Neural networks, 1

Last time: we started with logistic regression.

$$p(y=1|x) = \sigma(w^T x + w_0)$$

Aside: one way to derive this: let's have the difference in $p(y=1|x)$ and $p(y=0|x)$ be linear in log space:

$$\ln p(y=1|x) - \ln p(y=0|x) = \boxed{w^T x + w_0}$$

Now, we substitute $p(y=0|x) = 1 - p(y=1|x)$

$$w^T x + w_0 = \ln \frac{p(y=1|x)}{p(y=0|x)} = \ln \frac{p(y=1|x)}{1-p(y=1|x)}$$

and solve for $p(y=1|x)$:

$$\frac{p(y=1|x)}{1-p(y=1|x)} = \exp\{w^T x + w_0\}$$

$$p(y=1|x) = \frac{\exp\{w^T x + w_0\}}{1+\exp\{w^T x + w_0\}} = \frac{1}{1+\exp\{-1 \cdot \boxed{(w^T x + w_0)}\}}$$

$\downarrow$ $f(x, w, w_0)$

Write down the log likelihood:

$$ll(w) = \sum_{n=1}^{N} -y_n \ln(1+\exp\{-f_n\}) - (1-y_n) \ln(1+\exp\{f_n\}) \quad \text{to max.}$$

$$f_n = w^T x_n + w_0$$

(came from

$p(y=1|x)^y$

$y \ln p(y=1|x)$

$$\mathcal{L}(w) = \sum_{n=1}^{N} y_n \ln(1+\exp\{f_n\}) + (1-y_n) \ln(1+\exp\{f_n\}) \quad \text{to min}$$

define
$\mathcal{L}_n(w) =$
$y_n \ln(1+\exp\{-f_n\})$
$+ (1-y_n)\ln(1+\exp\{f_n\})$

Now we can take gradients $\rightarrow$ going to use chain rule

$$\frac{\partial \mathcal{L}_n(w)}{\partial w} = \frac{\partial f_n}{\partial w} \frac{\partial \mathcal{L}_n(w)}{\partial f_n}$$

$$\frac{\partial}{\partial w} w^T x_n + w_0 = x_n$$

$$\frac{\partial}{\partial f_n} \ln(1+\exp\{f_n\}) = \frac{\exp\{f_n\}}{1+\exp\{f_n\}} = p(y_n=1|x_n)$$

$$\frac{\partial}{\partial f_n} \ln(1+\exp\{-f_n\}) = \frac{-\exp\{-f_n\}}{1+\exp\{-f_n\}} = -p(y_n=0|x_n)$$

$\frac{\partial \mathcal{L}_n(w)}{\partial w} = \ldots p(y_n=0|x_n)(-1) + (1-y_n)\, p(y_n=1) \ldots x_n$

so: $\frac{\partial \mathcal{L}_n(w)}{\partial f_n} = y_n \, p(y_n=0|x_n)(-1) + \dots$

$$\frac{\partial \mathcal{L}_n(w)}{\partial w} = (x_n)\left(y_n \, p(y_n=0|x_n)(-1) + (1-y_n)\, p(y_n=1|x_n)\right)$$

so now, we can apply (S)GD:

1: sample a batch of data (size = M) (x_m, y_m)

2: compute $\frac{\partial \mathcal{L}_m(w)}{\partial w}$ for all x_m, y_m in M

3: finally, $w \leftarrow w - \underbrace{\eta}_{\text{step size}} \underbrace{\frac{N}{M}}_{\text{const}} \underbrace{\sum_m \frac{\partial \mathcal{L}_m(w)}{\partial w}}$] a lot of additional innovations here to reduce variance

→ ADAM (dim-wise step size)

→ momentum

assumption: our errors add. (log likelihoods, $\sum \text{err}_n^2$, ..)

are we done?

well, $w^T x + w_0$ isn't the most expressive function ...

choose a basis! $[x_1 \dots x_D] \rightarrow [\phi_1(\underline{x}) \cdots \phi_J(\underline{x})]$

↳ great if you know what ϕ should be

↳ but choosing ϕ can be hard

next: can we learn ϕ, alongside w, w_0?

(Neural Neworks are one way to do this!)

ASIDE
"Adaptive Basis" in stats
Regression

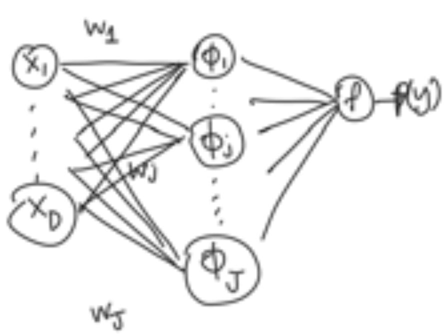
Neural Networks

idea: suppose we nest an LR in a LR!

$f(x_n) = w^T \phi + w_0$ } linear; will go through sigmoid $p(y=1|x_n) = \sigma(f_n)$

$\phi_j = \sigma(w_j^T x + w_{j0})$ for j in $1 \dots J$

in pictures:



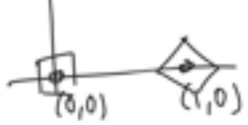
easy to put into matrix form:

if I take all the w_j's: $\underbrace{\left[w_1 \; w_2 \cdots w_j \cdots w_J \right]}_{W^1}$ with J columns and D rows

$$\phi = \underbrace{\sigma}_{\text{applied pair-wise}}\left(\underbrace{W^{1T} X}_{J \times N} \oplus \underbrace{w_0^1}_{\text{vector of size } J}\right)$$

examples (toy) suppose we want an X-OR function:

(0,1) (1,1) not linearly separable!



We can create basis functions that are of the σ form to make this linearly separable!

lets "pick out" [0,0] $\quad \phi_1 = \sigma(-x_1 - x_2 + \frac{1}{2})$

$\Rightarrow \quad f = -\phi_1 - \phi_2 + \frac{1}{2}$

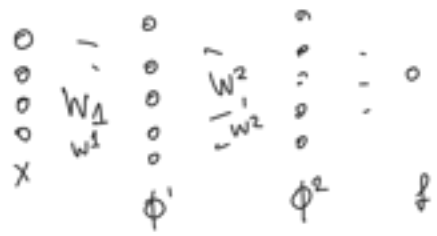
lets "pick out" [1,1] $\quad \phi_2 = \sigma(x_1 + x_2 - 1.5)$

Another example:

how could a linear set of basis functions be a good classifier for this?

pos: class is this intersection in the middle

Coming up:



Deep neural networks

multiple layers; other architectures

multiple outputs for multiple classes via softmax:

$p(y=1|x)$

$\vdots$

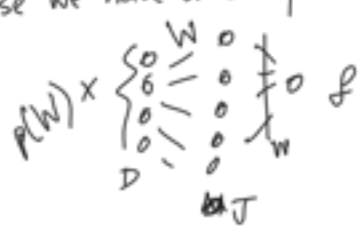
$p(y=k|x)$

$$\frac{\exp\{w_k^T x\}}{\sum_\ell \exp\{w_\ell^T x\}}$$

<u>Concept Exercise</u> : Bayesian Neural Networks

BNNs: lets put prob. over W, W^1 & compute posteriors...

suppose we have a 1-layer network w/ J-nodes



1. lets write down f as a function of feature outputs $\phi_j(x)$, w_j $\leftarrow w^T x$

$$f = \sum_j w_j \phi_j(x)$$

2. let $\phi_j(x)$ have some mean μ^J and variance σ_J^2

var(xy) =
var(x)var(y) +

2. ...

and $w_m \sim \mathcal{N}(0, \sigma_w^2)$

$\mathrm{var}(x)\mu_y^2 +$
$\mathrm{var}(y)\mu_x^2$

what is the mean & variance of f?

$E[f] = 0 \qquad \sigma_f^2 = J\left[\sigma_w^2\sigma_J^2 + \sigma_w^2\mu_J^2\right]$ assuming all units same μ^J, σ_J^2

3. Can you argue that f is Gaussian for large J?

CLT

↪ can be used to that an ooly broad 1-layer NN w/ these priors ... (dist over functions)