Machine Learning

Lecture 6

Unsupervised Learning with a bit of Supervised Learning

Clustering and nearest neighbour classifiers.

Justin Pearson[1]

mailto:it-1dl034@lists.uu.se

2021

[1] http://user.it.uu.se/~justin/Hugo/courses/machinelearning/

Supervised Learning

The paradigm so far

- Our data set consists of data

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(i)}, y^{(i)}), \ldots, (x^{(m)}, y^{(m)})$$

- Where $x^{(i)}$ is the data and $y^{(i)}$ is a label. Data is in general multi-dimensional and can consist of many variables.

Supervised Learning

The paradigm so far

- Our data set consists of data

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(i)}, y^{(i)}), \ldots, (x^{(m)}, y^{(m)})$$

- Data can
  - Categorical — Data comes from a discrete set of labels. For example you might have 'Passenger class' , 'first', 'second' or 'third'
  - Everything else, sometimes called continuous, or numeric. Even though you cannot have 0.5 of a person, if you are doing population counts it is easier to treat the data as continuous.

Supervised Learning

Two paradigms

Classification You labels $y^{(i)}$ are categorical data: "Cat", "Dog", "Hippo".

Regression You value $y^{(i)}$ is (possibly) multi-dimensional non-categorical data.

Supervised Learning — Multi class Classification

In general it is hard to train a multi-class classifier. Training a classifier that outputs "Dog", "Hippo", "Cat" from an image is quite hard.
Two strategies:

One-vs-Rest Train individual classifiers "Dog" vs "Not Dog", "Hippo" vs "Not Hippo", "Cat" vs "Not Cat" Pick the classifier with the best confidence.

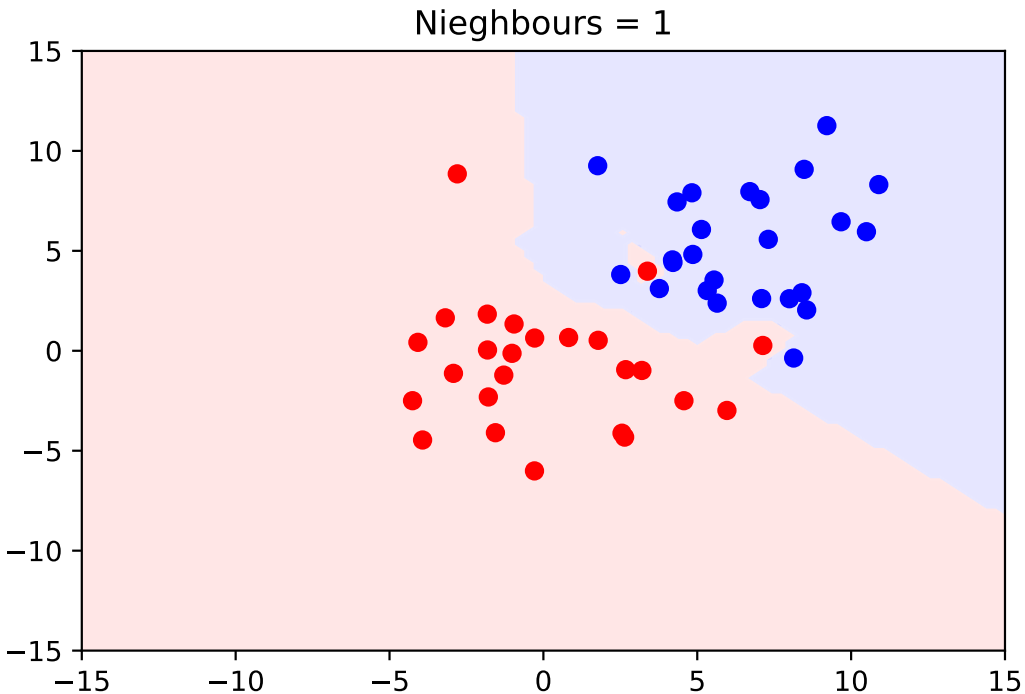
One-vs-One Given n classes train $n(n-1)/2$ binary classifiers: "Dog" vs "Hippo", "Dog" vs "Cat", "Hippo" vs "Cat". Given an unknown image let each classifier run and pick the class that gets the most votes.

k-nearest neighbour classifier

- Very simple classifier.
- Memory based, no model is learned you just have to remember the training data.
- To classify a point, look at the k-closest points look at their classes and take a vote.
- No need to do One-vs-Rest or One-vs-One.

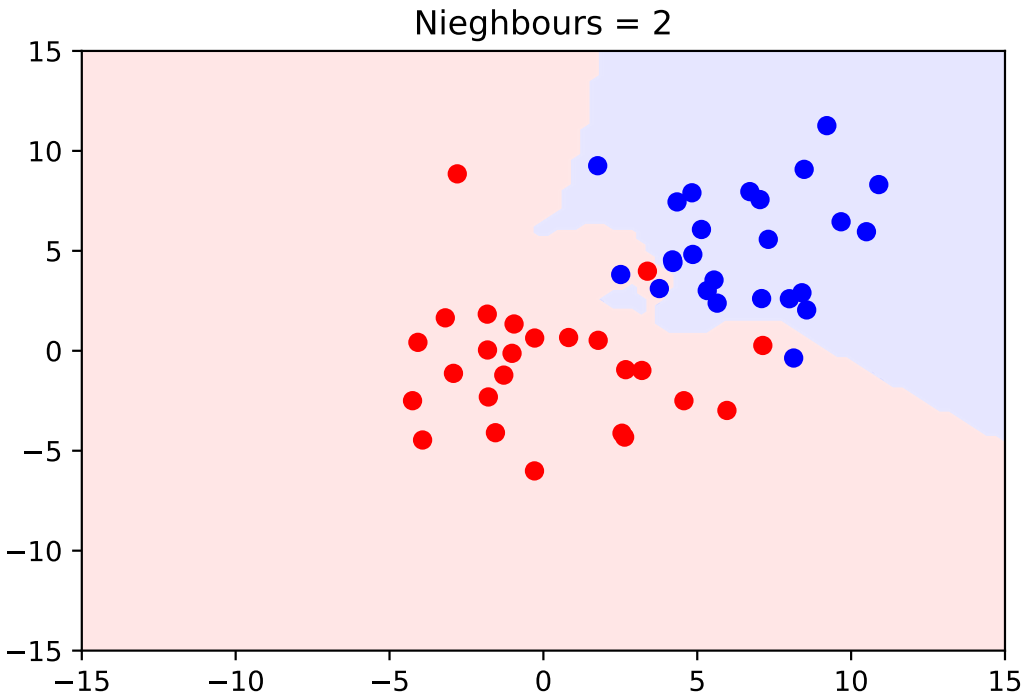
k-nearest neighbour classifier

Example two classes and k=1



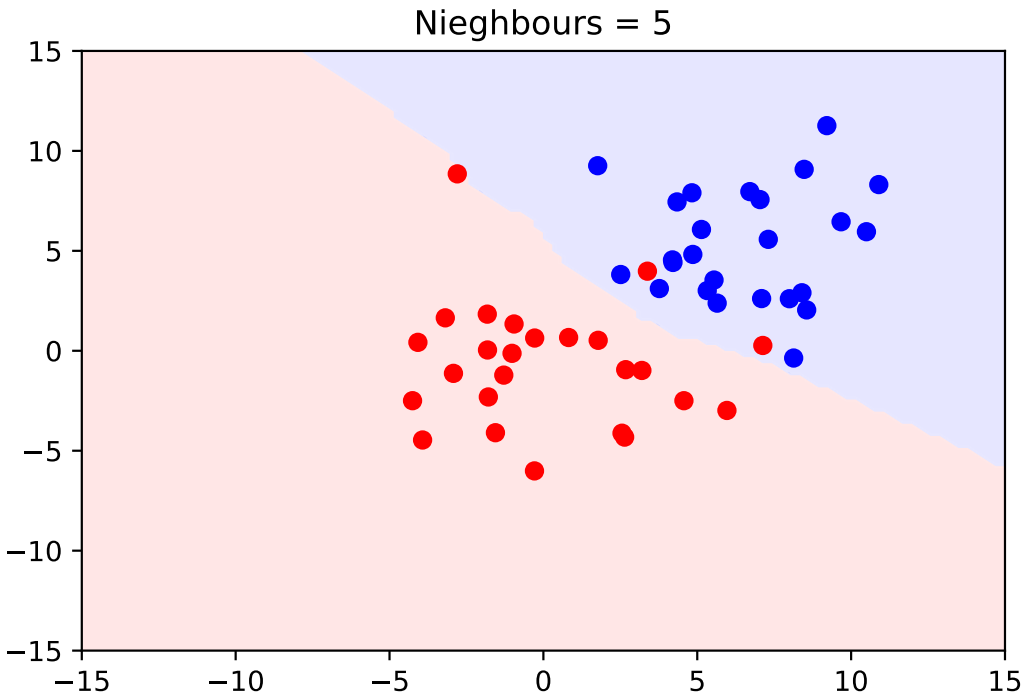
k-nearest neighbour classifier

Example two classes and k=2



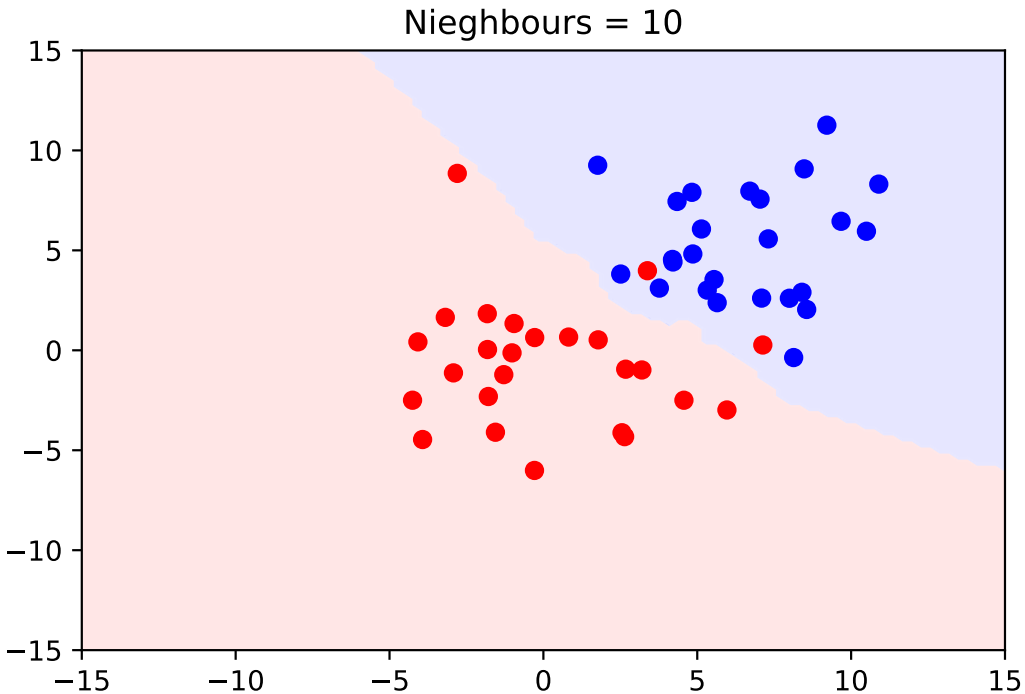
k-nearest neighbour classifier

Example two classes and k=5



k-nearest neighbour classifier

Example two classes and k=10



Why does it seem that points are miss-classified? How can this happen with the algorithm.

Other Metrics

All that we need for k-nearest neighbours is some function $d(x, y)$ that gives you the distance between two points x and y. As long as your function obeys the following axioms:

$$\forall x.d(x, x) = 0$$

$$\forall x, y.d(x, y) = d(y, x)$$

$$\forall x, y, z.d(x, y) + d(y, z) \leq d(x, z)$$

The k-nearest neighbour algorithm will work.

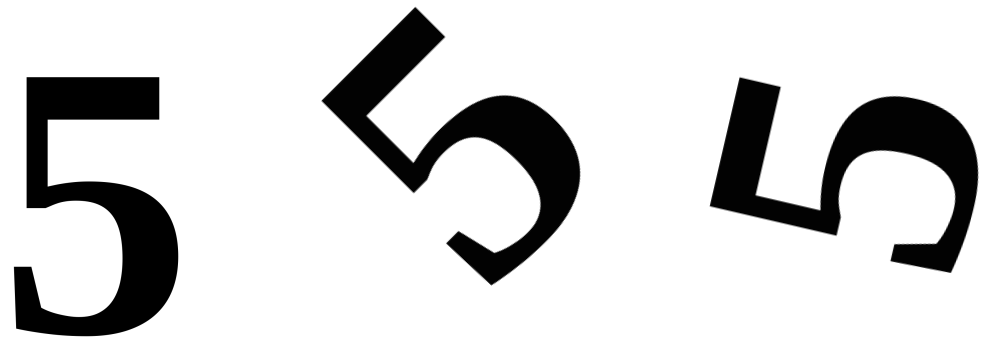
Other Metrics

For example you could use the edit (Levenshtein) distance metric that tells you the minimum number of insertions, deletions and substitutions to get from one string to another. For example $d(\text{kitten}, \text{sitting}) = 3$ by

$$\text{kitten} \rightarrow \text{sitten} \rightarrow \text{sittin} \rightarrow \text{sitting}$$

Transformation Invariant Metrics[3]

If you wanted to recognise numbers that can be rotated then you would want the distance between the following images to be zero.



One way would be to build a special metric[2], another idea is to add rotations (and other transformations) to your training data. With k-nearest neighbour you can run into memory problems, but with other learning algorithms this might not be too much of a problem.

---

[2] `https://ieeexplore.ieee.org/document/576916` Memory-based character recognition using a transformation invariant metric

[3] Image from Wikimedia

Problems with k-NN

- As the size of the data set grows, and the more dimensions of the input data the computational complexity explodes. This is sometimes referred to as the curse of dimensionality.
- With reasonable clever data structures and algorithms you can speed things up.

Even with these problems, k-NN can be a very effective classifier.

The problem of labels

- Good labels are hard to come by. Sometimes you will need a human to classify the data, and there is a limit to how much data you can get.

Unsupervised learning

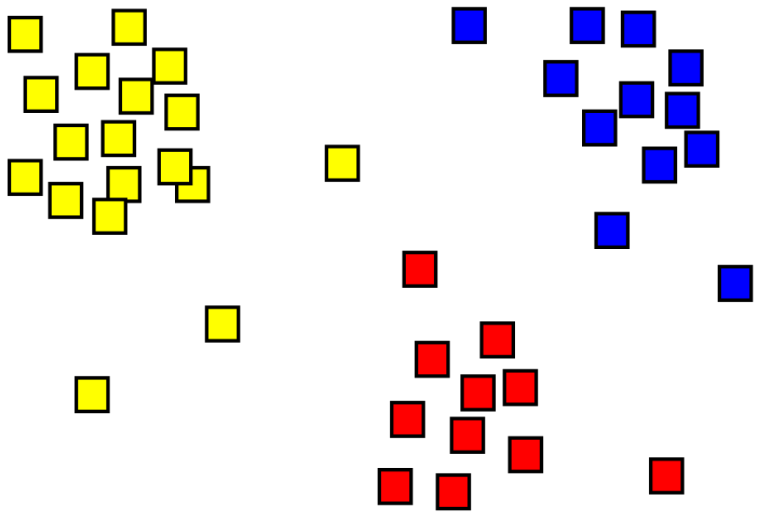
Simply put: how do we learn on data without labels? There are lots of motivations.

- Lack of good labels[4].
- Even with labels, there might be other patterns in the data that are not captured by the labels.
- Historical motivation, a big part of human learning is unsupervised. What can learn about biology and cognition from un-supervised learning algorithms. For example look at Hebbian Learning.
- Recommender systems. How does Amazon tell you which book to buy? Or how does Netflix tell you which film to watch? There are some specialised algorithms, but a lot of it is machine learning.
- Data mining. I have this data, what does it tell me?

[4] `https://www.theregister.co.uk/2020/02/17/self_driving_car_dataset/` Please check your data: A self-driving car dataset failed to label hundreds of pedestrians, thousands of vehicles.

Clustering[5]

General idea. Take the input data and group it into clusters of similar data points.

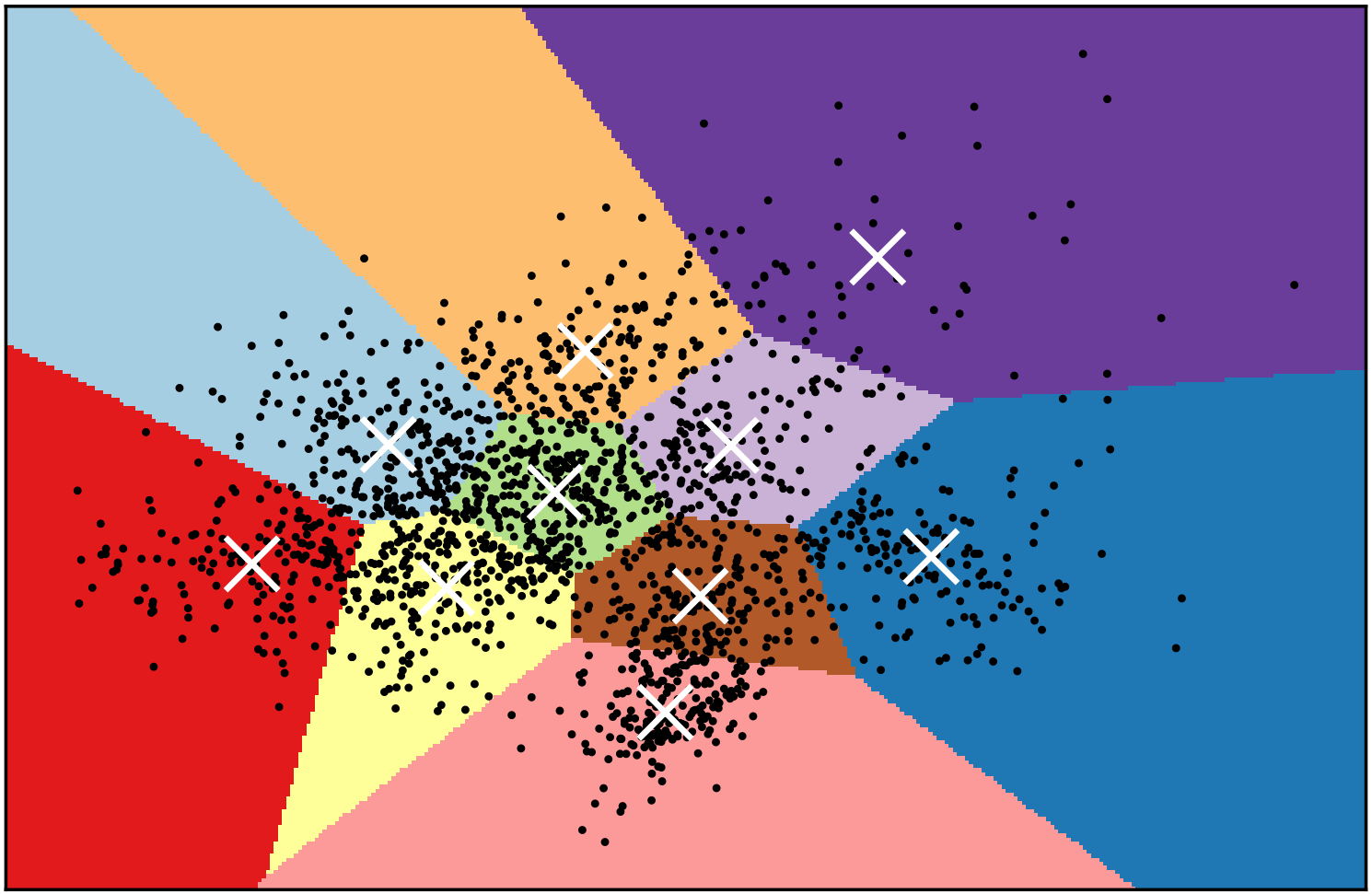


Each cluster becomes a class.

[5] Picture taken from `https://commons.wikimedia.org/wiki/File:Cluster-2.svg`

Clustering

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross



Clustering

Questions

- How do we decide how similar items are?
- How do we determine how many cluster there should be?
- What is the computational complexity of clustering?

k-means clustering

Given vectors $x_1, \ldots, x_n$ define the average as

$$\mu = \frac{1}{N} \sum_{i=1}^{n} x_i$$

This is also called the centroid or the barycentre.
You can prove that the point μ minimises the sum of the squared euclidean distances between itself and all the points in the set.
That is, it minimises

$$\sum_{i=1}^{n} ||x_i - \mu||^2$$

k-means clustering

Given a set of points $x_1, \ldots, x_n$ find a partition of the n-points into k sets $S = S_1, \ldots, S_k$ that minimises the objective

$$\sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

Where

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

So we want to find k centres $\mu_1, \ldots, \mu_k$ that minimises the spread or max distance in each cluster.
This is a declarative description of what we want, not an algorithm to find the clusters.

k-means clustering — Complexity

- NP-hard even for two clusters.

This means that you could in theory code up a SAT or a graph colouring problem as a clustering problem.
This also means that none of the heuristic algorithms are guaranteed to find the best clustering.

k-means clustering — Naive Algorithm

Initialise $\mu_1, \ldots, \mu_k$ with random values.
Repeat the following two steps until convergence:

Clustering For each data point x_i assign its cluster to be the mean point μ_j that it is closest to.

Recompute μ For each cluster j recompute the cluster recompute μ_j to be

$$\frac{1}{|S_j|} \sum x \in S_j x$$

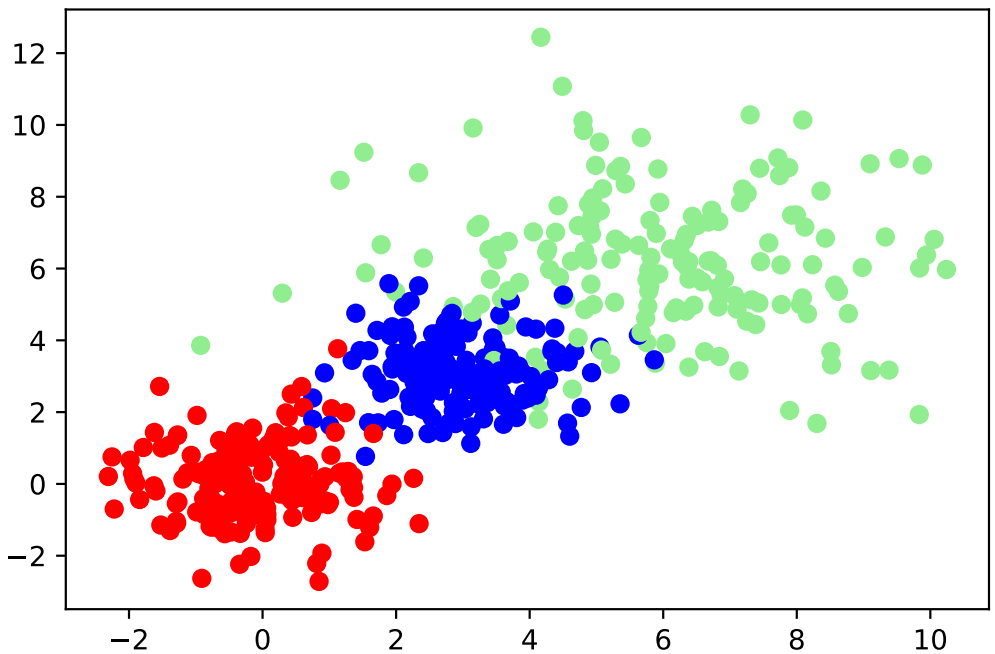
Note that the sets $S_1, \ldots, S_j$ change as the algorithm runs.

k-means clustering

Even though the naive algorithm is not guaranteed to converge to a global minimum it is often quite successful in practice. For example there are applications in

- Market segmentation, divide your customers into similar groups.
- Image segmentation
- In Astronomy, clustering similar observations.

What is the correct value of k?



How many clusters 1,2 or 3?

What is the correct value of k?

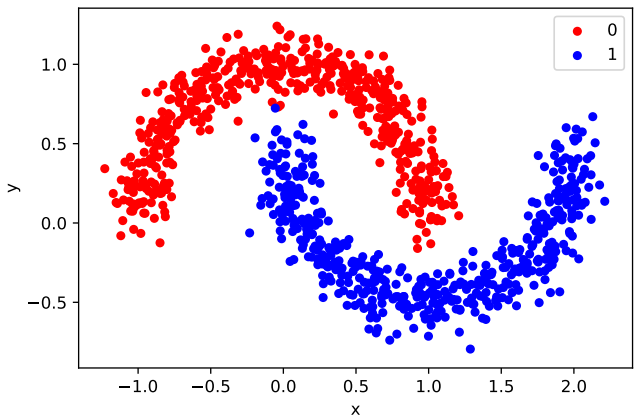
You can look at how objective changes

$$\sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

With different numbers of clusters, and stop increasing the number of clusters when you don't get any more big drops in the objective (the elbow method).
Often the number of clusters is application dependent and depends on what you doing with your clustering algorithm. There might be an application dependent method of evaluating the number of clusters.

Limitations of Clustering

- Clusters can only be blobs. You could not learn clusters for a data set like this:



- There are K-kernel methods to get around it.

k-means clustering only scratches the surface, there are a lot of extensions and more powerful clustering algorithms that attempt to cluster more complex data, but *k*-means clustering is always a good start.