Introduction to Machine Learning

Lecture 7:
Machine Learning Paradigms and Performance Metrics

Prof. Subrahmanya Swamy Peruru
Department of Electrical Engineering
IIT Kanpur

Lecture Content

1 Machine Learning Paradigms

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

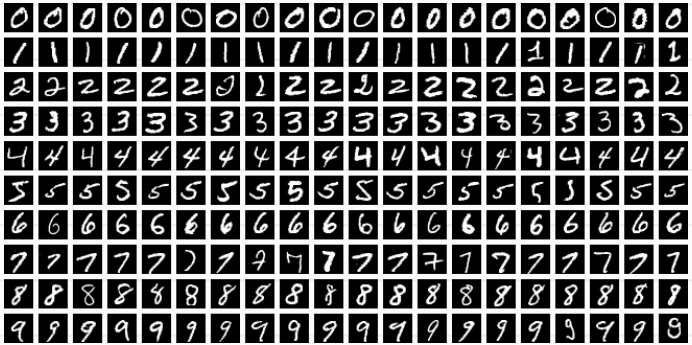
2 Performance Evaluation Metrics

- Mean square error
- Mean absolute error
- R^2 error
- Accuracy
- Precision, Recall, F-Score

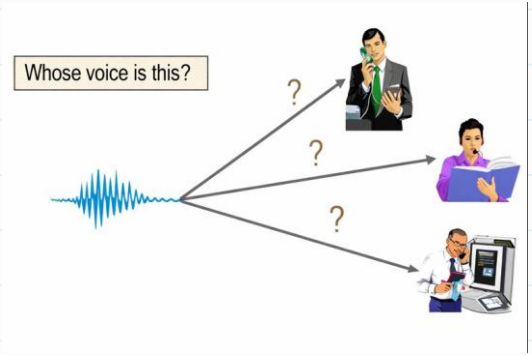
When should we use Machine Learning?

We typically use ML when:

- Explicit rules or formulas are hard to write
- Data is available, input–output relationship is complex

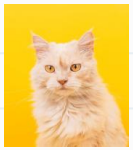


Recognize the digit



Supervised Learning

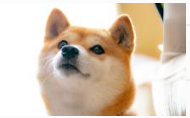
Labeled Training Data



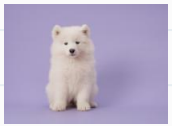
Cat

x

Cat



Dog



Dog

Labels

Training

New image

Trained model

$\hat{f}(x)$

??

Supervised learning: definition

Data:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Goal:

Learn $\hat{f}: \mathcal{X} \to \mathcal{Y}$

Predict y (response / target) accurately for unseen $\mathbf{x}$ (feature vector)

Typical formulation (Minimize Loss to find best fit):

$$\hat{f} \in \operatorname{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f(\mathbf{x}_i), y_i\right)$$

- $\mathcal{H}$ is hypothesis class
- $\ell(\hat{y}, y)$ is loss function
- **Note:** Performance of depends on how well it predicts on **unseen data**

Supervised learning: Regression vs Classification

Two common supervised settings:

Regression: $y \in \mathbb{R}$ (continuous)

Classification: $y \in \{1, \ldots, K\}$ (discrete)

Examples:

Regression: house size → price

Classification: email → spam / not spam

Regression

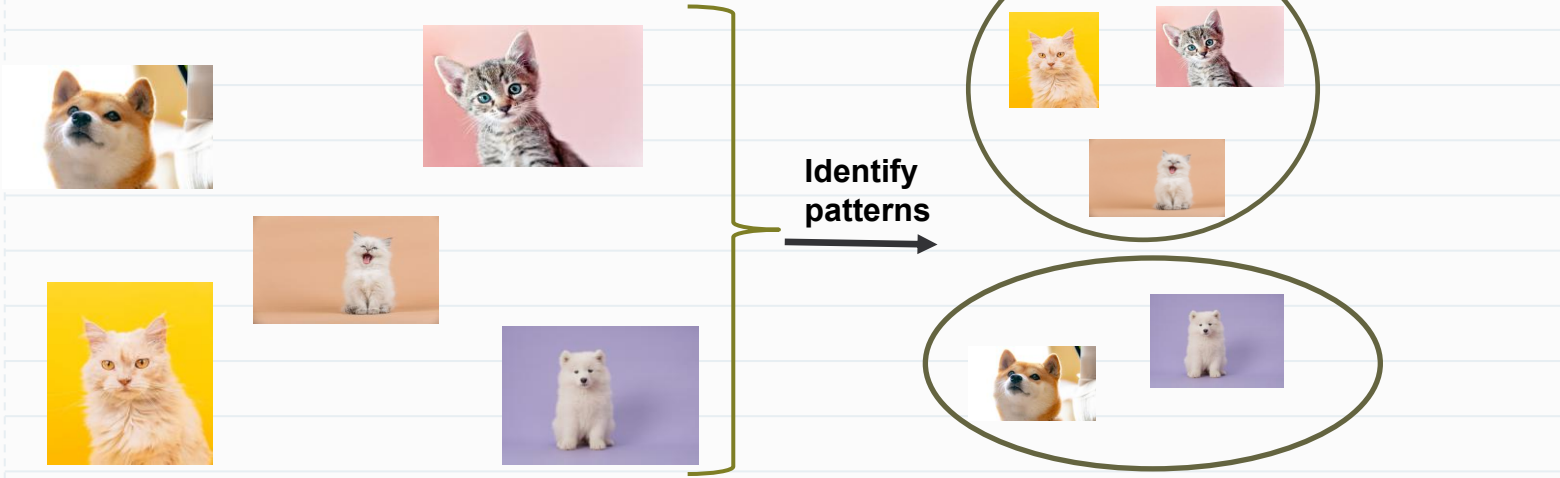
Size (sqft)	Price (Lakhs)
800	30
1000	38
1200	45
1500	60

Classification

Length (words)	Contains FREE?	Label
50	Yes	Spam
120	No	Not spam
30	Yes	Spam
200	No	Not spam

Unsupervised Learning

Unlabeled Data



What can we learn without labels?

Data:

$$\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{n}$$

There is **no target y to predict**.

So instead we try to **find structure**:

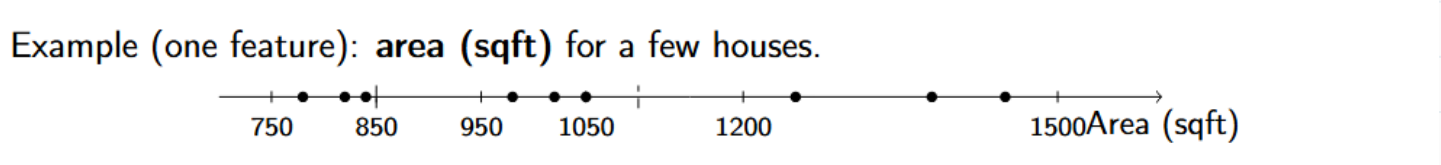
- clustering (group similar points)
- dimensionality reduction (compress / visualization)
- density estimation (model $p(\mathbf{x})$)

Clustering Example

Each data point corresponds to a house with only one feature: **area (sqft)**.

$$1020,\ 780,\ 1450,\ 840,\ 980,\ 1250,\ 820,\ 1380,\ 1050,\ \ldots$$

There are **no target label** such as price. The task is *not* to predict any label.



The algorithm groups (clusters) similar houses together.

- Cluster 1: houses around **750–850 sqft**
- Cluster 2: houses around **950–1050 sqft**
- Cluster 3: houses around **1200–1500 sqft**

Dimensionality reduction example

Suppose each house has features:

- area in sqft
- area in sqyd
- price per sqft
- total price

Even though $\mathbf{x} \in \mathbb{R}^4$,these features are correlated, so the data may lie near a lower-dimensional (say 2D) structure.

Goal: represent $\mathbf{x}$ using fewer dimensions while preserving information.

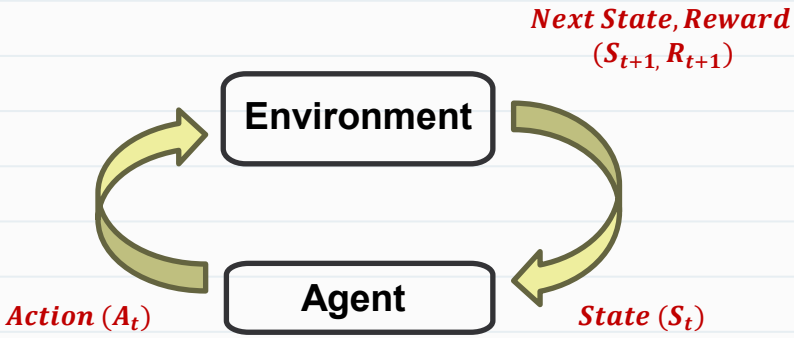
Reinforcement Learning

Feedback:
Score, new display

State:
Display

Actions:
Up / Left / Right / Down

Learn by Trial and Error



1. Agent observes the state and takes an action
2. Environment puts the agent in a new state &
3. Gives a reward based on the action taken

GOAL:
Learn policy to maximize the cumulative reward $\sum_t R\_t$

Paradigms of Machine Learning

Supervised Learning

- Fitting a function for the given labeled data (x, y)
- $y \approx f(x)$

Unsupervised Learning

- Identifying patterns or structure in unlabeled data
- E.g. Clustering, Dimensionality reduction

Reinforcement Learning

- Learning sequential tasks through trial and error
- Feedback through reward/penalty

Several Other paradigms exist!

Semi-supervised learning

In many real problems:

- Collecting data $\mathbf{x}$ is cheap
- Obtaining labels y is expensive or slow

Examples:

- Spam detection: millions of emails, few labeled
- Medical imaging: many scans, few expert annotations

We often have:

$$\mathcal{D}_L = \{(\mathbf{x}_i, y_i)\}, \quad \mathcal{D}_U = \{\mathbf{x}_j\}, \quad |\mathcal{D}_U| \gg |\mathcal{D}_L|$$

Goal:

Use additional unlabeled data to improve learning when labeled data is scarce.

Semi-supervised learning: a simple method

Self-training (idea):

- **Train** a classifier **using** a small **labeled dataset**
- Use it to **predict labels on unlabeled data**
- Keep only the **predictions** the model is very **confident** about
- **Add these to the labeled set** and retrain

This process is repeated until no new confident predictions are found.

Self-supervised learning

- Has access **only** to **unlabelled data**
- Learn representations from unlabeled data by creating training labels from the data itself.
- The learned representations are then reused for downstream tasks (classification, regression).

Example: masked word prediction in text

Original **unlabeled sentence**:

"Machine learning is very useful."

We create a training task by hiding a word:

Input: "Machine learning is very [MASK]."

Target: "useful" (the hidden word)

By learning to predict missing words,
the model learns meaningful language representations (syntax, semantics, context)
that transfer to many tasks.

Performance Evaluation Metrics

Regression metrics

Prediction error: $e_i = \hat{y}_i - y_i$

Common metrics:

MSE (Mean square error):

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

MAE (Mean Absolute error):

$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

MSE is more sensitive to outliers than MAE
MSE is differentiable everywhere

Outliers: MSE vs MAE

Tiny dataset (house size → price):

House	x (sqft)	y (Lakhs)
1	800	30
2	900	32
3	1000	34
4	1100	36
5	1200	80 (*outlier*)

x (sqft)	True y	$\hat{y}$ (trained with MSE)	$\hat{y}$ (trained with MAE)
800	30	21.6	30
900	32	32.0	32
1000	34	42.4	34
1100	36	52.8	36
1200	80	63.2	38

Train the same linear model $\hat{y} = w_0 + w_1 x$ with each metrics

Predictions from the learned models:

MSE training gets **pulled toward the outlier** (squared errors are very high for large mistakes).

MAE training **stays closer to the bulk** (errors grow linearly).

Takeaway:

MSE shifts the fitted line upward to reduce the outlier error, while MAE keeps typical points accurate instead of overreacting to one extreme case.

R^2 metric for regression

(R^2) coefficient of determination): measures how much better a model is compared to a simple baseline that always predicts the average output.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Baseline model:

$$\hat{y}_i^{\text{baseline}} = \bar{y}$$

(the same prediction for all inputs)

Interpretation:

$R^2 = 1$:perfect predictions

$R^2 = 0$:no better than predicting the mean

$R^2 < 0$:worse than the mean predictor

Another metric: **Adjusted R^2**

Classification Metrics: Accuracy

Suppose we have a classification dataset with n samples.

True labels: $y_1, y_2, \ldots, y_n$

Model predictions: $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$

For each sample:

$$y_i = \text{true class}, \qquad \hat{y}_i = \text{predicted class}.$$

Accuracy is defined as:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\hat{y}_i = y_i\}$$

Accuracy measures **how often the model predicts the correct label**

If accuracy is high, the model appears to perform well

Question: Is accuracy always **a reliable measure?**

Why accuracy is not enough

Consider a **spam detection problem (imbalanced data):**

- 1000 emails arrive in a day
- Only 50 are spam (5%)
- 950 are not spam (95%)

A very simple classifier:

Predict "not spam" for every email

What is its accuracy?

$$\text{Accuracy} = \frac{950}{1000} = 95\%$$

Question: Is this a good spam filter?

Precision and Recall

Confusion Matrix

Predicted	Actual (Ground Truth) Spam	Not Spam
Spam	TP = 0	FP = 0
Not Spam	FN = 50	TN = 950

- 1000 emails arrive in a day
- Only 50 are spam (5%)
- 950 are not spam (95%)

Two different questions matter in spam detection:

Recall (coverage of spam):

$$\text{Recall} = \frac{TP}{TP + FN}$$

Among all spam emails, how many did we catch?

Precision (quality of warnings):

$$\text{Precision} = \frac{TP}{TP + FP}$$

Among emails marked as spam, how many are truly spam?

For the "predict everything as not spam" classifier:

$$\textbf{Recall} = 0,$$

Precision is undefined

F1 score: balancing precision and recall

Precision and recall capture *different* types of errors:

- High precision, low recall: very cautious (misses many spam emails)
- High recall, low precision: very aggressive (many false alarms)

In many applications, we want **both** to be reasonably high.

F1 score combines precision and recall:

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Two contrasting cases:

$$\begin{aligned} &\text{Balanced classifier:} && \mathrm{Precision} = 0.75,\ \mathrm{Recall} = 0.60 && \Rightarrow\ \mathrm{F1} \approx 0.67 \\ &\text{Imbalanced classifier:} && \mathrm{Precision} = 1.00,\ \mathrm{Recall} = 0.10 && \Rightarrow\ \mathrm{F1} \approx 0.18 \end{aligned}$$

Interpretation:

F1 is high only when *both* precision and recall are high.

Same accuracy, very different F1

Consider 1000 emails:

- Spam (positive): 50
- Not spam (negative): 950

Classifier A: predicts "not spam" for all emails

- Predicted spam: 0
- Predicted not spam: 1000

$$TP = 0,\ FN = 50,\ FP = 0,\ TN = 950$$

$$\text{Accuracy} = 95\%, \qquad \text{F1} = 0$$

Classifier B: predicts "spam" for 60 emails

- Correctly catches 30 spam emails
- Incorrectly flags 30 non-spam emails

$$TP = 30,\ FN = 20,\ FP = 30,\ TN = 920$$

$$\text{Accuracy} = 95\%, \qquad \text{F1} \approx 0.55$$

Takeaway: Same accuracy, but completely different usefulness.

Thank You