(\underline{x}) = \sigma(\underline{v}^{(k)T}\underline{x} + b^{(k)})$$

Model f^n:

$$f(\underline{x}) = \underline{w}^T\underline{\phi}(\underline{x}) + b \quad \text{or} \quad f(\underline{x}) = \sigma(\underline{w}^T\underline{\phi} + b)$$

Function: $\theta = \{\{\underline{v}^{(k)}, b^{(k)}\}_{k=1}^{K}, \underline{w}^T, b\}$

Fit all of θ → call a "neural network"

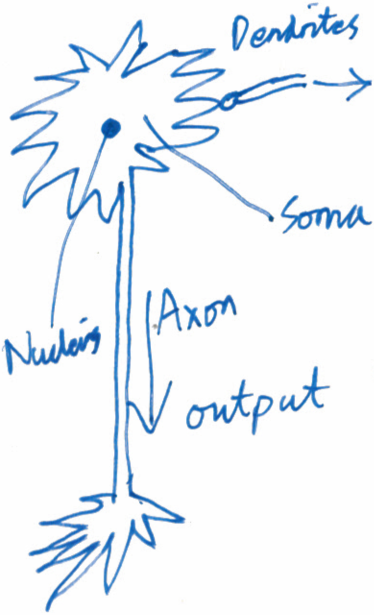


1 final neuron

K neurons

Data

Why "Neural Network"? (Non-examinable)



Inputs "$\underline{x}$"

output "$g(\underbrace{\underline{w}^T\underline{x}}_{\text{activation } a})$"

step $f^{\underline{n}}$

Soft step / Sigmoid

$g(a)=\sigma(a)$