Outline

• Supervised Learning Overview
• Formulation
• Train-Test Split
What is Machine Learning?

Given examples (training data), make a machine learn system behavior or discover patterns

Given to us

We need to design it

Final output which enables us to make predictions
Machine Learning: Overview

**Algorithms vs Model**

- Linear regression algorithm produces a model, that is, a vector of values of the coefficients of the model.

- Decision tree algorithm produces a model comprised of a tree of if-then statements with specific values.

- Neural network along with backpropagation + gradient descent: produces a model comprised of a trained (weights assigned) neural network.
Machine Learning: Overview

Nature of ML Problems

1. **Supervised Learning**
   
   The learning algorithm would receive a set of inputs along with the corresponding correct outputs to train a model.

   ![Diagram](image-url)
Supervised Learning

Regression

**Regression**: Quantitative Prediction on a continuous scale

**Examples**: Prediction of
- Age of a person from his/her photo
- Price of 10 Marla, 5-bedroom house in 2050
- USD/PKR exchange rate after one week
- Efficacy of Pfizer Covid vaccine
- Average temperature/Rainfall during monsoon
- Cumulative score in this course
- Probability of decrease in the electricity prices in Pakistan
- No. of steps per day

What do all these problems have in common?

**Continuous outputs**

Predicting continuous outputs is called regression
Supervised Learning

Classification

**Classification:** Given a data sample, predict its class (discrete)

**Examples:** Prediction of
- Gender of a person using his/her photo or handwriting style
- Spam filtering
- Object or face detection in a photo
- We will be Covid-free in 2025
- Temperature/Rainfall normal or abnormal during monsoon
- Letter grade in ML course
- Decrease expected in electricity prices in Pakistan next year
- More than 10000 steps taken today

What do all these problems have in common?

- Discrete outputs: Categorical
- Yes/No (Binary Classification)
- Multi-class classification: multiple classes

Predicting a categorical output is called classification
Supervised Learning Setup

Nomenclature

In these regression or classification problems, we have

- **Inputs** – referred to as Features
- **Output** – referred to as Label
- **Training data** – (input, output) for which the output is known and is used for training a model by ML algorithm
- **A Loss, an objective or a cost function** – determines how well a trained model approximates the training data
- **Test data** – (input, output) for which the output is known and is used for the evaluation of the performance of the trained model
Supervised Learning Setup

**Nomenclature - Example**

Predict Stock Index Price

- **Features (Input)**
- **Labels (Output)**
- **Training data**

<table>
<thead>
<tr>
<th>Interest_Rate</th>
<th>Unemployment_Rate</th>
<th>Stock_Index_Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.75</td>
<td>5.3</td>
<td>1464</td>
</tr>
<tr>
<td>2.5</td>
<td>5.3</td>
<td>1394</td>
</tr>
<tr>
<td>2.5</td>
<td>5.3</td>
<td>1357</td>
</tr>
<tr>
<td>2.5</td>
<td>5.4</td>
<td>1256</td>
</tr>
<tr>
<td>2.5</td>
<td>5.6</td>
<td>1254</td>
</tr>
<tr>
<td>2.5</td>
<td>5.5</td>
<td>1234</td>
</tr>
<tr>
<td>2.25</td>
<td>5.5</td>
<td>1195</td>
</tr>
<tr>
<td>2.25</td>
<td>5.5</td>
<td>1159</td>
</tr>
<tr>
<td>2.25</td>
<td>5.6</td>
<td>1167</td>
</tr>
<tr>
<td>2</td>
<td>5.7</td>
<td>1130</td>
</tr>
<tr>
<td>2</td>
<td>5.9</td>
<td>1075</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1047</td>
</tr>
<tr>
<td>1.75</td>
<td>5.9</td>
<td>965</td>
</tr>
<tr>
<td>1.75</td>
<td>5.8</td>
<td>943</td>
</tr>
<tr>
<td>1.75</td>
<td>6.1</td>
<td>958</td>
</tr>
<tr>
<td>1.75</td>
<td>6.2</td>
<td>971</td>
</tr>
<tr>
<td>1.75</td>
<td>6.1</td>
<td>949</td>
</tr>
<tr>
<td>1.75</td>
<td>6.1</td>
<td>884</td>
</tr>
<tr>
<td>1.75</td>
<td>6.1</td>
<td>866</td>
</tr>
<tr>
<td>1.75</td>
<td>5.9</td>
<td>876</td>
</tr>
<tr>
<td>1.75</td>
<td>6.2</td>
<td>?</td>
</tr>
<tr>
<td>1.75</td>
<td>6.2</td>
<td>?</td>
</tr>
<tr>
<td>1.75</td>
<td>6.1</td>
<td>?</td>
</tr>
</tbody>
</table>
Supervised Learning Setup

Formulation

We assume that we have \(d\) columns (features) of the input. In this example, we have two features; interest rate and unemployment rate, that is, \(d = 2\).

In general, we use \(\mathbf{x}_i\) to refer to features of the \(i\)-th sample, that is,

\[
\mathbf{x}_i = [x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,d}]
\]

If \(y_i\) is the label associated with the \(i\)-th sample \(\mathbf{x}_i\), we formulate training data in pairs as

\[
(\mathbf{x}_i, y_i), \quad i = 1, 2, \ldots, n
\]

Here, \(n\) denotes the number of samples in the training data. In this example, we have \(n = 21\).
Supervised Learning Setup

Formulation

Using the adopted notation, we can formalize the supervised machine learning setup. We represent the entire training data as

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

Here $\mathcal{X}^d$ - $d$ dimensional feature space and $\mathcal{Y}$ is the label space.

Regression: $\mathcal{Y} = \mathbb{R}$ (prediction on continuous scale)

Classification: $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{1, 2\}$ (Binary classification)

$\mathcal{Y} = \{1, 2, \ldots, M\}$ (M-class classification)
Supervised Learning Setup

Example

Data of 200 Patients:
- Age of the patient
- Cholesterol levels
- Glucose levels
- BMI
- Height
- Heart Rate
- Calories intake
- No. of steps taken

Model (h) ➔ Prediction of Oxygen Saturation
Supervised Learning Setup

Example

MNIST Data:
- Each sample 28x28 pixel image
- 60,000 training data
- 10,000 testing data
Supervised Learning Setup

Learning

Recall a problem in hand. We want to develop a model that can predict the label for the input for which label is unknown.

We assume that the data points \((x_i, y_i)\) are drawn from some \((\text{unknown})\) distribution \(P(X, Y)\).

Our goal is to learn the machine (model, function or hypothesis) \(h\) such that for a new pair \((x, y)\) \(P\), we can use \(h\) to obtain

\[
h(x) = y
\]

with high probability or

\[
h(x) \approx y
\]

in some optimal sense.
Supervised Learning Setup

**Hypothesis Class**

We call the set of possible functions or candidate models (linear model, neural network, decision tree, etc.) “the hypothesis class”.

Denoted by $\mathcal{H}$

For a given problem, we wish to select hypothesis (machine) $h \in \mathcal{H}$.

**Q: How?**

**A:** Define hypothesis class $\mathcal{H}$ for a given learning problem.

Evaluate the performance of each candidate function and choose the best one.
Supervised Learning Setup

**Q: How do we evaluate the performance?**

**A:** Define a loss function to quantify the accuracy of the prediction.

**Loss Function**

Loss function should quantify the error in predicting $y$ using hypothesis function $h$ and input $x$.

Denoted by $\mathcal{L}$. 
Supervised Learning Setup

0/1 Loss Function:

Zero-one loss is defined as

$$\mathcal{L}_{0/1}(h) = \frac{1}{n} \sum_{i=1}^{n} 1 - \delta_{h(x_i)-y_i}$$

Here $\delta_{h(x_i)-y_i}$ is the delta function defined as

$$\delta_k = \begin{cases} 1, & k = 0 \\ 0, & \text{otherwise} \end{cases}$$

Interpretation:
- Note normalization by the number of samples. This makes it the loss per sample.
- Loss function counts the number of mistakes made by hypothesis function $D$.
- Not used frequently due to non-differentiability and non-continuity.
Supervised Learning Setup

**Squared Loss Function:**

Squared loss is defined as (also referred to as mean-square error, MSE)

\[
L_{sq}(h) = \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2
\]

**Interpretation:**

- Again note normalization by the number of samples.
- Loss is always nonnegative.
- Loss grows quadratically with the absolute error amount in each sample.

**Root Mean Squared Error (RMSE):**

RMSE is just the square root of squared loss function:

\[
L_{rms}(h) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2}
\]
Supervised Learning Setup

**Absolute Loss Function:**

Absolute loss is defined as

\[ \mathcal{L}_{\text{abs}}(h) = \frac{1}{n} \sum_{i=1}^{n} |h(x_i) - y_i| \]

**Interpretation:**

- Loss grows linearly with the absolute of the error in each prediction.
- Used in regression and suited for noisy data.

* All of the losses are non-negative
Supervised Learning Setup

Learning

We wish to select hypothesis (machine) $h \in \mathcal{H}$ such that

$$h^* = \min_{h \in \mathcal{H}} \mathcal{L}(h)$$

(Optimization problem)

Recall We assume that the data points $(x_i, y_i)$ are drawn from some (unknown) distribution $P(X, Y)$.

We can come up with a function $h$ after solving this minimization problem that gives low loss on our data.

Q: How can we ensure that hypothesis $h$ will give low loss on the input not in $D$?
Supervised Learning Setup

To illustrate this, let us consider a model $h$ trained on every input in $D$, that is, giving zero loss. Such function is referred to as memorizer and can be formulated as follows

$$h(x) = \begin{cases} y_i, & \exists (x_i, y_i) \in D, \; x_i = x, \\ 0, & \text{otherwise} \end{cases}$$

**Interpretation:**
- 0% loss error on the training data (Model is fit to every data point in $D$).
- Large error for some input not in $D$
- First glimpse of **overfitting**.

**Revisit:**

**Q:** How can we ensure that hypothesis $h$ will give low loss on the input not in $D$?

**A:** Train/Test Split
Supervised Learning Setup

**Generalization: The Train-Test Split**

To resolve the overfitting issue, we usually split $D$ into train and test subsets:

- $D_{TR}$ as the training data, (70, 80 or 90%)
- $D_{TE}$ as the test data, (30, 20, or 10%)

**How to carry out splitting?**

- Split should be capturing the variations in the distribution.
- Usually, we carry out splitting using i.i.d. sampling and time series with respect to time

**You can only use the test dataset once after deciding on the model using training dataset**
Supervised Learning Setup

**Learning (Revisit after train-test split)**

We had the following optimization problem as

$$h^* = \min_{h \in \mathcal{H}} \mathcal{L}(h)$$

We generalize it as

$$h^* = \min_{h \in \mathcal{H}} \frac{1}{|D_{TR}|} \sum_{(x, y) \in D_{TR}} \mathcal{L}(x, y|h)$$

**Evaluation**

Loss on the testing data is given by

$$\epsilon_{TE} = \frac{1}{|D_{TE}|} \sum_{(x, y) \in D_{TE}} \mathcal{L}(x, y|h^*)$$
Supervised Learning Setup

**Generalization loss**

We define the generalized loss on the distribution $P$ from which the $D$ is drawn as the expected value (average value, probability weighted average to be precise) of the loss for a given $h^*$ as

$$
\epsilon = E[\mathcal{L}(x, y|h^*)]
$$

The expectation here is over the distribution $P$ of $(x, y)$.

Under the assumption that data $D$ is i.i.d (independent and identically distributed) drawn from $P$, $\epsilon_{TE}$ serves as an unbiased estimator of the generalized loss $\epsilon$. This simply means $\epsilon_{TE}$ converges to $\epsilon$ with the increase in the data size, that is,

$$
\lim_{n \to \infty} \epsilon_{TE} = \epsilon.
$$
Generalization: The Train-Test Split

At times, we usually split $D$ into three subsets, that is, the training data is further divided into training and validation datasets:

- $D_{TR}$ as the training data, (80%)
- $D_{VA}$ as the validation data, (10%)
- $D_{TE}$ as the test data, (10%)

Q: Idea:
Validation data is used to evaluate the loss for a function $h$ that is determined using the learning on the training data-set. If the loss on validation data is high for a given $h$, the hypothesis or model needs to be changed.
Supervised Learning Setup

Generalization: The Train-Test Split

More explanation* to better understand the difference between validation and test data:

- **Training set**: A set of examples used for learning, that is to fit the parameters of the hypothesis (model).

- **Validation set**: A set of examples used to tune the hyper-parameters of the hypothesis function, for example to choose the number of hidden units in a neural network OR the order of polynomial approximating the data.

- **Test set**: A set of examples used only to assess the performance of a fully-specified model or hypothesis.

Adapted from *Brian Ripley, Pattern Recognition and Neural Networks, 1996
Supervised Learning Setup

Generalization: The Train-Test Split (Example)

Cross validation simulates multiple train-test splits on the training data