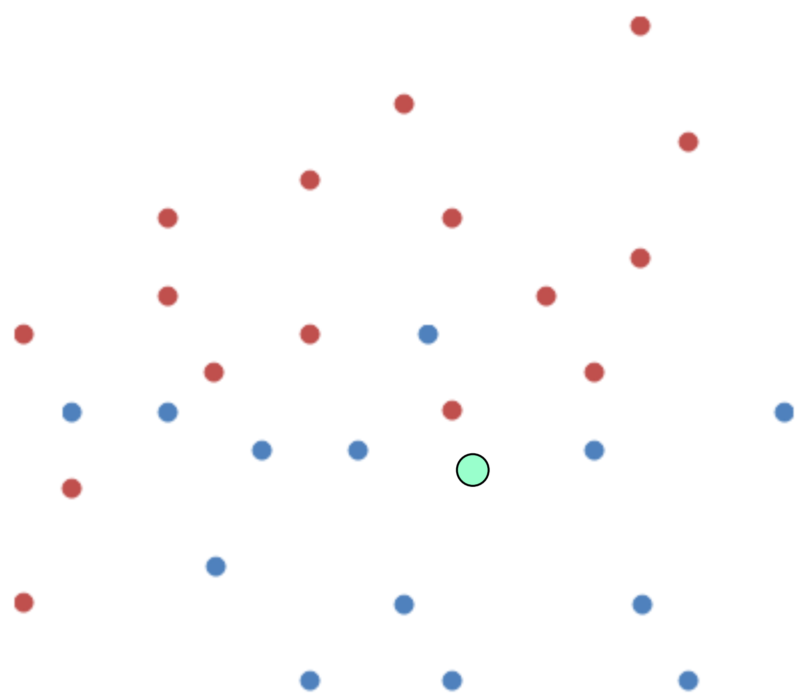
k-Nearest Neighbors (k-NN)

Contents

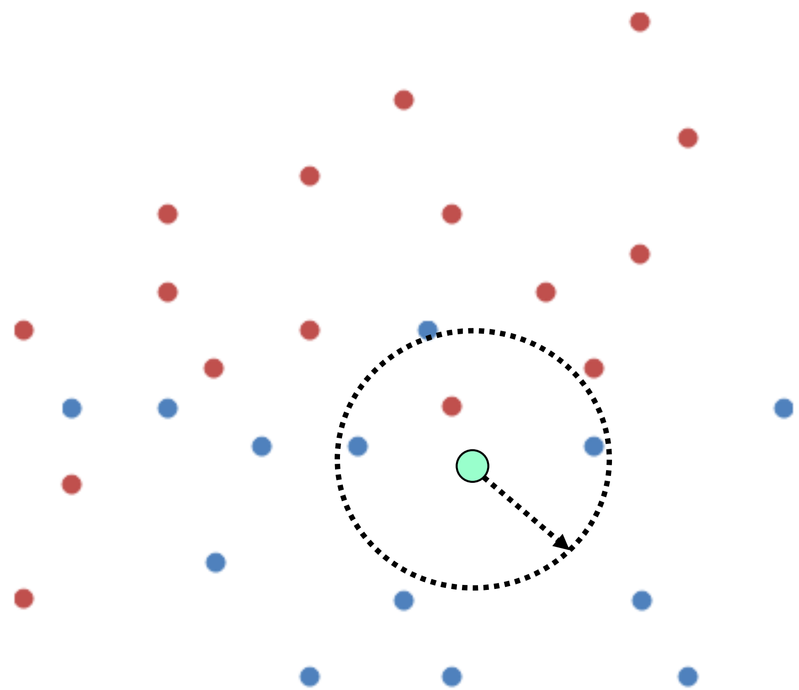
Classification

- **How to Predict Class of Unknown Data?**



Classification

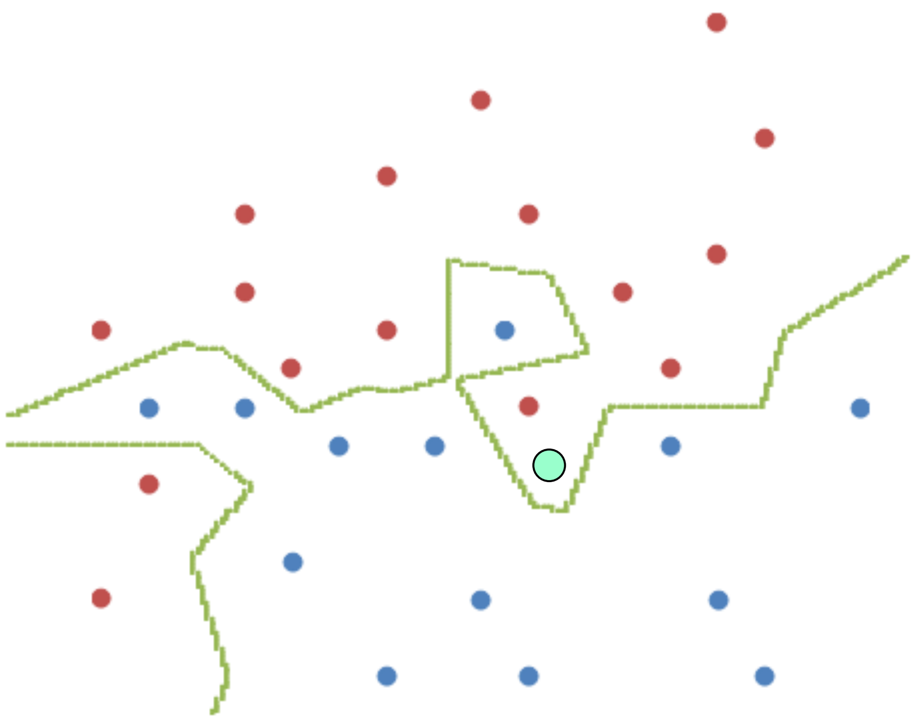
- **K-Nearest Neighbors**
  - Choose *k* nearest neighbors
  - Determine the class based on the majority



k=1, Red
k=3, Blue
k=5, Blue

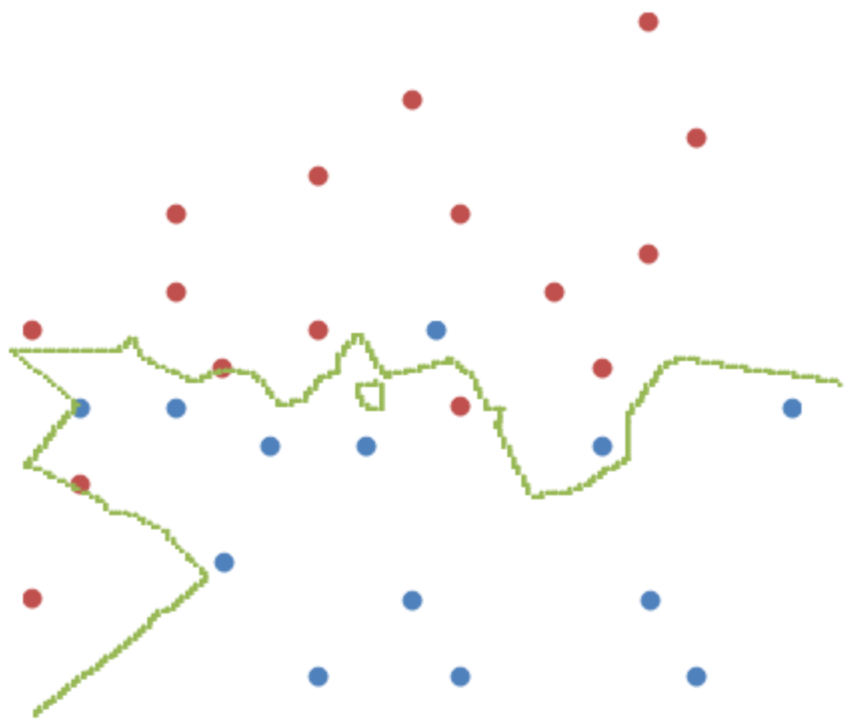
Classification

- **K-Nearest Neighbors**
  - K =1



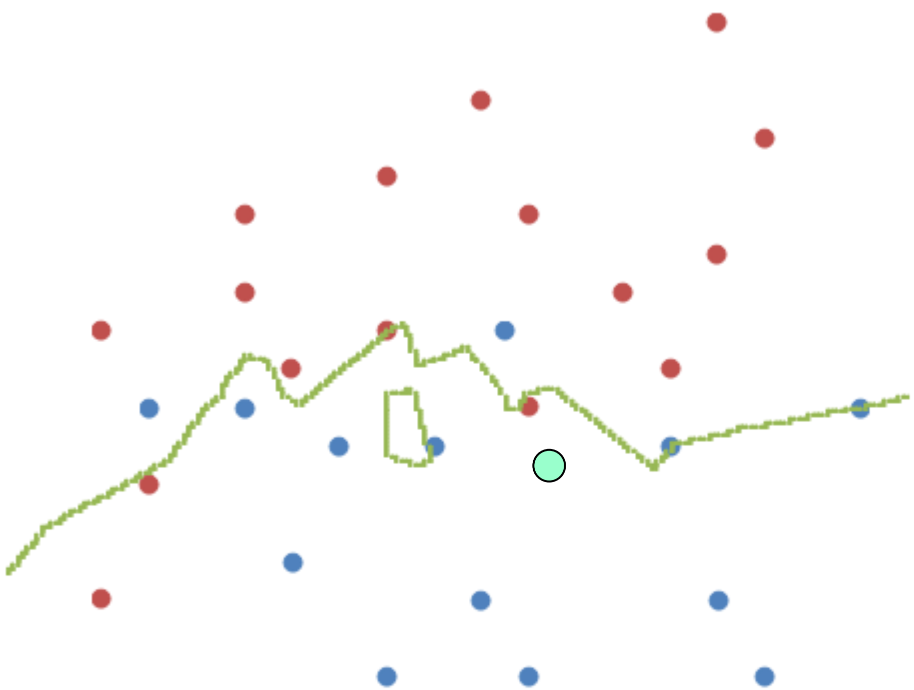
Classification

- **K-Nearest Neighbors**
  - K =3



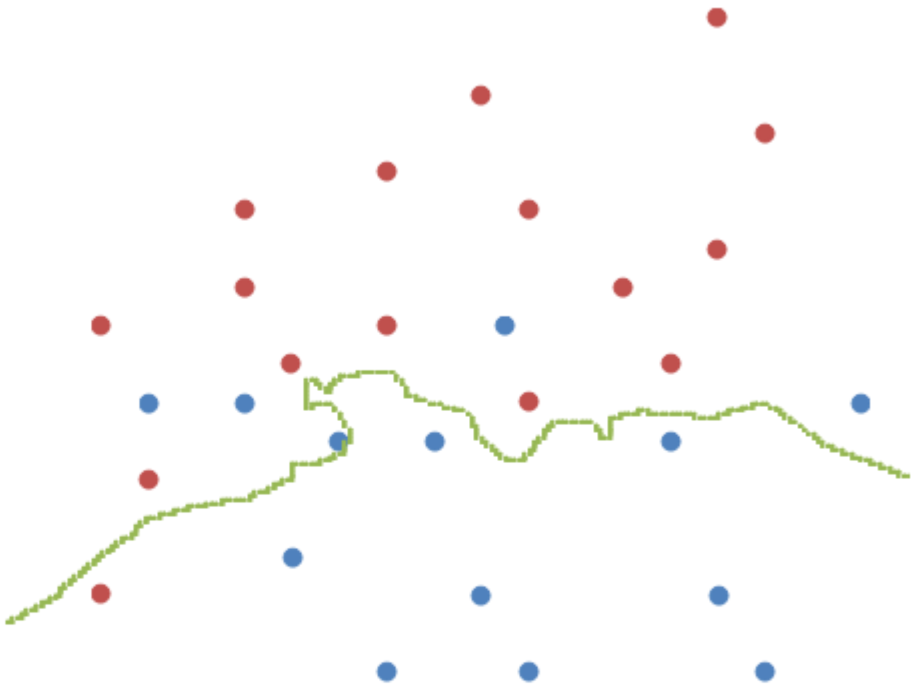
Classification

- **K-Nearest Neighbors**
  - K =5



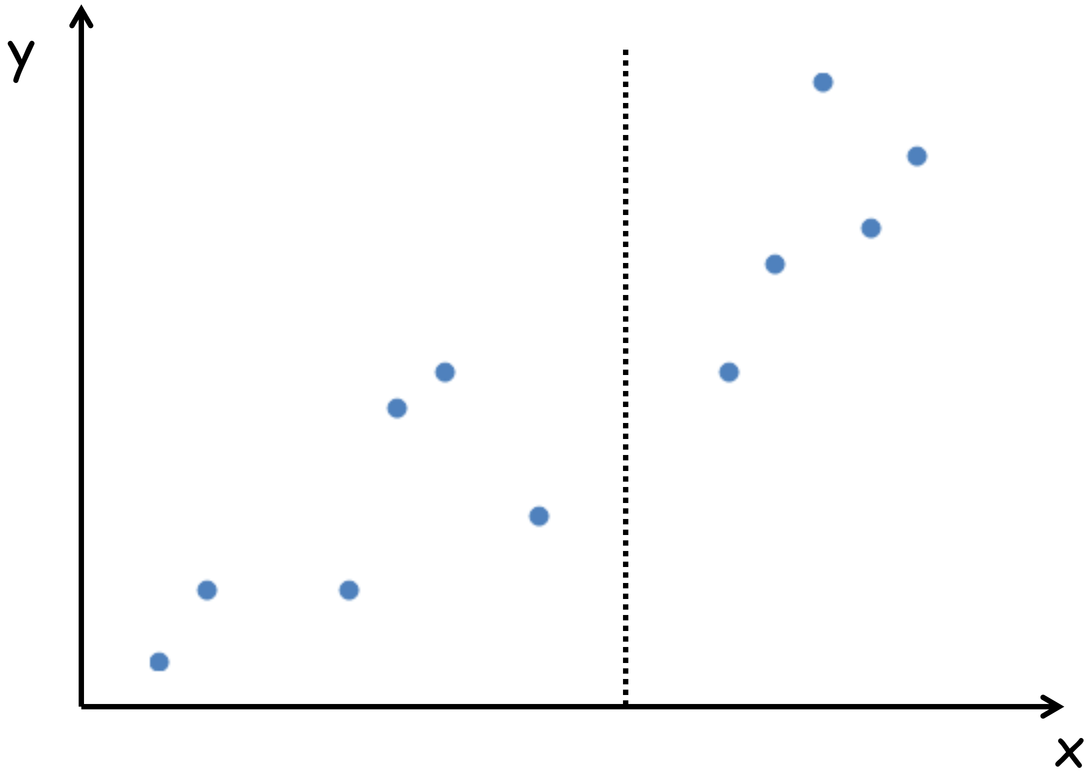
Classification

- **K-Nearest Neighbors**
  - K =10



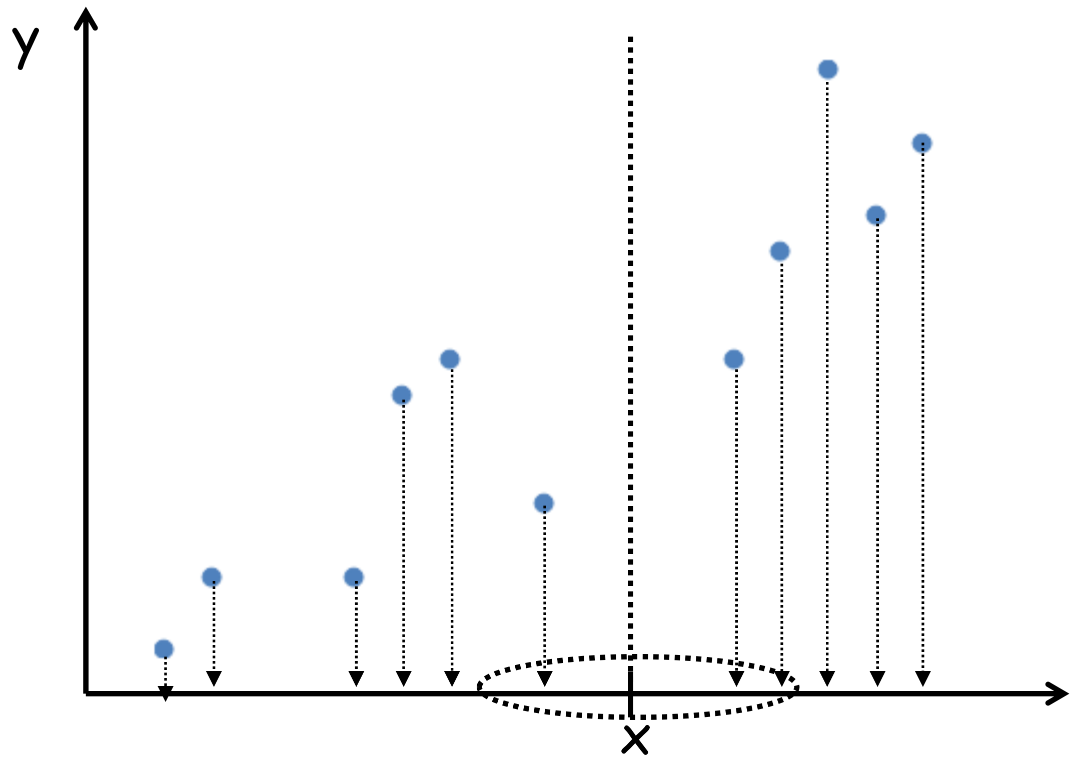
Regression

- **How to predict "y" value of unknown data**



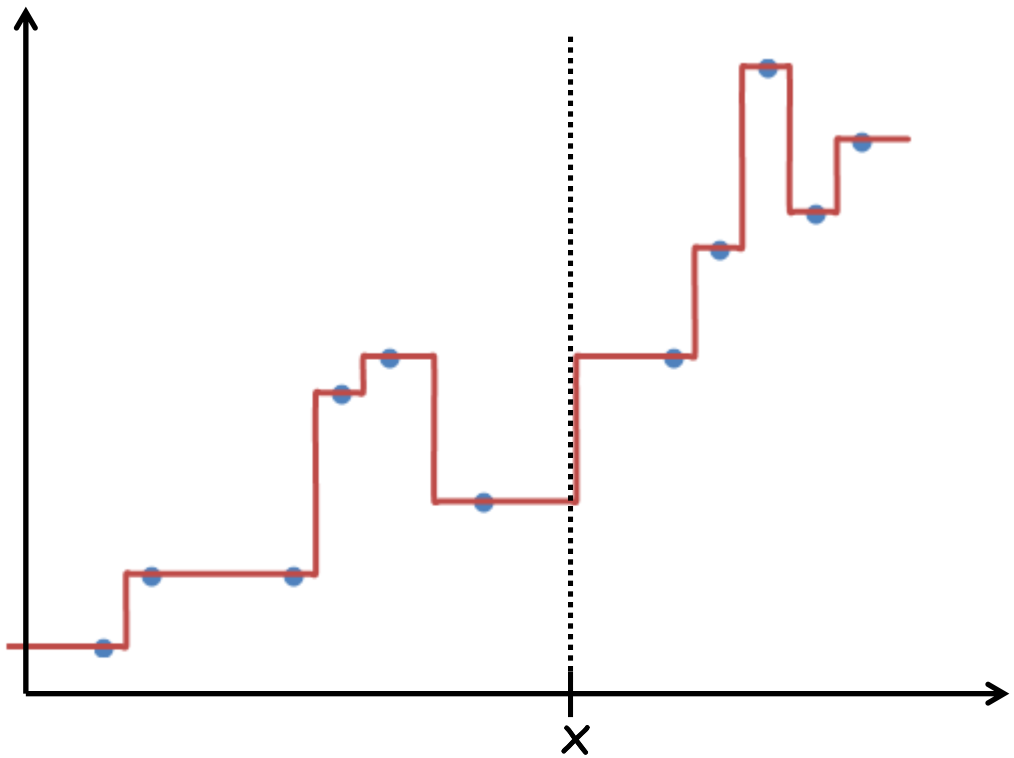
Regression

- **K-Nearest Neighbors**
  - Choose *k* nearest neighbors in X axis
  - Predict the average of their "y"



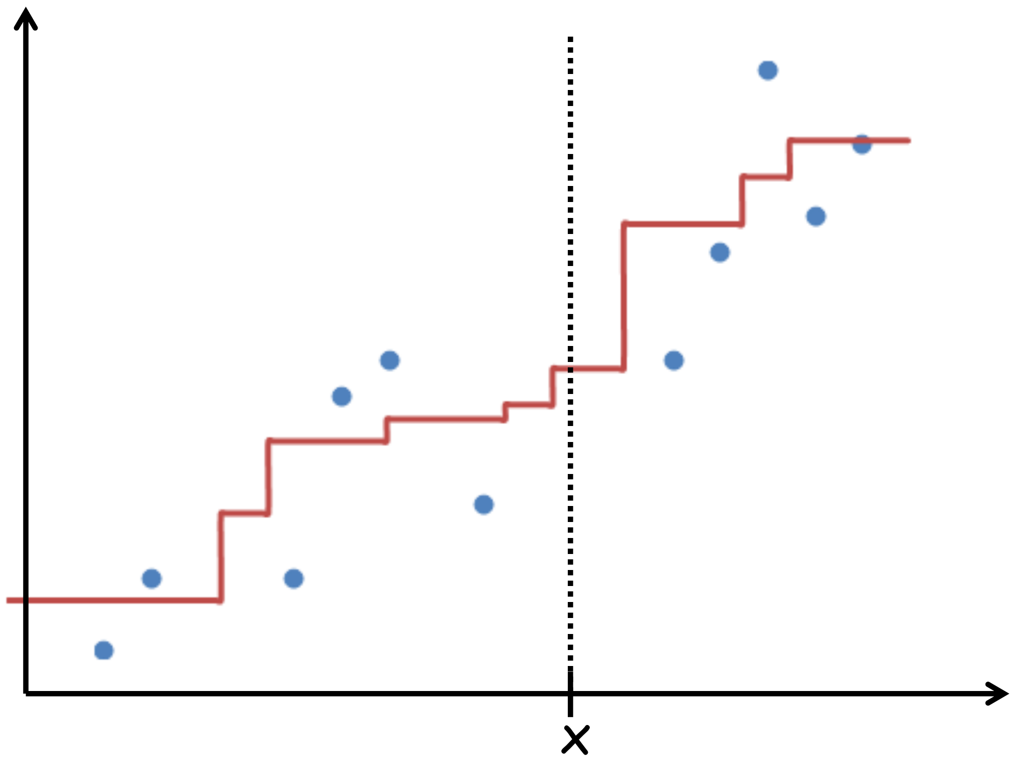
Regression

- **K-Nearest Neighbors**
  - K=1



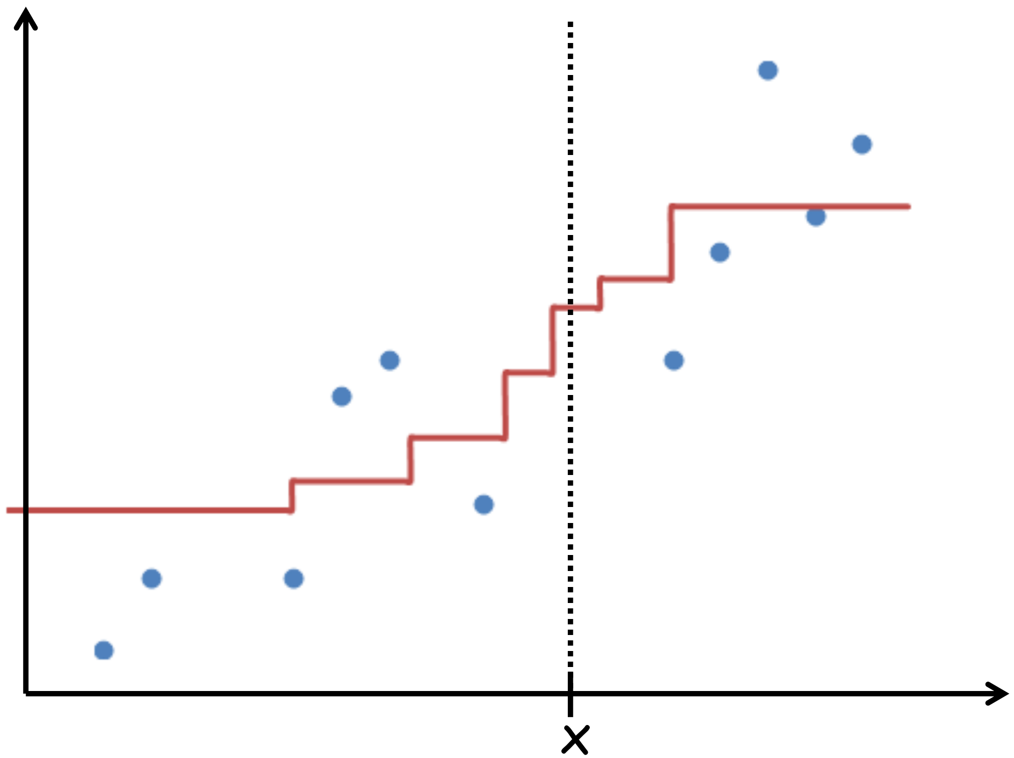
Regression

- **K-Nearest Neighbors**
  - K=3



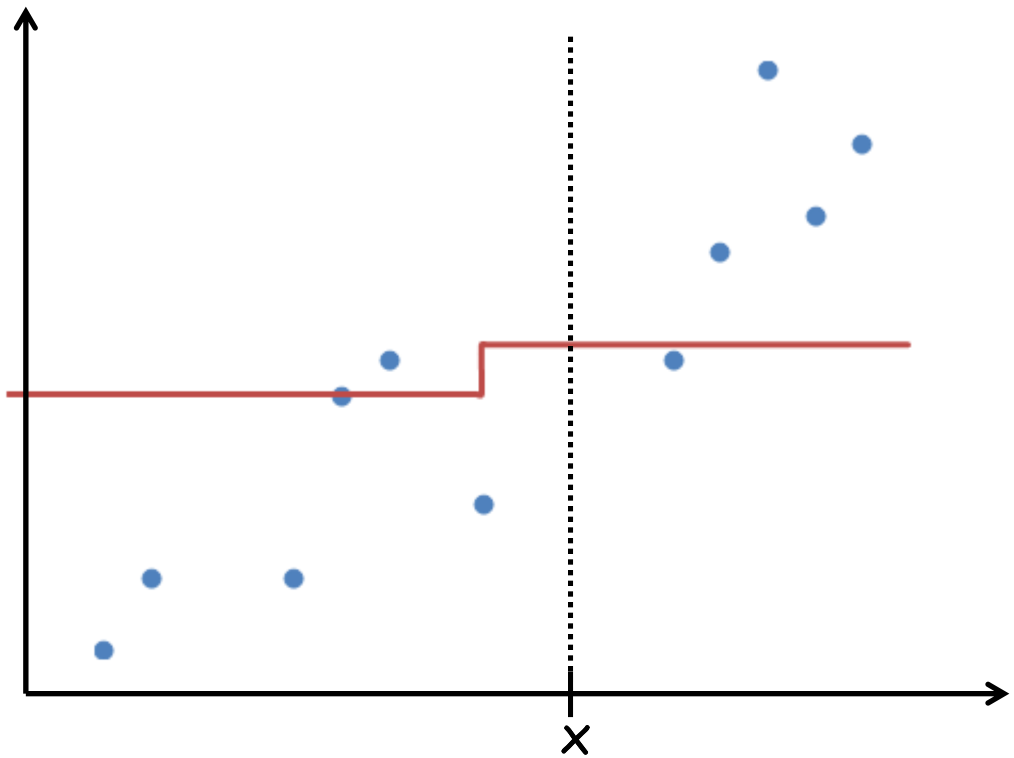
Regression

- **K-Nearest Neighbors**
  - K=5



Regression

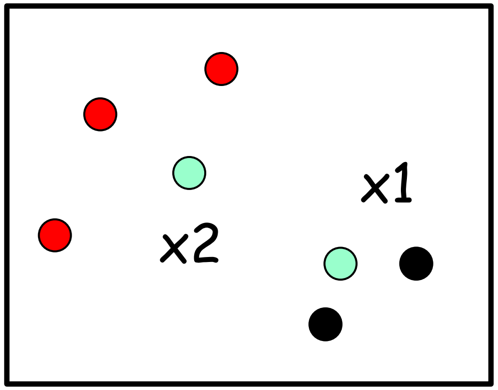
- **K-Nearest Neighbors**
  - K=10



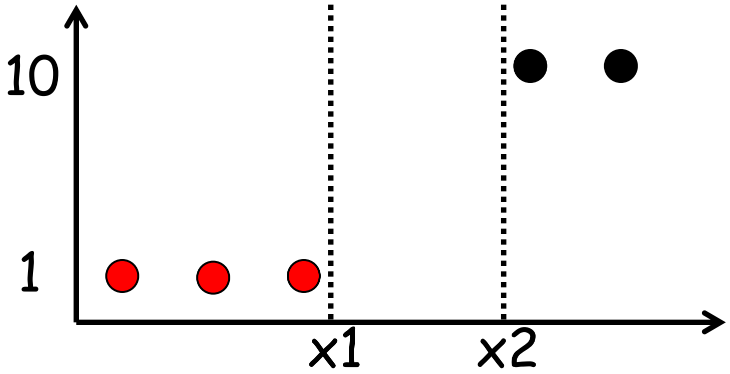
Variation

- **Why "just" counting or averaging?**

- **Classification (k=5)**
  - X1: Red or Black?
  - X2: Red or Black?



- **Regression (k=5)**
  - X1: close to 10 or 1
  - X2: close to 10 or 1



Variation

- **More weight to closer one**
  - Different weight depending on the distance from **x'**

- **Classification: Not just counting**

$$S(\mathbf{x}', R) = \sum_{\mathbf{x} \in N(\mathbf{x}', R)} w(\mathbf{x})$$

$$S(\mathbf{x}', B) = \sum_{\mathbf{x} \in N(\mathbf{x}', B)} w(\mathbf{x})$$

if $S(\mathbf{x}', R) > S(\mathbf{x}', B)$ then $\mathbf{x}'$ is R

else $\mathbf{x}'$ is B

$N(\mathbf{x}', R)$: the set of *Red* data among the nearest neighbors of $\mathbf{x}'$

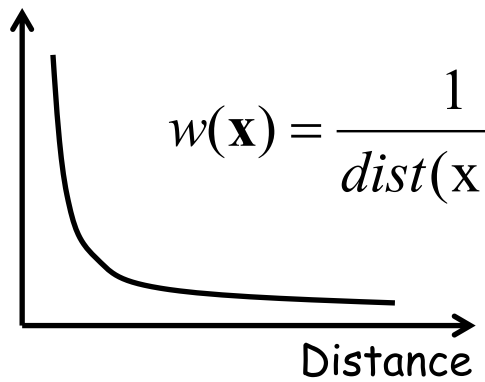
- **Regression: Not just averaging**

$$f(\mathbf{x}') = \frac{\sum_{\mathbf{x} \in N(\mathbf{x}')} w(\mathbf{x}) \cdot f(\mathbf{x})}{\sum_{\mathbf{x} \in N(\mathbf{x}')} w(\mathbf{x})}$$

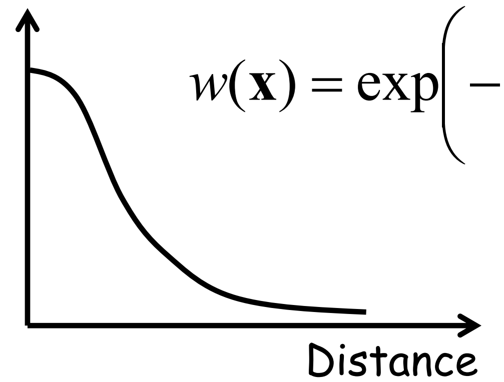
$N(\mathbf{x}')$: the nearest neighbors of $\mathbf{x}'$

Variation

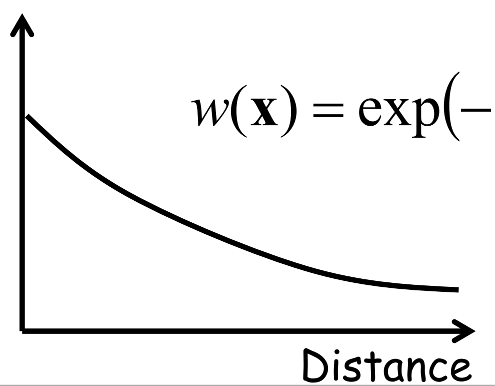
- **How to determine weight considering distance**



$$w(\mathbf{x}) = \frac{1}{dist(\mathrm{x}, \mathrm{x}')}$$



$$w(\mathbf{x}) = \exp\left(-\frac{dist(\mathrm{x}, \mathrm{x}')^2}{2\lambda^2}\right)$$

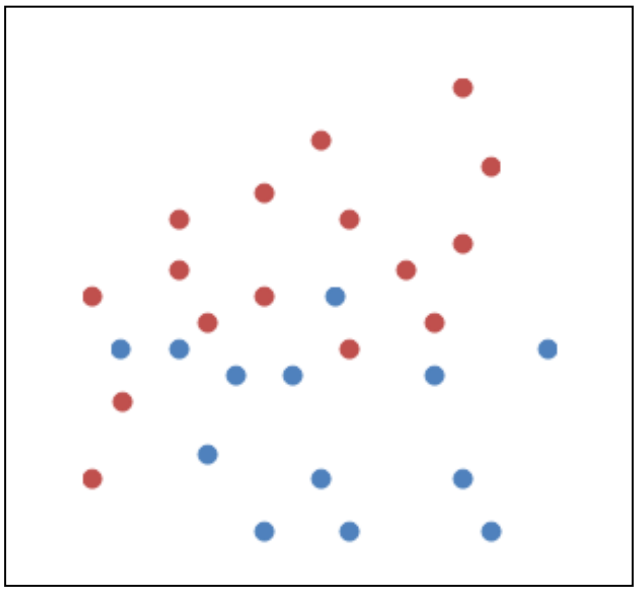


$$w(\mathbf{x}) = \exp\left(-dist(\mathrm{x}, \mathrm{x}')\right)$$

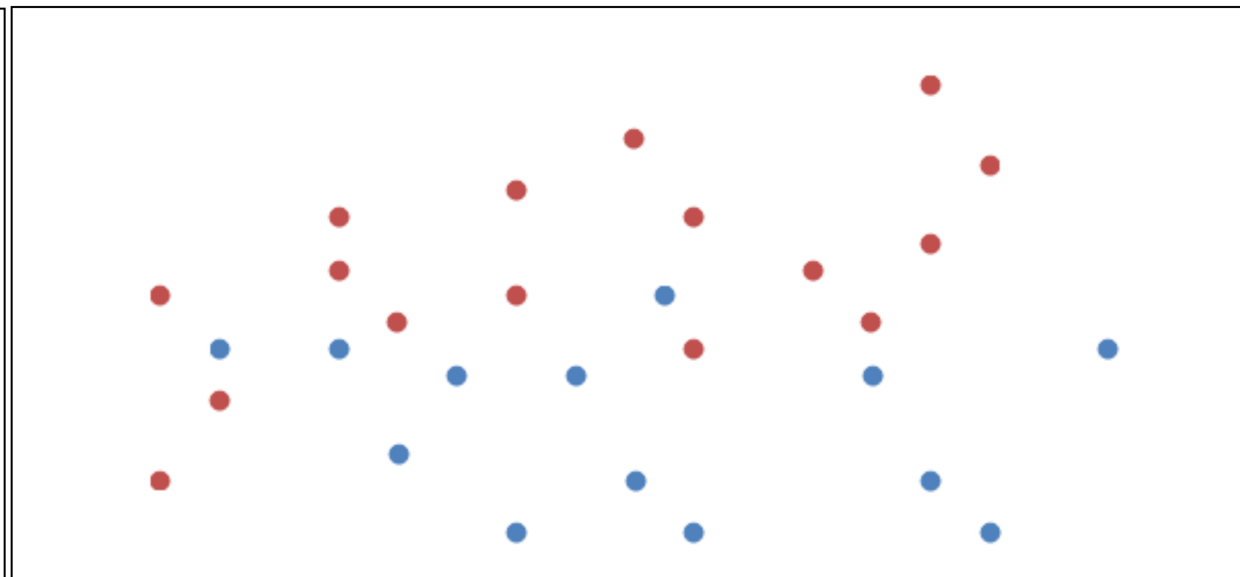
Do we need to choose k NNs?
-Yes, if you want
-Not necessarily

Distance Measure

- **Two data sets**
  - Compare the boundaries in D1 and D2 (1-NN)



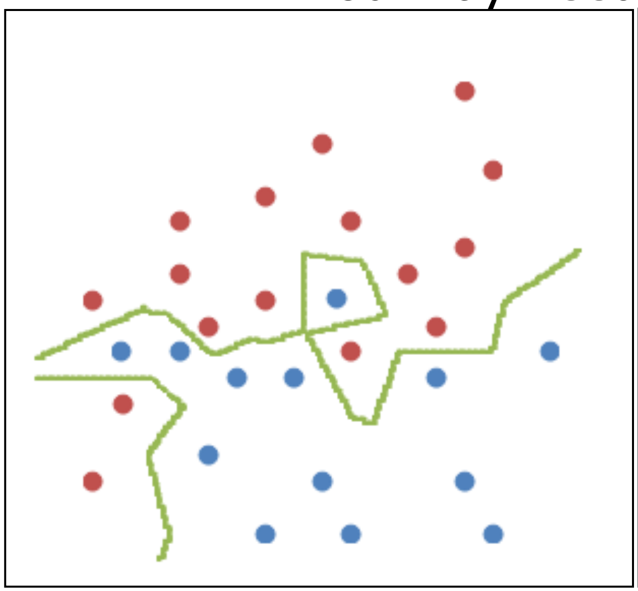
D1



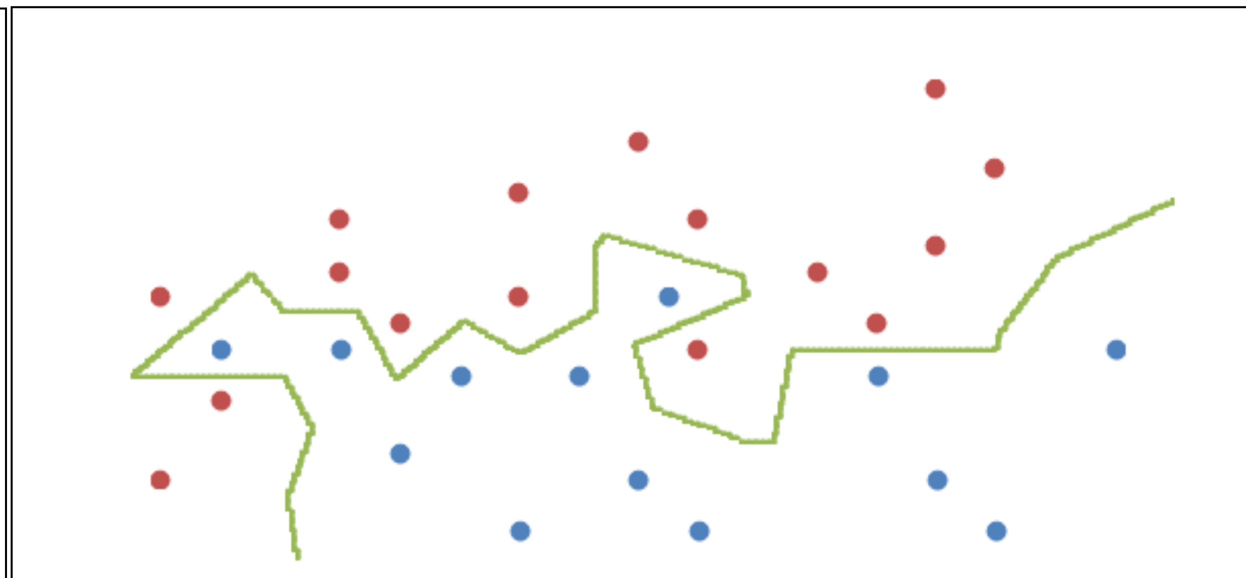
D2 $D2 = \{(x, y) \mid x = 2x', (x', y) \in D1\}$

Distance Measure

- **Two data sets**
  - Even though the data are linearly scaled, the boundary changes!!
  - You may need to choose other distance measures



D1



D2

$$D2 = \{(x, y) \mid x = 2x', (x', y) \in D1\}$$

Summary

- **Which k is better?**
  - Small k : higher variance (less stable) -> possibly overfitted
  - Large k : higher bias (less precise) -> possibly underfited

- **Proper choice of k**
  - Depending on the data
  - Use Cross-validation

Summary

- **Advantage**
  - No training (Only inference step)
  - Complexity of target functions do not matter
  - No loss of information

- **Disadvantage**
  - Have to keep all data -> Memory space
  - Sensitive to noise
  - If training data is imbalanced, major class may dominate
  - Need to calculate the distance from all training data -> Time
    - Especially in high dimensional space, expensive

Summary

- **Reducing Computational Cost**
  - Finding k nearest neibhbors is expensive: O(nd)
  - Space partitioning
    - quad-tree, locality sensitive hashing, etc.
  - Preprocessing
    - Reduce dimensions: Remove less important features, Vector quantization
    - Reduce size of data: Sampling, Clustering