$$P(y=c \mid x;\theta) \propto \frac{P_\theta(x \mid y;\theta)\, P_\theta(y)}{N}$$

$$\propto P_\theta(x, y;\theta)$$ ← generative classifier

$P_\theta(x, y)$ $\quad$ $P(x; y=c)$

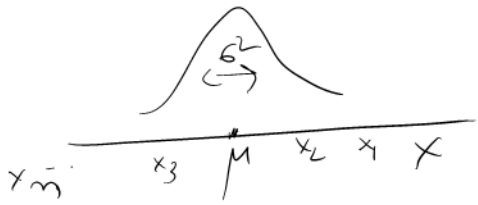
For class $c = \text{cat}$ feature of cat images

distribution of feature.

region of most probable feature for cat images

side

$= \mathcal{N}(x \mid \mu, \sigma^2)$



- Given probability distribution it easy to evaluate the probability in an interval (continuous random variable) or at a point (discrete random variable as
- But given probability distribution how do we generate sample $x_1, x_2 \ldots x_n$ (will cover sample generation later

$P(y=c \mid x;\theta)$

In discriminative classifier we model $P(y=c \mid x;\theta)$ (this) as a function $x;\theta$

We will write a direct function of x giving us the probability $P(y=c \mid x;\theta)$

(Note: here we don't model joint distribution $P(x,y)$)

For Binary case: logistic Regression (LR)

$$P(y=1 \mid x;w) = \mathrm{Ber}(y=1 \mid \sigma(w^T x))$$

[classification via regression]

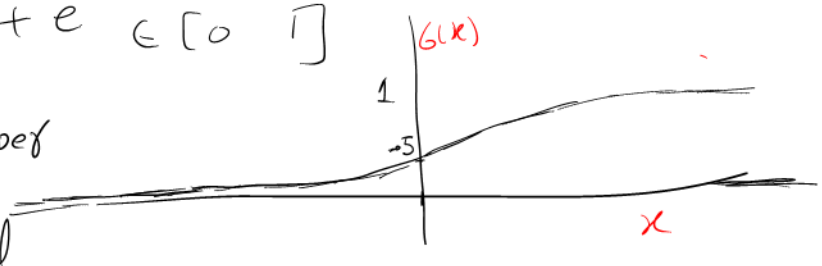
$= \sigma(w^T x)$

let $x \in \mathbb{R}^D$, $w \in \mathbb{R}^D$

Side what is σ (sigmoid function) $= \dfrac{1}{1+e^{-w^T x}}$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \text{here } x \in \mathbb{R}$$

$\in [0\ 1]$



sigmoid takes a real number and map it to $[0\ 1]$ interval

derivative $\sigma'(x) = \sigma(1-\sigma(x))$

LR is Popular

(1) will see that it is easy to extend for multi-class

(2) — using kernel trick $\phi(x)$ can model non-linear decision boundaries

what is decision surface of LR?

$$P(y=1 \mid x;w) = P(y=0 \mid x;w)$$

$$\frac{1}{1+e^{-w^T x}} = 1 - \frac{1}{1+e^{-w^T x}}$$

$$\frac{e^{w^T x}}{\cancel{e^{w^T x}+1}} = \frac{e^{-w^T x}}{1+e^{-w^T x}} = \frac{1}{\cancel{e^{w^T x}+1}}$$

$$e^{w^T x} = 1$$

take loge $\quad w^T x = 0$ or $w_1 x_1 + w_2 x_2 + \quad w_d x_d = 0$

Hence decision boundary is a line in high dim or is hyperplane

Note $x = \begin{bmatrix} x_1 \\ x_2 \\ x_d \\ 1 \end{bmatrix}$

modify add 1 as last feature

Model Fitting (estimate w)

let say we observe samples $D = \{(x_i, y_i)\}_{i=1}^{N}$

I.I.D samples

conditional log likelihood

$$\log P(D) = \log \prod_{i=1}^{N} p(y_i | x_i; w)$$

$$= \log \prod_{i=1}^{N} \sigma(w^T x_i)^{\mathbb{I}(y_i=1)} (1 - \sigma(w^T x_i))^{\mathbb{I}(y_i=0)}$$

$$= \sum_{i=1}^{N} \left(y_i \log \sigma(w^T x_i) + (1-y_i) \log(1 - \sigma(w^T x_i)) \right)$$

side

$Ber(x, \theta) = \theta$ if $x = 1$

$= 1 - \theta$ if $x = 0$

side OR

$Ber(x, \theta) = \theta^{x} (1-\theta)^{1-x}$ for $x \in \{0, 1\}$

$$NLL(D, w) = -\sum_{i=1}^{N} \left(y_i \log \sigma(w^T x_i) + (1-y_i) \log(1 - \sigma(w^T x_i)) \right)$$

$$Pred(x_i) = \begin{bmatrix} \sigma(w^T x_i) \\ 1 - \sigma(w^T x_i) \end{bmatrix}$$

prediction via LR

also see $\text{oneHot}(y_i) \begin{bmatrix} y_i \\ 1-y_i \end{bmatrix}$

ground truth

$\Sigma = 1$ vector sums up to 1

$$= -\sum_{i=1}^{N} \text{cross entropy}(Pred(x_i), \text{oneHot}(y_i))$$

$a = \begin{bmatrix} a_1 \\ \vdots \\ a_K \end{bmatrix}$ $b = \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}$

$\text{cross entropy}(a, b) = -\sum_{i=1}^{K} a_i \log b_i$

$-\sum_{i=1}^{K} a_i \log a_i$ entropy if $b = a$

can't write MLE solution in closed form?

But $NLL(D, W)$ is still a convex function in W

Solution: we use optimazation algorithms (iterative)

$$\theta_{K+1} = \theta_K - \eta \left(\frac{\partial NLL(\theta)}{\partial \theta} \right)_{K}^{at}$$

η: Stepsize = called Learning rate in deep learning

derivative (direction of steep descent)

effect of this is

$$NLL(\theta_{K+1}) \leq NLL(\theta_K)$$

Hence we are gaurenteed to not increase $NLL(\theta)$

$$\theta_{k+1} = \theta_k - \eta \left(\frac{\partial NLL(\theta)}{\partial \theta}\right)_k + \mu(\theta_k - \theta_{k-1})$$

Momentum update

$$0 \le \mu \le 1$$

$$D = \{(x_i, y_i)\}_{i=1}^{N}$$

data split into

$$T = \{(x_i, y_i)\}_{i=1}^{N/2} \qquad V = \{(x_i, y_i)\}_{i=N/2}^{N}$$

training set — validation set

model selection. Will cover in detail

8.3.6

$$f(w) = \underbrace{NLL(w)}_{+ve} + \underbrace{\lambda \|w\|}_{+ve}$$

quadratic penalty term

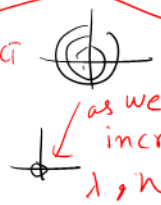
hyper parameter

regularization (called weight decay in deep learning)

can do LASSO too

or $f(w) = NLL(w) + \lambda |w|$

EFFECT of λ



as we increase λ, w decays towards 0

or can mix penalty

$$f(w) = NLL(w) + \lambda_1 \|w\|^2 + \lambda_2 |w|$$