ML and Optimization Lecture 2

- Challenges of optimization
- What is (supervised) Machine Learning
- Ex: Support Vector Machines for binary classification

Challenges of Optimization

$\theta^* \in \operatorname{argmin}_{\theta \in \Theta} J(\theta)$

constrained optimization

$\theta^* \in \operatorname{argmin} J(\theta)$

unconstrained optimization

- dealing w/ constraints (ex: OLS vs NNLS : Lec 1)
- ambiguity in matching the task w/ a specific objective function (ex: $\|\cdot\|_2^2$ or $\|\cdot\|_1$ for linear regression)
- these problems are extremely difficult to solve or even check a solution for optimality when there is not a specific structure we can exploit (ex: sparse linear regression)
- scale of the problem (ex: large-scale linear regression)

What is Machine Learning?

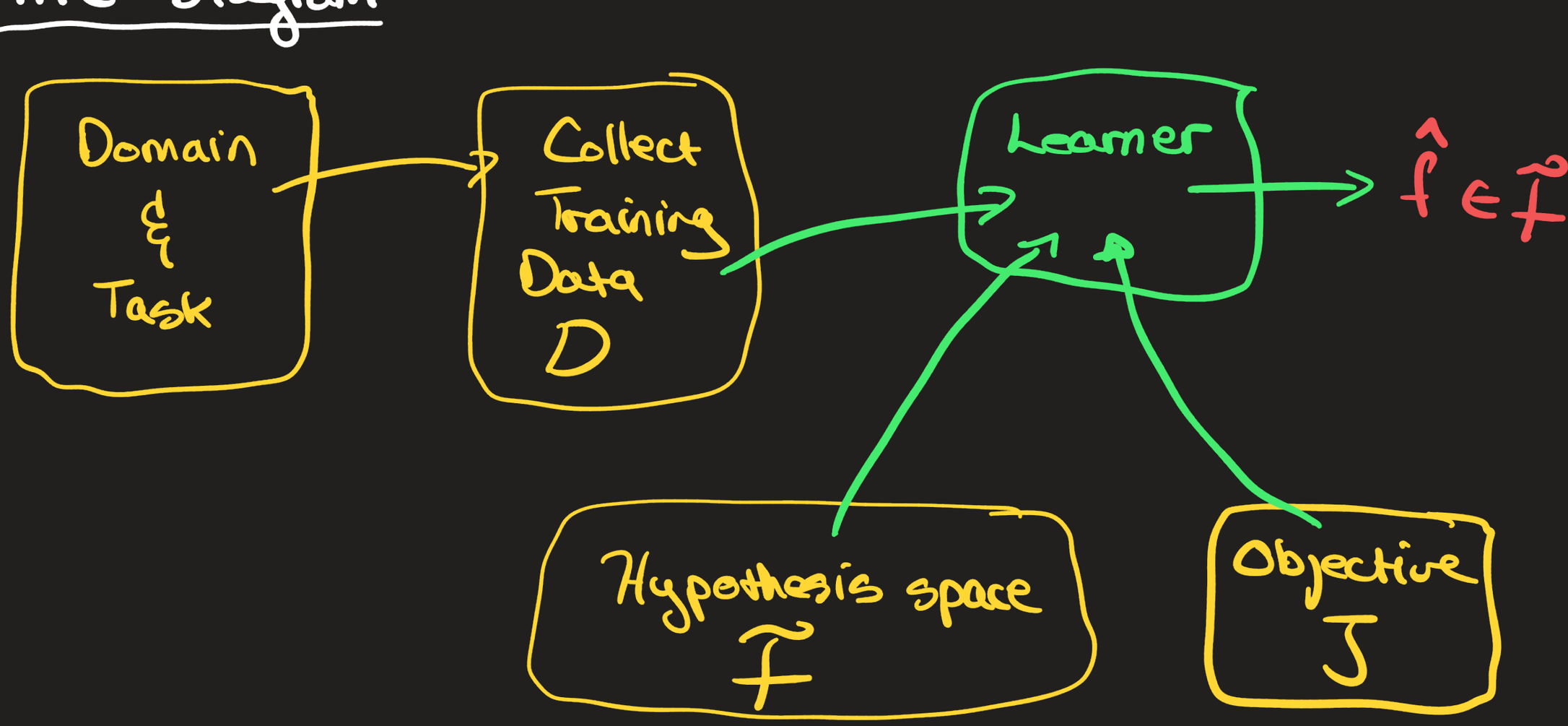
"Finding a predictable pattern from a set of data"

- predictable: accurate in the presence of randomness
- pattern: $f(x) = y$
- set of data: randomly selected observations of the form (x, y)

Assumptions that make ML possible:

1) there is an accurate function
2) that function can be learned from data

ML Diagram


Domain & Task
Collect Training Data D
Learner
$\hat{f} \in \tilde{f}$
Hypothesis space $\tilde{f}$
Objective J

Ex: Digit recognition (MNIST)

Given $x \in \mathbb{R}^{28\times 28} \sim \mathbb{R}^d$ where $d = 28\times 28$ grayscale image of digits 0–9

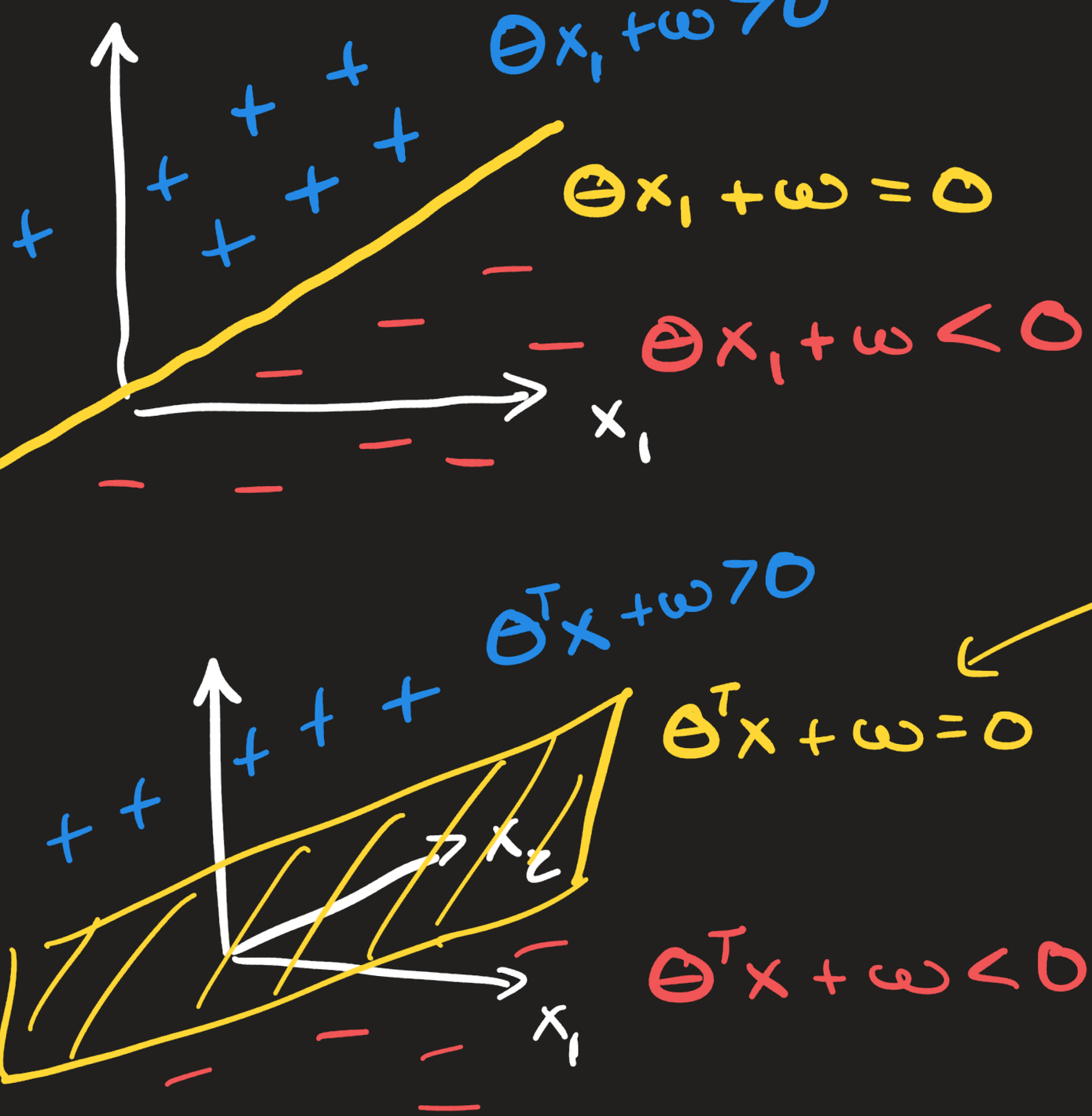
Task: predict which digit is in the image

Simplify for today to predicting if is 0 (class 1) or not (class -1)

Hypothesize that it suffices to consider thresholded affine combination so the pixels in x

$$\mathcal{F} = \{ f \mid f(x) = \text{sign}(\theta^T x + \omega) \text{ for } \theta \in \mathbb{R}^d \text{ and } \omega \in \mathbb{R} \}$$

Geometrically, this is equivalent to assuming linear separability of the positive and negative classes



This is an ASSUMPTION and its probably not the best one

decision surface OR separating hyperplane

How to measure how good a model is on the training data?

Maybe we can choose θ, ω to maximize accuracy

$$\hat{\theta}, \hat{\omega} = \text{argmax } J(\theta, \omega) = \frac{1}{n} \sum_{i=1}^{n} \text{sign}(\theta^T x_i + \omega) y_i$$

But optimizing 0-1 accuracy is difficult:
J is discontinuous w.r.t. θ and ω

We want a nice smooth objective: Idea: penalize models that make predictions close to the decision surface. Notice that if $\text{sign}(\theta^T x_i + \omega) = y_i$ and $|\theta^T x_i + \omega|$ is large then the model is confident

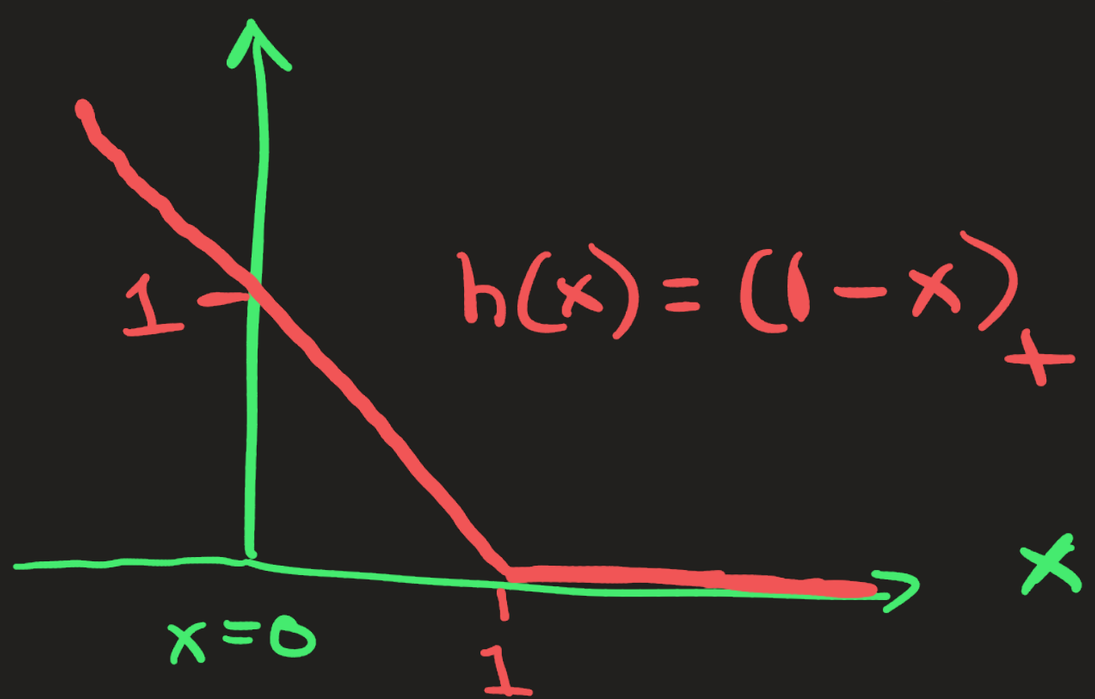
Notice that $|\theta^T x_i + \omega| = (\theta^T x_i + \omega) y_i$ when the model is correct

Take

$$J(\theta, \omega) = \frac{1}{n} \sum_{i=1}^{n} h\left((\theta^T x_i + \omega) y_i\right)$$

where h is the "hinge loss"

$$h(x) = (1-x)_+ = \begin{cases} 0 & \text{if } x > 1 \\ 1-x & \text{otherwise} \end{cases}$$



With this choice we choose our best model for binary classification by solving

$$(*) \quad \hat{\theta}, \hat{\omega} = \underset{\theta, \omega}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} (1 - (\theta^T x_i + \omega) y_i)_+$$

This objective is continuous and a.e. differentiable

One last issue to address in our choice of objective: if $\alpha > 0$, then the two models (θ, ω) and $(\alpha\theta, \alpha\omega)$ define the same decision surface:

$$\theta^T x + \omega = 0 \iff (\alpha\theta)^T x + \alpha\omega = 0$$

so $(*)$ is ill-posed: there are infinitely many equally well-performing models

We break this indeterminacy by using regularization

so our final objective is

$$\hat{\theta}, \hat{\omega} = \underset{\theta, \omega}{\operatorname{argmin}} \underbrace{\frac{1}{n}\sum_{i=1}^{n}(1-(\theta^T x_i+\omega)y_i)_+}_{\text{data-fitting term}} + \underbrace{\lambda\|\theta\|_2^2}_{\substack{\text{regularization} \\ \text{term}}}$$

Here $\lambda > 0$ is called the regularization parameter and $\|\theta\|_2^2$ is called ℓ_2-regularizer or weight decay

By varying λ (a hyperparameter) you can prioritize fitting the data more or having a less complex models

This general class of models for binary classification w/ targets in ± 1 is called Support Vector Machines (SVM)

$$\mathcal{F}_{\text{SVM}} = \{f \mid f(x) = \text{sign}(\theta^T x + \omega)\}$$

This particular objective is called soft-margin SVM.

$$J(\theta, \omega) = \frac{1}{n} \sum_{i=1}^{n} (1 - (\theta^T x_i + \omega) y_i)_+$$

Generalize to supervised ML

ℓ — loss function $\ell(\hat{y}, y) \in \mathbb{R}$ measures the loss in predicting $\hat{y}$ when the true value is y

$f(x)$ — prediction of a candidate model f for input x

$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{\hat{}}$ — training data set

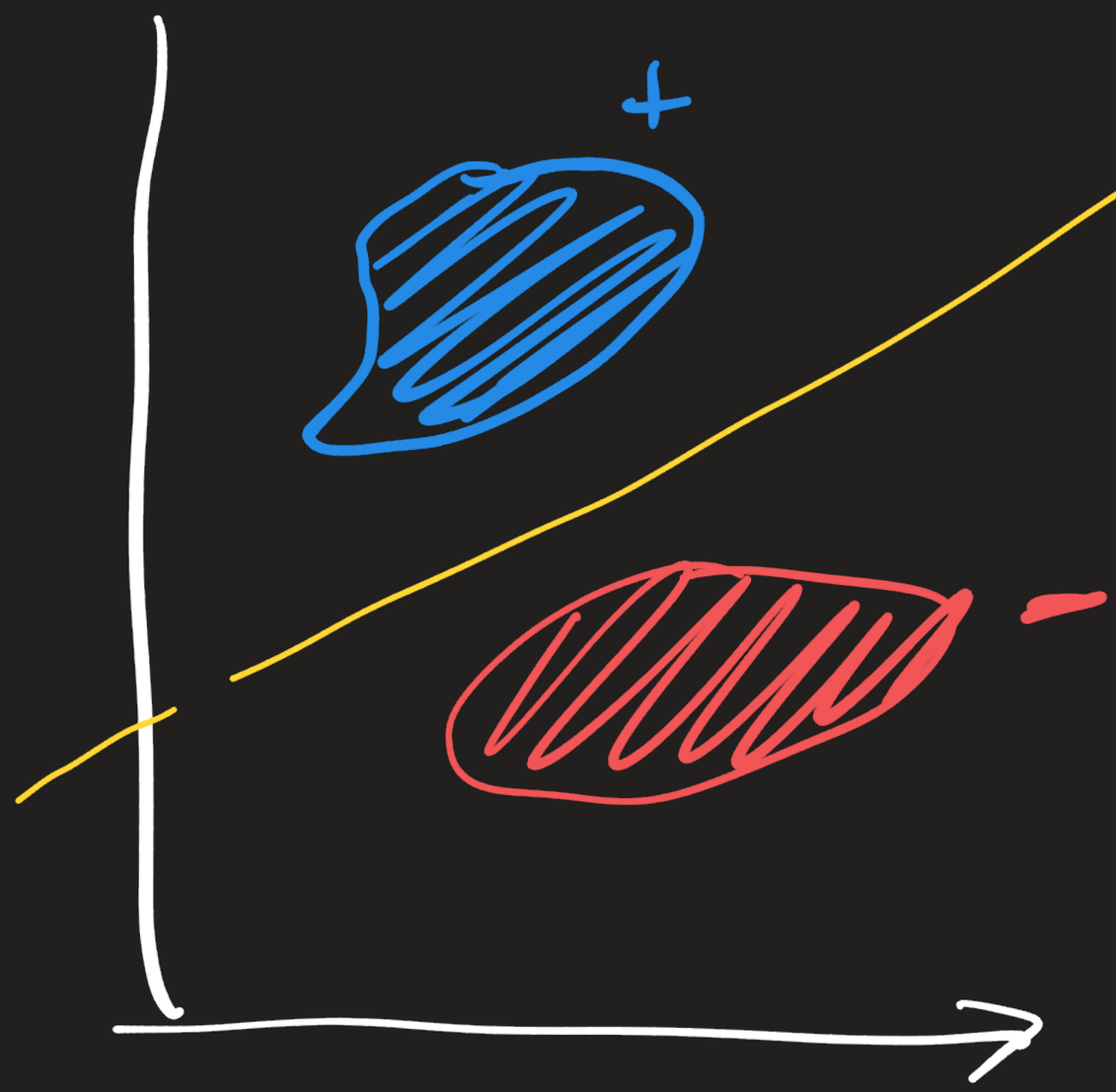
$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \underbrace{\mathbb{E}_{(x,y)\sim \mathbb{P}} \ell(f(x), y)}_{\text{population risk}}$$

In this class we use <u>Empirical Risk minimization.</u>

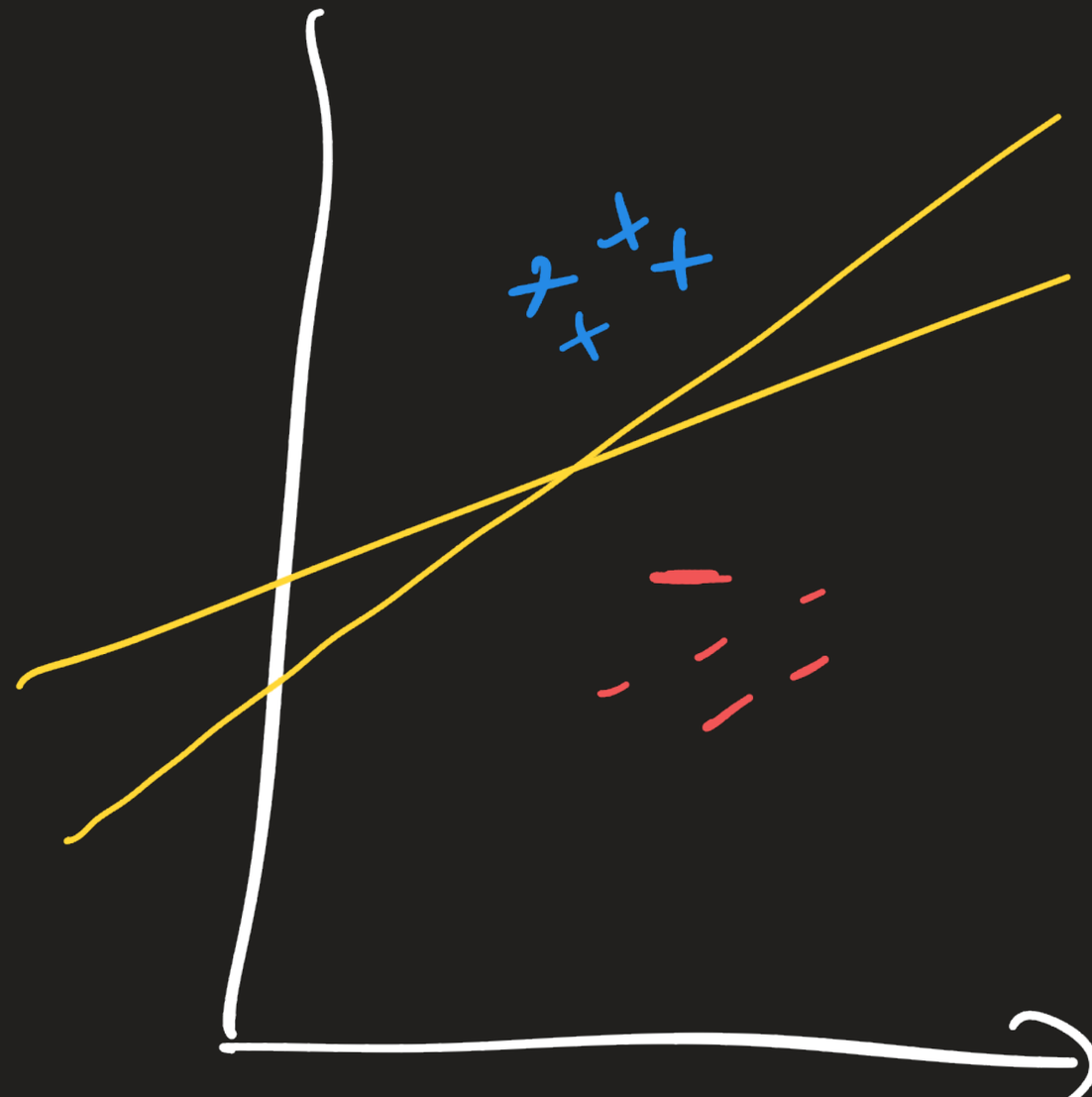
- Assume $\mathbb{E}_{(x,y)\sim P}\, \ell(f(x), y) \approx \frac{1}{n} \sum_{(x_i,y_i)\in D} \ell(f(x_i), y_i)$

and solve

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)}_{\text{empirical risk}}$$



population risk minimization



empirical risk minimization

High-level overview of ML

- formulate a task & pick a domain ($\mathbb{P}$)
- collect appropriate training data for ERM
  ($\mathbb{D}$ i.i.d. samples from $\mathbb{P}$)
- choose a good model class for the problem
  (SVM, logistic regression, multiclass LR,
  linear regression, Kernel versions of these, DL)
- pick a loss function appropriate to the task
- solve the ERM optimization problem