Example ① Logistic Regression

E: $X \in \mathbb{R}^{m \times n}$, $y \in \{0,1\}^m$ (supervised)

T: Produce a function $f: \mathbb{R}^n \to [0,1]$

from this, we define $\text{pred}_f: \mathbb{R}^n \to \{0,1\}$

$$\text{pred}_f(\vec{x}) = \begin{cases} 1 & f(\vec{x}) > .5 \\ 0 & f(\vec{x}) < .5 \end{cases}$$

P: Many choices exist, lets go with accuracy

$$\text{Acc} = \frac{\#\{i \mid \text{pred}_f(X_{i,*}^{(test)}) = y_i^{(test)}\}}{\#\text{ rows' in } X^{(test)}}$$

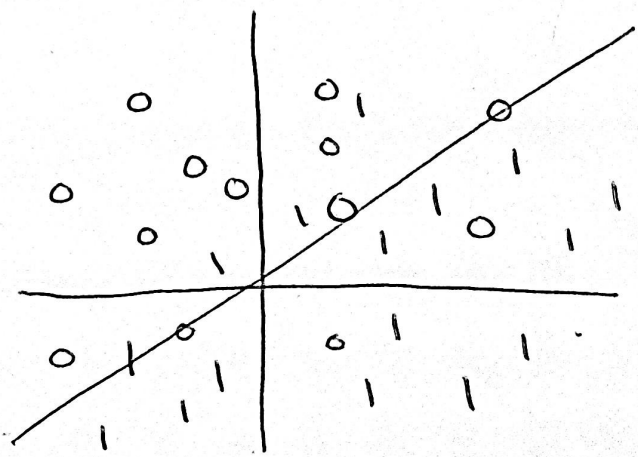
Note: Also could use precision or recall

A Solution

Case 1 We expect $y=1$ examples are reliably separated from $y=0$ examples by a hyperplane in $\mathbb{R}^n$

E.g. $X \in \mathbb{R}^{m \times 2}$

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 4 & 3 \\ \vdots & \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$



Goal: Find $\vec{w} \in \mathbb{R}^n$ $b \in \mathbb{R}$ s.t. the hyperplane $\vec{w}^T \vec{x} + b = 0$ separates 1's from 0's reliably

Question: How to find the best $\vec{w}, b$ using an optimization algorithm?

Want function $J(\vec{w}, b)$ which is minimized when accuracy is maximized. (Cost function)

Observations

$\vec{x} \cdot \vec{w} + b >> 0 \implies$ strong prediction for 1

" $<< 0 \implies$ " " " 0

This suggests we use a function g with the following properties

$$\left.\begin{aligned} &\lim_{z \to \infty} g(z) = 1 \\ &\lim_{z \to -\infty} g(z) = 0 \\ &g(0) = \tfrac{1}{2} \end{aligned}\right\} \quad (*)$$

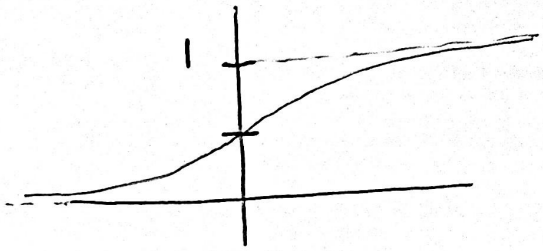
With this we can compute $g(\vec{x} \cdot \vec{w} + b)$

and predict 1 if $> .5$

0 if $< .5$

Def: The sigmoid function $g: \mathbb{R} \to [0,1]$ is

$$g(z) = \frac{1}{1+e^{-z}}$$



Prop ① g satisfies (*)

② $g'(z) = g(z)(1-g(z))$

Proof: Exercise

Again we use $\hat{X} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} X$ so $\hat{X} \cdot v$ instead of $Xw + \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix}$

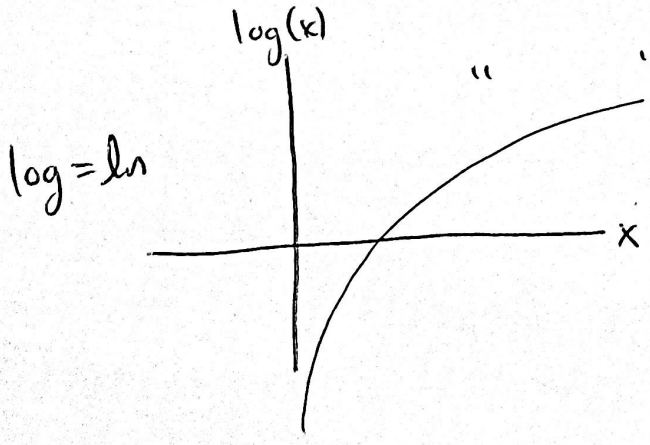
(i.e. $\vec{v} = (b \ \vec{w})^T$)

Idea: encourage $g(\vec{x}^T\vec{v})$ to be far from 0 if $y=1$

" " " " " 1 " $y=0$

$\iff$ penalize $\vec{v}$ if $y=1$ and $-\log(g(\vec{x}^T\vec{v}))$ is large

" " " $y=0$ " $-\log(1-g(\vec{x}^T\vec{v}))$ is large



$\iff$ penalize v if

$$-(1-y)\log(1-g(\vec{x}^T\vec{v})) - y\log(g(\vec{x}^T\vec{v})) \text{ is large}$$

Solution: Define the cost function $\left[\begin{array}{l}\text{Given dataset} \\ X, y = X^{(\text{train})}, y^{(\text{tra}}\end{array}\right.$

$$J(v) = \frac{1}{m}\sum_i -(1-y_i)\log(1-g(\hat{X}_{i,*}^T\vec{v})) - y_i\log(g(\hat{X}_{i,*}^T\vec{v}))$$

$$= \frac{1}{m}\left[-(\vec{1}-\vec{y})^T \text{Log}(\vec{1}-G(\hat{X}\vec{v})) - \vec{y}^T \text{Log}(G(\hat{X}\vec{v}))\right]$$

where $\text{Log}: \mathbb{R}^m \to \mathbb{R}^m$ $\text{Log}(x_1,\dots,x_m) = (\log(x_1),\dots,\log(x_m))$

$$G: \mathbb{R}^m \to \mathbb{R}^m \quad G(x_1,\dots,x_m) = (g(x_1),\dots,g(x_m))$$

and find $v^* = \underset{v}{\text{argmin}}\, J(v)$ using some form of gradient descent.

Proposition ① $\nabla_v J(v) = \frac{1}{m}(G(\hat{X}\vec{v}) - \vec{y})^T\hat{X}$

$$H(J)(v) = \frac{1}{m}\hat{X}^T D(v)\hat{X}$$

$$D(v) = \text{diag}(G'(\hat{X}v))$$

$$G'(x_1,\dots,x_m) = (g'(x_1),\dots,g'(x_m))$$

(2) J is twice differentiable, convex, and has Lipschitz continuous gradient.

Proof: HW 2 ∎

∴ Gradient descent reliably finds the global min (if it exists)

Case 2 Expect a nonlinear "decision boundary"

Solution Modify X by adding new columns corresponding to polynomial or other functions of current columns.

§5.2 Capacity, Overfitting, Underfitting

Recall Learning algo $(E, T, P, \text{program})$

the program is "trained" using E (e.g. $X^{(train)}, y^{(train)}$)

but P is determined by "test" data (not in E) (e.g. $X^{(test)}, y^{(test)}$)

Usually have some "cost function" and we minimize this w.r.t. training set to determine program

That is, minimize "test error" by minimizing "training error".

Question How can we statistically guarantee this approach works?

Need to make assumptions (i.i.d. assumptions): independent identically distributed

Both the training set and test set are gotten by independently randomly sampling the same probability distribution (call this P_{data})

Usually our task T is accomplished by choosing some set $\vec{w}$ of parameters found by minimizing some cost function $J(\vec{w})$ on training set.

E.g.