CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

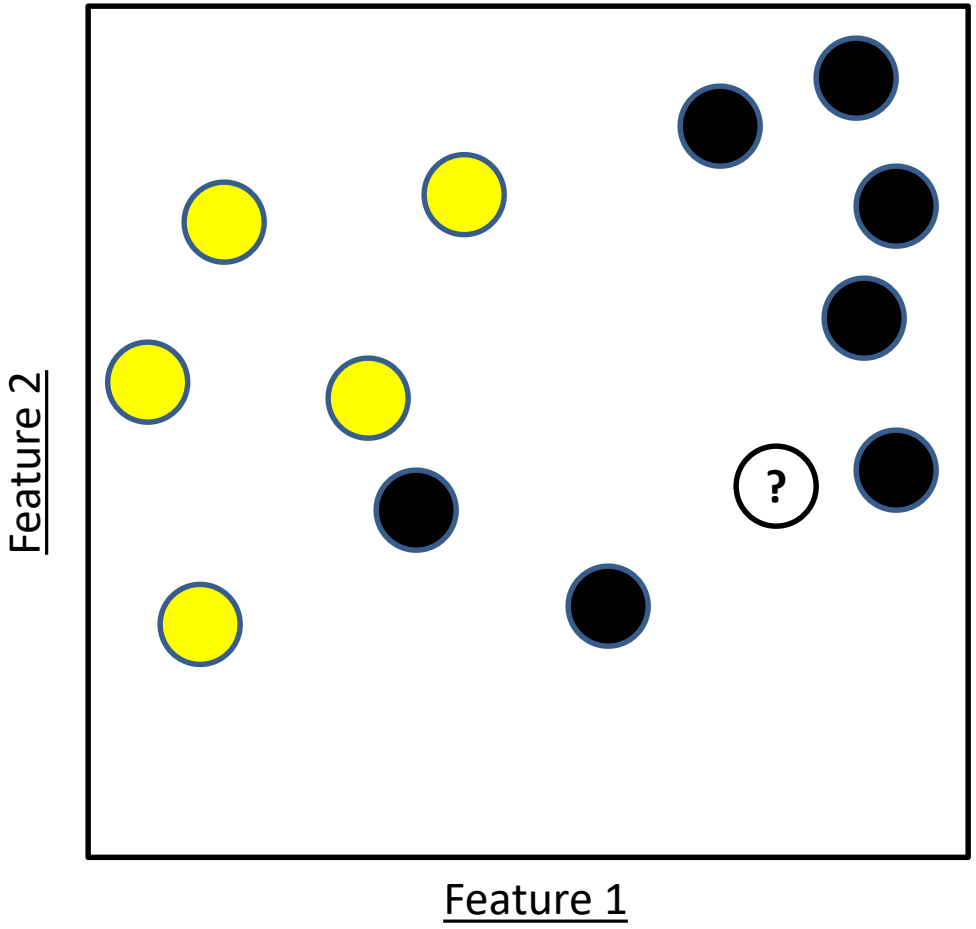
Machine Learning

Lecture: Instance-Based Learning (Part 2)

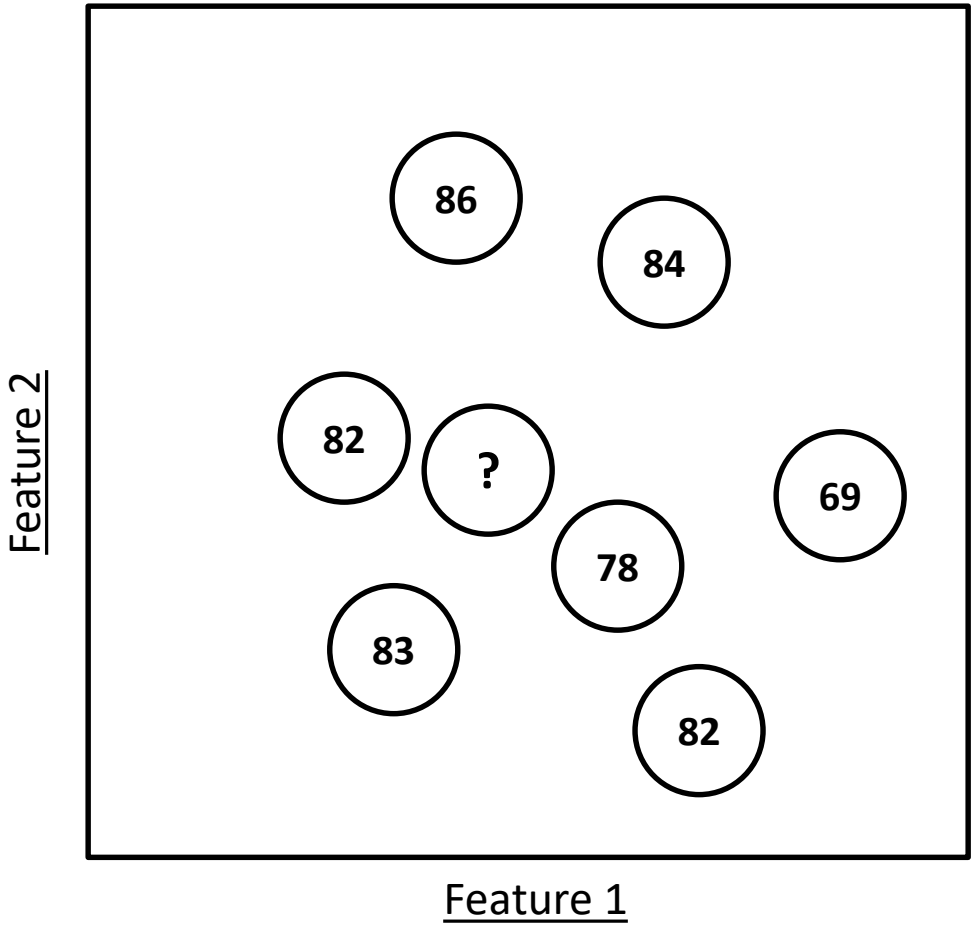
Ted Scully

Nearest Neighbour Example

Feature 1	Feature 2	Colour



Feature 1	Feature 2	Regression Target



Euclidean Distance Metric

- To help illustrate the various metrics let's assume we have the dataset below with n features and two instances p and q

	Feature 1	Features 2	….	Feature n
p	p_1	p_2	……	p_n
q	q_1	q_2	…….	q_n

If **p** = <p_1, p_2,..., p_n> and **q** = <q_1, q_2,..., q_n> are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by

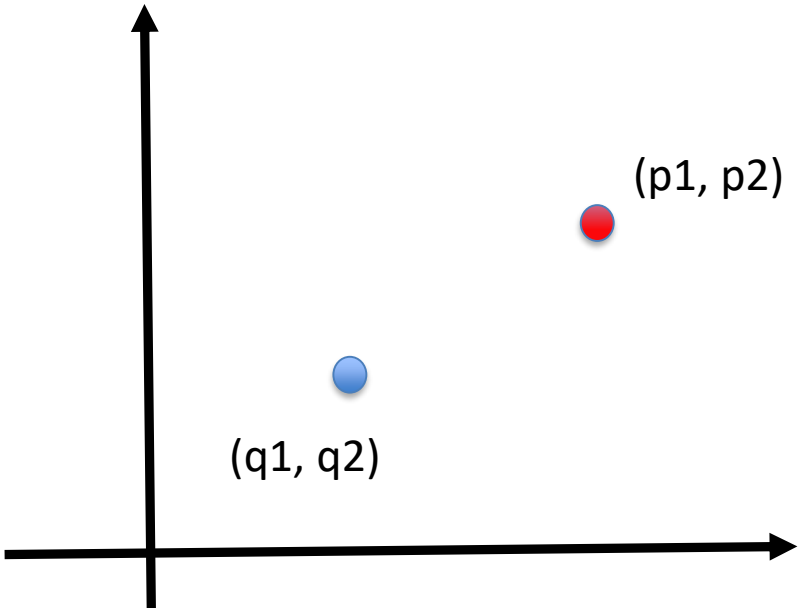
$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

Manhattan Distance Metric

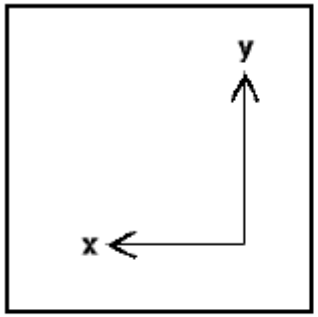
- Manhattan distance measures distance **parallel to each axis**, not diagonally (in downtown Manhattan, to get from one point to another you generally walk North-South and East-West, rather than 'as the crow flies').

- In other words take the sum of the absolute values of the differences of the coordinates
- $d(p, q) = |q_1 - p_1| + |q_2 - p_2| + |q_n - p_n| = \sum_{i=1}^{n} |q_i - p_i|$

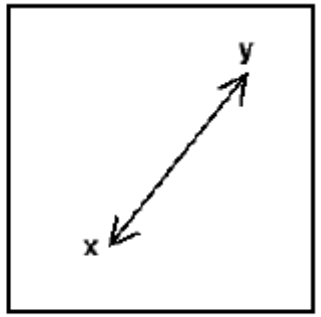


Manhattan Distance Metric

- Lets assume we have a simple dataset containing three instances as follows. We are also given the query instance below. Calculate the Manhattan and Euclidean distance between the **first training example** and the **query**



Manhattan



Euclidean

Area	Weight	Height	Capacity
10	8	4	14
12	10	6	12
14	9	4	11

Area	Weight	Height	Capacity
5	4	4	10

Area	Weight	Height	Capacity
10	8	4	8
12	10	6	12
14	9	4	11

Area	Weight	Height	Capacity
5	4	4	3

Manhattan Distance Metric

- Euclidean
  - $(10-5)^2+(8-4)^2+(4-4)^2+(8-3)^2$
  - 57
  - Square root of 57 = 7.55
- Manhattan
  - |10-5|+|8-4|+|4-4|+|8-3|
  - 5+4+0+5 = 14

Area	Weight	Height	Capacity
10	8	4	8
12	10	6	12
14	9	4	11

Area	Weight	Height	Capacity
5	4	4	3

Minkowski Distance

- The Minkowski distance between a feature vector **p** = <p_1, p_2,..., p_n> and another feature vector **q** = <q_1, q_2,..., q_n> is defined as:

- $d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^a\right)^{\frac{1}{a}}$

- In the above equation *a* is an integer. Consider the case when a = 1 or a = 2. Any comments.

Minkowski Distance

- The Minkowski distance between a feature vector **p** = <p_1, p_2,..., p_n> and another feature vector **q** = <q_1, q_2,..., q_n> is defined as:

- $d(p,q) = (\sum_{i=1}^{n} |p_i - q_i|^a)^{\frac{1}{a}}$

- In the above equation *a* is an integer. Consider the case when a = 1 or a = 2. Any comments.
  - For a = 1 we get the Manhattan distance and for a = 2 we get the Euclidean distance.
  - Minkowski Distance is a generalization of the Euclidean and Manhattan distance metrics.

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^a\right)^{\frac{1}{a}}$$

Larger values of *a* place more emphasis on large differences between feature values compare to smaller values. This is because the differences are raised to the power of *a*. Therefore, the Euclidean distance weights features with larger differences between feature value influence the final distance metric more than features with a smaller difference.

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^a\right)^{\frac{1}{a}}$$

As you might expect as you begin to decrease a the opposite effect happens. **The features with larger differences don't have the same level of impact.**

Problems Measuring Distance 1 -Scale

- Since the performance of k-NN is strongly dependent on the choice of distance metric, you need to be aware of some pitfalls.

- The first problem arises when the **features are different** from each other.
- For example, if one feature has a range between **0 and 1** and another feature has a range between **0 and 10, 000**, it hardly makes sense to add them as would happen with Euclidian or Manhattan distance metrics (for example, salary and age).

- What is the main problem that arises from the above situation?

Problems Measuring Distance (1)

- **Problem 1: Scaling**
  - Feature A has range 1-10
    Feature B has range 1-1000
  - Feature B will dominate calculations

- Example, lets calculate the distance between data instance 1 and 2
  - Data instance 1 = (5.5, 787)
  - Data instance 2 = (7.5, 567)

Problems Measuring Distance (1)

- **Problem 1: Scaling**
  - Feature A has range 1-10
    Feature B has range 1-1000
  - Feature B will dominate calculat
- Example, lets calculate the distance
  - Data instance 1 = (5.5, 787)
  - Data instance 2 = (7.5, 567)

We can see below that the second feature is entirely dominating the distance calculation simply because it has a larger range of values compared to the first feature.

We don't want our model to bias toward a particular feature simply because the range happens to be larger.

$$\sqrt{(5.5-7.5)^2+(787-567)^2}$$

$$\sqrt{16+48400}$$

Problems Measuring Distance (1)

- Solution:
  - Normalise all dimensions independently (scale data so that it has a maximum and minimum range)
  - Using range normalization we identifying the minimum and maximum value for a specific feature. We can then apply the following formula.
  - $$\boldsymbol{newValue} = \frac{\boldsymbol{originalValue} - \boldsymbol{minValue}}{\boldsymbol{maxValue} - \boldsymbol{minValue}}$$

- $$newValue = \frac{originalValue - minValue}{maxValue - minValue}$$

- Problem 1: Scaling
  - Feature A has range 1-10
    Feature B has range 1-1000
- Normalise variables
  - Feature A
    - (5.5 - 1)/(10-1) = 0.5
    - (7.5 -1)/(10 -1) = 0.72
  - Feature B
    - (787-1)/(1000-1) = 0.78
    - (567-1)/(1000-1) = 0.56

- Before Normalization
  - Data instance 1 = (5.5, 787)
  - Data instance 2 = (7.5, 567)

- After Normalization
  - Data instance 1 = (0.5, .78)
  - Data instance 2 = (0.72, 0.56)

- $newValue = \frac{originalValue - minValue}{maxValue - minValue}$

- Problem 1: Scaling
  - Feature A has range 1-10
    Feature B has range 1-1000
- Normalise variables
  - Feature A
    - (5.5 - 1)/(10-1) = 0.5
    - (7.5 -1)/(10 -1) = 0.72
  - Feature B
    - (787-1)/(1000-1) = 0.78
    - (567-1)/(1000-1) = 0.56

- Before Normalization
  - Data instance 1 = (5.5, 787)
  - Data instance 2 = (7.5, 567)

- After Normalization
  - Data instance 1 = (0.5, .78)
  - Data instance 2 = (0.72, 0.56)

$$\sqrt{(0.5-0.72)^2+(0.78-0.56)^2}$$

$$\sqrt{0.048+0.048}$$

Problems Measuring Distance (1)

- When we normalize the train data, it is also important to understand:
  - We normalize each feature **independently**
  - We must normalize the **test data** using the same parameters for max and min (that is we still use the minValue and maxValue from the original training set).

Problems Measuring Distance (2)

- Another problem with kNNs when measuring distance is that each individual feature is treated as being equally important. In other words each feature contributes equally to the determination of the final class.

- Of course certain features may be weak predictors and others strong predictors.

- We could look at this as an optimization problem where we assigns an weight to each feature and we optimize the weights to maximize model performance (computationally expensive).

- Typically, feature selection is performed before the data is even fed into the model in the first place.

Limitation of Standard KNN

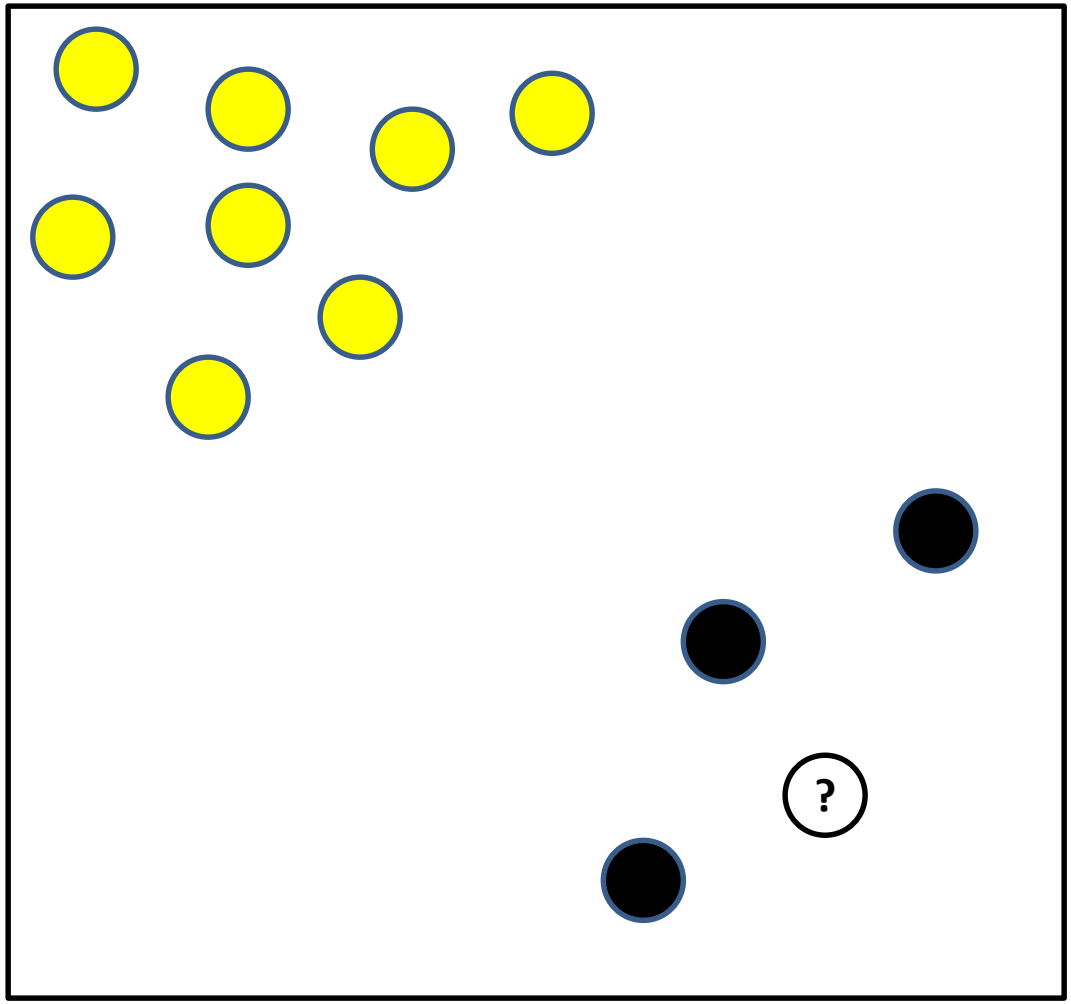
Assume in the following example that we are using **k=7**?

What are the potential problems?

What might you do to address this problem?

Distance-Weighted kNN

Give each neighbour weight: inverse of distance from target



Distance Weighted k-NN (Regression)

- It is the vote of each neighbour that is weighted according to how close it is to the target, so **closer neighbour's influence the prediction more**.

- We can if we wish use all cases as neighbours, since those very far from the target will have little influence on the prediction, but none will be completely ignored.
- We do the following for a regression problem.

Given a query instance x_q,

$$f(\mathbf{x}_q) := \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

Where

$$w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^n}$$

Distance Weighted k-NN (Regression)

- It is the vote of each neighbour that is weighted according to how close it is to the target, so **closer neighbour's influence the prediction more**.

- We can if we wish use all cases as neighbours, since those very far from the target will have little influence on the prediction, but none will be completely ignored.
- We do the following for a regression problem.

Given a query instance x_q,

$$f(\mathbf{x}_q) := \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

Where

$$w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^n}$$

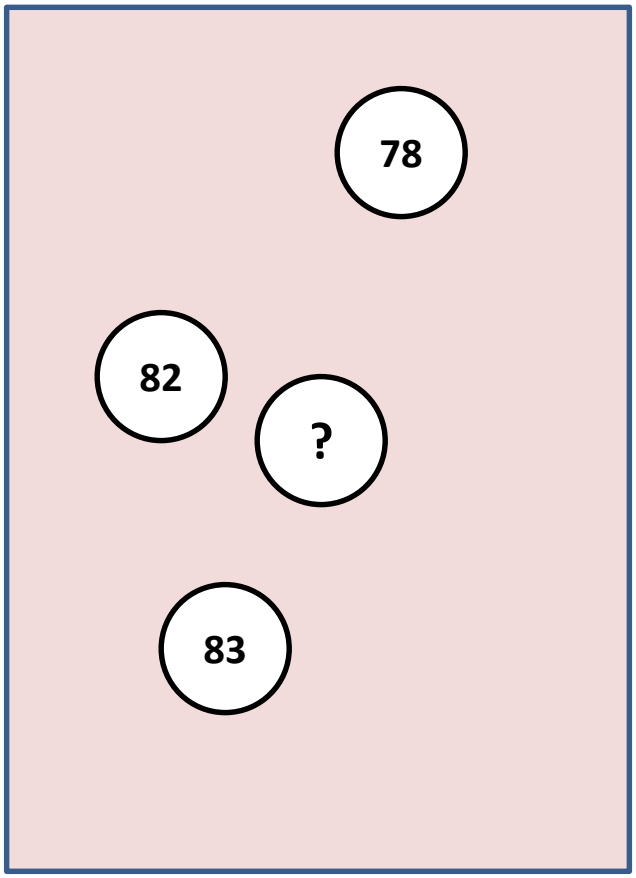
Here you will notice that we use the inverse distance to the power of n. Selecting a value n=2 is typical. However, we can also use n=1, n=3, etc

Distance Weighted k-NN Algorithm Regression Example

$$f(\mathbf{x}_q) := \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

- Consider the basic regression example depicted in the slide.
- Distance between query instance and:
  - Case 82 is 2 units
  - Case 83 is 4 units
  - Case 78 is 6 units

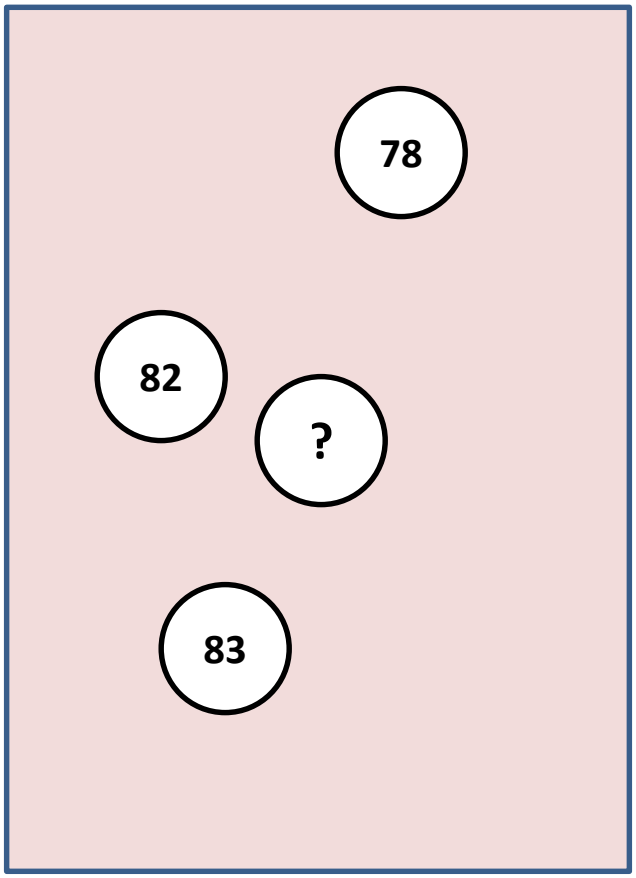
- What is the value of the query instance.



Distance Weighted k-NN Algorithm Regression Example

$$f(\mathbf{x}_q) := \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}$$

- Consider the basic regression example depicted in the slide.
- Distance between query instance and:
  - Case 82 is 2 units
  - Case 83 is 4 units
  - Case 78 is 6 units



- What is the value of the query instance.

- ((1/4)(82) + (1/16)(83) + (1/36)(78)) / (1/4 + 1/16 + 1/36)
- = 27.854/0.3402
- =81.856

Distance Weighted k-NN (Classification)

- We iterate through each class. For a specific class we identify each of the instances amongst the k nearest instances that belong to that class. We then add up the inverse distance for each of the identified instances.

- The class that results in the largest value is the selected class for the new query instance.

$$vote(c_j) := \sum_{i=1}^{k} \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^n} (c_i, c_j)$$

(y_i, y_j) returns 1 if the class labels match and 0 otherwise

What do you think will be the impact of n (n must be a positive number greater than or equal to 1)

Vote(Purple Class) (n=1)

Purple

1/10 +
1/9 +
1/5 = 0.41

Yellow
1/5+
1/6+
1/5 = 0.566

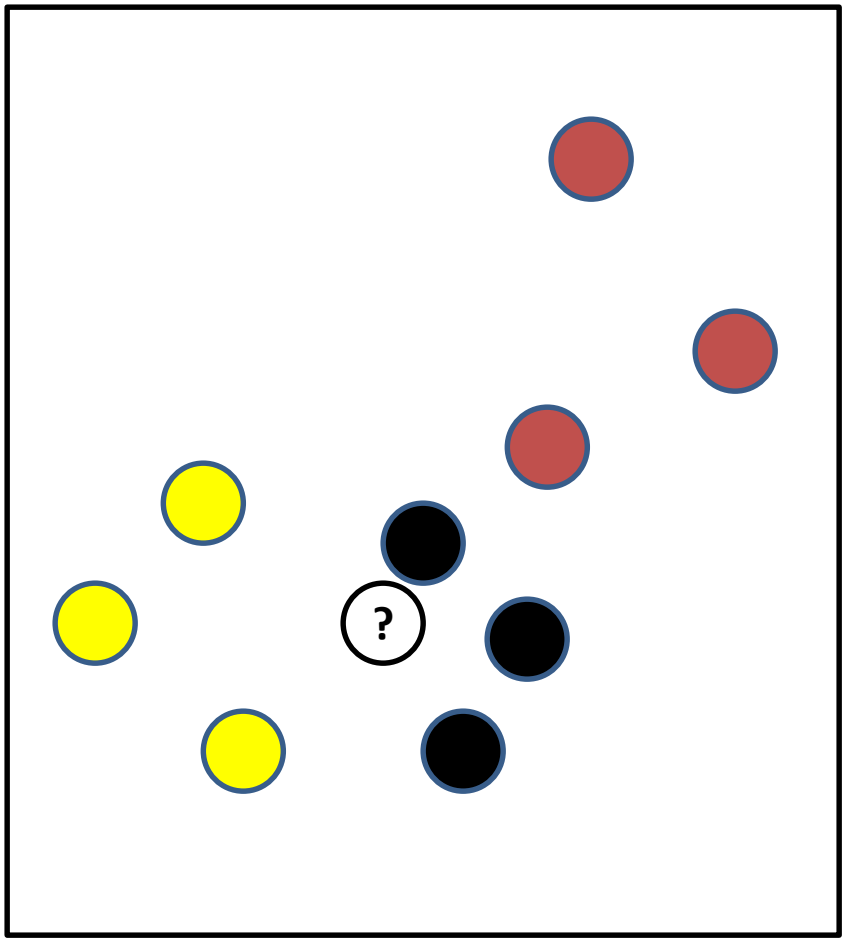
Black
1/1+
1/2+
1/2= 2

Purple

10
9
5

Yellow
5
6
5

Black
1
2
2



$$vote(c_j) := \sum_{i=1}^{k} \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^n} (c_i, c_j)$$

Result of Voting

- Result of voting is
- V(Purple) = 0.41
- V(Yellow) = 0.566
- V(Black) = 2

- Therefore the query instance is classified as a **Black class.**

Assessing the Performance of a Regression Model

- We have seen that we can apply a kNN to both classification and regression problems.

- However, so far we have only used a basic measure for classification performance (which is accuracy)
- Accuracy = number of test instances correctly classified/ total number of test instances.

- Later in the module we will look more comprehensively at a range of evaluation metrics.

- So what is a common **evaluation metric** we can use for **regression**?

Basic Measures of Error (Regression)

- The **R^2** coefficient compares the **performance of a model on a test set (sum of squared residuals)** with the performance of an imaginary model that always predicts the **average values from the test set (total sum of squares)**.

- The **R^2** coefficient is calculated as:

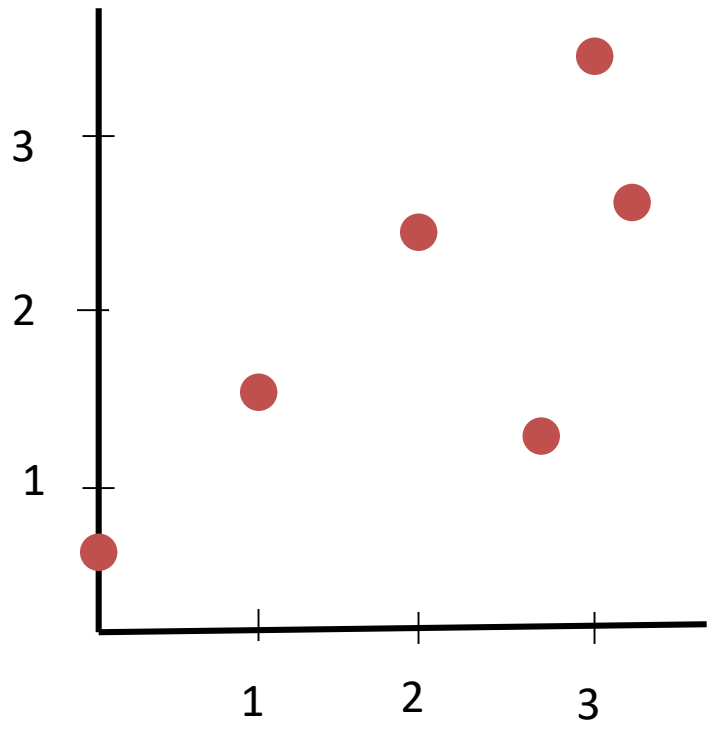
$$R^2 = 1 - \frac{sum\ of\ squared\ residuals}{total\ sum\ of\ squares}$$

- Where

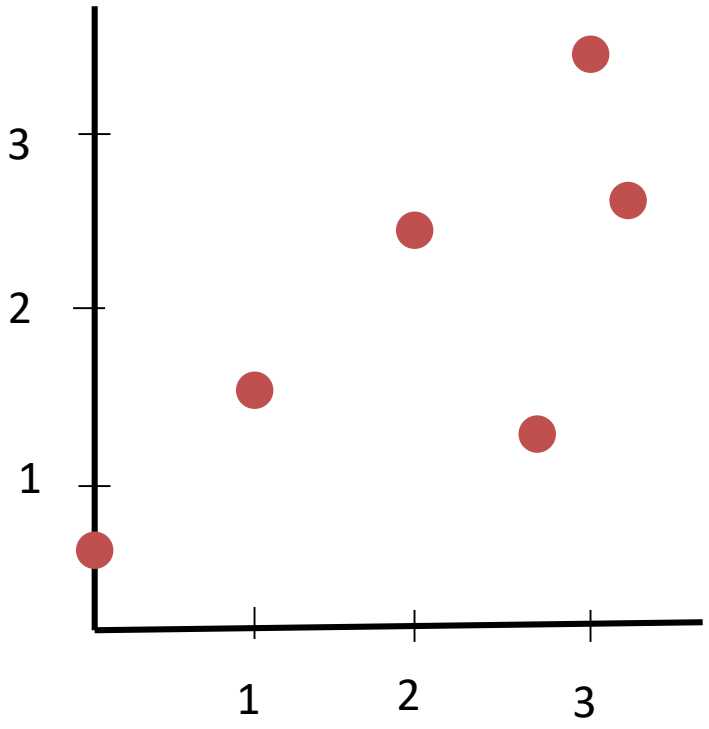
$$sum\ of\ squared\ residuals = \sum_{i=0}^{m} (f(x^i) - y^i)^2$$

$$total\ sum\ of\ squares = \sum_{i=0}^{m} (\bar{y} - y^i)^2$$

$$total\ sum\ of\ squares = \sum_{i=0}^{m} (\bar{y} - y^i)^2$$



$$sum\ of\ squared\ residuals = \sum_{i=0}^{m} (f(x^i) - y^i)^2$$



Basic Measures of Error (Regression)

- The **R^2** coefficient values typically fall in the range [0, 1) and larger values indicate better model performance.

- The **worse the model** produced, the closer the sum of square residuals value will be to the total sum of squares value. Consequently the **smaller the total R^2**.

- The **better the model** the smaller the squared residuals (smaller error in the model) and the **larger the R^2** value.

- While it is rare, the model produced could be worse than the total sum of squares. In this case the R^2 would be **negative**. The worse the model the lower the R^2 values. It means that whatever model that you came up with is worse than predicting the mean (not a good sign!).

When to use k-Nearest Neighbour

- Primary Benefits
  - **Comprehensibility**: easy to understand
  - Relatively straight-forward to **implement**
  - Can easily handle **multi-class** datasets
  - Effective classifiers for **complex target functions** => good for diverse concepts.
  - Can be used for both **regression and classification** problems

- Consider using when:
  - Moderate number of **training instances**
  - Moderate number of **features** per instance (< 20) [Note: If dealing with datasets with a large number of features we can perform dimensionality reduction and still use a k-NN approach]

When to use k-Nearest Neighbour

- Drawbacks:
  - Can only accommodate a **moderate number of features** (typically 20 or less)
  - Problems with **irrelevant features**. By default all features contribute equally to the similarity metric.
  - Can be very sensitive to **imbalanced datasets**.
  - It is called instance-based because it constructs hypotheses directly from the training instances themselves. This means that the hypothesis complexity can grow with the size of the data. Therefore it can be **slow in classification/regression if there is a large dataset.**

Curse of Dimensionality

- After over-fitting, the biggest problem in machine learning is the “curse of dimensionality”.

- The curse of dimensionality refers to various issues that arise when analyzing data in high-dimensional spaces (potentially with hundreds or even thousands of dimensions) and is particularly relevant to instance based learners.

- The common root of these difficulties lies in the fact that **as dimensionality increases, the volume of space increases so fast that the available data becomes sparse.**
  - This raises particular problems for ML techniques that rely on detecting areas where objects form groups with similar properties.

Curse of Dimensionality

- Many algorithms that work fine in low dimensions become intractable when the input is high-dimensional.
- Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the examples grows, because **a fixed-sized training set covers a dwindling fraction of the input space**.