Lecture 1

- Logistics
- What is Optimization?
- What is Machine Learning (ML)?
- Examples of ML formulated as optimization problems
what is Optimization?

$\chi$ - parameter space; e.g. a model space for ML

$\overline{J}: \chi \rightarrow \mathbb{R}$ : objective

Goal:

$x^* = \arg\min \ J(x) \quad x^* \text{ - best \solution}$

$x \in \chi$

Because we can equivalently find $x^*$ as the solution to a maximization problem

$x^* = \arg\max \ -J(x) \quad x \in \chi$

the difficulty of maximization & minimization in general is the same.
Challenges of optimization

- dealing with constraints
- ambiguity in matching the task w/ a specific objective function
- these problems are extremely difficult when there isn’t a specific structure we can exploit
- scale of the problem
**What is ML?**

"Finding a predictable pattern from a set of data."

- **Domain & Task**
- **Training data**
- **Learner (Opt alg)**
  - Hypothesis space $\mathcal{F}$
  - Objective $J$

**Observations used to learn $f$**

$\text{accurate } f(x) = y$
What does it mean for \( f \) to be optimal?

- **loss function** \( l(\cdot, \cdot) \)
- \( f(x) \) — prediction of a candidate model \( f \) for input \( x \)
- \( y \) — true label for that input

\( l \) — quantify the mismatch b/w the prediction & the ground truth

\[
\begin{align*}
\text{risk} & \quad f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(x, y) \sim P} l(f(x), y) \\
\text{error} & \quad \hat{\omega} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)
\end{align*}
\]

Called the population risk

Called the empirical risk

where \( (x_i, y_i) \) for \( i = 1, \ldots, n \) are training pairs, sampled i.i.d. from \( P \)
High-level overview of ML

- formulate a task & pick a domain

- collect appropriate data

- choose an appropriate model class for your problem

- pick a loss function and formulate our optim prob to find $f^*$

- solve the optim prob

(setup for classical ML that focuses only on accuracy)
Ex Binary classification using SVMs (support vector machines)

Assumption: the two classes are linearly separable

\[ f = \frac{1}{2} \mathbf{f}: f(x) = \text{sign}(\mathbf{w}^T x + b) \quad \text{for} \quad \mathbf{w} \in \mathbb{R}^d \text{ and } b \in \mathbb{R} \]

To design our loss for fitting an SVM:

- Penalize our model if \( y_i \) and \( \mathbf{w}^T x_i + b \) have different signs (means \( \ell(\mathbf{w}^T x_i + b, y_i) \) should be high)

- Penalize the model if \( y_i (\mathbf{w}^T x_i + b) < 1 \) (we want the model to predict a large number \( \mathbf{w}/ \) the correct sign)
- avoid trivially confident models
  (do not let $\|\omega\|_2^2 \to \infty$)

let $\phi(t) = \max (1-t, 0)$

Take $l(f(x_i), y_i) = \phi(y_i (\langle \omega, x_i \rangle + b))$

Two setups for the optimization problem of learning an SVM using hinge loss:

1) $\omega^*, b^* = \operatorname{argmin}_{\omega, b} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i (\langle \omega, x_i \rangle + b))$
   $\omega, b : \|\omega\|_2 \leq C$

2) $\omega^*, b^* = \operatorname{argmin}_{\omega, b} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i (\langle \omega, x_i \rangle + b)) + \lambda \|\omega\|_2^2$
Ex. Ordinary Least Squares Regression

Given a continuous target \( y \in \mathbb{R} \), the task is to predict \( y \) as a linear combination of the features \( x \in \mathbb{R}^d \).

E.g. \( y = \text{life expectancy} \)
\( x = \text{vector of demographic descriptors} \)

Assume that there are true coefficients \( \beta \in \mathbb{R}^d \) so that
\[
y = \langle x, \beta \rangle + b = \langle [x, 1], [\beta, b] \rangle
\]
\[
= \langle x_{\text{aug}}, \beta_{\text{aug}} \rangle \quad \text{(wLOG, assume we have augmented linear models w/ a bias term)}
\]

By augmenting we simplify to
\[
y = \langle x, \beta \rangle
\]
But our training data is corrupted by noise

\[ y = \beta x \]

\[ y_i = \langle \beta^*, x_i \rangle + \varepsilon_i \]

for \( i = 1, \ldots, n \) where

\[ \varepsilon \sim N(0, \frac{1}{n}) \]

Q: how can we recover \( \beta \)?

Alg for computing \( \beta_{ols} \approx \beta^* \)

\[ \beta_{ols} = \argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} (\langle x_i, \beta \rangle - y_i)^2 \]
Algorithm for computing $\beta_{\text{OLS}}$ (Ordinary Least Squares)

Notice

$$\frac{1}{n} \sum_{i=1}^{n} (\langle x_i, \beta \rangle - y_i)^2 = \frac{1}{n} \| y - X\beta \|_2^2$$

where

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

and note $\beta$ minimizes the objective $J(\beta) = \frac{1}{n} \| y - X\beta \|_2^2$

iff

$$\nabla_{\beta} J(\beta) = 0 = \frac{1}{n} X^T (X\beta - y) = 0$$

so we have

$$X^T (X\beta_{\text{OLS}} - y) = 0 \quad \text{characterizes } \beta_{\text{OLS}}$$

$$\Rightarrow \beta_{\text{OLS}} = (X^TX)^{-1} X^Ty$$