Lecture 8 Discriminative Learning – Linear Classification

Generative model

⇒ 1) learn CCD from training set ($p(x|y)$)

2) use BDT & BDR to get $p(y|x)$

$g(x) = \arg\max_y p(y|x)$

- The data is used to learn the CCD, the decision rule is secondary.
- Density estimation is an ill-posed problem.
  - which density to use? Gaussian, Laplacian, GMM, KDE

Vapnik's Advice: "When solving a given problem, try to avoid solving a more general problem as an intermediate step."

Discriminative Soln: use the data to directly estimate the decision rule.
- $g(x)$ or $p(y|x)$

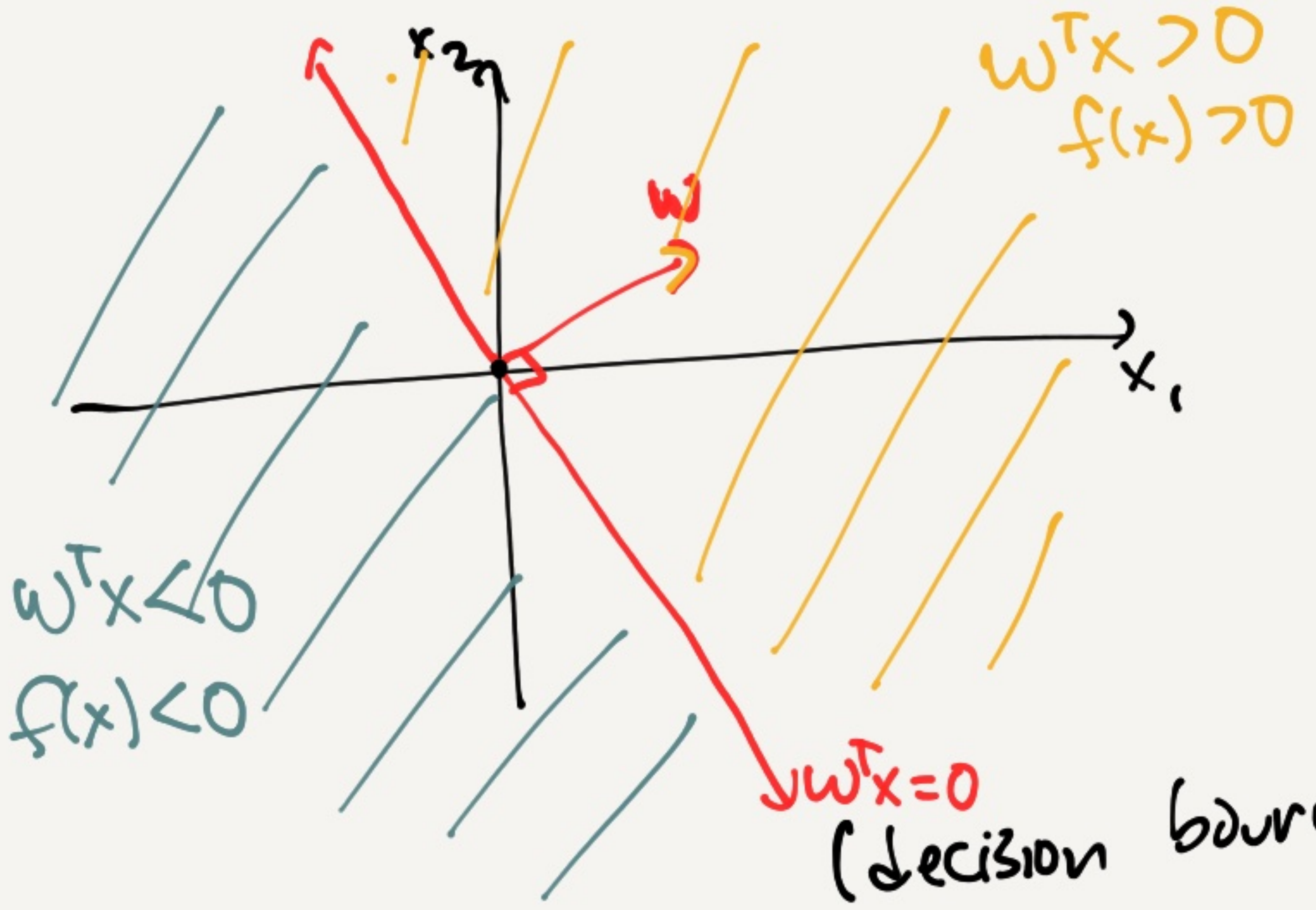
CS5487 Lecture Notes (2020)
Dr. Antoni B. Chan
Dept of Computer Science
City University of Hong Kong

Linear classifier

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

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

linear function: $f(x) = w^T x$, $w \in \mathbb{R}^d$ parameters of classifier.



w separates the space into 2 half-spaces. w points into the "positive" space.

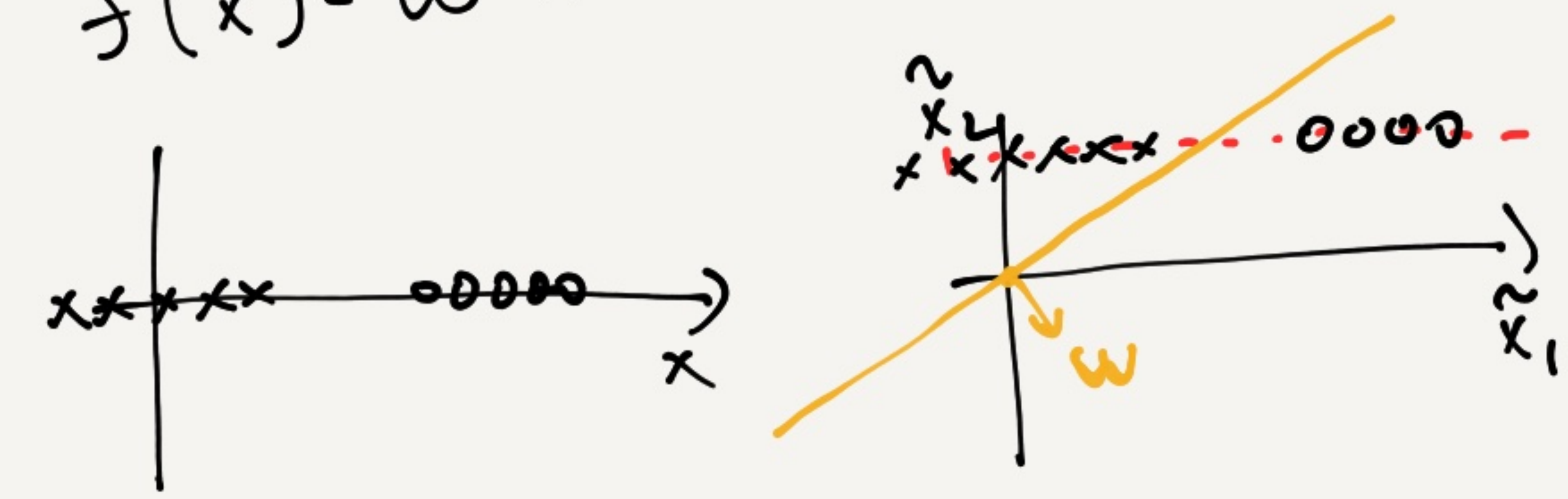
Decision Rule:

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

Note: bias term can be included into input.

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} \quad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}$$

$$f(\tilde{x}) = \tilde{w}^T \tilde{x} = w^T x + b$$



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

$= \{X, y\}$ $X = \begin{bmatrix} x_1 \cdots x_N \end{bmatrix}$ $y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$

Note:

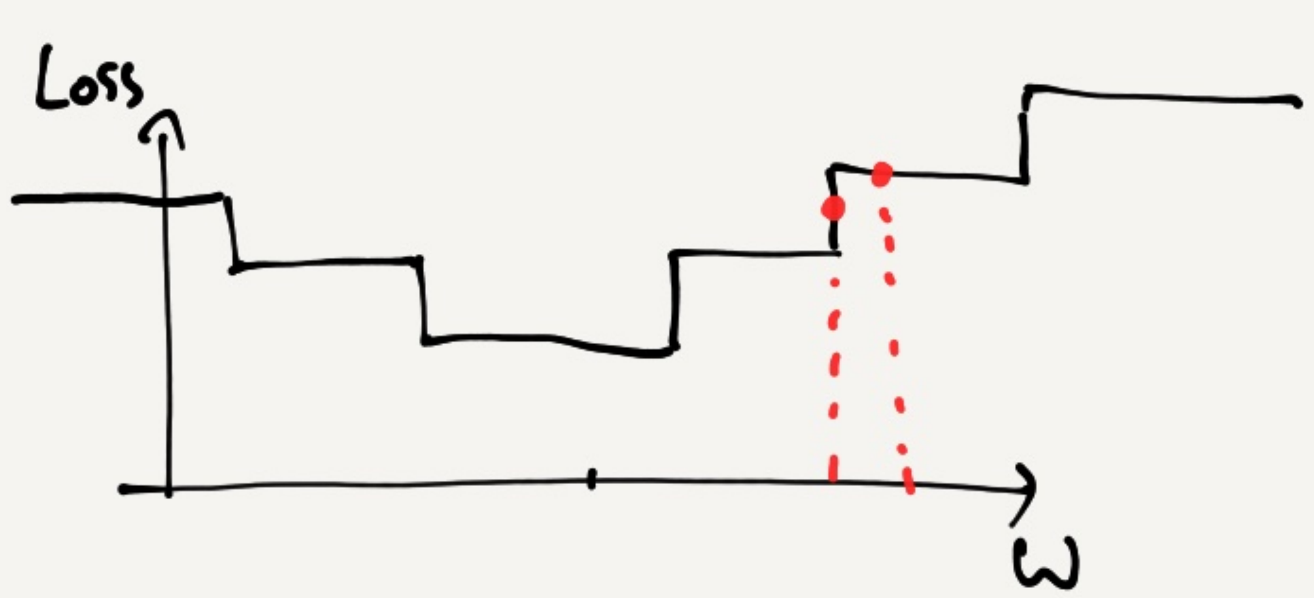
Given a w: $\underbrace{y_i}_{\substack{+1 \\ -1}} \underbrace{w^T x_i}_{\substack{>0 \\ <0}} > 0 \Rightarrow x_i$ correctly classified

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

Ideal Case: 0-1 loss function

Optimize the # of misclassifications.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n \begin{cases} 0, & y_i w^T x_i > 0 \quad \text{(correct)} \\ 1, & y_i w^T x_i < 0 \quad \text{(incorrect)} \end{cases}$$



Difficult to optimize!

Gradient is either 0, or undefined.

Least squares Classification (Label regression)

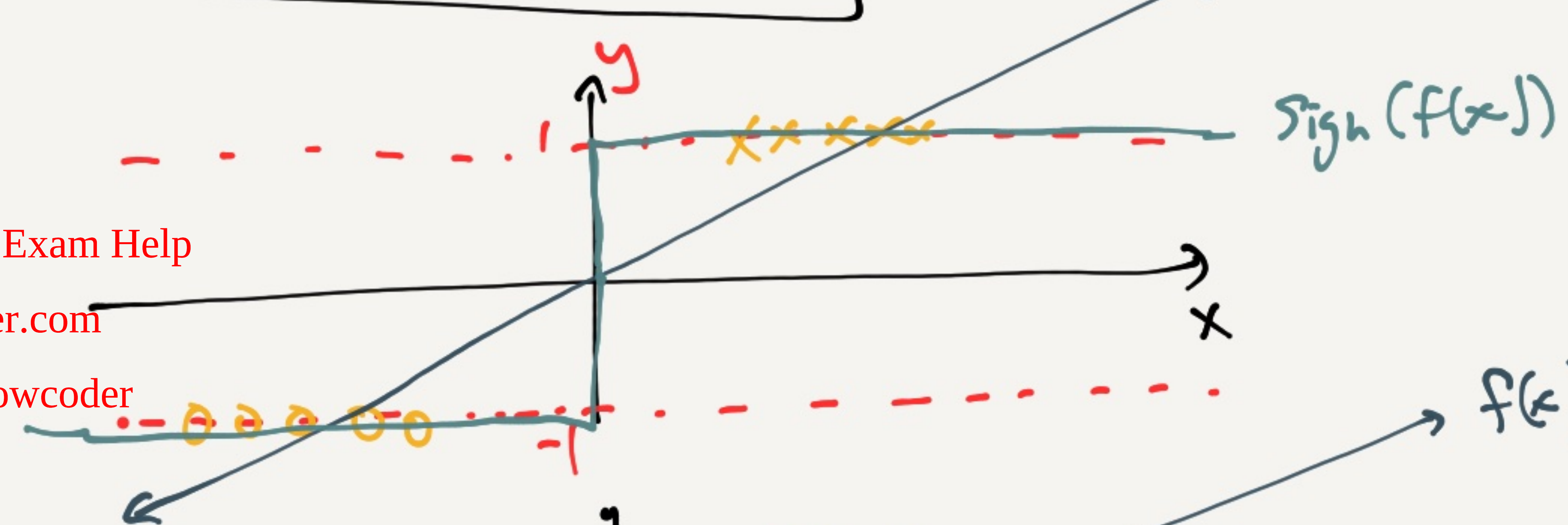
- Ignore the fact that y is discrete, & just apply LS regression.

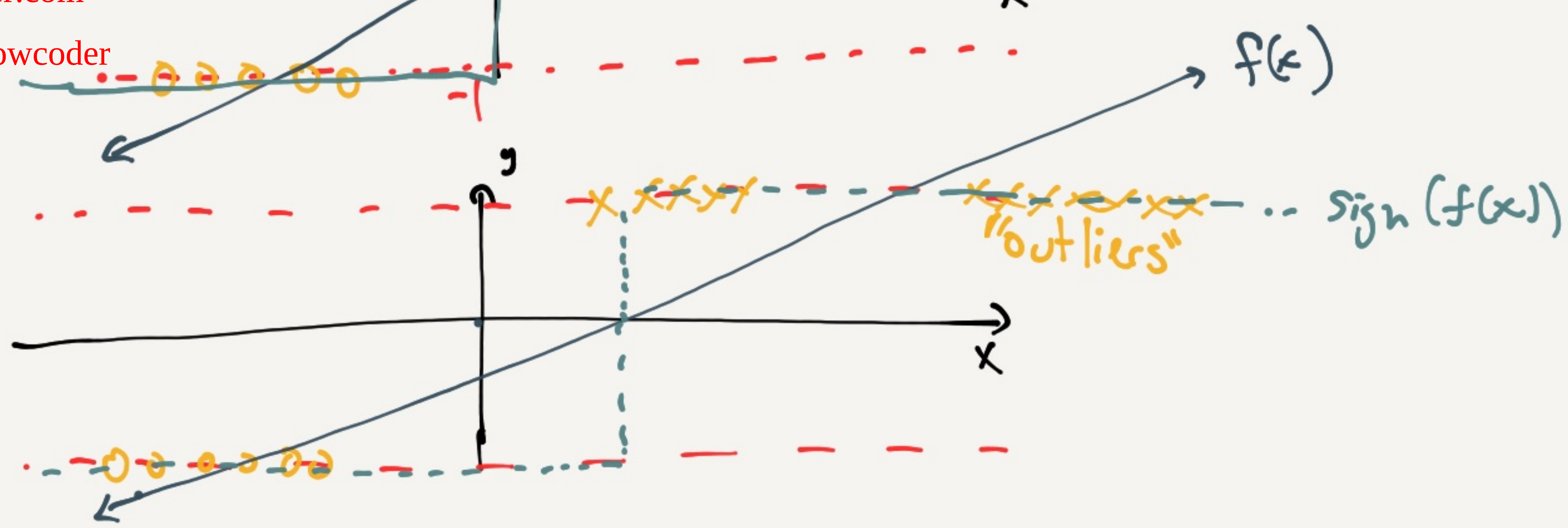
• Given $D = \{X, y\}$

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n \| w^T x_i - y_i \|^2$$ squared error

$$= \underset{w}{\operatorname{argmin}} \| X^T w - y \|^2$$

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





Note: not robust to "outliers"

- the squared penalizes predictions that are "too correct"

- FLD is a version of LSC (PS 7.7)

Perceptron (Rosenblatt 1962)

Perceptron criteria – only look at misclassified points.

$$E(w) = \sum_{i \in \mathcal{M}} -y_i w^T x_i \quad , \quad \mathcal{M} = \{\text{misclassified points}\} = \{ i \mid y_i w^T x_i < 0 \}$$

$E(w) = 0$ when all points 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 datapoint at a time

$$\Rightarrow w_{t+1} = w_t + \eta y_i x_i \quad , \text{ for some } i \in \mathcal{M}$$

rotate w to point towards the misclassified point ($y_i = 1$), & viceversa

rotation towards x_i

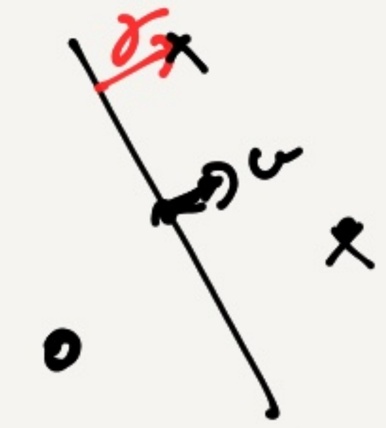
- How to set η the learning rate?

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

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

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

measurement of how "separable" the data is.



- it will not converge if the data is not linearly separable.

- many possible solutions, depending on the initialization.

equivalent w
$E(w) = 0$

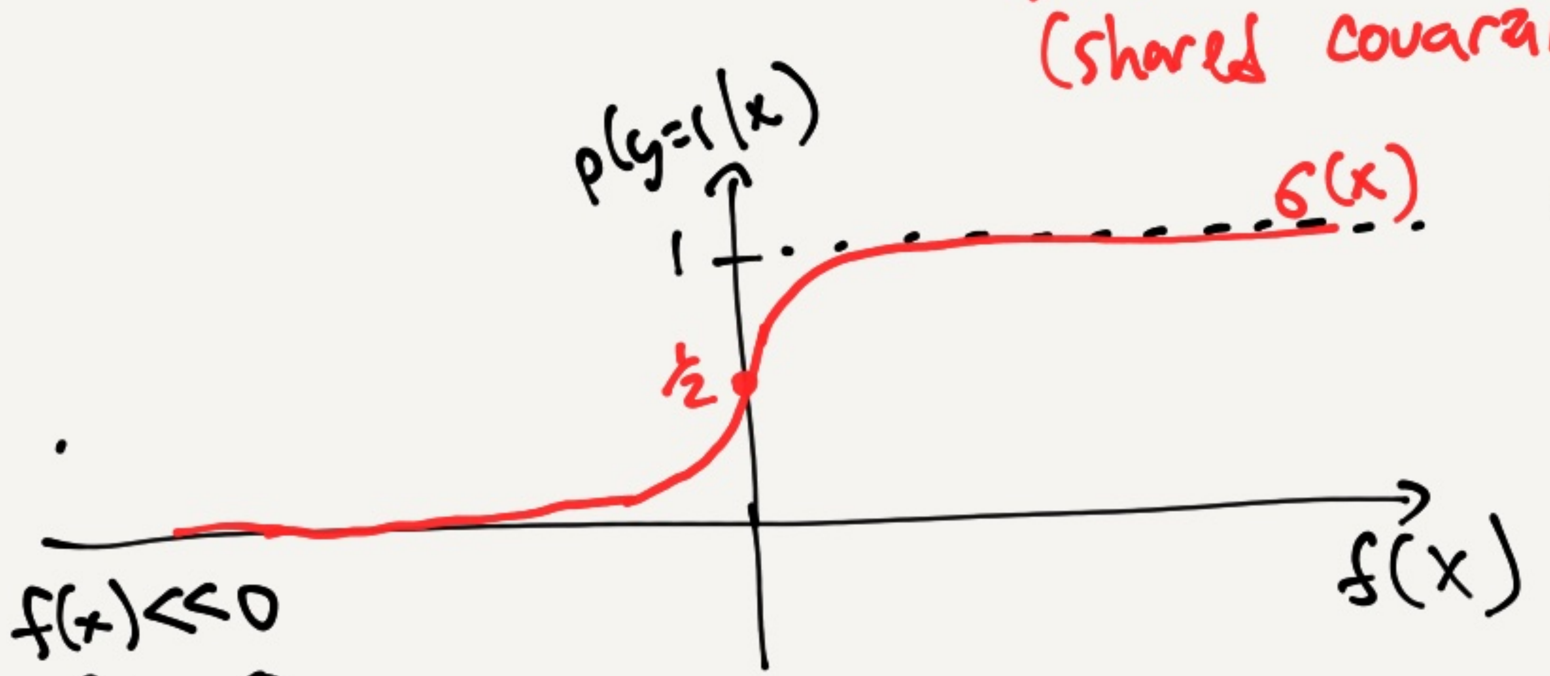
Logistic Regression (probabilistic approach)

- Consider 2-class binary problem: $y \in \{0, 1\}$
- From PS 6-7, if the CCDs are Gaussian $p(x|y) = N(x|\mu_y, \Sigma_y)$

then the posterior distribution $p(y|x)$ is a sigmoid fun.

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

$\underbrace{f(x)}$ is linear function if $\Sigma_y = \Sigma$ (shared covariance)



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

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

- with BDR, $f(x)$ is determined by the learned Gaussian parameters.
- Now, learn $f(x)$ directly.

Assume $f(x)$ is linear

$f(x) = w^T x$, $p(y=1|x) = \sigma(w^T x) = \pi$

Decision Rule:

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

Learning (2-class problem)

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

def $\pi_i = \sigma(w^T x_i)$ (probability of class 1 given x_i)

$$\underbrace{p(y_i|x_i, w)}_{\text{conditional prob. given } x_i \,\&\, w} = \pi_i^{y_i} (1-\pi_i)^{1-y_i}$$

BDR: Note we model $p(x|y)$

log-likelihood of data

$$\ell(w) = \sum_{i=1}^{n} \log p(y_i|x_i, w)$$

$$= \sum_i \left[y_i \log \pi_i + (1-y_i) \log(1-\pi_i) \right]$$

MLE

$$w^* = \underset{w}{\arg\max}\ \ell(w)$$

Error (loss) function

$$w^* = \underset{w}{\arg\min}\ -\ell(w)$$

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

• maximize $\ell(w)$

Apply Newton-Raphson method : (more in tutorial)

Iterate

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

⋮

$$\boxed{w^{(new)} = (XRX^T)^{-1} X R z}$$

← weighted least-squares using R, z, X

where

$$\begin{cases} R = \text{diag}\left(\pi_1(1-\pi_1), \ldots, \pi_N(1-\pi_N)\right) \\ z = \underbrace{X^T w^{(old)}}_{f^{old}(x)} - \underbrace{R^{-1}}_{\text{weight on each point}} \underbrace{(\pi - y)}_{\text{error btwn prediction \& class label}} \end{cases}$$

← depend current w & predicted π_i. higher weight to $\pi_i = \frac{1}{2}$

← target depends on w

" iterative reweighted least squares"

IRWLS / IRLS

Comparison of loss / error functions

All these methods optimize a similar form:

$$w^* = \underset{w}{\text{argmin}}\ E(w) = \underset{w}{\text{argmin}} \underbrace{\sum_i \overset{\text{loss function}}{L}(f(x_i), y_i)}_{\text{"empirical risk"}}$$

"empirical risk minimization" - all about training error.

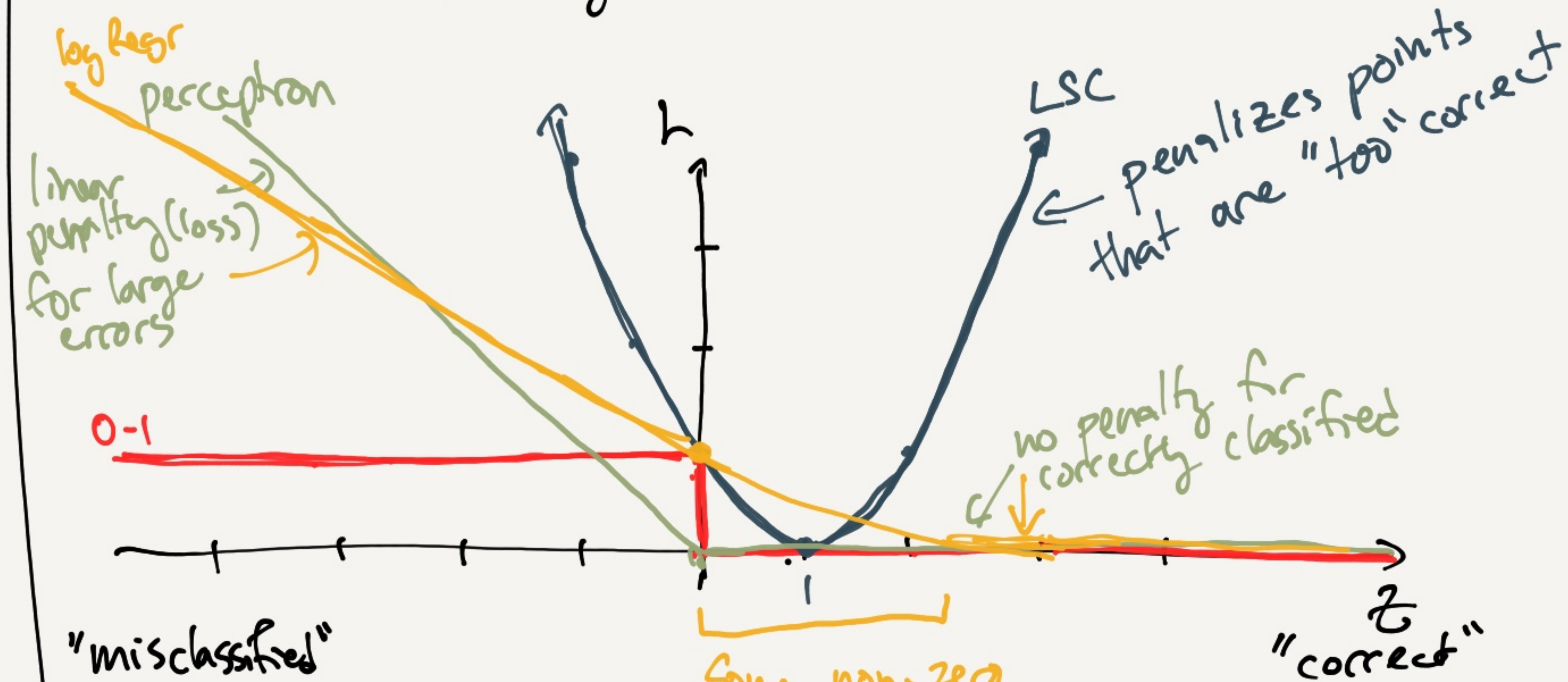
Let $z = y w^T x$

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

LSC: $L = (z-1)^2$

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

Log. Regr: $L = \log(1 + e^{-z}) \cdot \frac{1}{\log 2}$



Some non-zero loss for correctly classified points near the boundary.
⇒ push boundary away from these points