Machine Learning

- What is learning, understanding, attention, experience?
- How do we make technology achieve that?
- Math models? Based on hid. probability.

01. Unsupervised learning - from our experience (infant). Supervised - from a teacher.

Examples we have seen before:

(a) Unsupervised learning: clustering

\[ \text{Unlabeled data} \quad X_1, ..., X_n \in \mathbb{R}^d \]

- e.g. n cells, d genes

\( \Rightarrow \) two types of cells

(b) Supervised learning: classification

\[ \text{Labeled data} \quad (X_1, Y_1), ..., (X_n, Y_n) \in \mathbb{R}^d \times \{0,1\} \]

- e.g. n people, symptoms cancer/no "Training data"

Want to build an "oracle"

That makes a diagnosis

for a new patient: \( X_{n+1} \rightarrow Y_{n+1} \)
Supervised ML: a general framework

- A pair of random variables (or vectors) \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\) as above. Objective reality.

Ex. \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\) as above. Objective reality.

- The joint distribution of \((X, Y)\) is unknown. We only see:

  - Training data \((X_1, Y_1), ..., (X_n, Y_n)\): iid copies of \((X, Y)\).
  - Goal: predict \(Y\) from \(X\) as best as we can.

\(\Rightarrow\) We want to construct an oracle

\[ h: X \rightarrow Y: \quad h(x) \approx Y \quad (\star) \]

to make predictions for new, unseen data: \(h(X_{n+1}) = Y_{n+1}\)

How do we quantify the "goodness of fit" (\(\star\))?

- We fix a loss function \(l: \mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}\), eg. \(l(t-s) = (t-s)^2\), and define the risk (a.k.a. test error)

\[ R(h) := \mathbb{E} l(h(x), Y) = \mathbb{E} (h(X_{n+1}), Y_{n+1}) \]

Examples:

(a) quadratic loss \(l(t,s) = (t-s)^2\) \(\Rightarrow R(h) = \mathbb{E} (h(x)-Y)^2\)

(b) logistic loss

(c) hinge loss (svm)
How do we construct an oracle \( h \)?

1. **\( h \) too complex:** overfitting

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

2. **\( h \) too simple:** underfitting

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

3. **\( h \) is OK:** good fit

\[
\begin{array}{cccccccc}
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

**Our Strategy:**

1. Fix some collection of functions \( \mathcal{H} \), called a **hypothesis class**.
2. Select \( h \in \mathcal{H} \) that best fits the training data.

**Examples:**

- (a) \( \mathcal{H} = \{ \text{all functions } h : X \to Y \} \) \( \Rightarrow \) overfitting (1)
- (b) All linear functions: \( \mathcal{H} = \{ h(x) = \langle w, x \rangle + b : w \in \mathbb{R}^d, b \in \mathbb{R} \} \)
  \( \Rightarrow \) linear regression
- (c) \( \mathcal{H} = \{ h(x) = \text{sign} (\langle w, x \rangle + b) : w \in \mathbb{R}^d, b \in \mathbb{R} \} \) \( \Rightarrow \) SVM (2)
- (d) \( \mathcal{H} = \{ \text{all polynomials } p(x) \text{ of degree } \leq 2 \} \) (3)
- (e) \( \mathcal{H} = \{ h_1, h_2 \} \) \( \Rightarrow \) hypothesis testing

No systematic way to choose \( \mathcal{H} \). ("Model selection")
• The best \( h \in \mathcal{H} \) is the one that minimizes the risk

\[
R(h) = \mathbb{E} \ell(h(x), y).
\]

\[h^* = \arg\min_{h \in \mathcal{H}} R(h).
\]

• But \( R(h) \) can’t be computed (can’t take \( \mathbb{E} \) over the population)

empirical risk (a.k.a. training error)

\[
R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i).
\]

\[h_n^* = \arg\max_{h \in \mathcal{H}} R_n(h)
\]

\( \downarrow \) can be computed from training data \( \uparrow \) (can be NP-hard)

\[
\text{Ex (Binary classification) quadratic loss } \Rightarrow \quad R_n(h) = \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 = \% \text{ of misclassified training data}.
\]

\[
\begin{cases}
1 & \text{if } h(x_i) \neq y_i \\
0 & \text{otherwise}
\end{cases}
\]

Empirical Risk Minimization (ERM) Algorithm

1. **Training**: for input data \((x_1, y_1), \ldots, (x_n, y_n)\); compute \( h_n^* \).

2. **Prediction**: on query \( X \), output \( h_n^*(X) \) "oracle"

How do we measure the quality?
• Generalization error := $R(h_n^*)$

measures how well the algorithm generalizes to unseen data.

• Examples:

(a) $h = \{\text{all functions}\}$, $Y = f(x)$.

$\exists$ a perfect fit to the training data: $h_n^*(x) = \{ Y_i \text{ if } x = x_i \}$

training error $R_n(h_n^*) = 0$. (Overfitting)

BUT does NOT generalize well:

$R(h_n^*)$ is large. Memorized, not generalizes.

(b) $h = \{\text{all linear functions}\}$, quadratic loss $\Rightarrow$

$w_n^* = \arg\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} (\langle w, x_i \rangle + b - Y_i)^2$

= linear regression. OK.

Our goal: bound the generalization error. How does it depend on the "complexity" of $h$?