

07-280: ML Problem Formulation

Nihar B. Shah, CMU

Historical Context

Long time ago: **Rule-based AI**.

- E.g., AT&T identifying faults in telephone lines based on logs.
- OK for some applications. Not very well otherwise, e.g., how to describe a picture in pixel space?

Instead, **Machine Learning**: Rely primarily on examples.

Input lies in some set $\mathcal{X}$, and output (labeled) in some set $\mathcal{Y}$.

Goal: Come up with a function $\mathcal{X} \rightarrow \mathcal{Y}$, say $h : \mathcal{X} \rightarrow \mathcal{Y}$, such that for a new input $x \in \mathcal{X}$, whose true label is y , we have $h(x) = y$.

We are given examples: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})$

These are called **training data / samples / datapoints**. Denoted as $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$.

- N = sample size
- $x^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y} \forall i \in \{1, 2, \dots, N\}$

Use these to come up with function $h : \mathcal{X} \rightarrow \mathcal{Y}$.

(“Training” the model)

Two key questions:

(Q1) How do we come up with h ?

(Q2) How do we measure its performance?

Broadly two types of tasks:

- **Classification:** $\mathcal{Y}$ is finite, elements unordered. Special case is binary classification with $|\mathcal{Y}| = 2$.
- **Regression:** $\mathcal{Y}$ is continuous, ordered.

Re. Q2: Measuring Performance — “Loss Function”

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \quad (\text{usually } \geq 0)$$

Lower is better. Loss is $\ell(h(x), y)$ where:

- First argument: your guess
- Second argument: true value.

E.g., Classification: Is it right or wrong?

$$\ell(\hat{y}, y) = \mathbf{1}\{\hat{y} \neq y\}$$

where $\mathbf{1}\{\cdot\}$ is the **indicator function**:
$$\begin{cases} 1 & \text{if true} \\ 0 & \text{if false.} \end{cases}$$

E.g., Regression: Squared error

$$\ell(\hat{y}, y) = (\hat{y} - y)^2.$$

For any $h : \mathcal{X} \rightarrow \mathcal{Y}$ that you choose, you care about its performance on a new, previously unseen datapoint (“test data”) which we will denote as $x^{(\text{new})}$.

Applying h to $x^{(\text{new})}$ to predict its label is called **inference**.

Given $x^{(\text{new})} \in \mathcal{X}$ with true label $y^{(\text{new})} \in \mathcal{Y}$ (unknown to you), you would like $\ell(h(x^{(\text{new})}), y^{(\text{new})})$ to be small. Called “**generalization error**” or “**test error**”.

Generalization: Performance on unseen data.

Frequently, we assume that all datapoints (both training & test) are drawn i.i.d. from some unknown distribution on $\mathcal{X} \times \mathcal{Y}$.

Then generalization error is:

$$\mathbb{E}_{(x^{(\text{new})}, y^{(\text{new})})} \left[\ell \left(h(x^{(\text{new})}), y^{(\text{new})} \right) \right]$$

“**Risk**” = Expected Loss

1 Re. Q1: How do we choose h ?

We will discuss one popular approach here. Want to choose h which minimizes $\mathbb{E} [\ell(h(x^{(\text{new})}), y^{(\text{new})})]$.

But we don’t know $y^{(\text{new})}$ nor the distribution.

We do have access to $(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})$. These are assumed to be drawn i.i.d. from some unknown distribution, and further $(x^{(\text{new})}, y^{(\text{new})})$ is also assumed to be drawn independently from that distribution.

Idea: Minimize average loss on training data.

Empirical Risk Minimization

Choose h that minimizes:

$$\frac{1}{N} \sum_{i=1}^N \ell(h(x^{(i)}), y^{(i)}) \longleftarrow \text{Empirical Risk}$$

This is called “**Empirical Risk Minimization**”.

From what class of functions should we draw the h ? How to compute the minimizing h efficiently? These will be discussed in future lectures.

Unsupervised Learning

So far we did **Supervised Learning**: There are labels $y^{(1)}, \dots, y^{(N)}$.

We will now discuss unsupervised learning. Here, you have only $x^{(1)}, \dots, x^{(N)} \in \mathcal{X}$.

Then what can you do?

- **Clustering**: Group similar $x^{(i)}$ ’s together.
- **Dimensionality Reduction**: Suppose $\mathcal{X} = \mathbb{R}^d$, say $d = 10,000$.
Convert $x^{(1)}, \dots, x^{(N)}$ to $\tilde{x}^{(1)}, \dots, \tilde{x}^{(N)} \in \mathbb{R}^{\tilde{d}}$ where $\tilde{d} \ll d$, e.g., $\tilde{d} = 10$.
You preserve some properties (e.g., $x^{(i)} \approx x^{(j)} \Rightarrow \tilde{x}^{(i)} \approx \tilde{x}^{(j)}$).
Make it more human interpretable, computationally faster processing.

- **Learning Representations, and Generation**:

Generate new images or generate text ...

- “**Self-Supervised Learning**”: Convert unsupervised problem to supervised. E.g., take text from the Internet. There are no labels. Hide a word and let the word be the label y . Let the preceding sentence be the x . Now do supervised learning — you have built a next word predictor which forms the basis of LLMs.