Logistics

- One TA: Owen Xie
- Office hours on course website

  www.cs.rpi.edu/~gittea/teaching/fall2020/m/andopt.htm

- We will Piazza for general discussions
- Grading breakdown:

  HWs (50%) - 11 hws

  Projects (35%) - more details in a couple of weeks

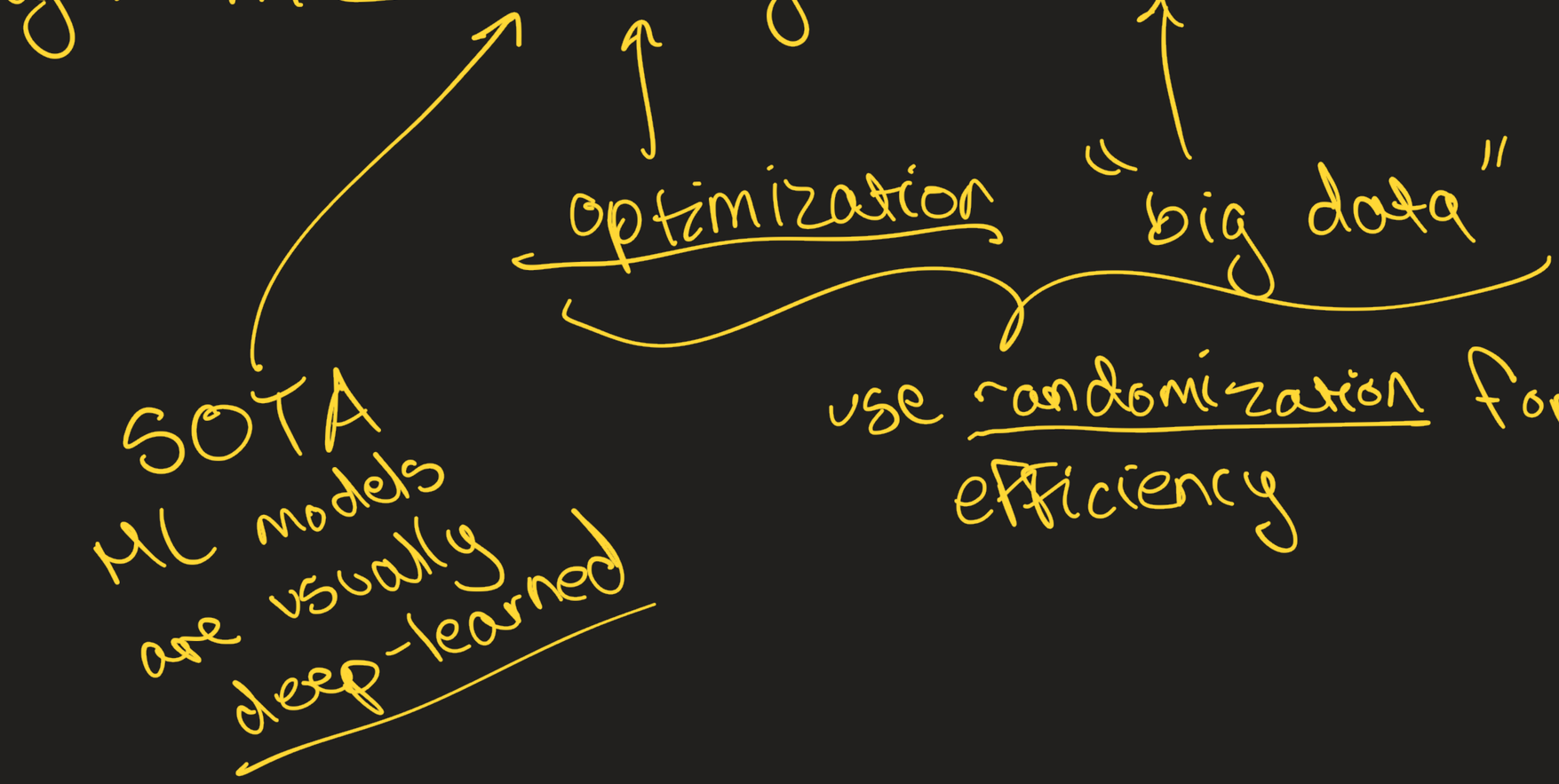
  Weekly Participation (15%)

  ≥ 14

What is this class?

A survey of topics that you may see separately (in more details) in ML, optimization, randomized algorithms, and Deep Learning courses.

Why? ML = learning from data



Lecture breakdown

- 4 basic probability theory & how to phrase ML problems as optimization problems (ERM framework)
- 9 general techniques for solving these optimization problems (convex optimization)
- 5 randomized algorithms for some specific ML applications
- 9 deep learning (specific problem classes & architectures)
- ~1 topics we didn't cover (teaser & motivation to research)

Resources

qual online

- Introduction to Applied Linear Algebra. Boyd & Vandenberghe
- Deep Learning w/ Python. Chollet.

Examples

unsupervised :
- k-means clustering
- spectral clustering
- k-nn
- language modeling

supervised :
- svm
- image recognition
- perceptron
- naive Bayes

} classification

- linear regression
- logistic regression

At a high-level ML consist of:

- formulate a task
* — collect appropriate data
- choosing an appropriate model for your prob & data
- formulate an appropriate optimization problem to fit that model

*** — solving the opt prob, potentially after modifying it to make it more tractable

Ex k-nearest neighbors (for 2-class classification problem)
k-nn

Data Given $X = \{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^d$
are features and $y_i \in \{+1, -1\}$

Prob We want to learn y as a function of x:
$y = f(x)$

Think: we want to determine whether a person will default on their loan.

Model we assume that inputs that share similar features should fall in similar classes

let $\mathcal{N}_k(x)$ = k nearest neighbors of x in our training data

predict

$$y = f(x) = \text{sign}\left(\sum_{x_i \in \mathcal{N}_k(x)} y_i\right)$$

- Optimization

Given a new point $x \in \mathbb{R}^d$, we need to find $\mathcal{N}_k(x)$, then sum the corresponding y_i and take the sign

What is the cost of this operation?

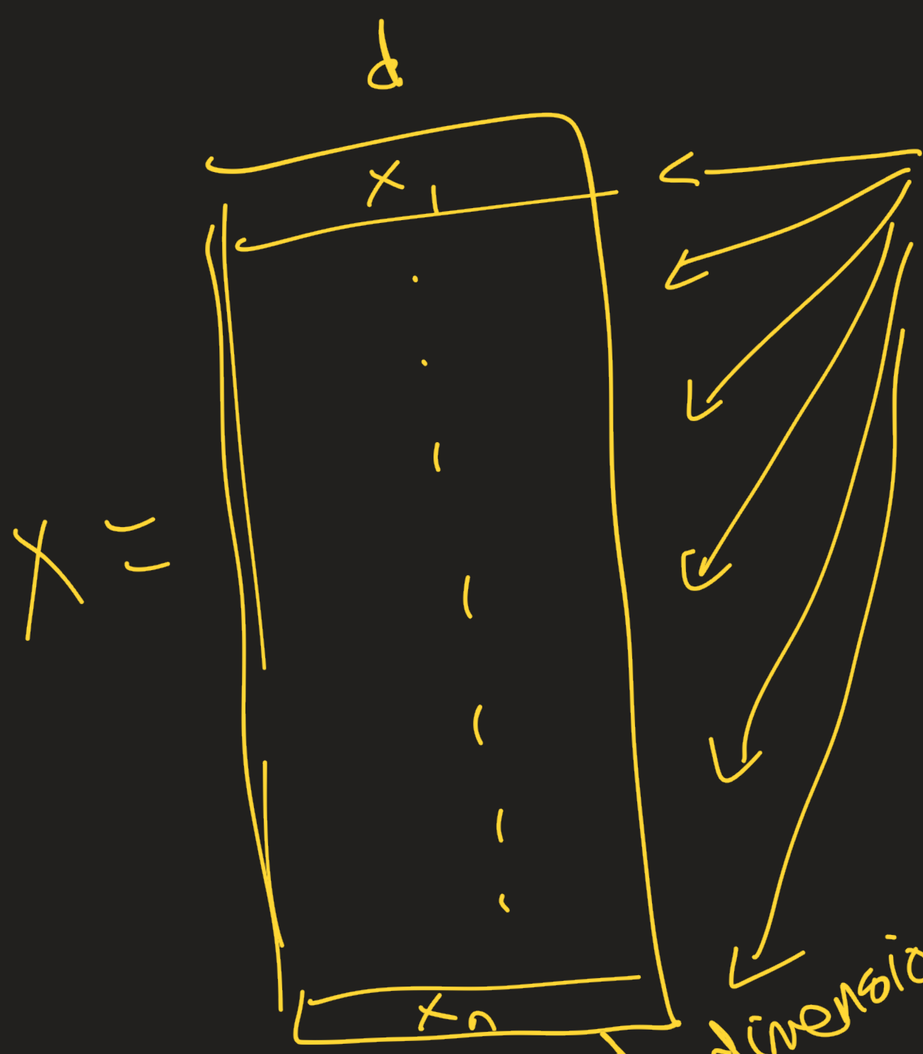
$y = \text{sign}(1 + 1) = 1$

$k = 2$

$y = \text{sign}(-1 - 1)$

$= -1$

$$\text{sign}(x) = \begin{cases} 1 \text{ if } x > 0 \\ -1 \text{ if } x < 0 \\ 0 \text{ if } x = 0 \end{cases}$$



determining $\|x_i - x\|_2^2$

$$= \sum_{j=1}^{d} ((x_i)_j - x_j)^2$$

is $O(d)$

dimensions do this for n points

$\Rightarrow O(nd)$ to compute distances

$+ O(n \log n)$ to find k smallest distances

$+ O(k)$ to sum and take the sign

$\Rightarrow$ Cost of K-nn in d dimensions

$O(nd + n \log n + K)$

When n & d are too large (e.g. $n \sim O(10^6)$ and $d \sim O(10^3) \Rightarrow nd \sim O(10^9)$)

this is too expensive to use directly —

How to reduce the cost? We use <u>approximate nearest neighbors</u>: project the points from $\mathbb{R}^d$ down to a lower dimensional space, $\mathbb{R}^c$ and find the nearest neighbors there.

In practice: fix a matrix $P \in \mathbb{R}^{c \times d}$

then compare $\|\underbrace{Px_i}_{\in \mathbb{R}^c} - Px\|_2^2$ instead of $\|x_i - x\|_2^2$ ($x_i, x \in \mathbb{R}^d$)

This is only useful if P is such that

$$\|P(x_i - x)\|_2^2 \approx \|x_i - x\|_2^2$$

and can be found efficiently, then

$\mathcal{N}_K(x)$ is preserved

Algorithm 1) precompute P

2) precompute and store $\{Px_i\}_{i=1}^n$ — offline

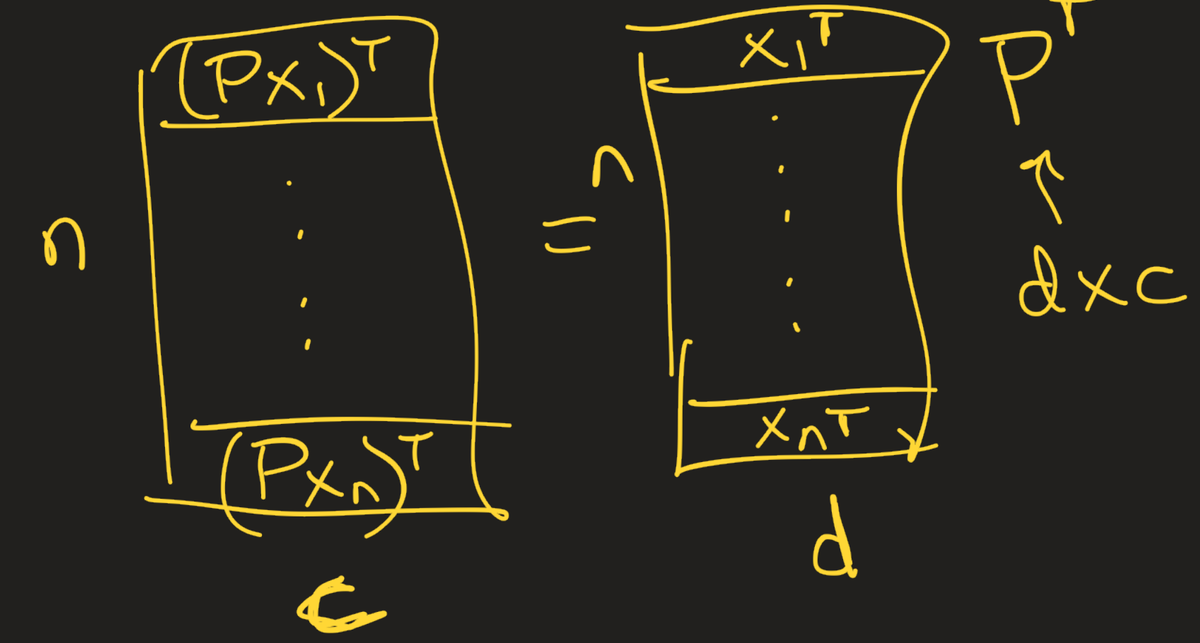
3) given a new x, compute Px

and use k-nn on $\{Px_i\}_{i=1}^n$ and Px

Costs

offline costs: $O(dc + ndc)$

computing P



online cost: $O(dc + nc + n\log n + K)$

computing Px

Takeaway:

k-nn: $O(nd)$

K-nearest approx neighbors: $O(nc + dc)$

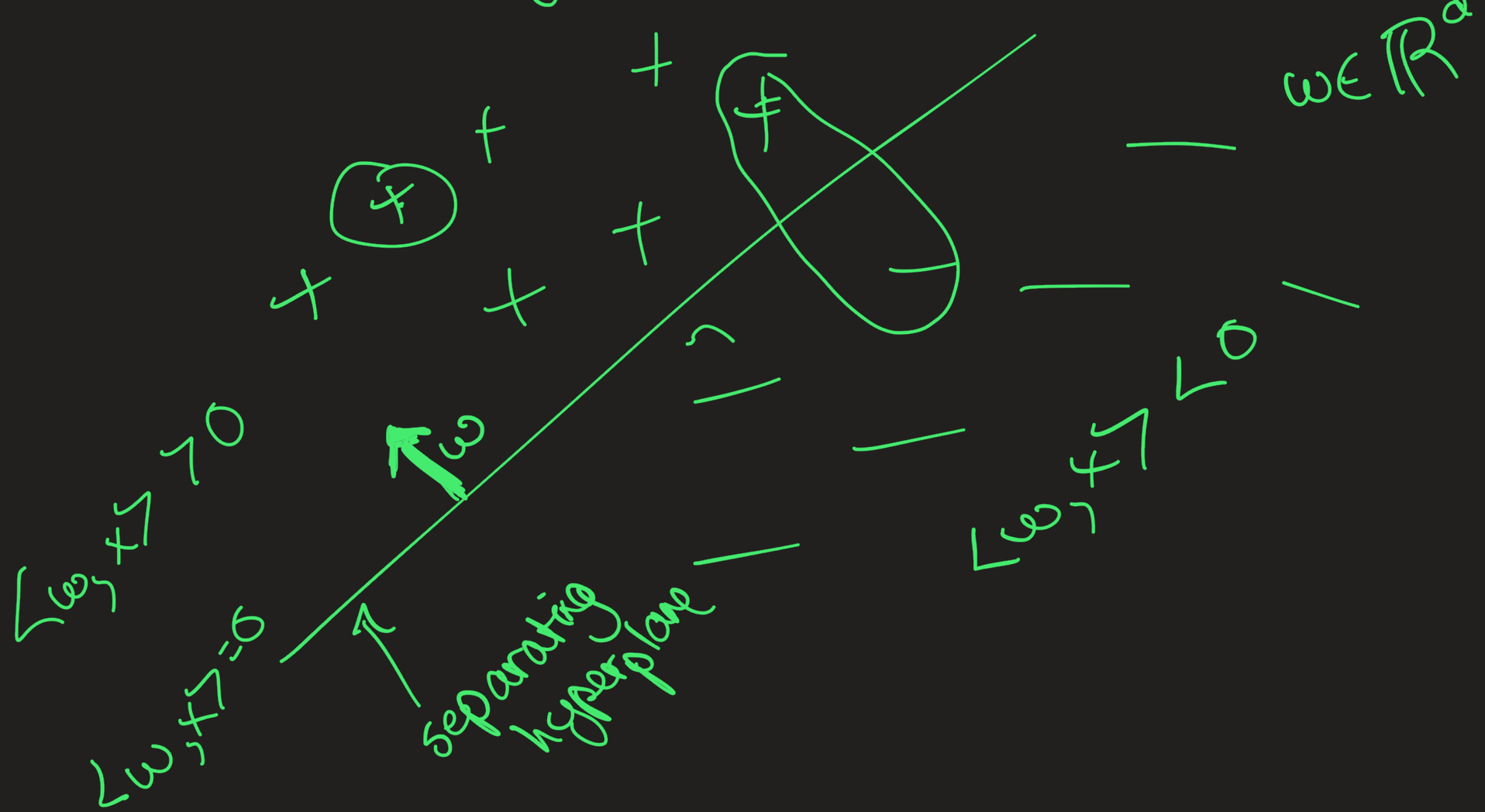
$n = O(10^6)$

$d = O(10^3)$

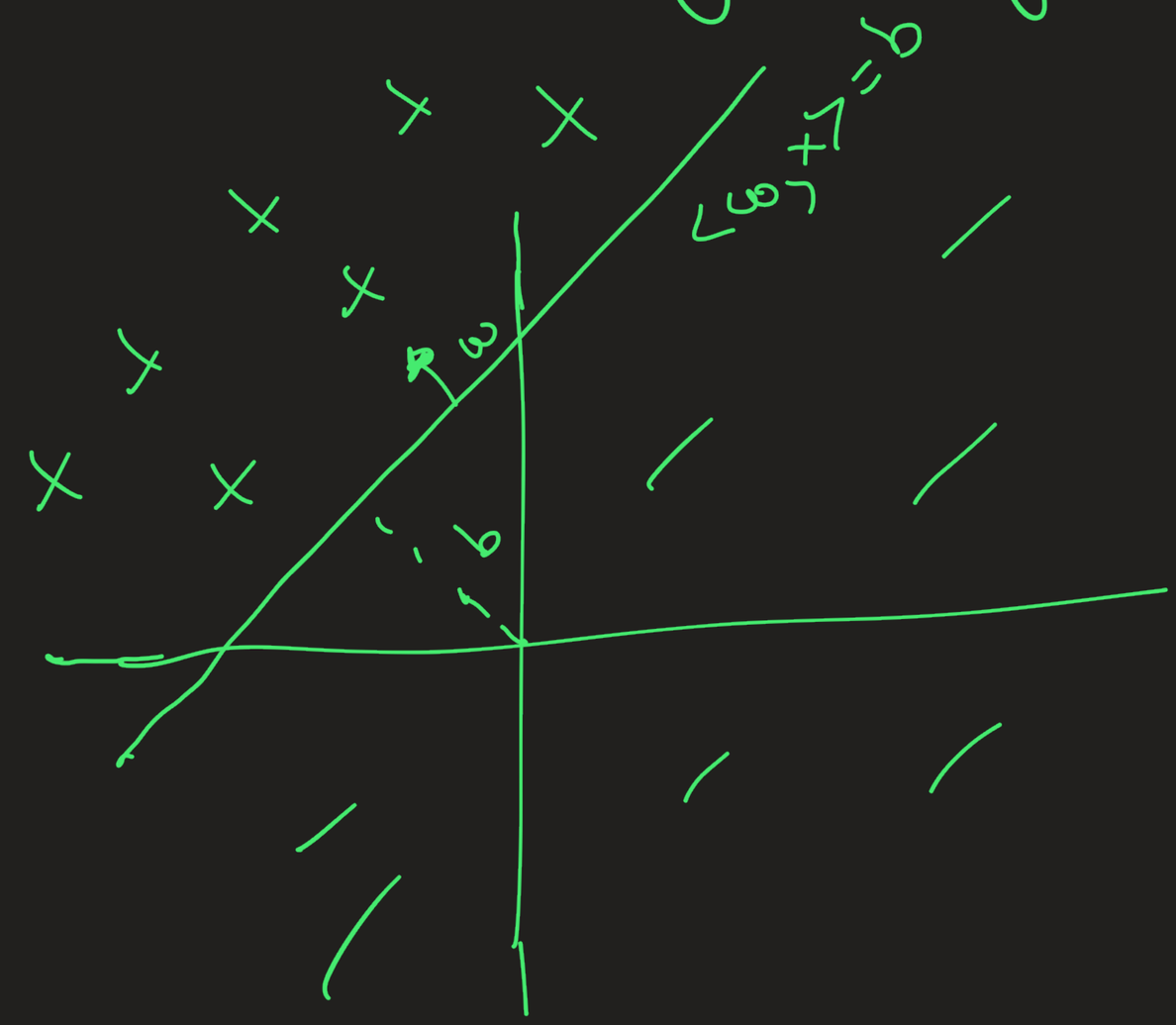
$c = O(10)$

Ex Binary classification via SVM

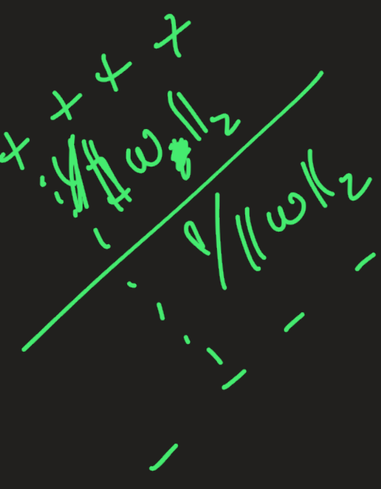
Goal: given positive & negative examples learn a linear separation boundary so that all/most positive examples fall on one side and all/most negative examples fall on the other



model: $g \approx \text{sign}(\langle \underline{w}, x \rangle + b)$



Optimization Problem



- we penalize the current w if

$$y_i \langle w, x_i \rangle < 0$$

for any training pair (x_i, y_i)

- in fact, we want the model to be confident

$$y_i \langle w, x_i \rangle > 1$$

for all training pair (x_i, y_i)

- to avoid simply achieving confidence by letting $\|w\|_2 \to \infty$ (this is an unstable model!) we want $\|w\|_2$ to be small

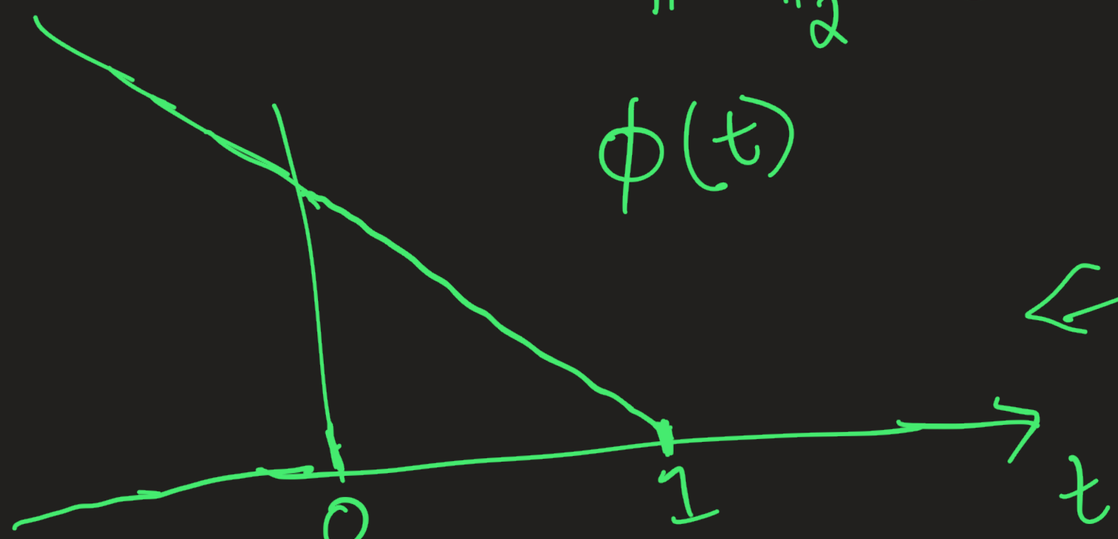
This leads to two possible optimization problems

$$w_* = \underset{\substack{w \in \mathbb{R}^d \\ \|w\|_2 \le r}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max\{1 - y_i \langle w, x_i \rangle, 0\}}_{\text{called the hinge loss}}$$

$$\phi(y_i \langle w, x_i \rangle)$$

where

$$\phi(z) = \max\{1 - t, 0\}$$



(constrained optimization)

(penalized optimization)

$$w_* = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \phi(y_i \langle w, x_i \rangle) + \frac{\lambda}{2} \|w\|_2^2}$$

In ML we prefer penalized optimization