Lecture 8 Discriminative Learning - Linear classifiers

Generative model: 1) learn CCD from data $p(x|y)$

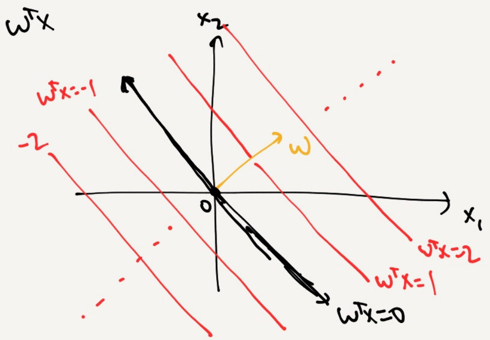
2) BDR to get classifier $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$

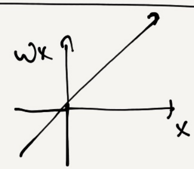
- **Note:** data is used only in step (1) to get the CCDs. classifier is secondary.
- Density estimation is an ill-posed problem.
  Gaussian, GMM, Gamma, ...

Vapnik advice: "when solving a given problem, avoid solving a more general problem as an intermediate step."

Discriminative Solution: solve the decision boundary or $p(y|x)$ directly.

"use the data to learn to discriminate classes, rather than generating data"





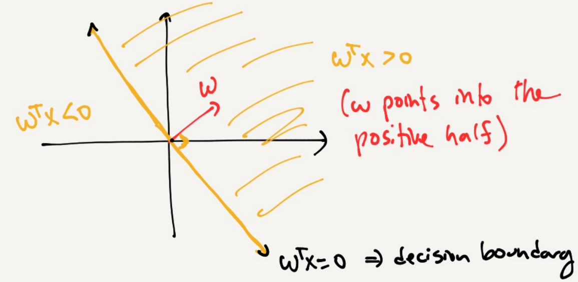
Linear classifier

output: $y \in \{+1, -1\}$ binary class

input: $x \in \mathbb{R}^d$

Linear function: $f(x) = w^Tx$

w separates the space into 2 half-spaces



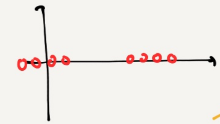
Decision rule

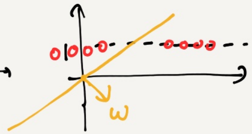
$$y^* = \mathrm{sign}(w^Tx) = \begin{cases} +1, & w^Tx \geq 0 \\ -1, & w^Tx < 0 \end{cases}$$

Note: bias can be included as another dimension

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad f(\tilde{x}) = \tilde{w}^T\tilde{x} = w^Tx + b$$

$$\tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}$$





Training Set

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

$D = \{X, y\}$, $X = [x_1 \cdots x_n]$, $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

Note:

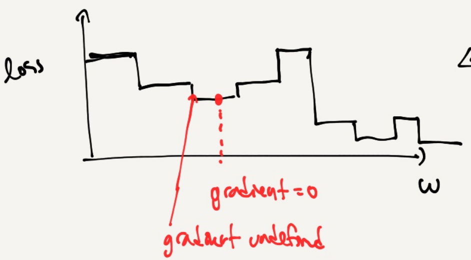
Given w, $\underbrace{y_i}_{\substack{a)\ +1 \\ b)\ -1}} \underbrace{w^T x_i}_{\substack{>0 \\ <0}} > 0 \Rightarrow$ correctly classified x_i

$y_i w^T x_i < 0 \Rightarrow$ misclassified x_i

Ideal case: 0-1 loss function

Optimize the # of misclassified samples.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \underbrace{\begin{cases} 0, & y_i w^T x_i > 0 \\ 1, & y_i w^T x_i < 0 \end{cases}}_{\text{0-1 loss function}}$$



difficult to optimize.

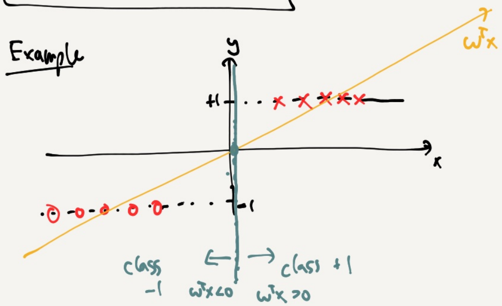
Least squares classification (Label regression)

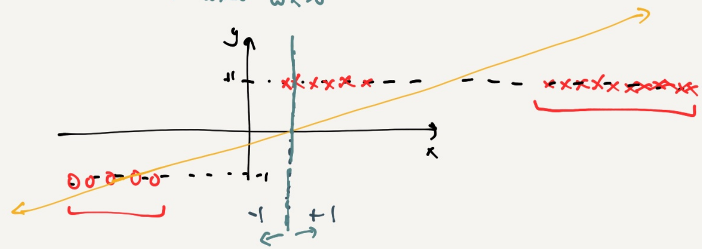
- Ignore the fact that y is discrete & just apply LS regression.

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - w^T x_i)^2 = \underset{w}{\operatorname{argmin}} \|y - X^T w\|^2$$

$$\boxed{\hat{w} = (XX^T)^{-1} X y}$$

Example





- not robust to outliers.
- Squared error penalizes predictions that are "too correct".
- FLD is a version of least-squares classifier (PS7-7)

Perceptron (Rosenblatt, 1962)

Perceptron criteria – only look at misclassified points.

$$E(w) = \sum_{i \in \mathcal{M}} \underbrace{-y_i w^T x_i}$$

$\mathcal{M}$: misclassified points

$-y_i w^T x_i$: higher loss for x_i that are badly misclassified, $y_i w^T x_i \ll 0$

$E(w) = 0$ when data correctly classified.

Perceptron Algorithm

$$w^* = \underset{w}{\operatorname{argmin}}\, E(w) = \underset{w}{\operatorname{argmin}} \sum_{i \in \mathcal{M}} -y_i w^T x_i$$

- computers were slow in 60's...
- Apply "stochastic gradient descent" (SGD)
  - use one data point at a time:

$$w^{(t+1)} = w^{(t)} + \eta\, \underbrace{y_i x_i}, \quad \text{for some } i \in \mathcal{M}.$$

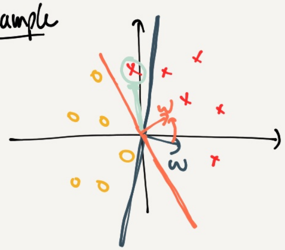
η: learning rate

$y_i x_i$: rotates w to point towards

1) the positive class

2) w gets longer as we iterate $\Rightarrow$ each sample has diminishing effect.

example

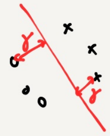


- Rosenblatt proved that perceptron algo converges in $\left(\frac{R}{\gamma}\right)^2$ iterations if the data is linearly separable.

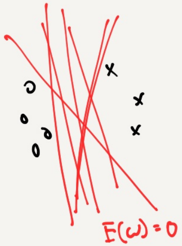
$$R = \max_i \|x_i\|$$

$$\gamma = \text{"margin"} : \|\hat{w}\|^2 = 1, \quad y_i \hat{w}^T x_i \geq \underline{\gamma}, \ \forall i$$

(how separable is the data)



- will not converge if the data is not linearly separable.
- many possible solutions with loss $= 0$, depends on initialization.



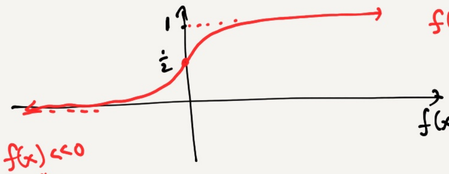
- Note: learning rate doesn't matter.

Logistic Regression (probabilistic approach)

- consider 2-class problem: $y \in \{0,1\}$
- PS 6-7: for CCDs of Gaussians
  $\Rightarrow$ posterior $p(y|x)$ is a sigmoid function.

$$p(y=1|x) = \frac{1}{1+e^{-f(x)}} = \sigma(f(x))$$

sigmoid



$f(x) \gg 0 \Rightarrow \sigma(f(x)) \to 1$
$p(y=1|x) \to 1$

$f(x) \ll 0$
$\Downarrow$
$\sigma(f(x)) \to 0$
$p(y=1|x) \to 0$

- when CCD Gaussians have same covariance $\Rightarrow f(x)$ is a linear function.
- with BDR, $f(x)$ is determined from the CCD parameters.

$\Rightarrow$ now, learn the parameters of $f(x) = w^T x$ directly.

Setup

$$f(x) = w^T x$$

$$p(y=1|x) = \frac{1}{1+e^{-w^T x}} = \sigma(w^T x) = \pi$$

Decision rule

$$\hat{y} = \begin{cases} 1, & p(y=1|x) > \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} = \begin{cases} 1, & w^T x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Look at the # of parameters

LR: $w \in \mathbb{R}^d \to d$ parameters.

much better for high dimensional data (need less data compared to BDR)

BDR: $\mu \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d \times d} \to O(d^2/2)$ parameters

Learning

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

let $\pi_i = \sigma(w^T x_i) = p(y_i = 1 | x_i)$

distribution: $p(y_i | x_i, w) = \pi_i^{y_i} (1-\pi_i)^{1-y_i}$

log-likelihood of data:

$$\ell(w) = \sum_i \log p(y_i | x_i, w)$$

$$= \sum_i y_i \log \pi_i + (1-y_i) \log (1-\pi_i)$$

MLE

$$w^* = \underset{w}{\operatorname{argmax}} \; \ell(w)$$

Alternatively

$$w^* = \underset{w}{\operatorname{argmin}} \; -\ell(w)$$

$$= \underset{w}{\operatorname{argmin}} \underbrace{\sum_i - y_i \log \pi_i - (1-y_i) \log (1-\pi_i)}_{\text{"cross-entropy loss"}}$$

- Maximize $\ell(w)$
- Apply Newton-Raphson method

$$w^{(new)} = w^{(old)} - [\underbrace{\nabla^2 \ell(w)}_{\text{Hessian}}]^{-1} \underbrace{\nabla \ell(w)}_{\text{gradient}}$$

tutorial

$$w^{(new)} = (XRX^T)^{-1} XRz$$

← weighted least squares w/ R, z, X

where

$R = \operatorname{diag}(\pi_1(1-\pi_1), \ldots, \pi_n(1-\pi_n))$ ← weights depend on current w.

$z = \underbrace{X^T w^{(old)}}_{\text{current prediction}} - R^{-1} \underbrace{(\pi - y)}_{\text{error}}$ ← target depends on current w

iterative reweighted least squares (IRWLS) (IRLS)

Comparison of loss (error) functions.

All of these methods are of the form:

$$w^* = \arg\min_{w} \underbrace{\sum_{i=1}^{n} L(f(x_i), y_i)}_{\text{empirical risk}}$$

loss function

"empirical risk minimization" — all about training error.

Let $z_i = y_i w^T x_i \Rightarrow \begin{cases} z_i > 0 \Rightarrow \text{classify correctly } (x_i, y_i) \\ z_i < 0 \Rightarrow \text{misclassify} \end{cases}$

loss functions

Ideal 0-1 loss: $L(z) = \begin{cases} 0, & z > 0 \\ 1, & z < 0 \end{cases}$

LS classifier: $L(z) = (z-1)^2$

perceptron: $L(z) = \begin{cases} 0, & z > 0 \\ -z, & z < 0 \end{cases} = \max(0, -z)$

logistic regression $L(z) = \log(1 + e^{-z}) \cdot \frac{1}{\log(2)}$

$z = 0 \Rightarrow L(z) = 1 \Rightarrow$ still have loss idea, to seperate further.

Consider $y = 1$

$\Rightarrow \begin{array}{ll} w^T x > 0 & 1 \\ w^T x < 0 & 0 \end{array}$

LSC & LR loss are convex approx to the ideal 0-1 loss.