Week 1 :-

* **What is machine Learning?**

Field of study that gives computers the ability to learn without being explicitly programmed.

* **Types of Learning :-**

- Unsupervised Learning - Trained on unlabeled data

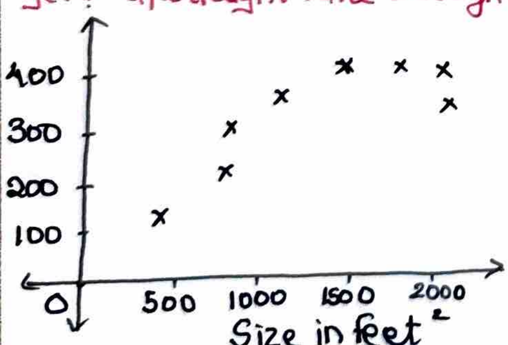
- Supervised Learning - Trained on labeled data

→ Probably the most common problem type of problem in machine learning.

→ Ex:- How do we predict housing prices?

- Collect data regarding housing prices and how they relate to size in feet.

Someone has a house - 700 sq ft, how much can they get? a) Straight Line through - 150k b) Second order polynomial maybe 200k

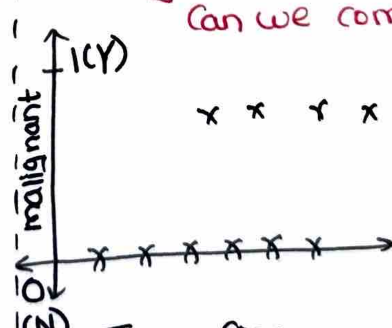


Regression

Price → Continuous

Looking at some data

Can we come to a prognosis?



Classification

Benign/Malignant

No in-between

Learning

- Supervised — Better (Human intervention reqd) Actually Predict
  - Classification — Tumor? malignant ↗ Y ↘ N — Discrete
  - Regression — Continuous
- Unsupervised — Good (only group data) → don't predict
  - Clustering — Group Similar experiences together
  - Association — Relationships between variables — Market Basket
  - Dimension Reduction — Preprocessing - Remove noise from image

* KNN k nearest neighbor :-

Similarity

Similarity points ~ inverse distance

(add points to con text)

* Farther two points are from each other, the more dissimilar they are
* Conversely, closer points are considered to be more similar.

Euclidean

$$\|x - x'\|_2 = \sqrt{(x_1 - x_1')^2 + \ldots\ldots (xd - xd')} \quad \Big| p=2$$

Certain places this distance is not valid.
- Like two points on a city grid

2-norm
L2 norm

Manhattan Distance

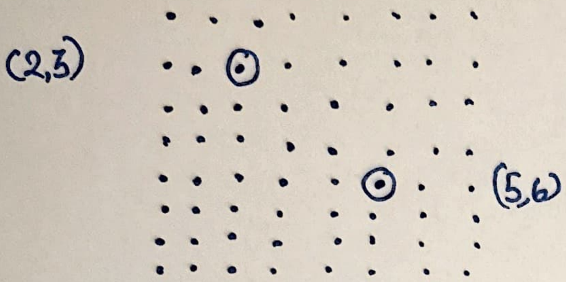
no root in manhattan distance.

$$\|x - x'\|_1 = (|x_1 - x_1'|^p + \ldots\ldots + |xd - xd'|^p)^{1/p} \quad \Big| p=1$$

* P-norm
Minkowski Distance $= \|x - x'\|_p = (|x_1 - x_1'|^p + \ldots\ldots |xd - xd'|)^{1/p}$

How about p becoming very big like $p = \infty$

* Thats called Chebyshev distance.



Consider king → can move one step

→ horizontally

→ vertically

→ diagonally

King is currently at (2,3)

row ↙ ↘ column

King wants to move to (5,6)

So, it can move diagonally 3 steps → this can be implied

but it can be found too

How?

$$\max(|x_2 - x_1|, |y_2 - y_1|)$$

row difference – $|5-2| = 3$

column difference – $|6-3| = 3$

$\max(3,3) = 3$ → Same as diagonal.

(Greatest Difference in Rows and Columns)

* **Kernels**

$\|x - x'\|_2$ ↳ distance

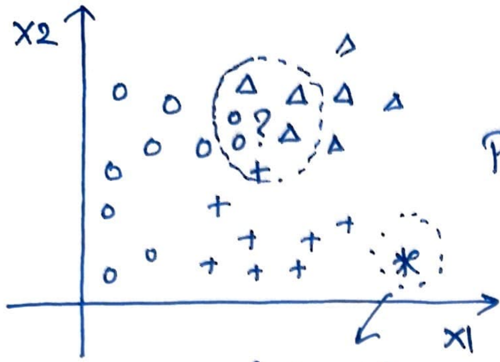
$K(x, x') = \exp(-\|x - x'\|_2)$

↳ kernel

→ if distance is less $\exp(0) \sim 1$ (similar).

→ if distane more or close to infinity $\exp(-\infty) \sim 0$ (not similar)

K-nearest neighbors



fails with K=1 or mislabeled.

Predict ?

K nearest neighbors
what is most appropriate label.

majority vote
[△△△ = 3
+ = 1
o = 1] no of instances

5 closest instances.
doesn't mean all the instances in circle

Similarity score
[△ : [0.2 0.1 0.1] → most similar = 0.4
o : [0.1]
+ : [0.2]] aggregate of inverse distance

→ Knn is a hyperalgorithm :- (informal term)

* K=5 depends on us to set 7, 10, 15
different set of predictions on different unseen points } we get to decide.

* majority vote
aggregate similarity] different setting
↳ } 1norm
2norm] another hyperparameters

ALL THESE SETTINGS DECIDED BEFORE RUNNING ARE HYPERPARAMETERS

→ K
Whenever I saw a face → I am able to retrieve from memory - 100% accuracy
SUPER RECOGNIZER

In knn, how to? can we? If I set K=1, I will be correct
nearest neighbor is X, closest to me
distance is 0

Is it a good thing?
It affects our ability to perform later on unseen points.
Outlier * → we miss this intuitively OVERFITTING

$k=2$

near to * closest are

$\left.\begin{matrix}+\\+\end{matrix}\right\}$ majority → +

$k=3$ $n=150$ (no of points)

Δ	70
O	30
+	50

→ $k=150$ → look at entire data

majority rule gives Δ

→ generalized it → which is wrong

↓

UNDERFITTING

not being faithful to labelled data

* K-NN algorithm

majority rule, assume $k=5$, binary classification (two labels 1 or 0)

```
                              datametrics      0 or 1 (labels for nearest)
                                   ←           ↗     → prediction for
function KNN-predict (X, y, x̂)
S = []
for j = 1 to n            % n → number of rows in X ⟶ (A)
    append to S the distance ||X^(j) − x̂||_2
S, prm = sort(S)          → prm is original positions of elements in S
C = 0  // initialize counter
for j=1 to k
    idx = prm[k]
    if y[idx] == 1 then C = C+1 } how many knns are 1 } increment counter
if C > k/2 then return 1 else return 0 } majority rule
```

(A) goes and looks in the entire dataset.
inefficient

$10^6 \cdot 10^3$
$\sim 10^9$

$\boxed{n = 10^6 \quad O(n \cdot d) \quad \text{distances}}$ that not how our brain works for similarity

In reality, we do KNN fit
make it a data structure (efficient search of NN)] nice placement $O(n \cdot d)$
only once

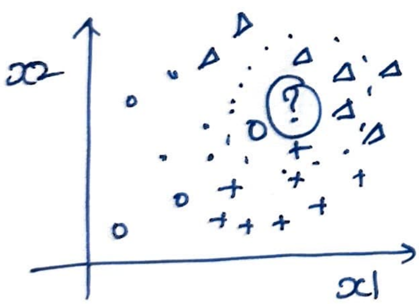
Other following predictions will happen
$O(d \cdot \log n)$

$10^3 \cdot (10^6 \simeq \log_{10} 10^6 \Rightarrow 6)$

$10^3 \cdot 6 \sim$ in thousands

*KNN for regression:-

Instead of categorical variables we have numerical values



$k = 5$

S1	Δ	0.5
S2	Δ	-0.4
S3	Δ	0.3
S4	O	1
S5	+	-1

one method → average

$$\frac{0.5 - 0.4 + 0.3 + 1 - 1}{5}$$

similarity → weighted average

$$\frac{S_1\, 0.5 + S_2\,(-0.4) + \ldots S_5}{5}$$

* KNN regressor GPT explanation.