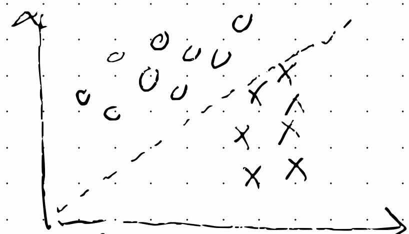
Logistic Regression in Python

Classification → supervised learning
(supervised)

decide between classes
↳ cat vs dog, numbers from pics

2d classification → 

classes can be sep.
by a straight line

↳ $y = x + 0$
↳ $0 = x - y$

Linear line can also be written as

if $h(x,y) = x - y$ then $h(x,y) \begin{cases} > 0 = \text{'O'} \\ < 0 = \text{'X'} \end{cases}$

difficulty: if $h(x,y) = 0$ then how do we define point?

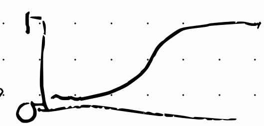
$h()$ is a linear combo of weights $\theta_1 \theta_2 \theta_3 \ldots = \underset{D\times 1}{\Theta}$
inputs $x_1 x_2 x_3 = \underset{n\times D}{X}$ and targets $y_1 y_2 y_3 = \underset{n\times 1}{\vec{y}}$
↳ $\underset{n\times 1}{\hat{y}} = X\Theta$

↳ note how in 3d, decision boundary becomes a line.

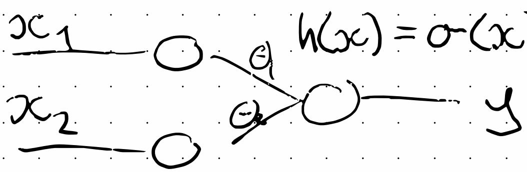
↳ in 4d; 3d-hyperplane

- Can we calculate the output of a neuron?

  ↳ neurons, inactive until threshold is passed & becomes active.



inputs



$h(x) = \sigma(x) \longrightarrow \tanh(x) \in \{-1, 1\}$

$\sigma(x) \in \{0, 1\}$

$= \dfrac{1}{1+e^{-x}}$

thus, if $\theta^T x$ ↑↑↑ then closer to 1.

if $\theta^T x$ ↓↓↓ then closer to 0.

note that $\sigma(\theta^T X) \in \{0,1\} = P(y=1 | X)$

where $P(y=1|X) + P(y=0|X) = 1$

↳ we predict the class w/ higher likelihood.

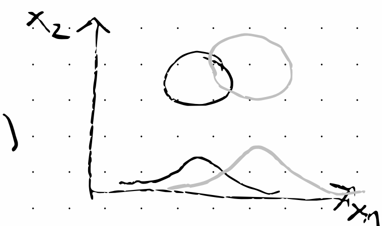
Assumption of Logistic Regression

= data can be separated by a plane.

= "linearly separable"

How do we find weights, θ?

consider 2 gaussian dist. clouds



$$p(x) = \frac{1}{\sqrt{2(\pi)^D |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

↳ from bayes rule $P(Y=1|X) = \dfrac{P(X|y=1)\,P(y=1)}{P(X)}$

& $P(y=0|X) = \dfrac{P(X|y=0)\,P(y=0)}{P(X)}$

if we combine the two probabilities...

$$P(y=1|X) = \frac{1}{1 + \dfrac{P(X|y=0)\,P(y=0)}{P(X|y=1)\,P(y=1)}}$$

already looks a little like logistic reg.

where $\ln\left(\dfrac{P(X|y=0)\,P(y=0)}{P(X|y=1)\,P(y=1)}\right) = -\theta^T x + b$

Let $p(y=1) = 1-\alpha$ & $p(y=0) = \alpha$

then $= \ln(p(X|y=0)) + \ln(\alpha) - \ln(p(X|y=1)) - \ln(1-\alpha)$

Gaussian dist.

if we add in gaussians then we get...

$$= \frac{-1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_0 - \mu_0^T\Sigma^{-1}x + \mu_0^T\Sigma^{-1}\mu_0)$$

$$\frac{+1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_1 - \mu_1^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}\mu_1)$$

$$+\ln\left(\frac{\alpha}{1-\alpha}\right)$$

$\hookrightarrow$ $w^T = (\mu_1^T - \mu_0^T)\Sigma^{-1}$

$b = \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \ln\frac{\alpha}{\alpha-1}$

Notes:

- Linear Discriminant analysis performed above.

... This is totally useless!

$\hookrightarrow$ gradient descent used instead.

→ Loss function, cross entropy

$$J = -\sum_{n=1}^{N} y_n \log(\hat{y}_n) + (1-y_n)\log(1-\hat{y}_n)$$

note that when $y=1$, ② $=0$ & when $y=0$, ① $=0$

↳ if $y=1, \hat{y}=1 \ldots J=0$

$y=0, \hat{y}=0 \ldots J=0$

$y=1, \hat{y}=.99 \ldots J=0.11$ ← small error

$y=1, \hat{y}=.5 \ldots J=0.69$ ← larger error

Maximum Likelihood

↳ $p(y=1)=p \quad p(y=0)=1-p$

say $N=10$, $N_1=7$, $N_0=3$

$L = p^7(1-p)^3$ ← maximize L w.r.t. p

$$\log L = \log[p^7(1-p)^3]$$

$$= 7\log p + 3\log(1-p)$$

$$\frac{\partial \log L}{\partial p} = \frac{7}{p} + \frac{-3}{1-p} = 0 \ldots \frac{1-p}{p} = \frac{3}{7}$$

$\therefore \frac{1}{p} - 1 = \frac{3}{7}$ ☺

$$p = \frac{7}{10} = p(H)$$

↳ Now that we understand this w/ 1s & 0s, apply to logistic reg.

$$p(y=1|X) = \sigma(\theta^T X) = y$$

$$L = \prod_{n=1}^{N} y_n^{t_n} (1-y_n)^{1-t_n}$$ ← target $\hat{y}$

↳ take log likelihood, get cross entropy - loss function

<u>How to optimize weights</u>

↳ LDA shown before only works when:

1. data is normally dist.
2. equal cov.

Try LR approach? can't solve for the weights

Solution, gradient descent!

↳ Start w/ loss function

$$J = -\sum_{n=1}^{N} y_n \log(\hat{y}_n) + (1-y_n)\log(1-\hat{y}_n)$$

↳ $$\frac{\partial J}{\partial \theta_i} = \sum_{n=1}^{N} \frac{\partial J}{\partial \hat{y}_n} \frac{\partial \hat{y}_n}{\partial a_n} \frac{\partial a_n}{\partial \theta_i} \leftarrow a_n = \theta^T x_n$$

$$\boxed{\frac{\partial J}{\partial y_n} = -y_n \frac{1}{y_n} - (1-y_n)\frac{1}{1-\hat{y}_n}}$$

deriv of σ

$$y_n = \sigma(a_n) \qquad \boxed{\frac{\partial \hat{y}_n}{\partial a_n} = \hat{y}_n(1-\hat{y}_n)}$$

$$\boxed{\frac{\partial a_n}{\partial \theta_i} = x_{n_i}}$$

assemble...

$$\frac{\partial J}{\partial \theta_i} = -\sum_{n=1}^{N} \frac{y_n}{\hat{y}_n} \hat{y}_n(1-\hat{y}_n) x_{n_i} - \frac{1-y_n}{1-\hat{y}_n} \hat{y}_n(1-\hat{y}_n) x_{n_i}$$

$$\therefore \boxed{\frac{\partial J}{\partial \theta} = \sum_{n=1}^{N} (y_n - \hat{y}_n) x_n}$$

vectorize save more: $\boxed{\frac{\partial J}{\partial \Theta} = X^T(y-\hat{y})}$ ✓

bias term? add column of 1s to X

↳ remember from Lin. Reg class:

$$\Theta := \Theta - \eta \frac{\partial J}{\partial \Theta} J(\Theta)$$

where $\frac{\partial J}{\partial \Theta} J(\Theta)$ = boxed equation above

pseudocode:

- import data
- $z = X\Theta$
- learning rate = 0.1 (or else)
- Steps

↳ for i in steps

$$\Theta := \Theta - \eta * [X^T(y - \sigma(X\Theta)]$$

Practical Issues

Most essential for any ML model

So far... ① training & ② predicting

↳ problems w/ overfitting
↳ problems where ideal weights are inf
↳ problems w/ non-linear decision boundaries

Interpreting the weights

↳ Still similar to lin. reg. except features w/ higher weights contribute more to $P(y=1|X)$

you can also look @ odds $\boxed{\frac{P(y=1|X)}{P(y=0|X)}}$

$= \exp(w^T X)$

log odds : $= w^T X$

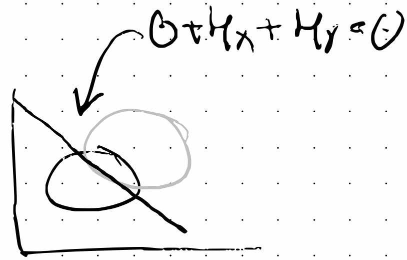
linear regression

→ L2 regularization

↳ prevent overfitting

Why use regularisation?

↳ circle problem



for the line, $x = -y$, do weights matter?

↳ due to the math, the best weights are

inf x = -inf y ← difficult to calculate

use regularization to penalize large weights

re $$J_{reg} = J + \lambda \frac{1}{2} w^T w = J + J_{L2}$$

λ ↳ smoothing param chosen manually.

$$\frac{\partial J_{L2}}{\partial w} = \lambda w$$

$$\therefore \quad \frac{\partial J_{reg}}{\partial w} = X^T(\hat{Y} - Y) + \lambda w$$

· Probabilistic perspective

↳ normally, maximize likelihood

regularization, maximize $\underbrace{\text{posterior}}_{\text{max. a posteriori (MAP)}}$

→ L1 regularization

↳ normally we want X to be tall
but what if X is wide?

· Goal: select features w/ most importance
set others to 0

once again $J_{reg} = J + \lambda w$

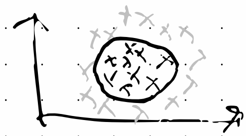
↳ $\frac{\partial}{\partial w} = \lambda \, \text{sign}(w)$

$$\therefore \frac{\partial J_{reg}}{\partial w} = X^T(\hat{Y} - Y) + \lambda \, \text{sign}(w)$$

→ Donut Problem

↳ Logistic Regression is good at classifying linearly seperable data.

Problem: non-linear sets



How to draw this line?

~~Similar to how we solve for our linear problems w/ linear regression~~

↳ in dataset add bias term & radius of each point

↳ grad desc based on L2 reg.