Lecture 9 Discriminative Learning – Linear classifier

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 estimator 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 generate data"

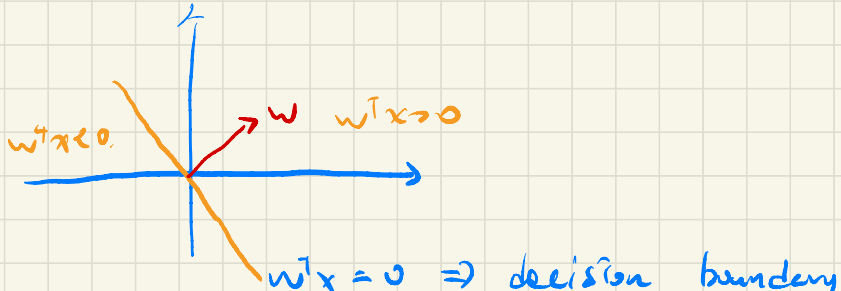
Linear classifier

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

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

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

w seperates the space into 2 half-spaces.



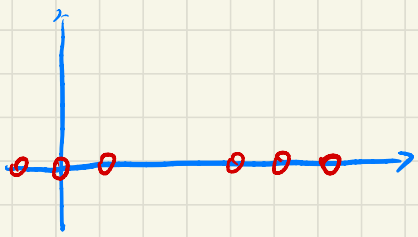
w points into the positive half. (?)

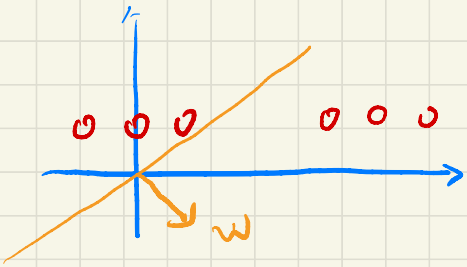
Decision rule:

$$y^* = \text{sign}(w^T x) = \begin{cases} +1 & w^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Note: bias can be included as another dimension

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





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, $y_i w^T x_i > 0 \Rightarrow$ correctly classified x_i

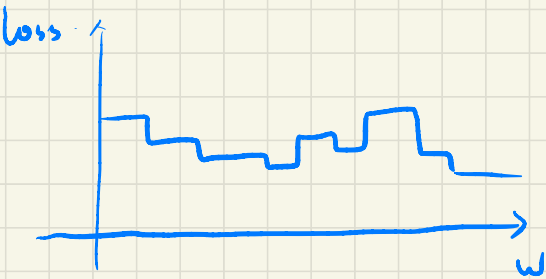
$y_i w^T x_i < 0 \Rightarrow$ misclassified x_i

Ideal case: 0-1 loss function:

Optimize the number of misclassified samples.

$$w^* = \arg\min_w \sum_{i=1}^{n} \begin{cases} 0, & y_i w^T x_i > 0 \\ 1 & y_i w^T x_i < 0 \end{cases}$$

0-1 loss function.



★: the gradient is 0 or undefined. So it's difficult to optimize.

Least square classification (Label regression)

- Ignore the fact that y is discrete and just apply

LS regression

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

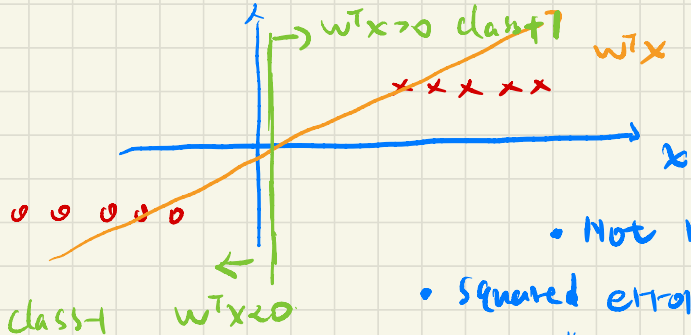
$$\|y - X^T w\|^2 = (y - X^T w)^T (y - X^T w)$$

$$= y^T y - y^T X^T w - (X^T w)^T y + w^T X X^T w$$

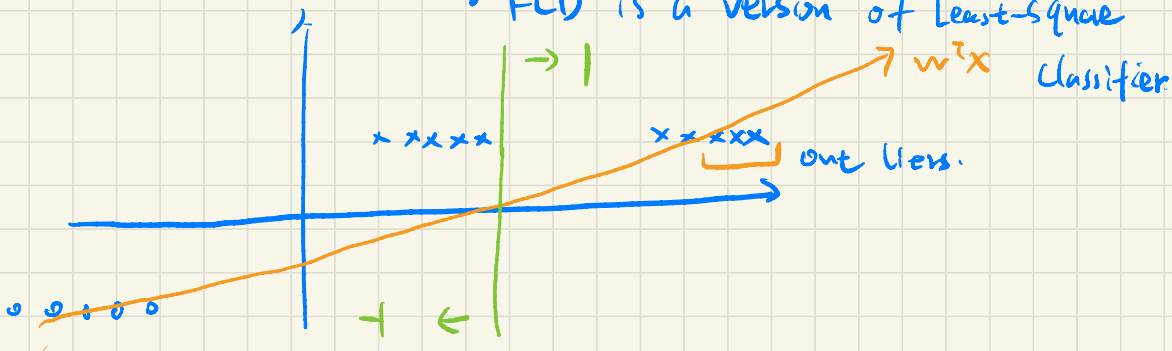
$$= y^T y - 2 y^T X^T w + w^T X X^T w$$

$$\frac{\partial}{\partial w} = -2Xy + 2XX^T w = 0 \Rightarrow w = (XX^T)^{-1} Xy$$

Example:



- Not robust to outliers
- Squared error penalizes that are "too corrected"
- FLD is a version of Least-square classifier



Perceptron

perceptron criteria — only look at misclassified points

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

higher loss for x_i that are badly misclassified: $y_i w^T x_i << 0$

M: misclassified points

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

Perceptron Algorithm:

$$w^* = \arg\min_w E(w) = \arg\min_w \sum_{i \in M} -y_i w^T x_i$$

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

$$w^{t+1} = w^t + \eta\, y_i x_i, \text{ for some } i \in M$$

η: learning rate.

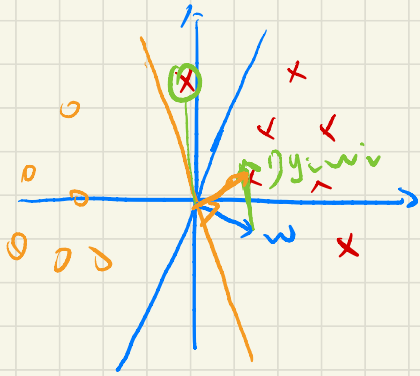
rotates w to point towards

1) the positive class.

2) w gets longer as we iterate

⇒ each sample has diminishing effect

example:



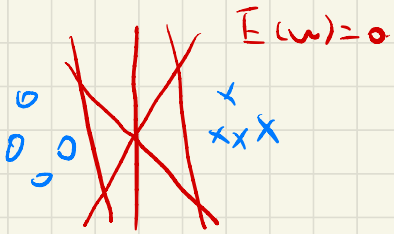
- Rosenblatt proved that perceptron algorithm converges in $\left(\frac{R}{\gamma}\right)^2$ iterations if the data is linearly seperable

$R = \max_i \|x_i\|$

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

how seperate is the data

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



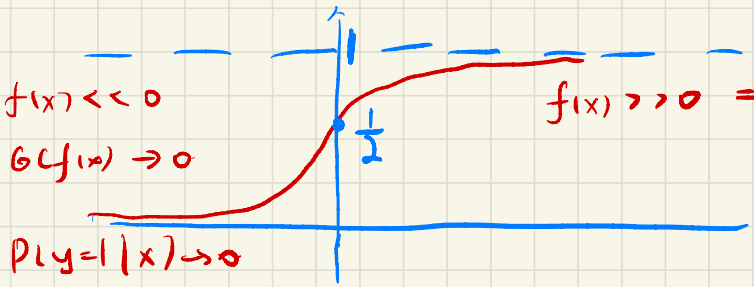
- Note: learning rate doesnot matter.

Logistic Regression (probabilistic appwach)

- Consider two class problem: $y \in \{0, 1\}$
- Ps 6-7: for CCD of Gaussians $\Rightarrow$ Postorior $P(y|x)$ is a sigmoid function.

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

sigmoid.



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

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

Set up:

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

Decision rule:

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

Look at the number of parameters

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

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

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

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

Let $\pi_i = \sigma(w^T x_i) = p(y_i = 1 \mid x_i)$

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

log-likelihood of data:

$$L(w) = \sum_i \log p(y_i \mid x_i, w)$$

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

MLE:

$$w^* = \arg\max_w L(w)$$

Alternatively:

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

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

- Minimize $L(w)$
- Apply Newton-Raphson method.

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

tutorial ↓

$$w^{new} = (X R^T X)^{-1} X R z$$ ← weighted least square with R, z, X

where: iterative reweighted least squares (IRWLS) (IRLS)

$R = \operatorname{diag}(\pi_1(1-\pi_1), \ldots, \pi_n(1-\pi_n))$ ← weighted 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

Comparison of loss (error) functions

All of these methods are of the form:

$$w^* = \arg\min_w \sum_{i=1}^{n} L(f(x_i), y_i)$$

L: Loss function

$\sum_{i=1}^{n} L(f(x_i), y_i)$: empirical risk

"empirical risk minimization" → all about training error

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

Loss function:

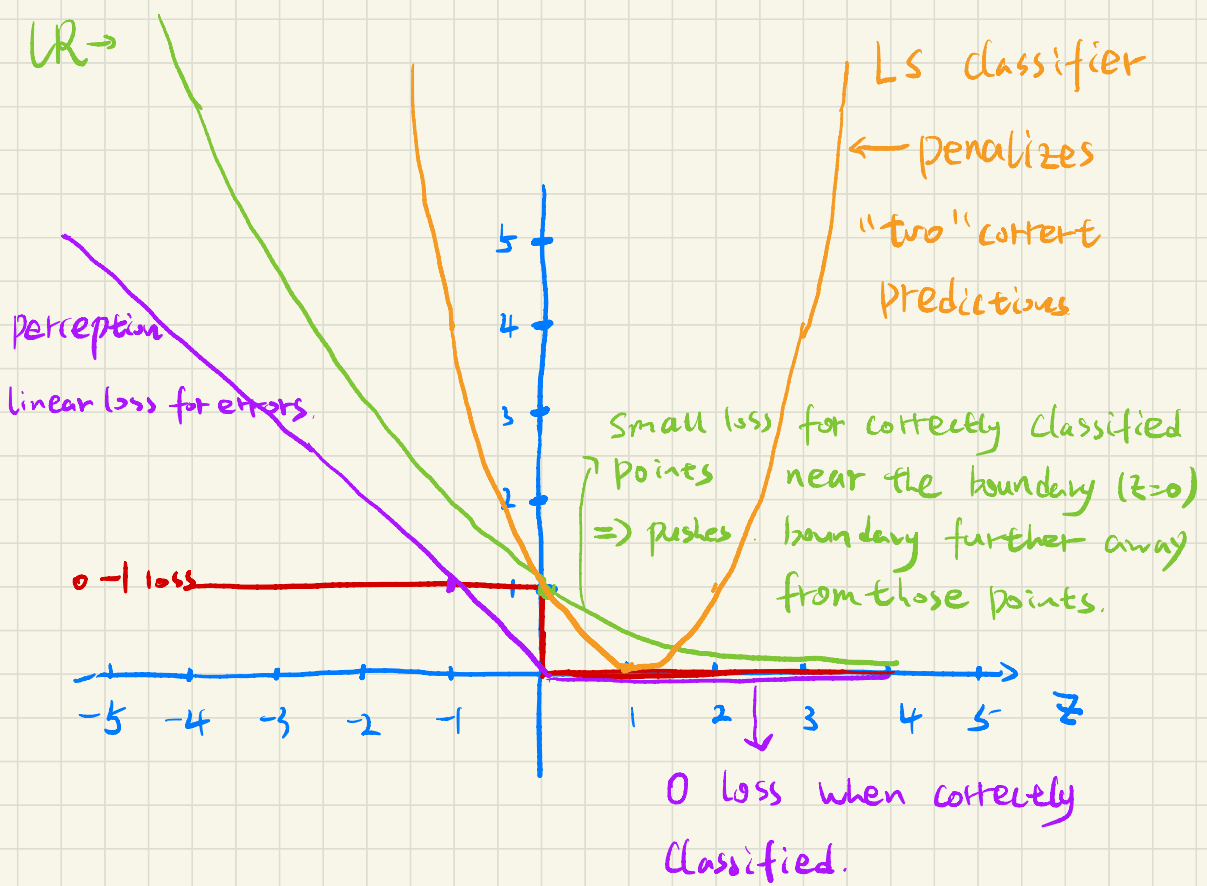
Ideal 0-1 loss: $L(z) = \begin{cases} 0, & z > 0 \\ 1, & z < 0 \end{cases}$

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

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

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

$z = 0, \quad L(z) = 1$



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