K-Nearest Neighbour

MACHINE LEARNING

K-Nearest Neighbour

- The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- Also, It is known as:
  - ✓ Memory-Based Reasoning
  - ✓ Example-Based Reasoning
  - ✓ Instance-Based Learning
  - ✓ Case-Based Reasoning
  - ✓ Lazy Learning

Different Learning Methods

- Eager Learning
  - Explicit description of target function on the whole training set
- Instance-based Learning
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as “Lazy” learning

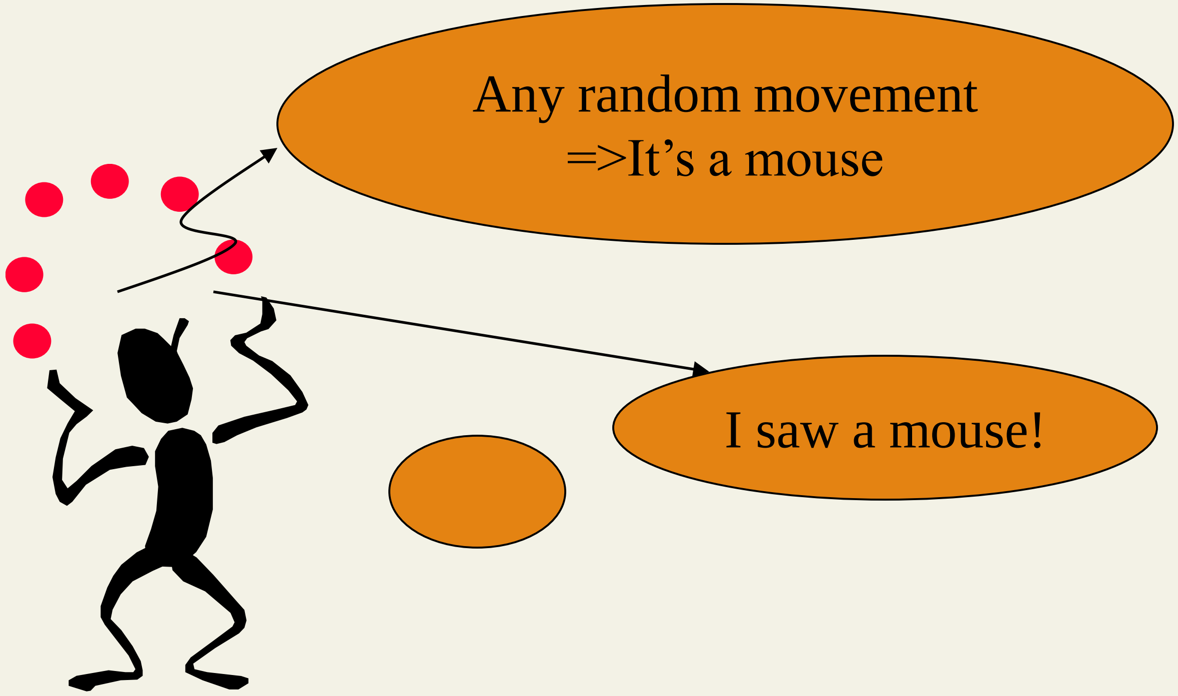
K-NN Fundamentals

- **Requires three things**
  - The set of stored records.
  - Distance Metric to compute distance between records.
  - The value of k, the number of nearest neighbors to retrieve.

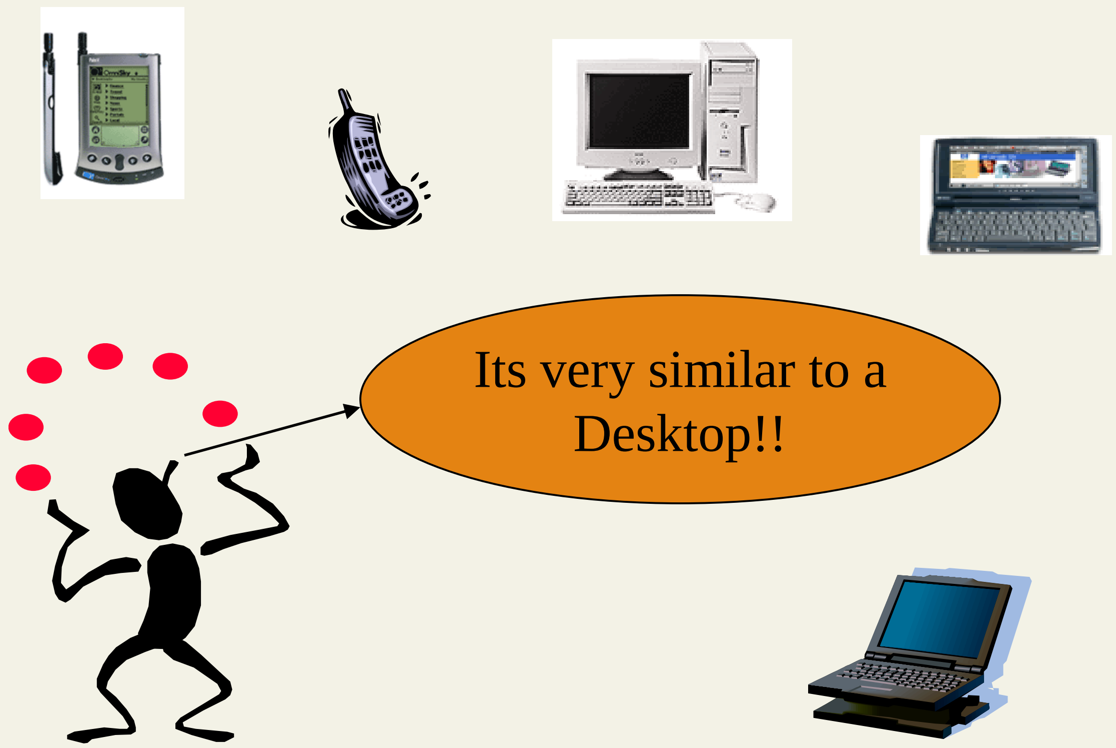
- **To classify an unknown record:**
  - Compute distance to other training records.
  - Identify K- nearest neighbors.
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote).

Different Learning Methods

- Eager Learning



Instance-based Learning



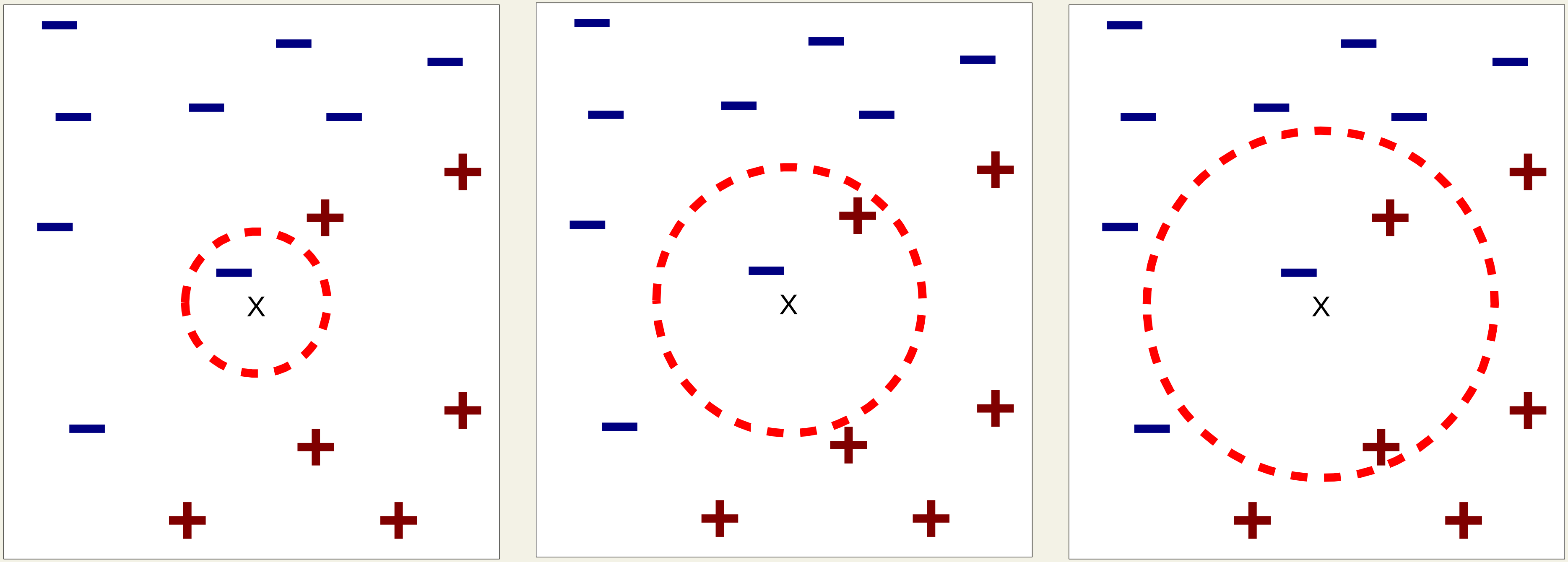
Instance-based Learning

- K-Nearest Neighbor Algorithm
- Weighted Regression
- Case-based reasoning

K-Nearest Neighbor

- Features
  - All instances correspond to points in an n-dimensional Euclidean space
  - Classification is delayed till a new instance arrives
  - Classification done by comparing feature vectors of the different points
  - Target function may be discrete or real-valued

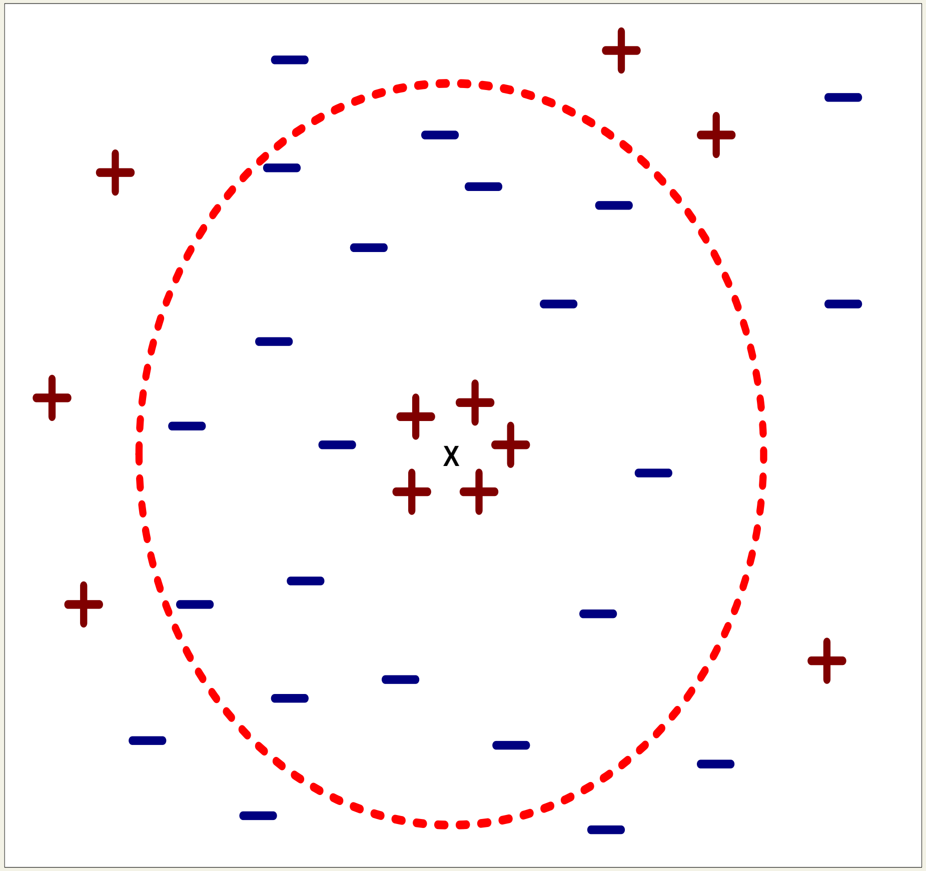
K-? Nearest Neighbor



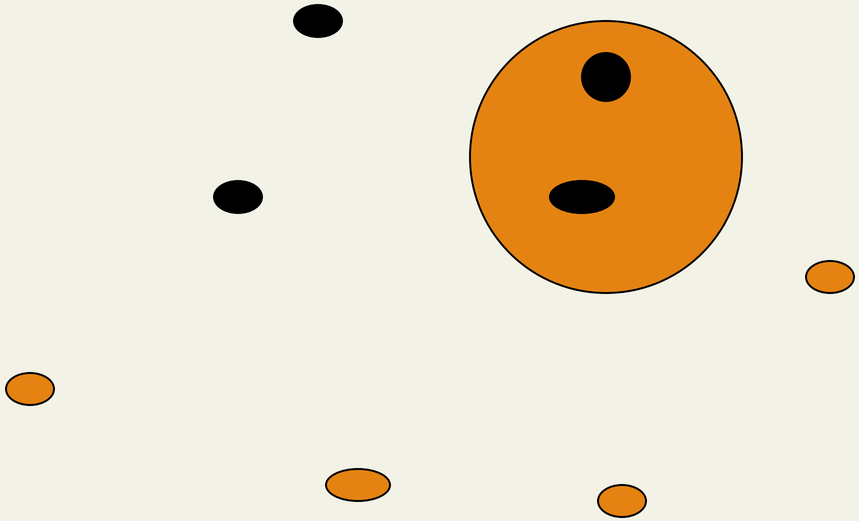
1-NN 2-NN 3-NN

Select K

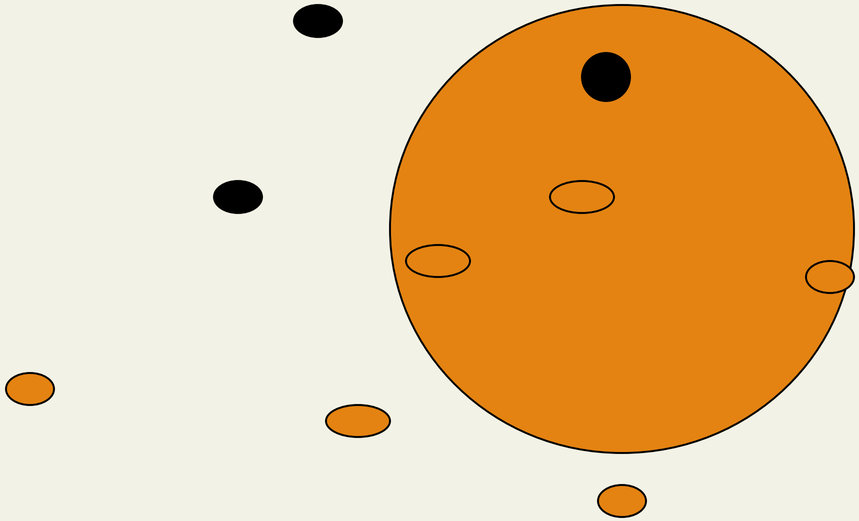
- If K is too small, sensitive to noise points.
- If K is too large, neighborhood may include points from other classes.
- Coming to your question, the value of **k** is non-parametric and a general rule of thumb in choosing the value of **k** is **k = sqrt(N)/2**, where **N** stands for the **number of samples in your training dataset**.
- Keep the value of k odd, so that there is no tie between choosing a class but that points to the fact that training data is highly correlated between classes and using a simple classification algorithm such as k-NN would result in poor classification performance.



1-Nearest Neighbor



3-Nearest Neighbor



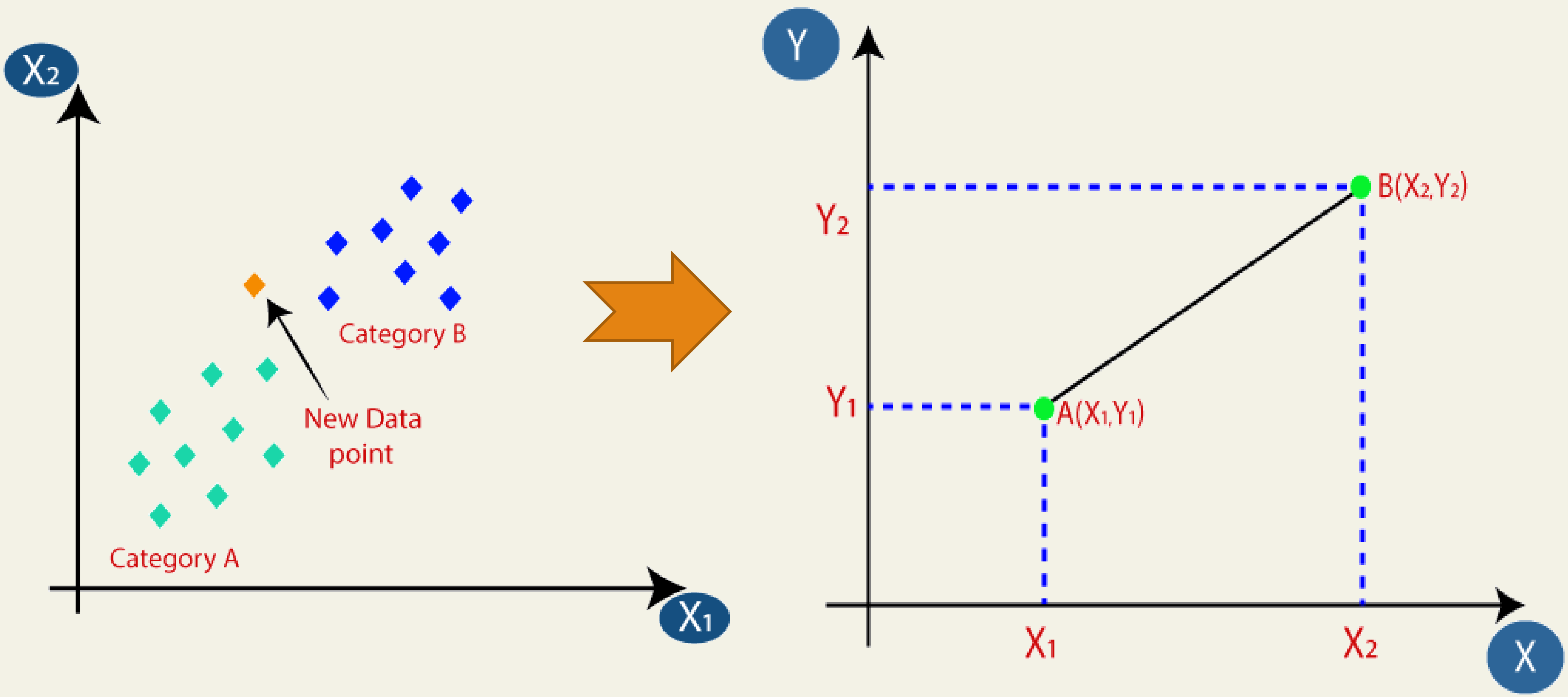
K-Nearest Neighbor

- An arbitrary instance is represented by $(a_1(x), a_2(x), a_3(x),.., a_n(x))$
  - $a_i(x)$ denotes features
- Euclidean distance between two instances

$d(x_i, x_j)$=sqrt (sum for r=1 to n $(a_r(x_i) - a_r(x_j))^2$)

- Continuous valued target function
  - mean value of the k nearest training examples

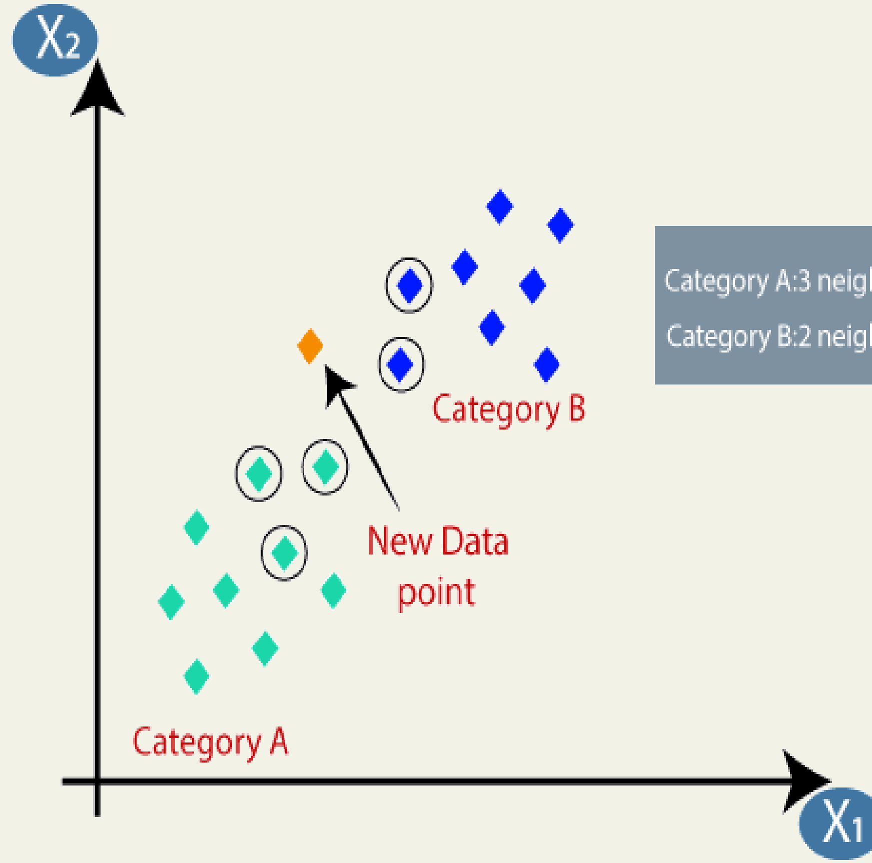
K-Nearest Neighbor…



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

K-Nearest Neighbor...

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.
- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.



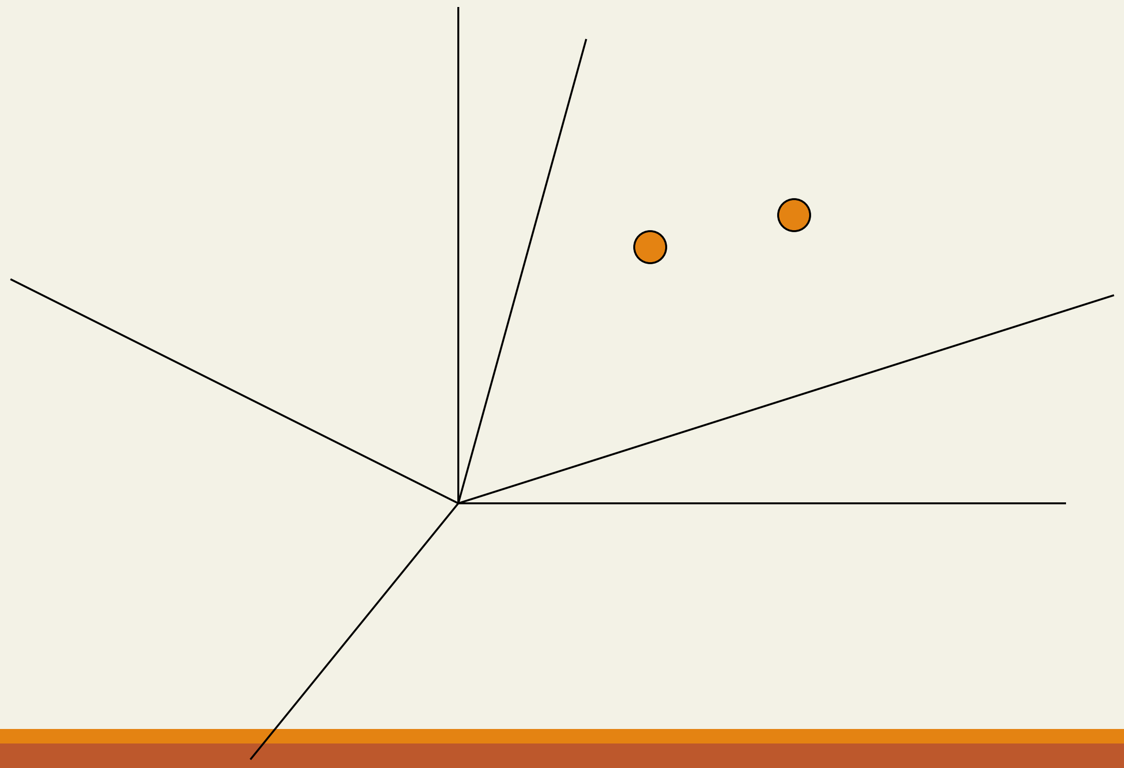
Distance-Weighted NN Algorithm

- Assign weights to the neighbors based on their 'distance' from the query point
  - Weight 'may' be inverse square of the distances

➔ All training points may influence a particular instance
  - Shepard's method

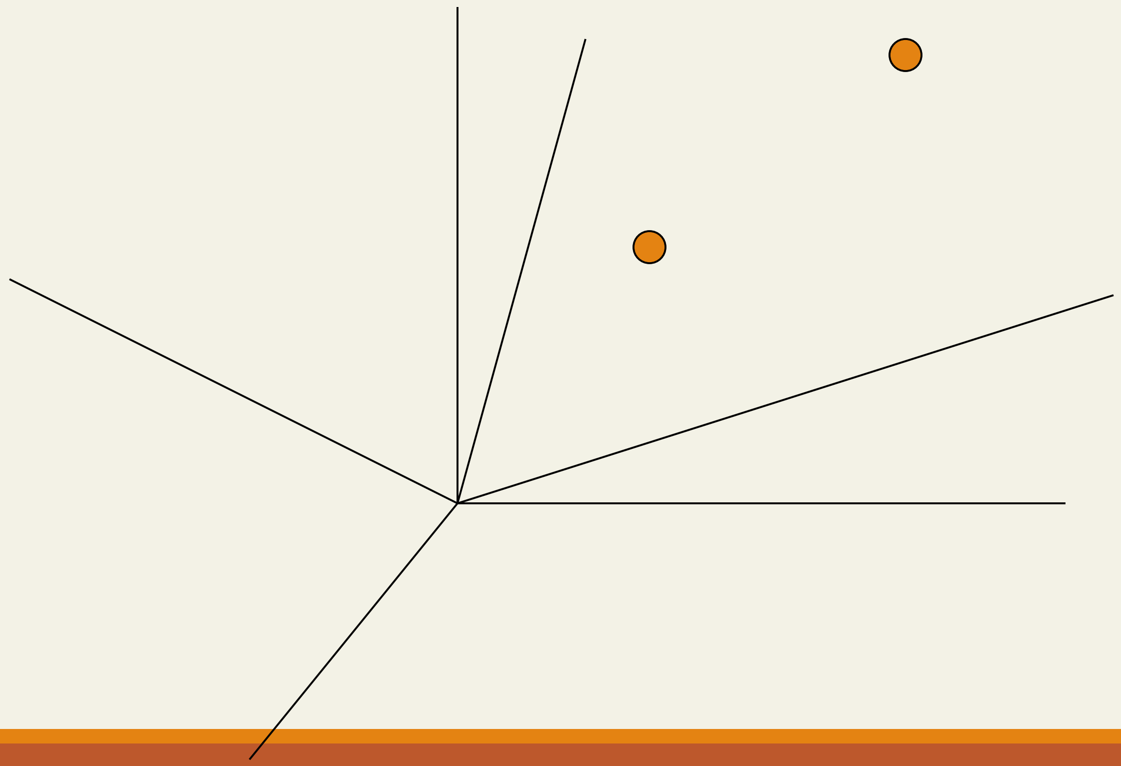
Remarks

- Curse of Dimensionality



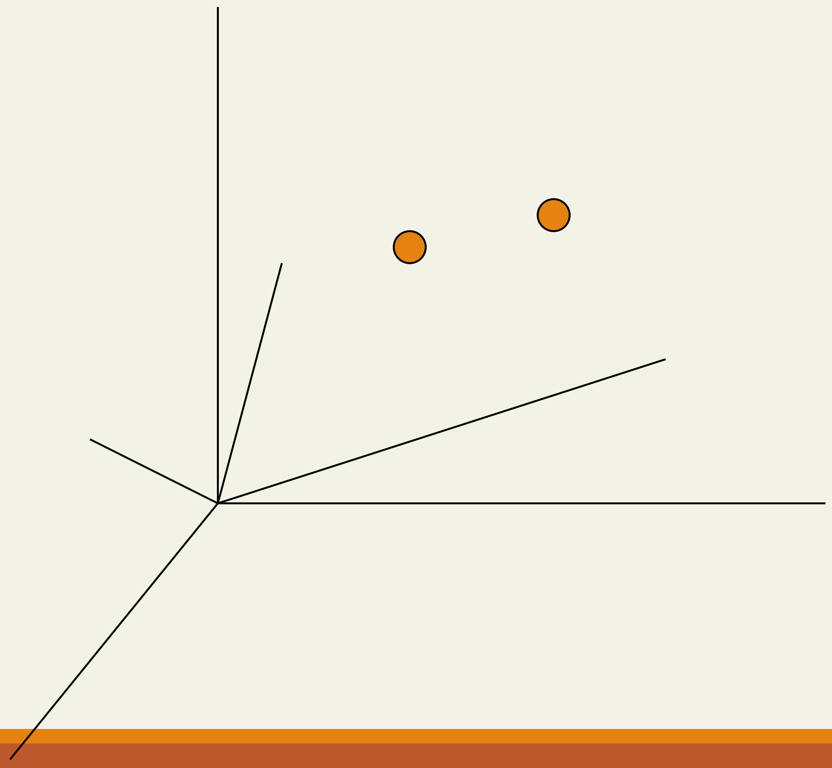
Remarks

- Curse of Dimensionality



Remarks

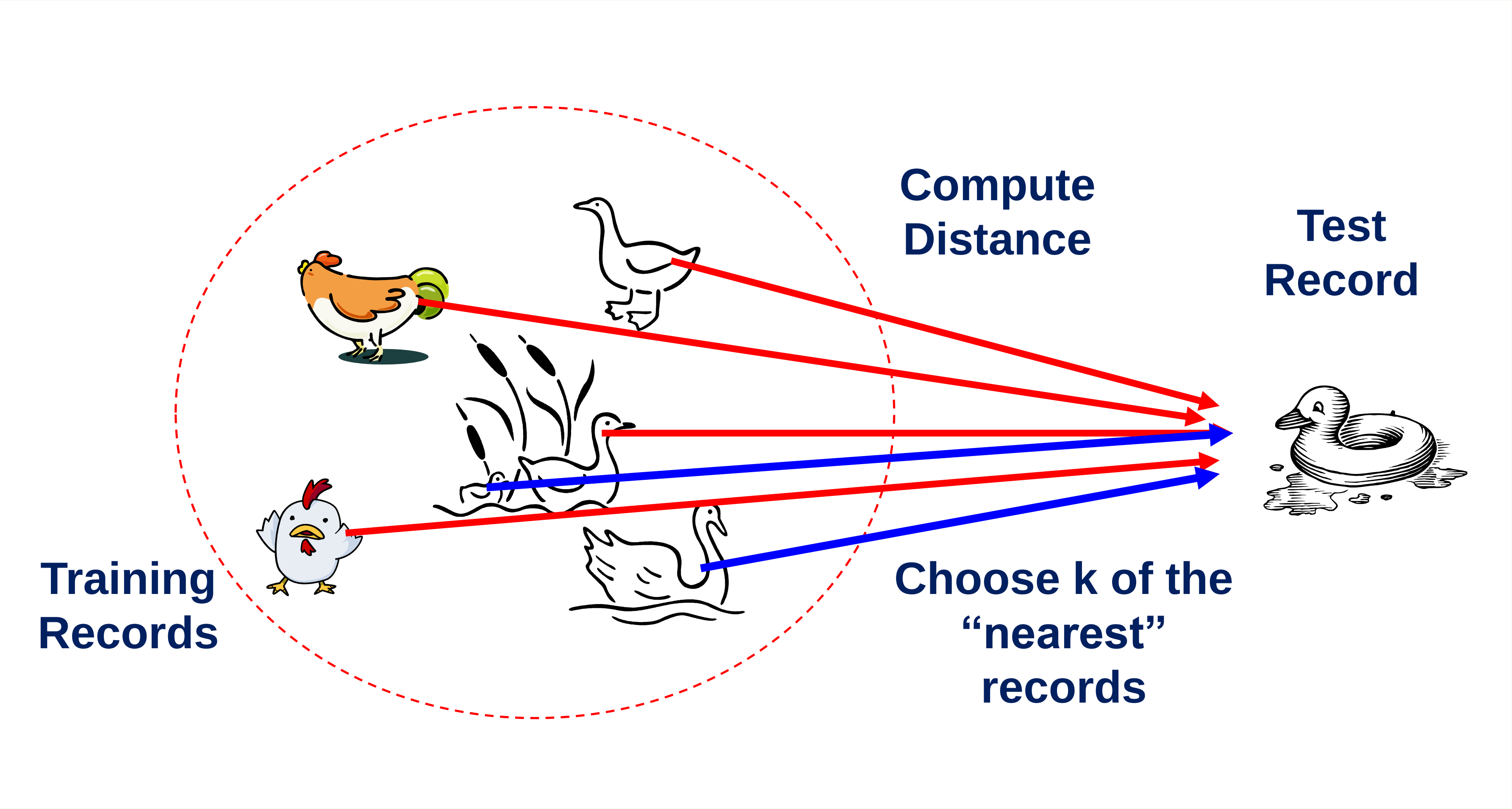
- Curse of Dimensionality



Remarks

- Efficient memory indexing
  - To retrieve the stored training examples (kd-tree)

K-NN Model


Compute
Distance
Test
Record
Training
Records
Choose k of the
“nearest”
records

K-NN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
    1. Calculate the distance between the query example and the current example from the data.
    2. Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
5. Pick the first K entries from the sorted collection.
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages

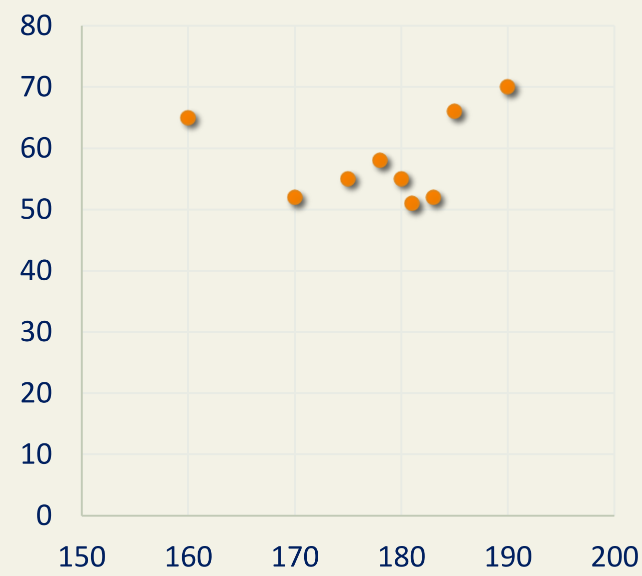
- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.
- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Example 1

- Consider a dataset of health status as given in Table. What will be the status of a sample ***(180, 65).***

S.No.	Height	Weight	Status
1	160	65	Unhealthy
2	170	52	Healthy
3	175	55	Healthy
4	178	58	Healthy
5	180	55	Unhealthy
6	181	51	Unhealthy
7	183	52	Unhealthy
8	185	66	Healthy
9	190	70	Healthy

Distance
20
16.40122
11.18034
7.28011
10
14.03567
13.34166
5.09902
11.18034



$$\text{Euclidean Distance } (D) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Example 1…

Test Sample :(180, 65).

S.No.	Height	Weight	Status	Distance
8	185	66	Healthy	5.09902
4	178	58	Healthy	7.28011
5	180	55	Unhealthy	10
3	175	55	Healthy	11.18034
9	190	70	Healthy	11.18034
7	183	52	Unhealthy	13.34166
6	181	51	Unhealthy	14.03567
2	170	52	Healthy	16.40122
1	160	65	Unhealthy	20

K=1

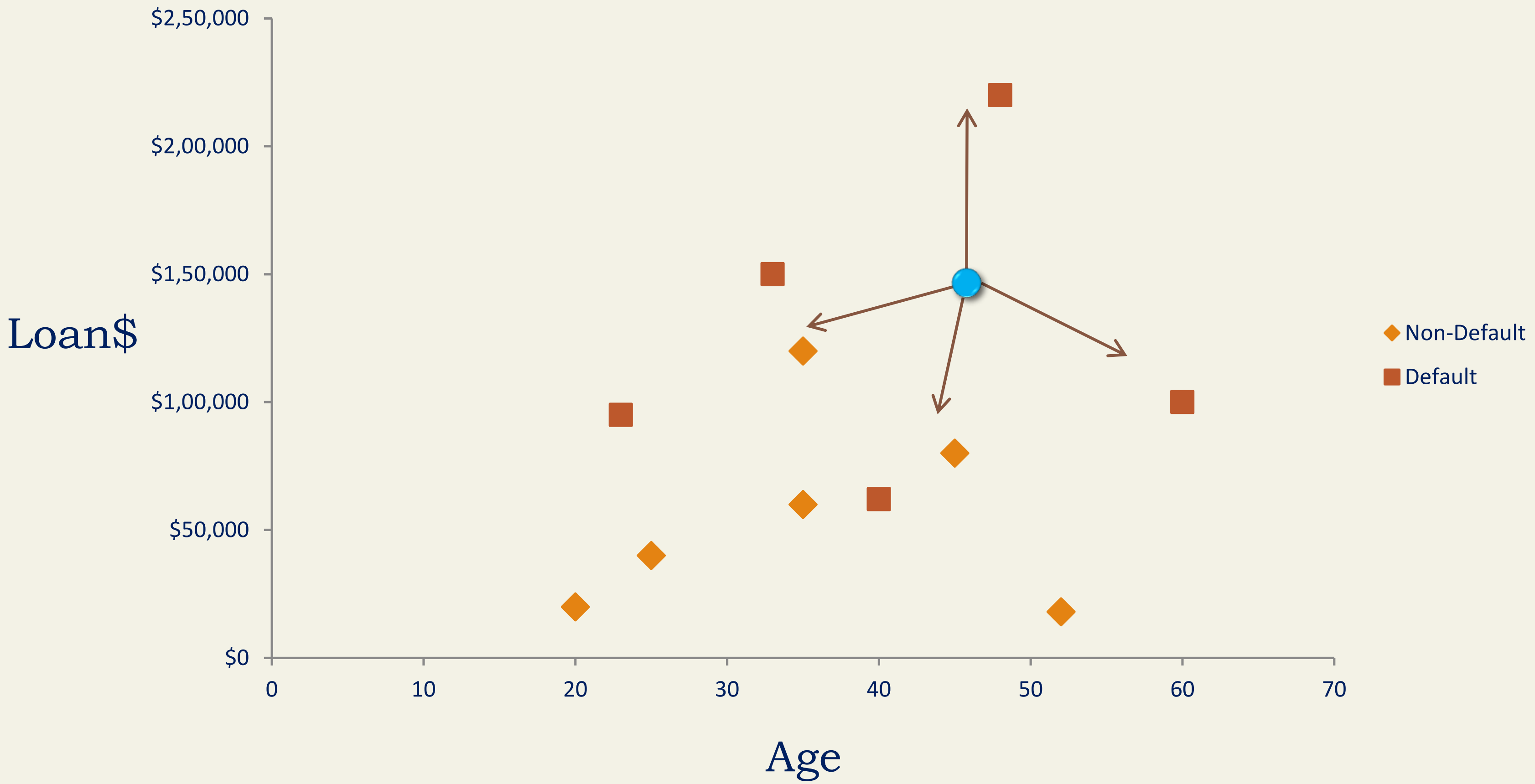
K=2

K=3

K=1, Healthy; K=2, Healthy; K=3, Healthy

Example 2

- **Sanction of loan amount**



K-NN Classification

Age	Loan	Default	Distance
25	$40,000	N	102000
35	$60,000	N	82000
45	$80,000	N	62000
20	$20,000	N	122000
35	$120,000	N	22000
52	$18,000	N	124000
23	$95,000	Y	47000
40	$62,000	Y	80000
60	$100,000	Y	42000
48	$220,000	Y	78000
33	$150,000	Y	8000
48	$142,000	?	

$$\text{Euclidean Distance } (D) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$