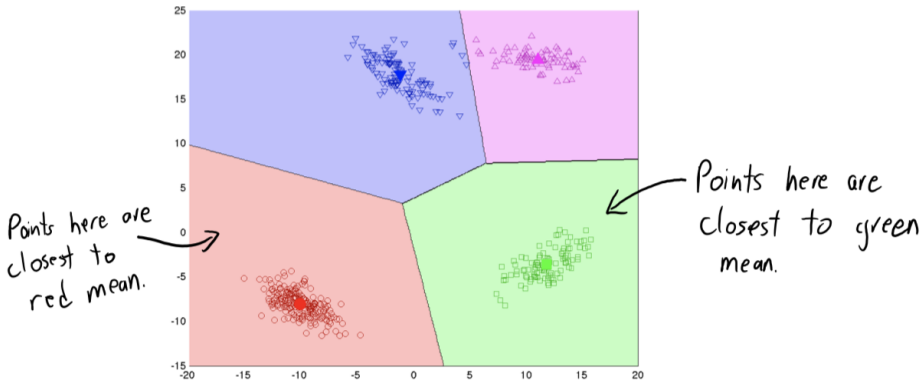
https://www.cs.ubc.ca/~fwood/CS340/

Lecture IX

January 27th, 2020
https://www.cs.ubc.ca/~fwood/CS340/lectures/L9.pdf

Shape of K-means clusters

K-means partitions the space based on the "closest mean":

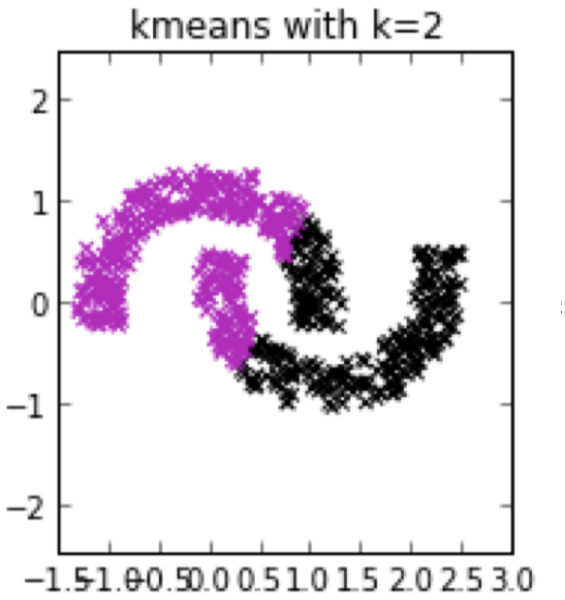


We observe that the clusters are convex regions

Convex set: A set is convex if line between two points in the set stays in the set.

K-Means with non convex clusters

Example of non-convex banana-shaped data points:



Density based clustering

This is a different approach for clustering where the clusters are defined by *dense* regions. The examples in non-dense regions do not get clustered.

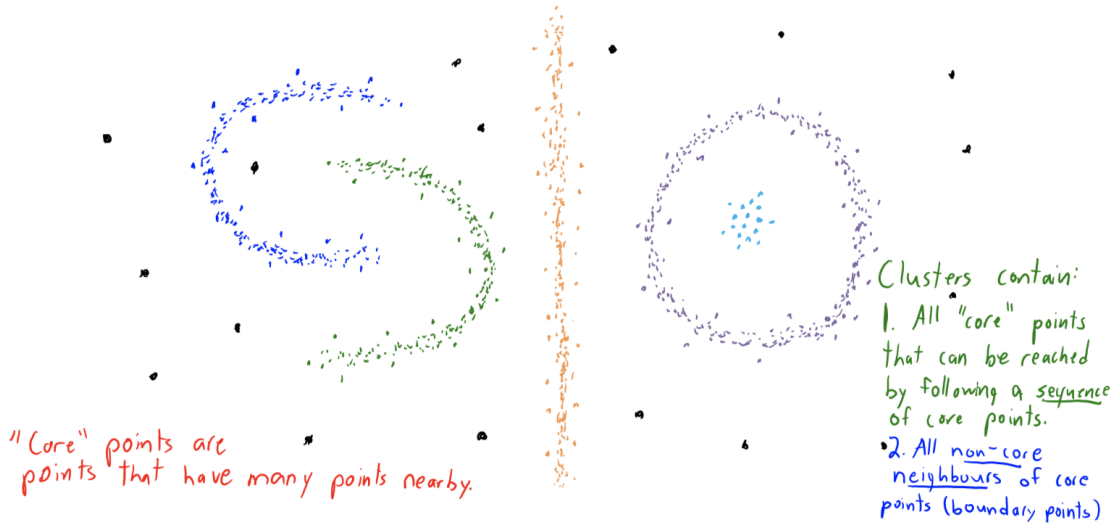
This means that the clusters can be non-convex, and it is a non-parametric clustering method (there is no fixed k value for the number of clusters).

The **Density based clustering** algorithm (DBSCAN) has two hyper-parameters:

1. Epsilon ε, used to determine if another point is a neighbor:

2. Min-neighbors: number of neighbors needed to say that a region is *dense*. If you have at least min-Neighbours "neighbours", you are called a "core" point.

The main idea is to merge all neighboring core points to form clusters:



Pseudo-code

```
For each example x[i]:
   if x[i] is already assigned to a cluster:
      do nothing
```

```
else:
   Test whether x[i] is a core point.
   if x[i] is not a core point:
      do nothing
   else:
      make a new cluster
      call the expand cluster_function
```

The expand cluster function:

```
Assign to this cluster all x[j] within distance 'eps' of core point xi
For each new core point found, call 'expand cluster' (recursively)
```

Density based clustering problems

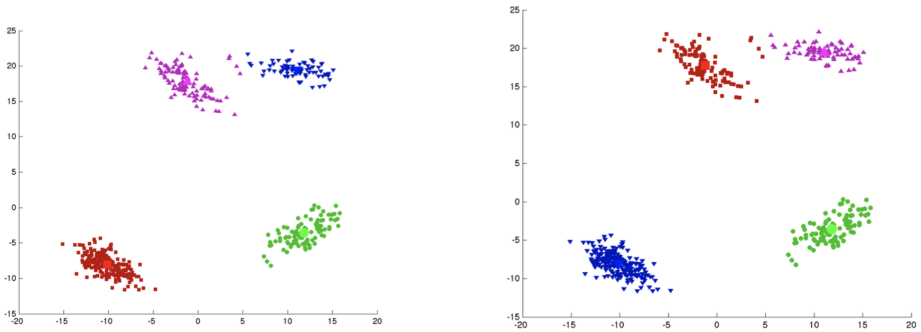
- Some points are not assigned to a cluster.
  - Tis can be good or bad depending on the application
- There is an ambiguity of non-core 'boundary' points.
- This method is sensitive to the choise of the ε and `minNeighbors` arguments
  - Original paper proposed an "elbow" method (see bonus slide).
  - Otherwise, not sensitive to initialization (except for boundary points).
- If you get a new example, then finding a cluster is expensive.
  - Need to compute distances to core points (or maybe all training points).
- In high-dimensions, need a lot of points to 'fill' the space.

Ensemble clustering

We can consider ensemble methods for clustering: *Consensus clustering*. Bootstrapping is widely used, but we need to be careful about how we combine the models.

An example of this is running K-means 20 times, and then cluster using the mode of each $\hat{y}_i$. Normally, averaging across models doing different things is good, but overall this is a bad ensemble method, specifically because of the label switching problem. This because:

- The cluster labels $\hat{y}_i$ are meaningless.
- We could get same clustering with permuted labels ("exchangeable"):



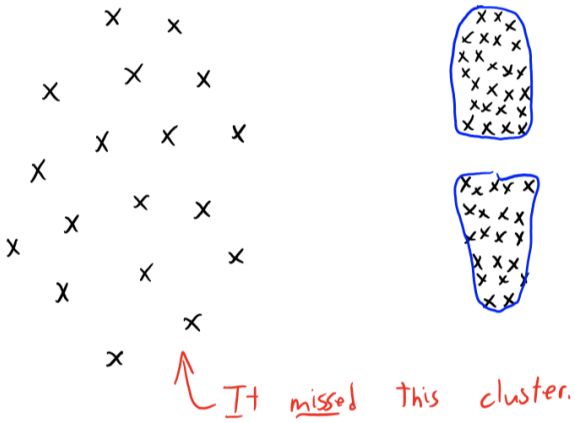
- All $\hat{y}_i$ become equally likely as number of initializations increases.

This problem is addressed using the following idea: ensembles can't depend on label "meaning":

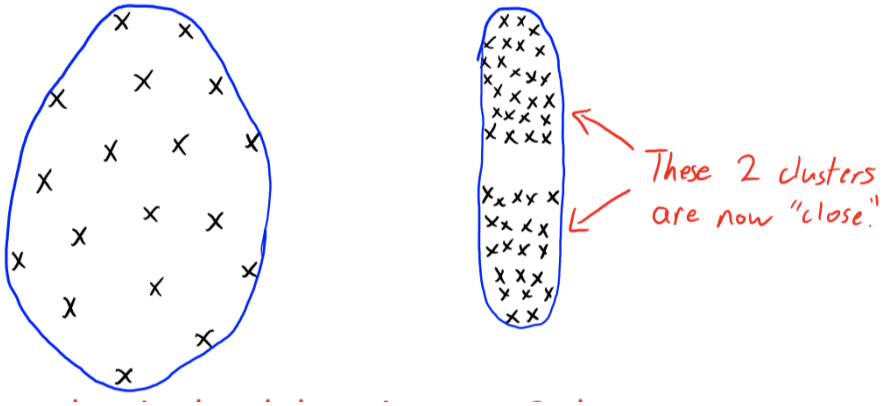
- Don't ask "is point x_i in red square cluster?", which is meaningless.
- Ask "is point x_i in the same cluster as x_j?", which is meaningful

Differing densities

Consider density-based clustering on this data:



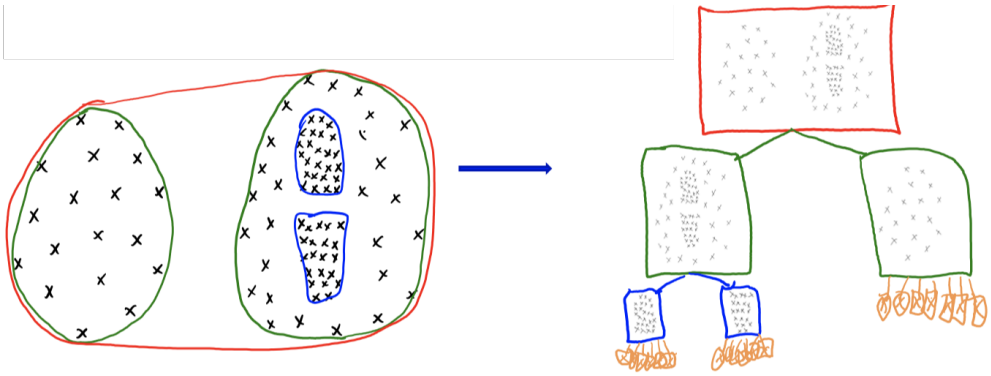
However, if we increase epsilon and run it again, we get this:



In this case there may not be an ε value that gives you 3 distinct clusters. Examples exist, where this may be even worse. A solution to this is **Hierarchical Clustering**

Hierarchical Clustering

This is a method that produces a tree of clusterings. Each node in the tree splits the data into 2 or more clusters. This provides more information than using fixed clusters. Often, these have individual data points as leaves:



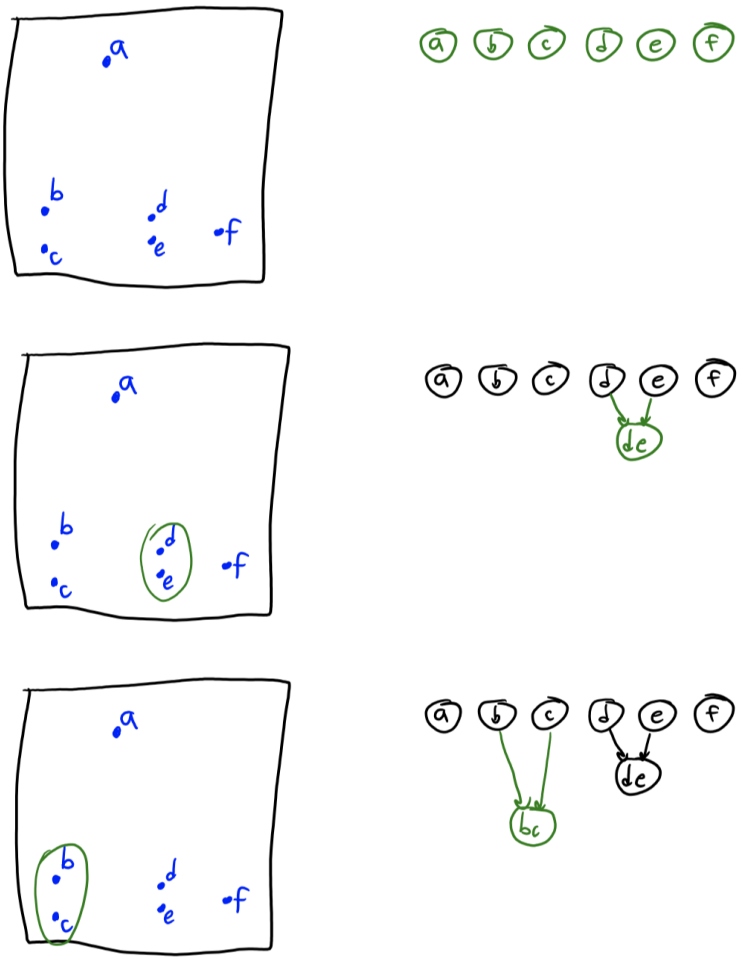
Application: Phylogenetics

See slides

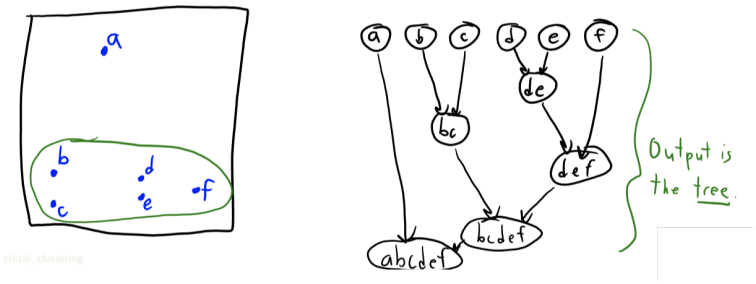
Agglomerative (bottom-up) clustering

This is the most common hierarchical method.

1. Starts with each point in its own cluster.
2. Each step merges the two "closest" clusters.



...
...



This method has been reinvented by different fields under different names("UPGMA"). It also needs a "distance" between two clusters, thus a standard choice is the distance between the means of the clusters (Not necessarily the best, many choices exist - *bonus slide*).

Cost is $O(n^3d)$ for basic implementation. Each step costs $O(n^2d)$, and each step might only cluster 1 new point

Lecture X

January 29th, 2020 `https://www.cs.ubc.ca/~fwood/CS340/lectures/L10.pdf`

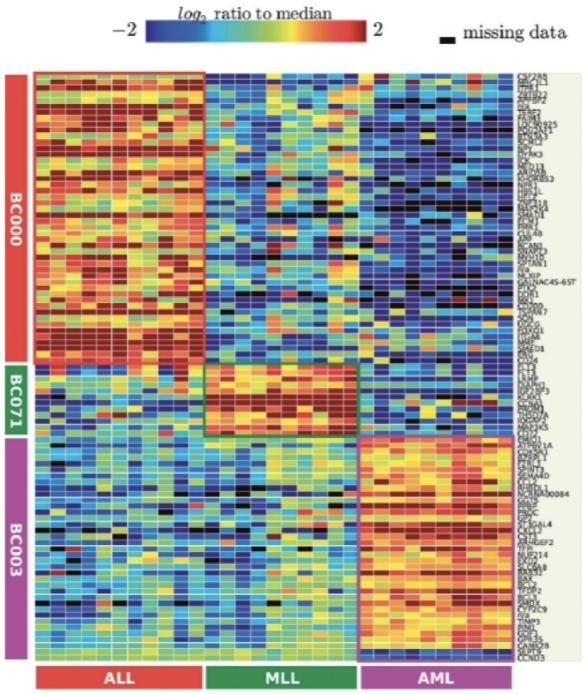
Biclustering

Biclustering is the process of clustering the training examples **and** the features. Thus this also gives information about the relationship of the features. The most popular method is:

- Run clustering method on X (examples).
- Run clustering method on X^T (features).

This method is often plotted with X as a 'heatmap':

- Where rows/columns are arranged by clusters.
- This helps us 'see' why things are clustered



Look at slides for significant applications fo the clustering methods.

Outlier Detection

The goal of outlier detection is to find observations that "unusually different" from the others. This is also known as "anomaly detection". We may be interested in removing the outliers, since they can skew the data, or actually be interested in the outliers themselves for security reasons. Outliers can occur for several reasons, some of which may be:

- Measurement errors
- Data entry errors
- Contamination of data from other sources
- Rare events

Applications of outlier detection may include:

- Datacleaning.
- Security and fault detection (network intrusion, DOSattacks).
- Fraud detection(credit cards, stocks, voting irregularities).
- Detecting natural disasters (underwater earthquakes).
- Astronomy (find new classes of stars/planets)
- Genetics (finding individuals with new/ancient genes)

There are several classes of methods used for outlier detection:

1. Model-based methods
2. Graphical approaches
3. Cluster-based methods
4. Distance-based methods
5. supervised-learning methods

But before anything is done with regards to outlier detection, it is good practice to do a simple sanity check:

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	Peanuts	Sick?
0	0.7	0	0.3	0	0	0	1
0.3	0.7	0	0.6	-1	3	3	1
0	0	0	"sick"	0	1	1	0
0.3	0.7	1.2	0	0.10	0	0	2
900	0	1.2	0.3	0.10	0	0	1

We as our-self the following question:

- Would any values in the column cause a Python/Julia "type" error?
- What is the range of numerical features?
- What are the unique entries for a categorical feature?
- Does it look like parts of the table are duplicated?

These types of simple errors are VERY common in real data.

Model-Based outlier detection

This method revolves around fitting a probabilistic model, and then we have that the outliers are the examples with low probability. For example:

Example

Let us assume that the data follows a standard normal distribution. Then the z-score for the 1-D data is given by:

$$z_i = \frac{x_i - \mu}{\sigma}$$

where:

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

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

We say that an example is an outlier if $|z|> 4$, or in other words, it is more than 4 standard deviations from the mean.

Problems with Z-Score

Several problems occur with the Z-score. Norably, we have that the mean and variance are sensitive to outliers. A good fix for that would be to use quantiles, or to sequentially remove the worst outliers. Additionally, we have that the Z-score assumes that the data is "uni-modal", i.e.: that the data is centered around the mean.

Global vs. Local Outliers

Is the red point an outlier? What if we add the blue points?

In this case we have that the red point has the lowest z-score:

- In the first case it is a global outlier
- In the second case it is a local outlier (it is within the normal range of the data, but it is far from all the other data points)

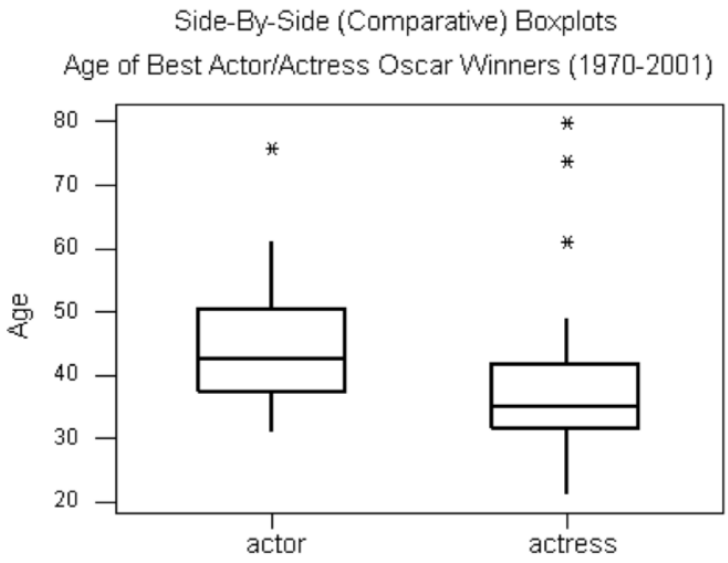
Overall it is very difficult to properly define outliers. However we can have things such as **outlier groups**, which are repeating patterns of outliers:

Graphical Outlier detection

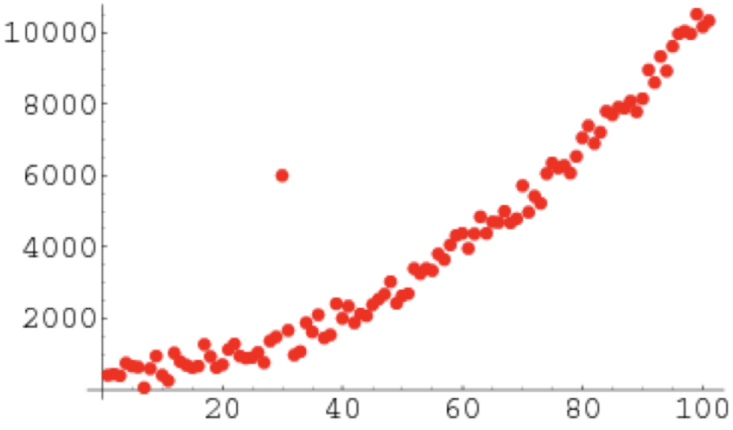
The **graphical approach** to outlier detection consists of looking at a graph of the data, and have a human decide whether a data point is an outlier.

Examples:

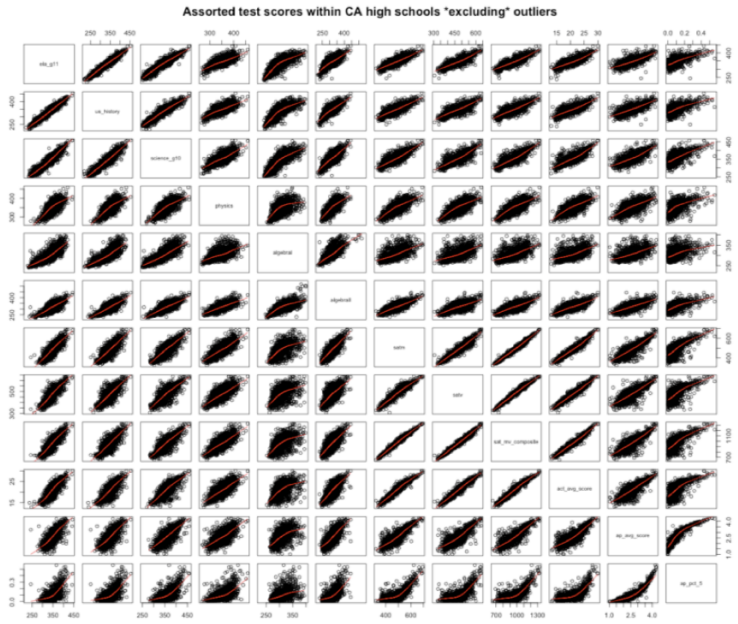
1. ***Box-plot***:
    - Visualization of quantiles/outliers.
    - Only 1 variable at a time.



2. ***Scatter-plot***:
   - Can detect complex patterns.
   - Only 2 variables at a time.



3. ***Scatterplot array***:
   - Look at all combinations of variables.
   - But laborious in high-dimensions.
   - Still only 2 variables at a time.



4. ***Scatter-plot of 2-dimensional PCA***
    - Details discussed later in the course.

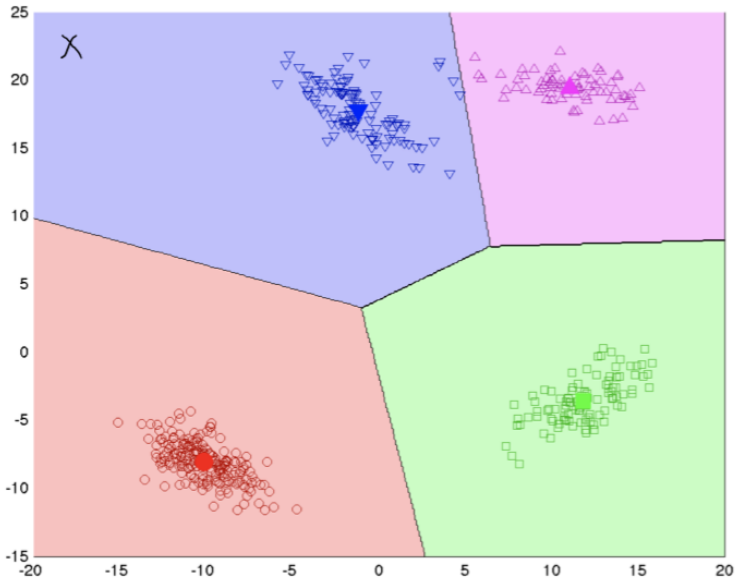
Cluster-based outlier detection

This method detects outliers based on clustering:

- We cluster the data
- Find points that don't belong to clusters

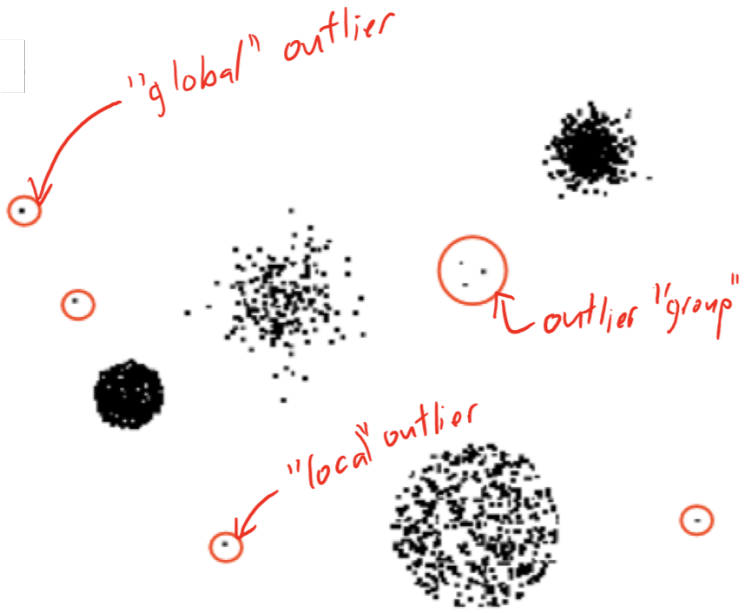
Examples:

1. ***K-means***:
    - Find points that are far away from any mean.
    - Find clusters with a small number of points.



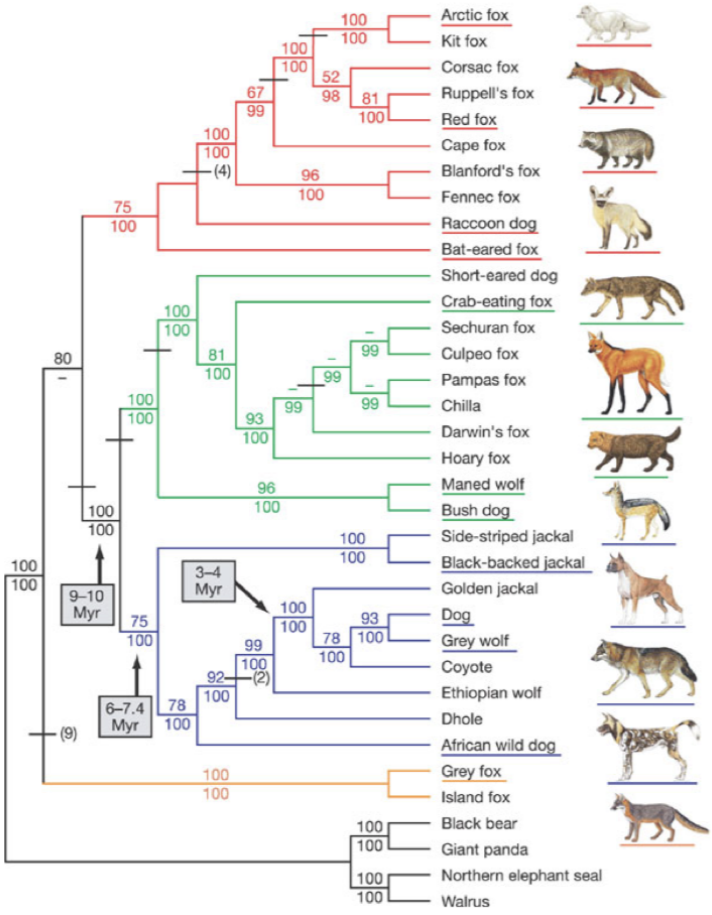
2. ***Density-based clustering***:

   - Outliers are points not assigned to cluster.



3. Hierarchical clustering:

   - Outliers take longer to join other groups.
   - Also good for outlier groups.



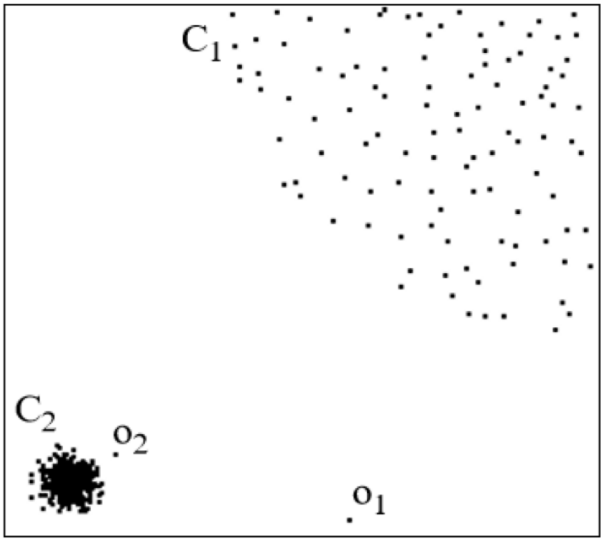
Distance-Based Outlier Detection

However, we have that most outlier detection techniques are based on distances. For instance using **KNN outlier detection**, for each point we compute the average distance to its **KNN**.

- We choose points with biggest values (or values above a threshold) as outliers

Local Distance-Based Outlier Detection

As with density-based clustering, we encounter a problem with differing densities:



In the above example, we have that the outlier o_2 has the same approximate density as the points in cluster C_1. Thus the basic ides around cluster-based models is to recognize that o_2 is **relatively far** compared to its neighbors.

We therefore construct the outlierness ration for a specific example i:

$$\frac{\text{average distance of } i \text{ to its KNNs}}{\text{average distance of neighbors of } i \text{ to their KNNs}}$$

We have that if outlierness > 1, x_i is further away from neighbours than expected.

Look at slides for more notes on ***isolation forests*** *and problems with* ***Unsupervised Outlier Detection***

Supervised Outlier Detection

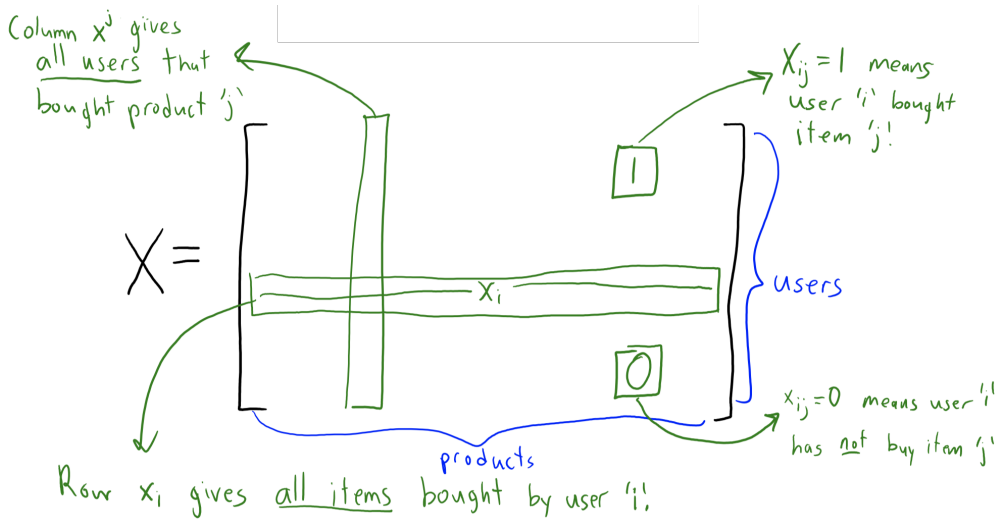
The final approach to outlier detection is to use supervised learning:

- $y_i = 1$ if x_i is an outlier.
- $y_i = 0$ if x_i is a regular point.

Here we can use out regular supervised learning models, and we can potentially find very complicated outlier patterns. However this methods needs supervision, which means that we need to know beforehand what an outlier looks like, and it may make it hard to detect 'new' types of outliers.

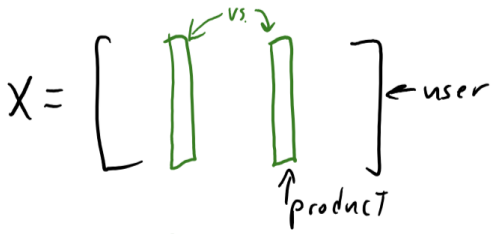
User-product Matrix

Motivation: Product Recommendation
If a customer comes to out website looking to buy an item, we want to find similar items that they migh also want to buy. This yields a ***user-product matrix***:



Amazon Recommendation Method

This method used a User-product matrix as follows:



This method returns the KNNs across the columns:

- We find the values of j that minimize $\|x^i - x^j\|$
- The products bought by similar users.

But first we have to divide each column by its norm, to normalize the vector. This is necessary to reflect if a product has been bought by a lot of users, or a few users.

Lecture XII

January 31st, 2020 `https://www.cs.ubc.ca/~fwood/CS340/lectures/L12.pdf`

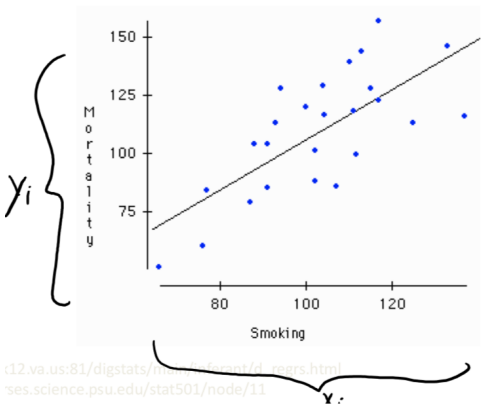
Supervised Learning Round II: Regression

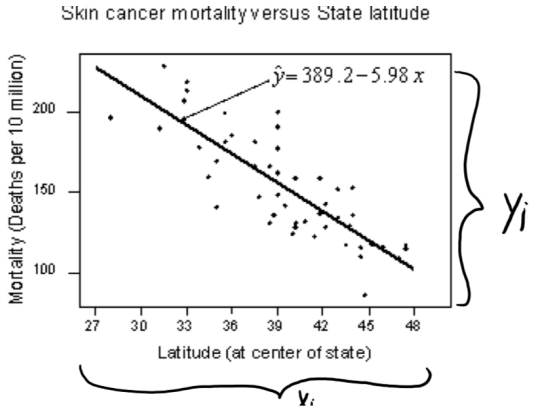
Previously we have considered classification, which assumed that y_i was discrete. Now we will consider regression, where we allow y_i to be numerical.

Example: Dependent vs. Explanatory variables

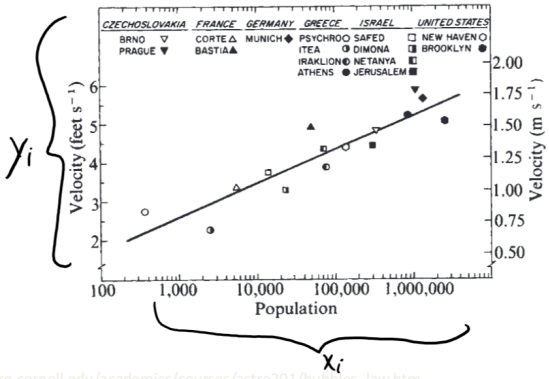
The goal for us is to discover the relationship between numerical variables:

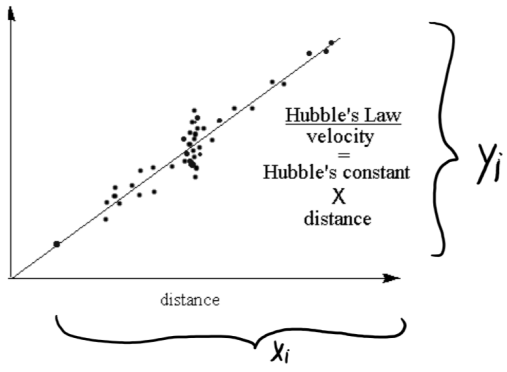
- Does number of lung cancer deaths change with number of cigarettes?
- Does number of skin cancer deaths change with latitude?



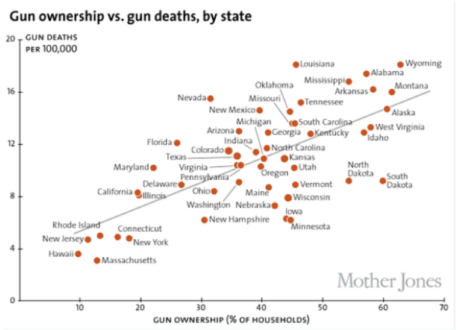


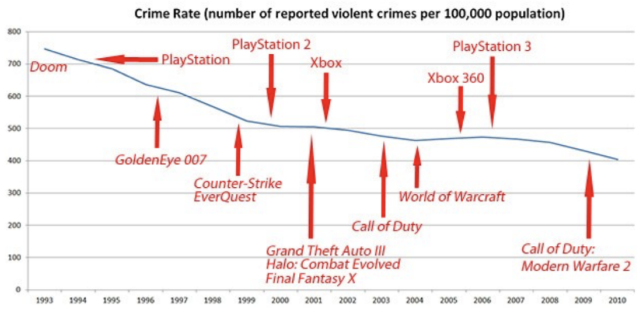
- Do people in big cities walk faster?
- Is the universe expanding or shrinking or staying the same size?





- Does number of gun deaths change with gun ownership?
- Does number violent crimes change with violent video games?





Essentially, out goal is to discover a relationship between numerical variables. We must note that we're doing supervised learning, i.e.: we are trying to predict the value of 1 variable, instead of measuring the correlation between 2.

Supervised learning does not give casualty. (**Confused about this bit**)

Handling Numerical Labels

One way to handle numerical labels, is to discretize them.

- E.g., for 'age' could we use {'$age \leq 20$', '$20 < age \leq 30$', '$age > 30$'}.
- This allows us to apply the methods of classification to do regression.
- **However:** But *coarse discretization* loses resolution and *fine discretization* requires lots of data.

There exist regression versions of classification methods such as Regression trees, probabilistic models, non-parametric models, however one of the oldest but still most popular methods in use is *Linear regression based on squared error*.

Linear Regression in 1-D

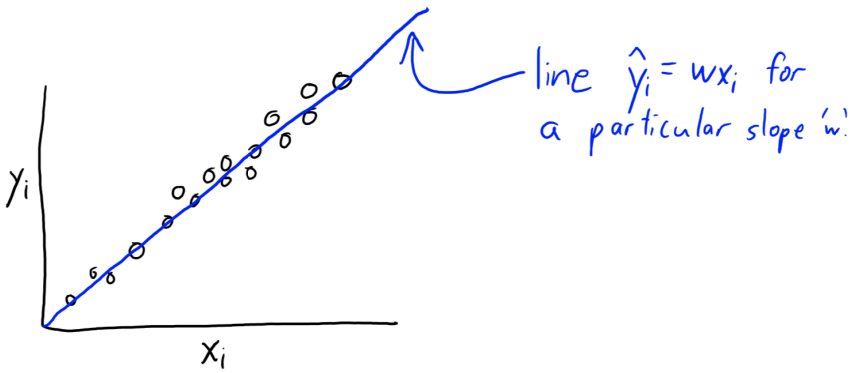
In this case, we assume that we only have 1 feature ($d = 1$). **Linear regression** makes predictions $\hat{y}_i$ using a **linear function** of x_i:

$$\hat{y}_i = wx_i$$

The parameter w is referred to as the weight, or regression coefficient of x_i.

As x_i changes, slope w affects the rate that $\hat{y}_i$ increases/decreases:

- Positive w: $\hat{y}_i$ increases as x_i increases.
- Negative w: $\hat{y}_i$ decreases as x_i increases.



Least Squares Objective

The linear model is given by:

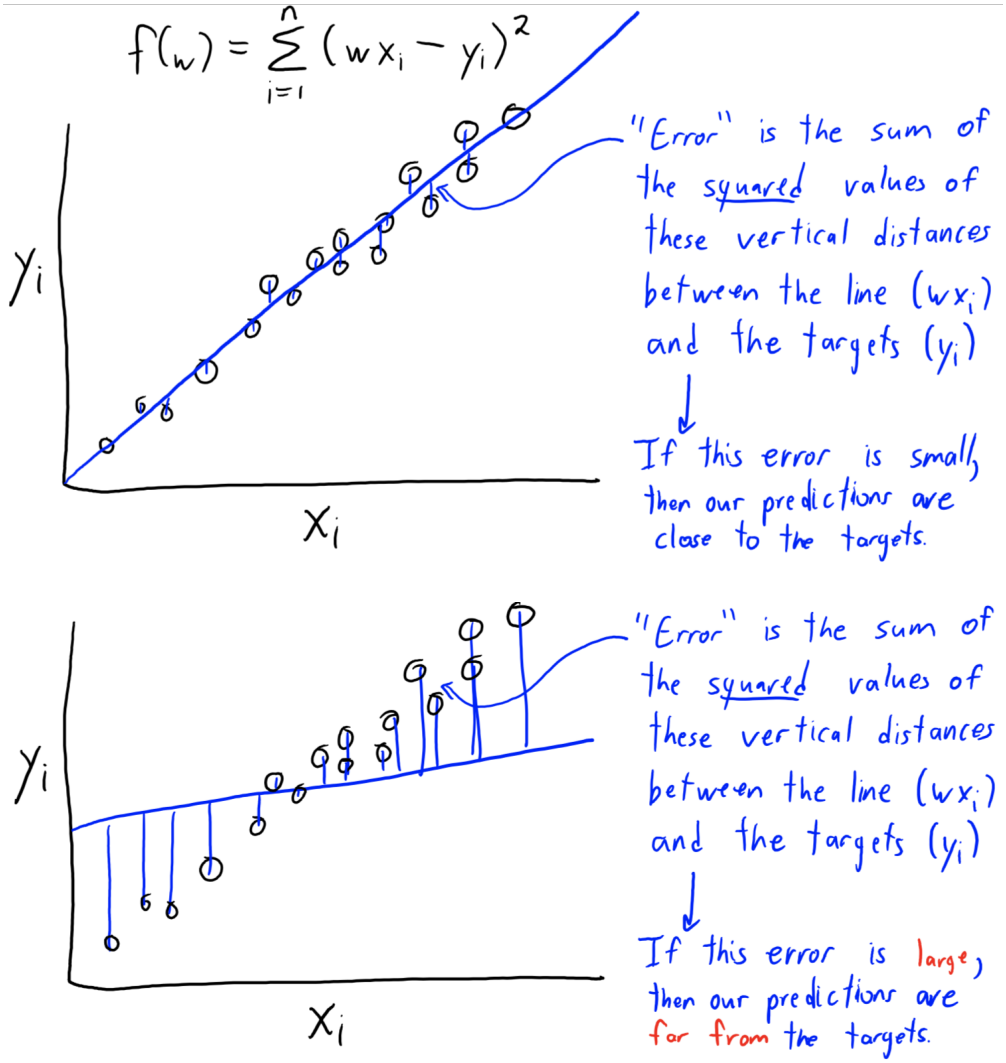
$$\hat{y}_i = wx_i$$

Thus to make predictions on a new example, we use:

$$\hat{y}_i = w\tilde{x}_i$$

However it is not possible to use the same error as before, because it is unlikely to find a line where $\hat{y}_i = y_i$ exactly for many points. Thus the best model will be when $|\hat{y}_i - y_i|$ is small, but not exactly 0.

Consequently, instead of evaluating y_i exactly, we also evaluate the "size of the error" in the prediction. And the classic way of doing so, is to set w such that we minimize the sum of squared errors.



Finding a w that minimizes the sum of squared errors

We have the following derivation:

$$\begin{aligned} f(w) &= \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\left[w^2x_i^2 - 2wx_iy_i + y_i^2\right] \\ &= \frac{w^2}{2}\underbrace{\sum_{i=1}^{n}x_i^2}_{\text{constant } a} - w\underbrace{\sum_{i=1}^{n}x_iy_i}_{\text{constant } b} + \frac{1}{2}\underbrace{\sum_{i=1}^{n}y_i^2}_{\text{constant } c} \\ &= \frac{w^2}{2}a - wb + \frac{1}{2}c \end{aligned}$$

If we take the first derivative, we have that:

$$f'(w) = wa - b + 0$$

And thus setting $f'(w) = 0$, yields:

$$w = \frac{b}{a} = \frac{\sum_{i=1}^{n}x_iy_i}{\sum_{i=1}^{n}x_i^2}$$

This exists if we have a non zero feature. We then also have to check if we are actually minimizing, by seeing if the second derivative is negative:

$$f''(w) = a = \sum_{i=1}^{n}x_i^2$$

Since any squared value is positive, and we are considering a non zero feature, then we have that the second derivative is strictly positive. Thus $f''(w) > 0$ implies that the value found is a minimizer.

Least-square in 2 Dimensions

Motivating example

2D model

The linear model looks as follows:

$$\hat{y}_i = w_1x_{i1} + w_2x_{i_2}$$

This defines a 2-Dimensional Plane

Notation
If we have d features, then the d-dimensional model is:

$$\hat{y}_i = w_1x_{i1} + w_2x_{i_2} + w_3x_{i_3} + \cdots + w_dx_{i_d}$$

In words, our model is that the output is a weighted sum of the inputs. We can re-write this in summation notation:

$$\hat{y}_i = \sum_{j=1}^{d}w_jx_{ij}$$

Or in vector notation:

$$\hat{y}_i = w^Tx_i$$

Here we assume that w and x_i are column vectors.

Least squares in d-dimensions

In d-dimensions we try to minimize the following function of a vector w:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2$$

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2$$

'w' is now a vector

prediction is inner product of 'w' and 'x_i' (linear combination of features)

"Error" is still the sum of squared differences between "true" y_i and our "prediction" $w^T x_i$