CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Machine Learning

Machine Learning

Lecture: Instance-Based Learning

Ted Scully

Instance Based Learning

- Instance-based learning is a family of learning algorithms that **compare new problem instances with existing instances in the training data.**

- Predictions for new instances are based on their **similarity to stored instances** (the basis of the similarity measure is typically distance)

Nearest Neighbour Algorithm (1)

- The Nearest Neighbour algorithm is the simplest form of IBL
- Nearest Neighbour algorithm:
  - Given a test case with a value to be predicted, identify which stored case it is nearest.
  - Assigns the **new test case the same class as the nearest neighbour**
  - Requires a distance metric.
- This very simple algorithm is very susceptible to noise.

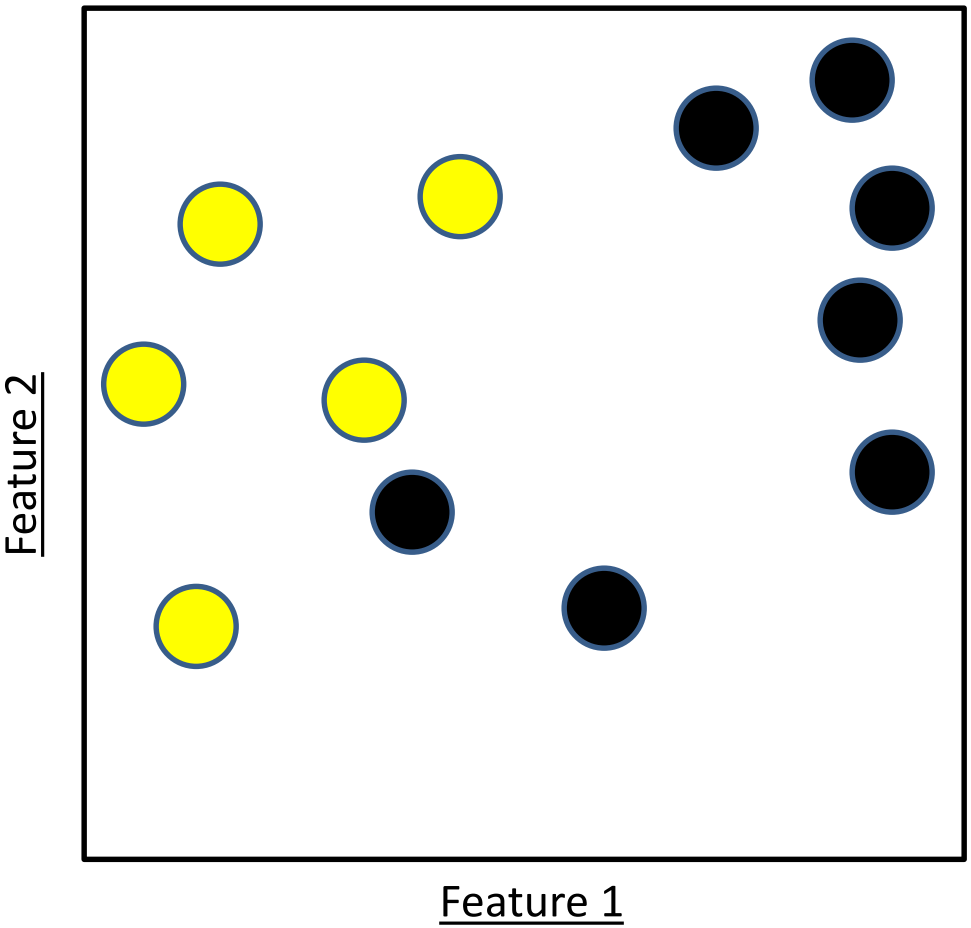
Given a query instance $\mathbf{x}_q$,

first locate the nearest training example $\mathbf{x}_n$

then $\mathbf{f}(\mathbf{x}_q) := \mathbf{f}(\mathbf{x}_n)$

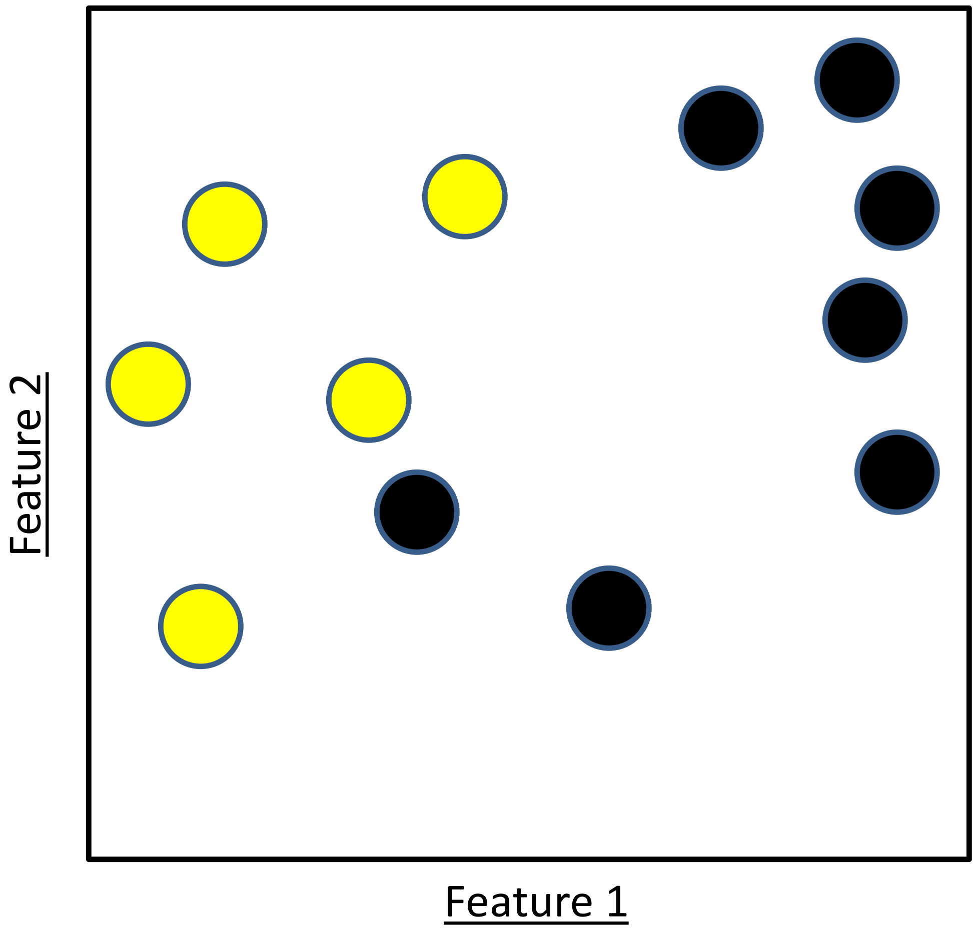
Where $\mathbf{f}(\mathbf{x}_n)$ is the class associated with the data item $\mathbf{x}_n$

Feature 1	Feature 2	Colour



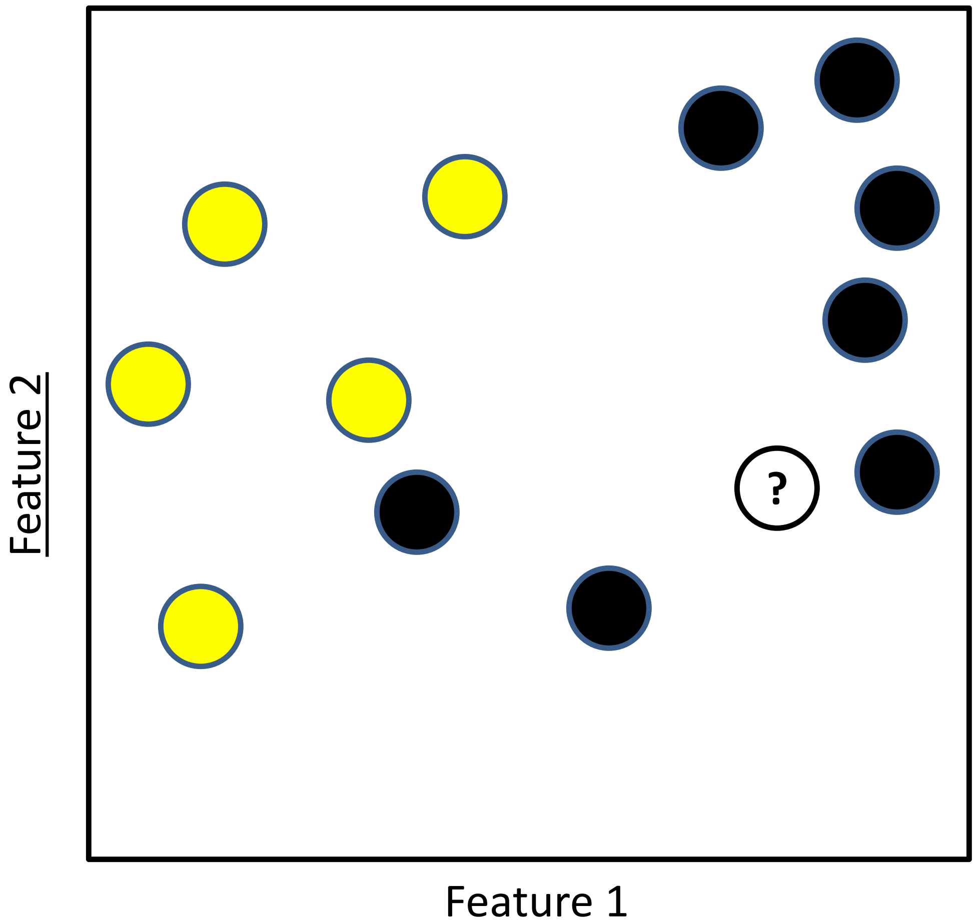
Feature 1	Feature 2	Colour

- We can represent a dataset in an IBL by mapping all instances to a **feature space,** that is, using each descriptive feature as an axis of a coordinate system.
- We can then place each instance within the feature space based on the value of it's features.

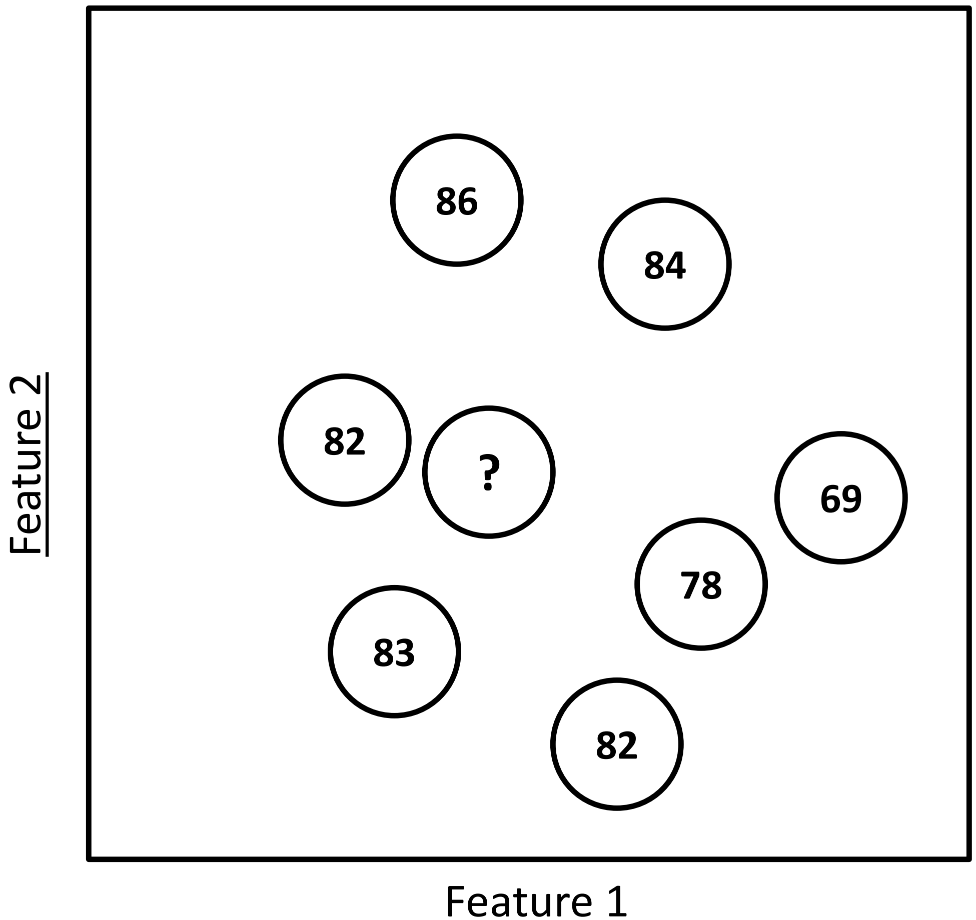


Nearest Neighbour Example

- We wish to classify the new case, which is the white circle with the question mark.
- The nearest neighbour in feature space is selected and the example instance is assigned the same class.



Feature 1	Feature 2	Regression Target



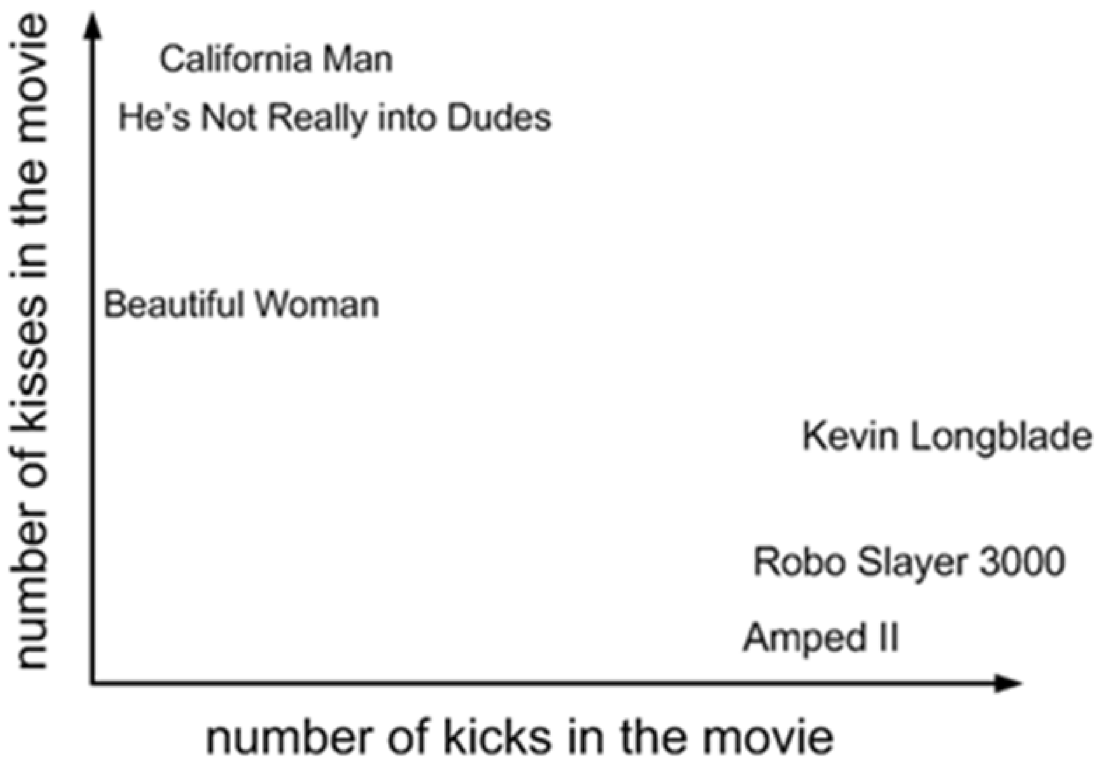
Sample Dataset

Movie title	# of kicks	# of kisses	Type of movie
California Man	3	104	Romance
He's Not Really into Dudes	2	100	Romance
Beautiful Woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action
Amped II	98	2	Action

Machine Learning In Action – Peter Harrington

Feature Space

Here we can see the feature space for our simple movie dataset.

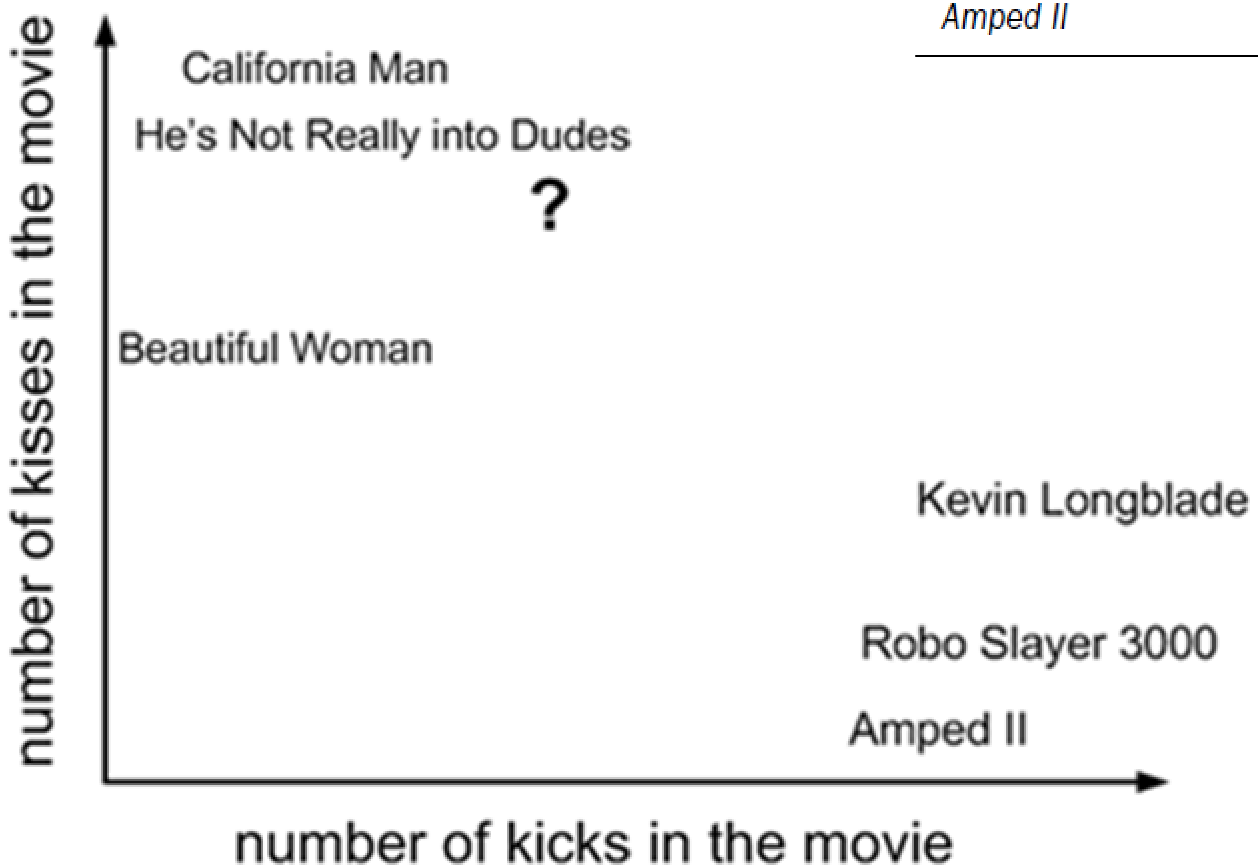


Movie title	# of kicks	# of kisses	Type of movie
California Man	3	104	Romance
He's Not Really into Dudes	2	100	Romance
Beautiful Woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action
Amped II	98	2	Action
?	18	90	Unknown

Assume we get an unseen movie and we have to classify it as a Romance or action based on it's feature values.

Given a query instance $\mathbf{x}_q$, first locate the nearest training example $\mathbf{x}_n$ then $\mathbf{f}(\mathbf{x}_q) := \mathbf{f}(\mathbf{x}_n)$

Movie title	Distance to movie "?"
California Man	20.5
He's Not Really into Dudes	18.7
Beautiful Woman	19.2
Kevin Longblade	115.3
Robo Slayer 3000	117.4
Amped II	118.9



As the query is closest to the film "He's Not Really into Dudes" then it is classified as a **Romance**

Movie title	# of kicks	# of kisses	Type of movie
California Man	3	104	Romance
He's Not Really into Dudes	2	100	Action
Beautiful Woman	1	81	Romance
Kevin Longblade	101	10	Action
Robo Slayer 3000	99	5	Action
Amped II	98	2	Action
?	18	90	Unknown

Noise: An incorrect classification

What would happen if "He's no really into dudes" was incorrectly classified as an Action movie.
Our new query instance '?' would also get **incorrectly classified** as an Action.

The simple nearest neighbour approach is very likely to over-fit on the training data.

Any ideas of how we might go about improving the algorithm?

K-Nearest Neighbour

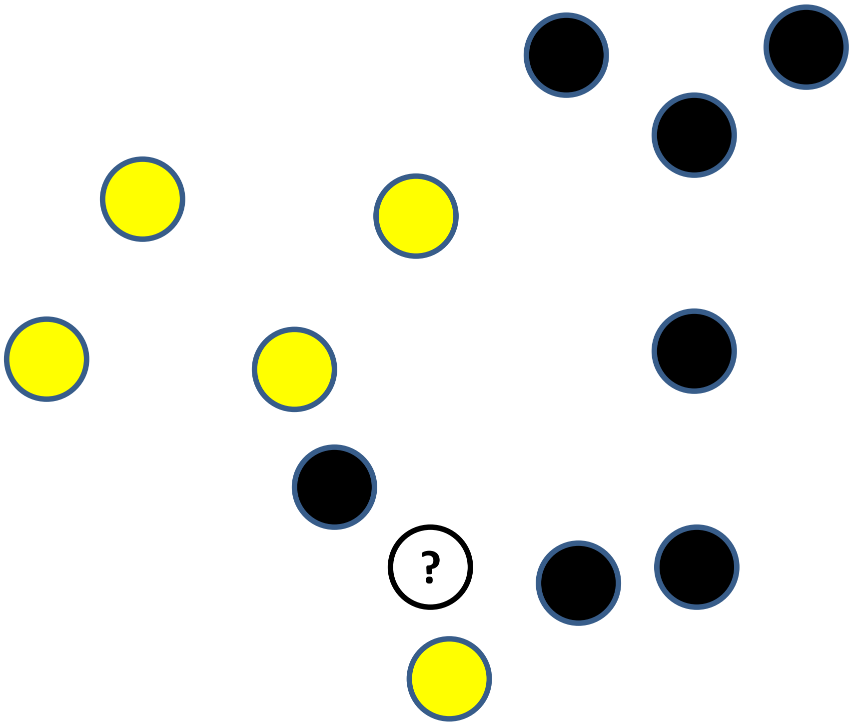
- A simple extension is to consider not just the nearest neighbour, but **several nearest neighbours**.
- This requires defining a neighbourhood; the standard approach is to use a neighbourhood that is just large enough to include a fixed number of points, k.
- But how do we decide on **appropriate class** or **regression value** if we are consider more than one neighbour in feature space???

K-Nearest Neighbour

- A simple extension is to consider not just the nearest neighbour, but **several nearest neighbours**.
- This requires defining a neighbourhood; the standard approach is to use a neighbourhood that is just large enough to include a fixed number of points, k.
- Prediction is based on these k nearest neighbours.
  - If this is a **regression** problem then use the **average** of k-nearest neighbours.
  - If it is a **classification** problem then take a **vote** amongst the k-nearest neighbours
  - This approach is less sensitive to noise.

k - Nearest Neighbour Algorithm - Classification

- k–NN can be applied to classification problems
- If it is a classification problem then take a **majority vote** amongst the k-nearest neighbours
- What is the classification of the query instance if k =3? What if k = 5



k - Nearest Neighbour Algorithm - Regression

We assume a set of training examples <x_i, $f(x_i)$>

Given a query instance x_q,

Identify k nearest training examples

If it is regression problem, then average values of k-nearest neighbours

$$f(x_q) := \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

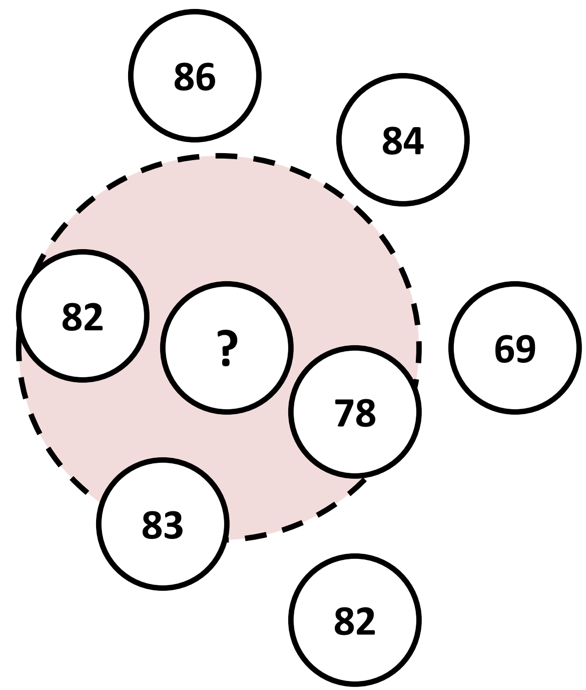
k - Nearest Neighbour Algorithm - Regression

- k – NN can be applied to regression problems

- The answer is the <u>average</u> value of k of the neighbours

- What is the answer if: k = 3?

  (82+78+83)/3 = 81.

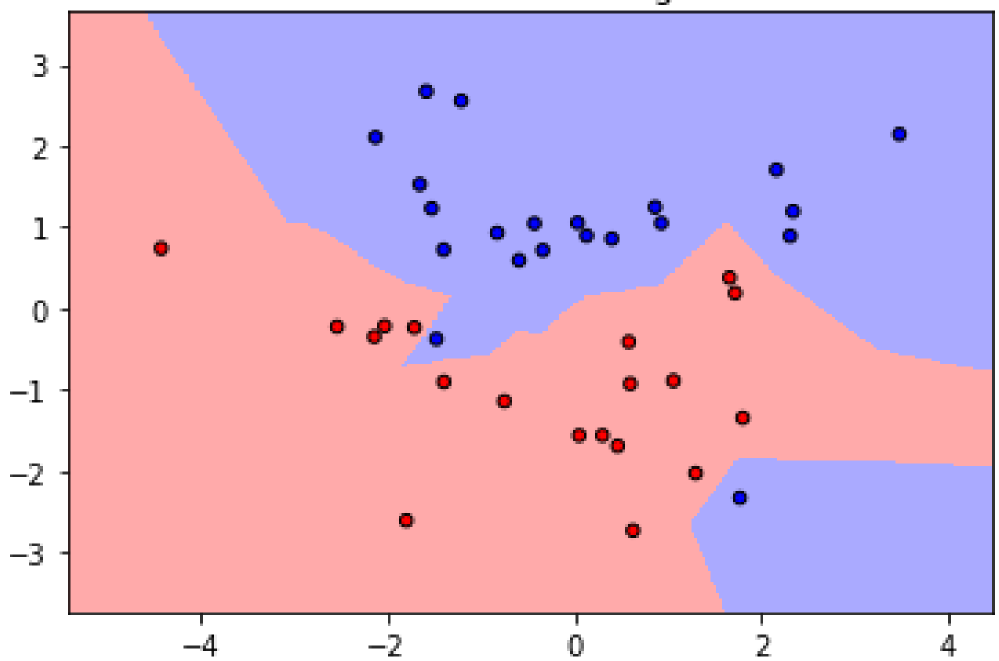


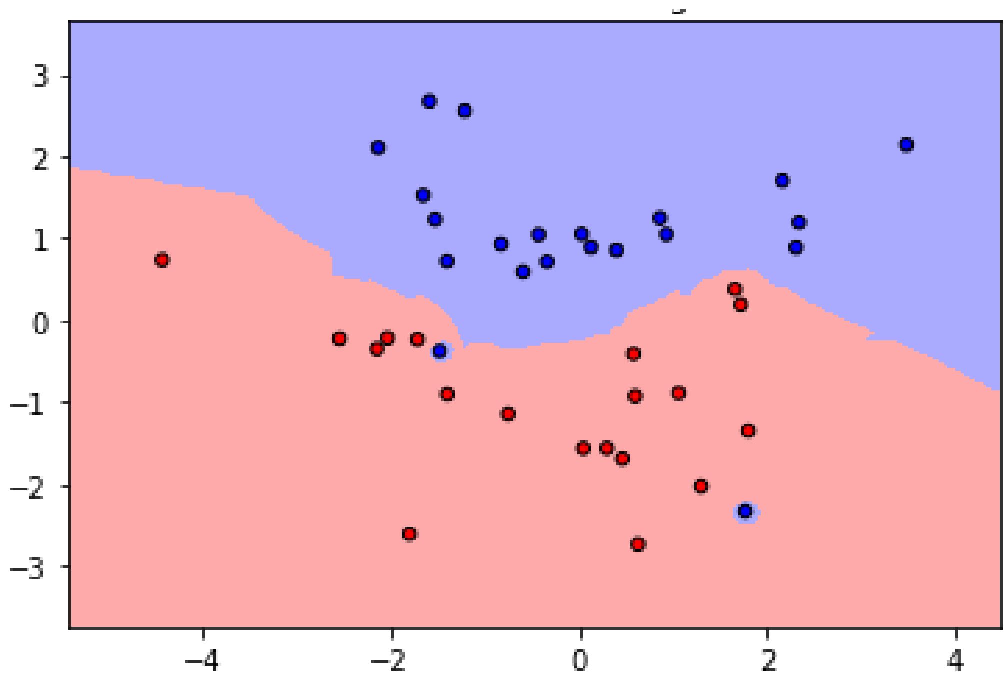
Selecting an Appropriate K Value

- The selection of an appropriate value of k is very important. The parameter k is what we refer to as a hyper-parameter
- Selecting **too small** a value can make the algorithm **susceptible to noise** and can **overfit on the training data**.

- Selecting larger values of k lessen the impact of noise on the classification but make **boundaries between classes less distinct (in other words the model can underfit and the boundary fails to capture the patterns in the data)**.

- There are many techniques for selecting a k value.
  - Certain rules-of-thumb such as use the square root of the number of classified instances [<u>not recommended</u>].
  - Instead you should select a **range of different k values** and assess the performance of your model for these values (later when we cover scikit learn we will look at using N-fold cross validation and search to identify good values for k)


3
2
1
0
−1
−2
−3
−4
−2
0
2
4

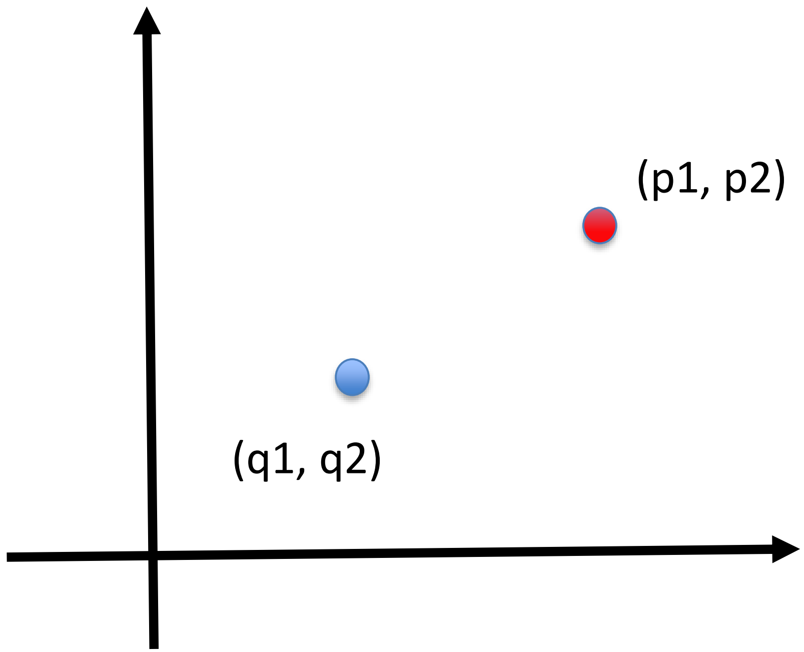

3
2
1
0
−1
−2
−3
−4
−2
0
2
4

Distance Metrics

- An important aspect of k-NN algorithms is how we determine which instances are the nearest to the target case. Thus, the <u>distance metric is a measure of the similarity between two cases</u>.

- Common distance metrics include:
  - Euclidean
  - Manhattan
  - Minkowski

Distance Metrics - Euclidean



- 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

Euclidean Distance Metric

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

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

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

It is important to understand that p and q here represent two data instances. Each instance consisting of a finite set of features. The instance q has the features q_1, q_2, ... q_n.

Euclidean Distance Metric

Movie title	# of kicks	# of kisses	
California Man	3	104	
He's Not Really into Dudes	2	100	
Beautiful Woman	1	81	
Kevin Longblade	101	10	
Robo Slayer 3000	99	5	
Amped II	98	2	Action
?	18	90	Unknown

Movie title	Distance to movie "?"
California Man	20.5
He's Not Really into Dudes	18.7
Beautiful Woman	19.2
Kevin Longblade	115.3
Robo Slayer 3000	117.4
Amped II	118.9

Distance between **California man** (3, 104) and the **query** instance (18, 90) would be:

Euclidean Distance Metric

Movie title	# of kicks	# of kiss	
California Man	3	104	
He's Not Really into Dudes	2	100	
Beautiful Woman	1	81	
Kevin Longblade	101	10	
Robo Slayer 3000	99	5	
Amped II	98	2	Action
?	18	90	Unknown

Movie title	Distance to movie "?"
California Man	20.5
He's Not Really into Dudes	18.7
Beautiful Woman	19.2
Kevin Longblade	115.3
Robo Slayer 3000	117.4
Amped II	118.9

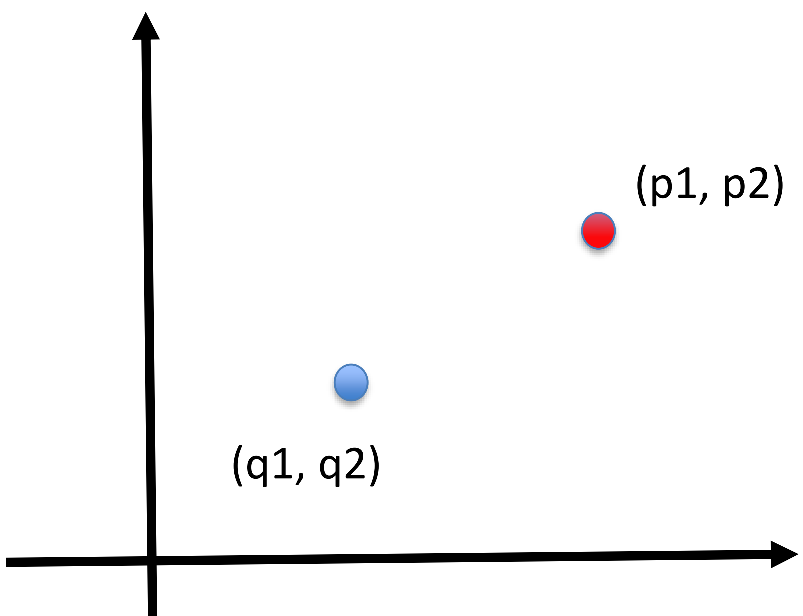
Distance between **California man** (3, 104) and the **query** instance (18, 90) would be

$$\sqrt{(3-18)^2 + (104-90)^2} = \sqrt{225+196} = 20.5$$

Manhattan Distance Metric

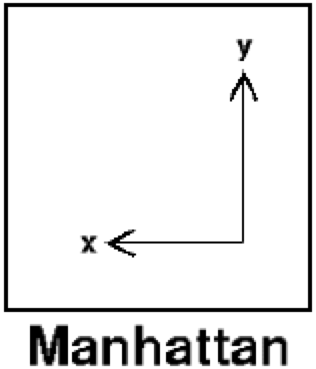
- Manhattan distance measures distance **parallel to each axis**, not diagonally (in downtown Manhattan, to get from one point to another you generally walk North-South and East-West, rather than 'as the crow flies').

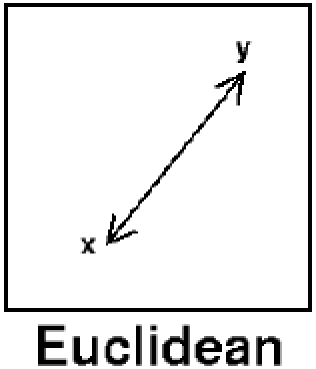
- In other words take the sum of the absolute values of the differences of the coordinates
- d(p, q) = $|q_1 - p_1| + |q_2 - p_2| + |q_n - p_n| = \sum_{i=1}^{n} |q_i - p_i|$



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**





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
  - $(10\text{-}5)^2+(8\text{-}4)^2+(4\text{-}4)^2+(8\text{-}3)^2$
  - 57
  - Square root of 57 = 7.55
- Manhattan
  - $|10\text{-}5|+|8\text{-}4|+|4\text{-}4|+|8\text{-}3|$
  - 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:
- $d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^{a}\right)^{\frac{1}{a}}$
- In the above equation *a* is an integer. Consider the case when a = 1 or a = 2. Any comments.
  - 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.

Building a K Nearest Neighbour Classifier

- **Step 1. Read information from a dataset**
  - Read data from a dataset containing classified instances. Read each feature of the dataset as well as corresponding class.

- **Step 2. Determine distance between each dataset entry and the query instance**
  - Use a suitable distance metric to calculate the distance between the query instance and all k neighbours

- **Step 3. Classify the query instance**
  - Identify k nearest data instances. Assign query instance category corresponding to most common category.

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 **<u>features are different</u>** 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 calculations

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

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

$$\sqrt{16+48400}$$

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 formul.
    - $\boldsymbol{newValue = \frac{originalValue - minValue}{maxValue - minValue}}$

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

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

- Problem 1: Scaling
  - Feature A has range 1-10
    Feature B has range 1-1000
- Normalise variables
  - Feature A
  - Feature B

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

- 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 – Irrelevant Features

- The other principal problem is that **all features are included equally** in the calculations we have looked at, even though some features may be **redundant** or **less relevant**.
- Therefore, a number of features may skew the result even through they might of little or no impact to the classification.

- **Solution 2A**:
  - **Assign weighting** to each dimension (Optimise weighting to minimise error)
- **Solution 2B**:
  - Give some dimensions **0 weight** (Feature subset solution)

- Either way, since we cannot know in advance what weighting to give dimensions, systematic repeated experiments are needed to optimise them.
- We will look feature selection in more detail later in the module.