AIL/ML Overview

2. General Classes of problems:
   1. Supervised (i.e. Predictive) Learning
   2. Unsupervised Learning

Supervised: Goal is to learn a mapping
From inputs: \( X \) \( \rightarrow \) \( Y \)

With supervised learning, we are given labels.

\[ D = \{(x_i, y_i)\}_{i=1}^n \]
where \( x_i \in \mathbb{R}^d \)

\( X \) usually denotes the design matrix

where \( X \) \( n \times d \),
\[ y = \begin{bmatrix} x \end{bmatrix} \]
When \( y \in \mathbb{R} \), the problem is referred to as \textbf{Regression}.

When \( y \in \mathbb{E}_1, \ldots, \mathbb{E}_3 \), the problem is referred to as \textbf{Classification}.

E.g., predict expected income from education level (Regression).

E.g., predict whether digital image contains a pedestrian (Binary Classification).

Goal: \( Y = f(X) + \varepsilon \), where \( \varepsilon \) is error.

Why estimate \( f \)?

1. Prediction
2. Inference

\( \hat{Y} = f(X) \) (Our estimate)
How to quantify the proximity of \( f \) to \( \hat{f} \)?

Use a loss function.

Examples of loss functions:

1. \( \text{0-1 Loss} \) (for classification)

\[
L(f(x), \hat{f}(x)) = \begin{cases} 
0 & \text{if } f(x) = \hat{f}(x) \\
1 & \text{if } f(x) \neq \hat{f}(x)
\end{cases}
\]

2. \( \text{Quadratic Loss} \)

\[
L(f(x), \hat{f}(x)) = \int \left( f(x) - \hat{f}(x) \right)^2
data
Flow Chart

I. Collect Data: \( D = \{(x_i, y_i)\}_{i=1}^n \)

II. Partition into Train and Test Sets

\[ D_1: \text{Train Set} \quad D_1 \subset D \]

\[ D_2: \text{Test Set} \quad D_2 \subset D \]

\( D_1 \cap D_2 = \emptyset \)

Note

III. Train a Model: \( f \) using \( D_1 \)

(Regression, NN, SVM)

IV. Evaluate Model with Loss function, using \( D_2 \)

Big Idea: The smaller (Test-1) Loss on the Test set, the better the model (Ideally).

We use the results on the Test set to approximate how well the model will generalize to new data.
With unsupervised learning, we are just given data without labels.

In this case, we are interested in discovering "interesting structure" in the data. This is sometimes called knowledge discovery or cluster analysis.

Note: Reinforcement Learning offers a 3rd problem class in AI/ML - where an "agent" learns how to act or behave when given occasional reward or punishment signals. (e.g., Alpha Go, Atari w/ Deep Q-learning) (2016; 2014)

Parametric Models vs. Non-Parametric Models

Parametric models consist of a finite (or fixed) of parameters: $\Theta = (\theta_1, \theta_2, ..., \theta_n)$. 
Idea: Using the training data, we learn values for the parameters.

**Ex.** Polynomial Regression: Fit a polynomial curve to a data set (e.g., using OLS, etc.)

**Linear Regression**

\[ f(x) = \theta_0 + \theta_1 x \]

**Quadratic Regression**

\[ f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \]

Model Parameters: \( \mathbf{\theta} = [\theta_0, \theta_1, \theta_2]^T \).

**Polynomial Regression**

\[ f(x) = \sum_{i=0}^{d} \theta_i x^i \quad (d+1 \text{ parameters}) \]

**Note:** If we use a model with a small number of parameters, it is usually easier to train (requires less time for data).
However, a low dimensional model might not be sufficiently "complex" to capture all the interesting & useful patterns in our data!

(This is called **underfitting**)

On the other hand, a large dimension/complex model requires more computation & time on average. Moreover, an excessively complex model will be "overly tuned" to the training data.

(This is called **overfitting**)

**Conclusion:** There is no "free lunch" in data science!

---

((underfitting))

low complexity model

((overfitting))

high complexity model
How do I know when I get it right?

This is the "art" of AI/ML/Data Science!

(In other words, there is no simple answer.)

In general, however, remember that we can assess our model accuracy with a loss function:

\[
\text{Mean-Squared Error (MSE)} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2
\]

\[
\text{(0-1 Loss)} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i))
\]

Note: Unfortunately, having a low training error (e.g., MSE) does not guarantee low test error in general.

One common remedy for imperfect models:

Train several models of varying complexity (e.g., linear regression, quadratic, cubic regression), compute MSE for each test set, choose model with lowest MSE.
Common "U-shape" error graph.

**Bias-Variance Trade-off**

The "U-shape" phenomenon in the test MSE is indicative of two competing properties of learned models: Bias & Variance.

- **Low-dimensional (simple models):** High Bias & Low Variance.
- **High-dimensional (complex/flexible models):** Low Bias & High Variance.
More concretely, the expected MSE for a given value $x_0$, can always be decomposed into the sum of 3 fundamental quantities:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \text{Var}[\hat{f}(x_0)] + [\text{Bias}(\hat{f}(x_0))]^2$$

From above, we see that the ideal model will simultaneously achieve [low variance & low bias].

Ex. Unsupervised Learning

Suppose we have $D = \{(x_1, x_2, \ldots, x_n)\}$ with no class labels (i.e., no $y$ values), 2-d data.
We will use a clustering method to first cluster the data (let k = # of clusters) then classify a new datum based on a nearest centroid criterion – this is usually called k-means.

Idea: Suppose we obtain biometric data from men & women (so k = 2), where $x \in \mathbb{R}^2$.

We use the k-means algorithm to identify each training datum as belonging to either cluster 1 or cluster 2. (Don't worry about k-means details now).

$$M_1 = \frac{1}{|C_1|} \sum_{x \in C_1} (x_1, x_2) = \text{centroid of } C_1$$

$$M_2 = \frac{1}{|C_2|} \sum_{x \in C_2} (x_1, x_2) = \text{centroid of } C_2$$

How to classify new test datum $x^*$?

Class for $x^*$ = $\arg \min_{i \in \{1, 2\}} \| x^* - M_i \|^2$ based on nearest cluster center.
Note that the previous examples are considered parametric models because the number of parameters is fixed once we "test" these parameter values. The training data can be discarded when we perform test prediction/classification.

What would a non-parametric model look like?

Two conditions:
1. The number of parameters is not fixed, and typically grows as the training set gets larger;
2. The training data is retained for prediction/classification (conditionally).

(Ex. Kernel density estimation)

Idea: fit a smooth, flexible "histogram" to data. Produce a probability density function. Model complexity increases as the number of data points increases.
Logistic Regression: A parametric (Binary) Classification Model.

Def. The sigmoid function:

\[ \text{sign}(x) = \frac{1}{1 + e^{-x}} \]

Note:
\[
\lim_{x \to -\infty} \text{sign}(x) = 1 \\
\lim_{x \to +\infty} \text{sign}(x) = 0
\]
Ideas:
Parameter Models $\theta = (\theta_0, ..., \theta_m)$

$$P(y | \mathbf{x}, \theta) = \text{Ber} \left( y \mid \text{sign} \left( \theta^T \mathbf{x} \right) \right)$$

- Bernoulli
- Model parameters
- Classifiers for classification

Yes 0.13

Steps:
1. With training, linear model parameters $\theta$.

2. Evaluate: $\text{sign} \left( \theta^T \mathbf{x} \right) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$

3. Apply a decision rule (Threshold)
   - e.g. $f(x) = 1 \iff P(y = 1 | x) > 0.5$

Toy example:
- Training Data
- Model fit

Data in one class
Data in zero class

Recall: Yes 0.13