Data Analytics

Lec03. Data Modeling with Python

Week 2 Recap: Exploring Two Variables

		Response	
		Categorical	**Quantitative**
Explanatory	**Categorical**	**C→C**	**C→Q**
	Quantitative	**Q→C**	**Q→Q**

- C→Q: use box-and-whisker plots
- C→C: use two-way tables
- Q→Q: use scatter-plot
- Q→C: use other methods (e.g. statistic tests)

Week 2 Recap: From Data Visualization to Visual Analytics

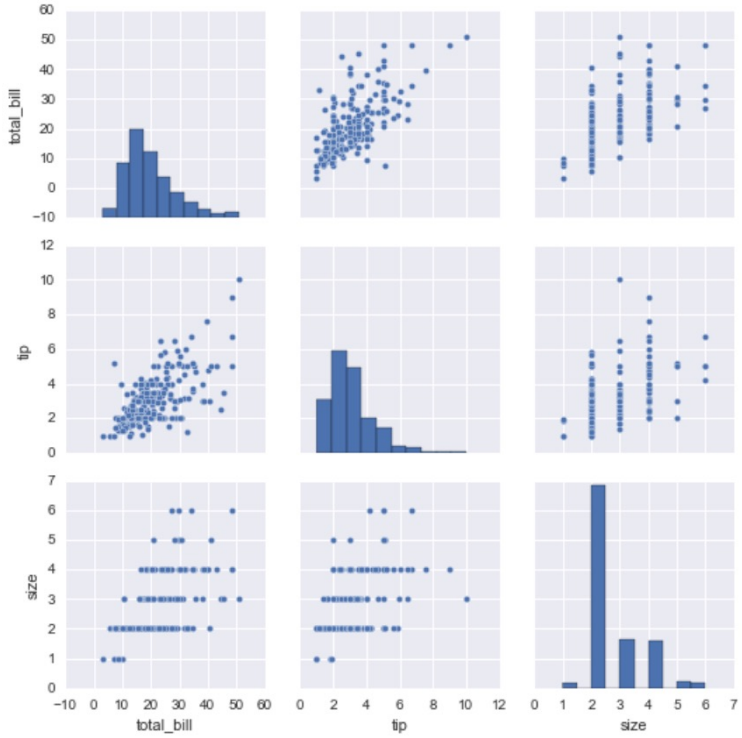
- Interactive viz = Old-fashioned viz + Interaction Scheme
  - Enable **visual analytics** via interactive and reproducible results
  - Easy and fast to develop and customize

- Old-fashioned viz
  - Great for data exploration, developed throughout the last few centuries
  - Rapid data exploration
  - Focus on most important details

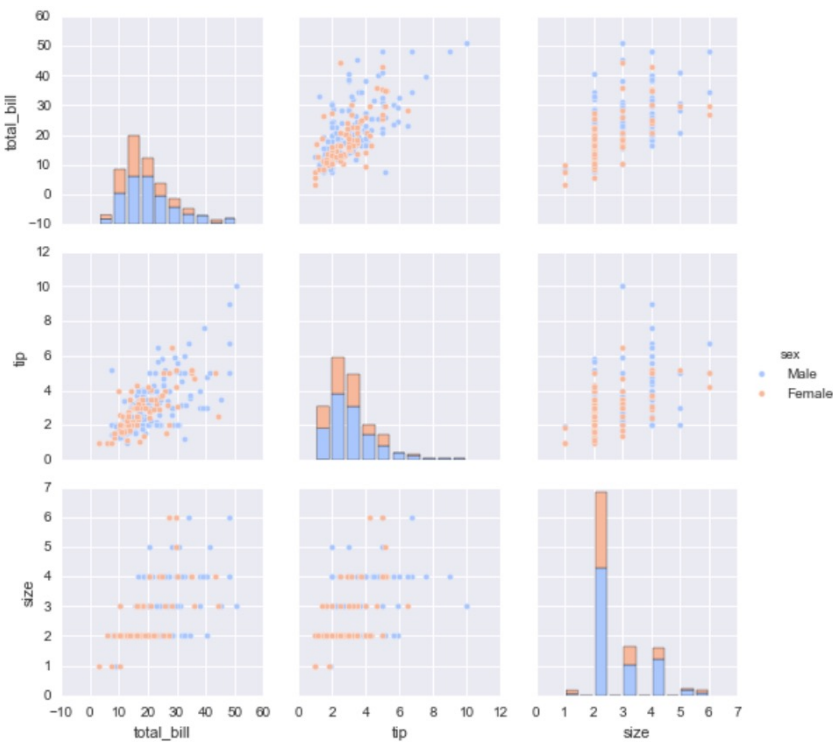
- Interactive viz
  - More and more common nowadays. New frameworks are the key enabler.
  - Support multiple analyses
  - Focus on more dimensions

Week 2 Recap: Exploratory Data Analysis with Python

- Pairplot automatically analyze pairwise relationships



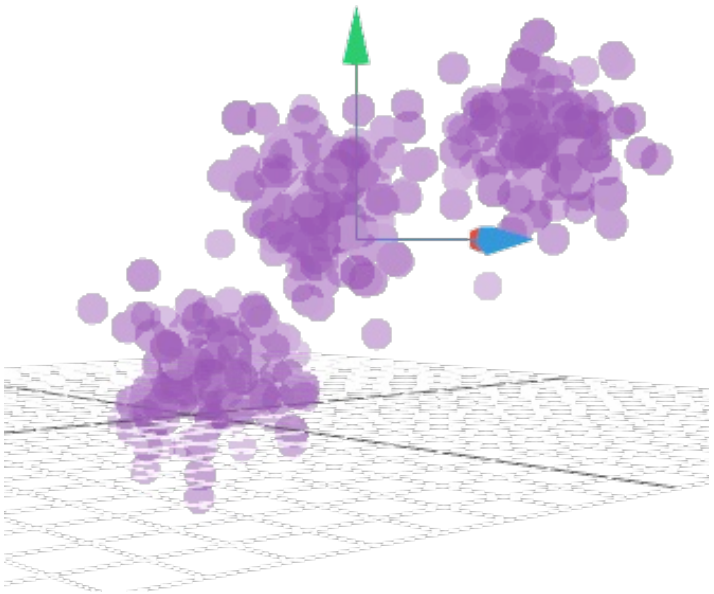
sns.pairplot(tips)

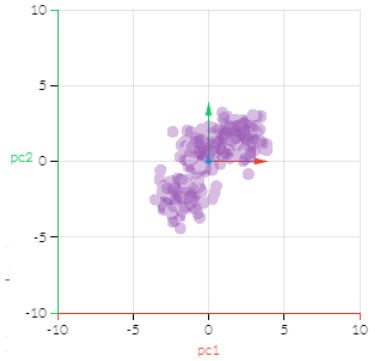


sns.pairplot(tips,hue='sex',palette='coolwarm')

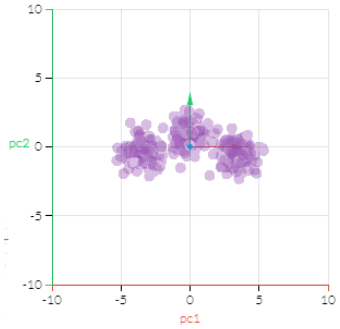
Week 2 Recap: Dimensionality Reduction

- http://setosa.io/ev/principal-component-analysis/



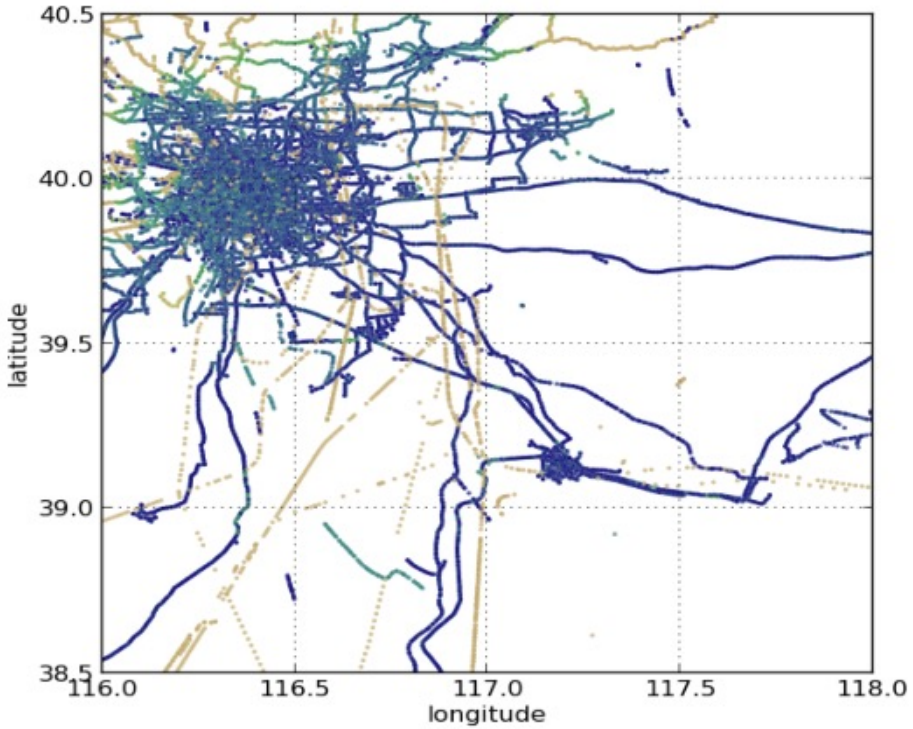


Bad feature reduction

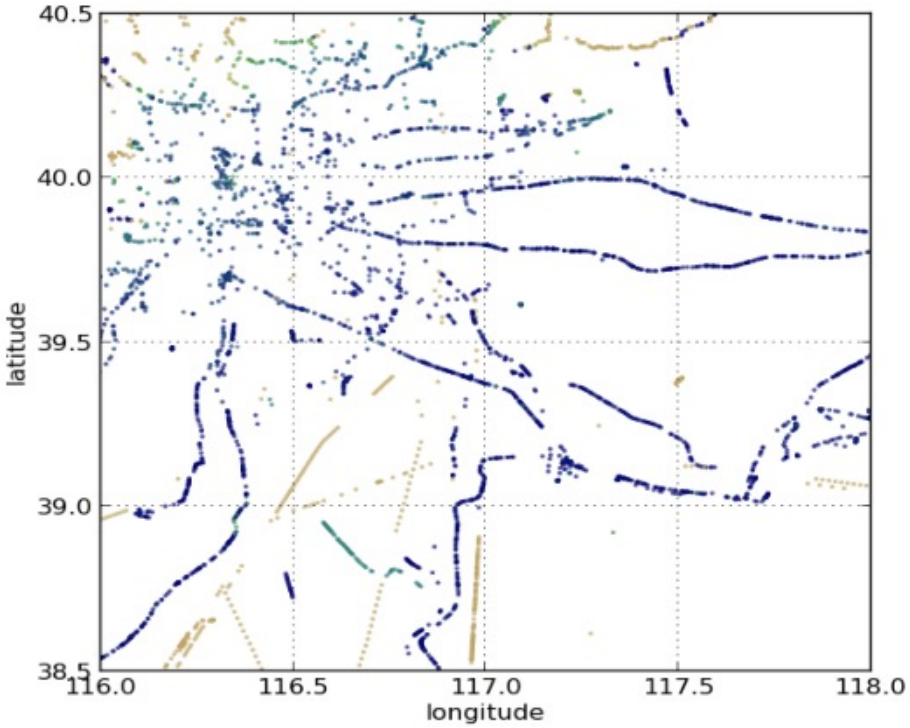


Good feature reduction

Week 2 Recap: Data Sampling



original data



a good sample

Course structure

W1. Data Processing with Python

W2. Data Exploration with Python

W3. **Data Modeling with Pytyhon**

W4. Data Analytics for Timeseries

W5. Holiday

W6-7. Data Analytics for Texts

W8. Data Analytics for Images

W9. Data Analytics for Graphs

W10-11. Data Analytics for Other Data

W12. Revision

Data Modeling - Statistical Analysis

Revise the content of pre-requisite knowledge and focus on data modeling

I. Clustering – Unsupervised Learning
II. Classification – Supervised Learning
III. Regression – Supervised Learning
IV. Model Evaluation

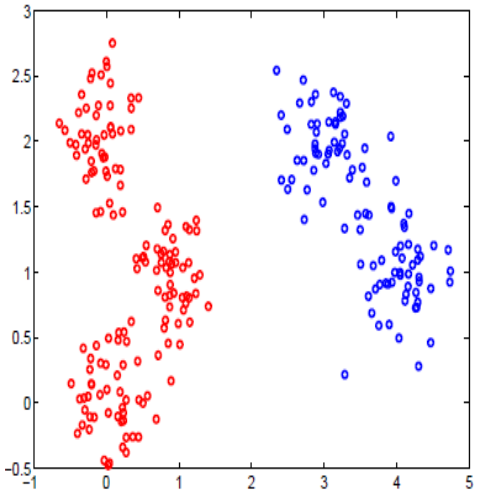
I. Clustering – Unsupervised Learning

Exploring Data Similarity

- **Clustering**: group together "similar" instances in the data sample
  - Distribute data into different groups such that similar instances are in the same group
  - Similarity between data entries is defined in terms of some distance metric (can be chosen)
  - Constraint: instances from the same group must be similar than instances from different groups
- Common techniques:
  - K-means clustering
  - DBScan
  - Co-clustering

Problem Statement

- Cluster: A collection/group of data objects/points
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis
  - find *similarities* between data according to characteristics underlying the data and grouping similar data objects into clusters
  - unsupervised learning: no predefined classes for a training data set
  - Two general tasks: identify the "natural" clustering number and properly group objects into "sensible" clusters
- Why clustering?
  - A stand-alone tool to gain an insight into data distribution
  - A pre-processing step of other data analysis routines



how many clusters?

Clustering: Applications

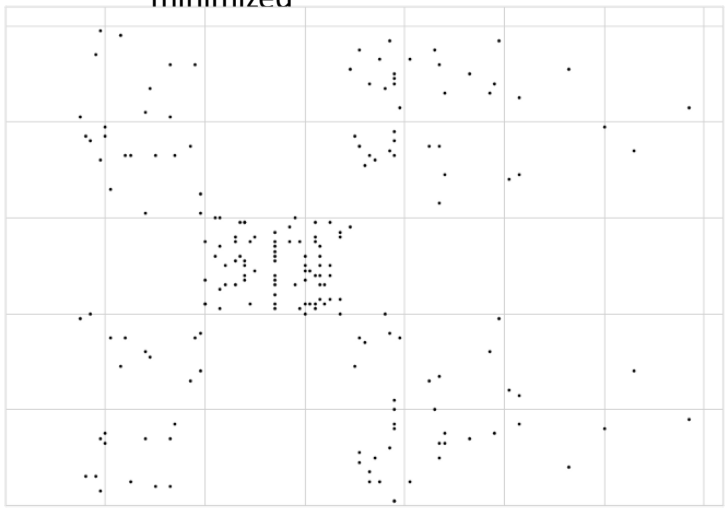
- Applications:
  - Bank Security: fraud pattern discovery
  - Biology: taxonomy of living things such as kingdom, phylum, class, order, family, genus and species
  - City-planning: Identifying groups of houses according to their house type, value, and geographical location
  - Climate change: understanding earth climate, find patterns of atmospheric and ocean
  - Finance: stock clustering analysis to discover correlation between shares
  - Information retrieval/organisation: Google search, topic-based news
  - Land use: Identification of areas of similar land use in an earth observation database
  - Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
  - Social network mining: special interest group automatic discovery (community recognition)

Clustering: Methods

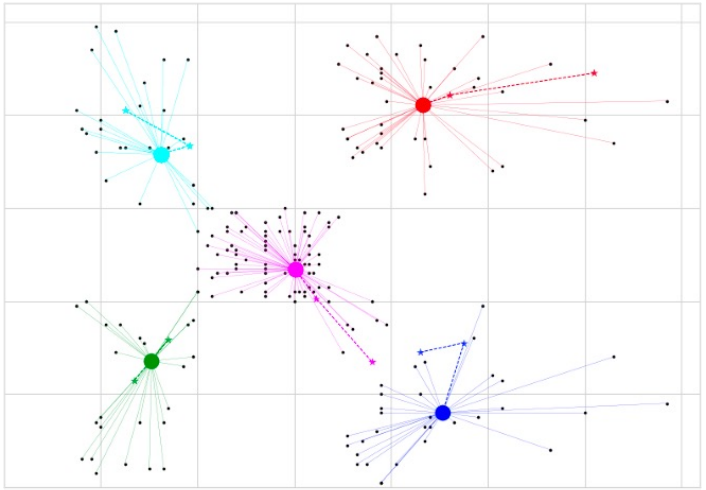
- Partitioning approach (1 variable)
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square distance
  - Typical methods: K-means, K-medoids, CLARANS, ……
- Hierarchical approach (1 variable)
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Agglomerative, Diana, Agnes, BIRCH, ROCK, ……
- Density-based approach (1 variable)
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue, ……
- Co-clustering (2 variables)

K-means clustering

- A simple greedy algorithm (usually called Lloyd's algorithm):
  - Divide data into K clusters, each of which has a center (centroid)
  - Each data point belongs to a single cluster only
  - K centroids are chosen such that distance (usually Euclidean) from data points to their centroids is locally minimized



Input



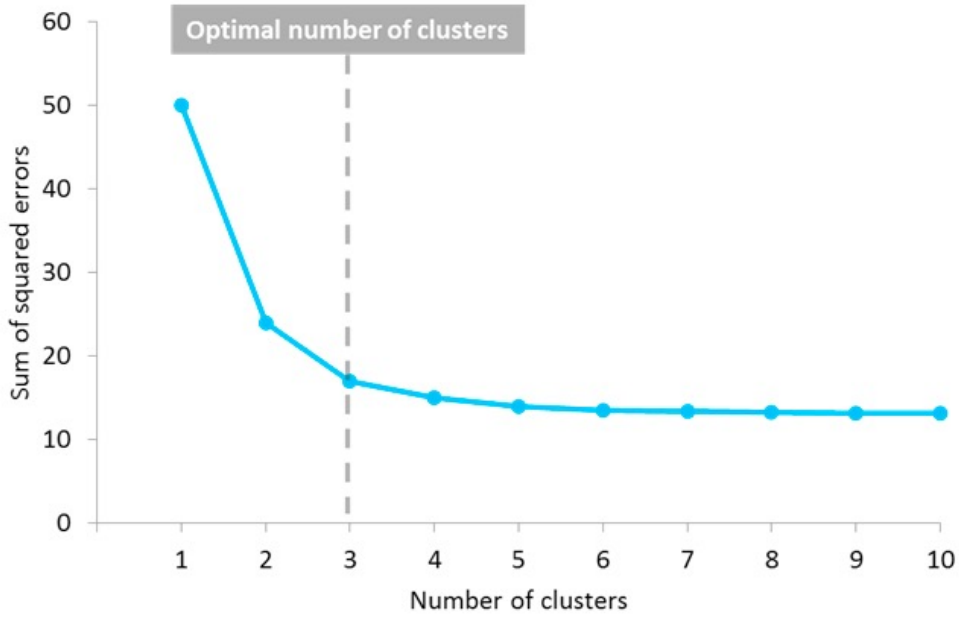
Output

K-means algorithm

- General steps:
  1. Choose the number K of clusters
  2. Select K points as initial cluster centers
  3. Assign each data point to the nearest centroid
  4. Compute new centroid and recompute cluster assignment
  5. Repeat step 4 until new centroids are the same as old centroids

Step 1: Choose the number of clusters

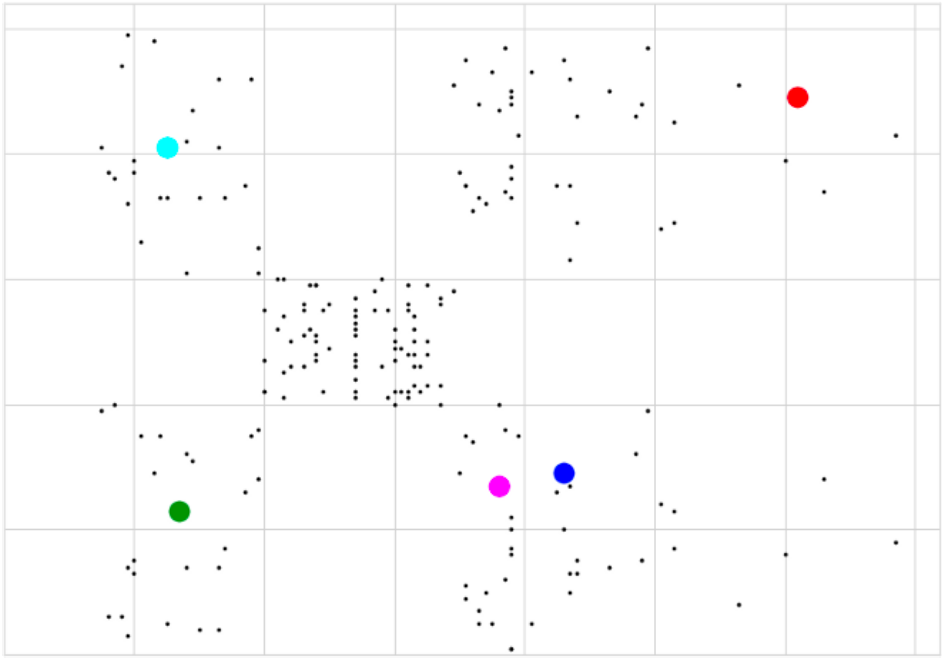
- No ultimate way to find the exact no. clusters
  - K too small: data points inside a cluster are too different from each other (high deviations)
  - K too large: the whole dataset is fragmented
- Best practice: trial and error
- Normally, you will be given K before-hand



It all depends on your dataset

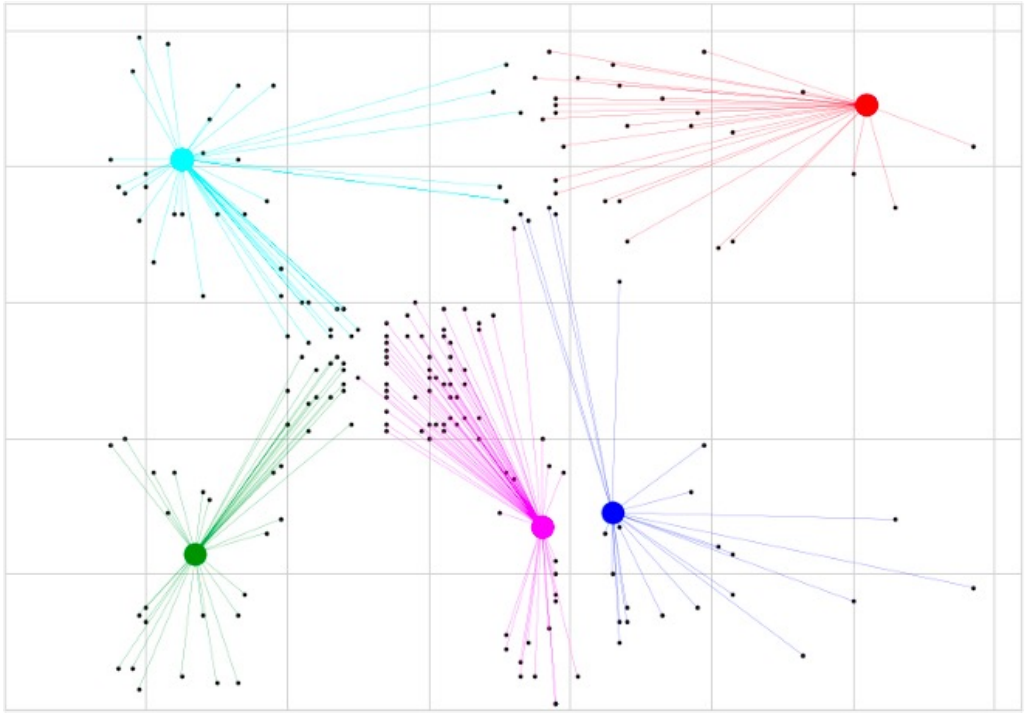
Step 2: Select K points as initial centroids

- Assume $K = 5$
- Most popular: random initialization
- OPTIONAL: heuristic initialization
  - Maxmin (further point heuristic)
  - Sometimes can improve performance



Step 3: Assign data point to the nearest cluster

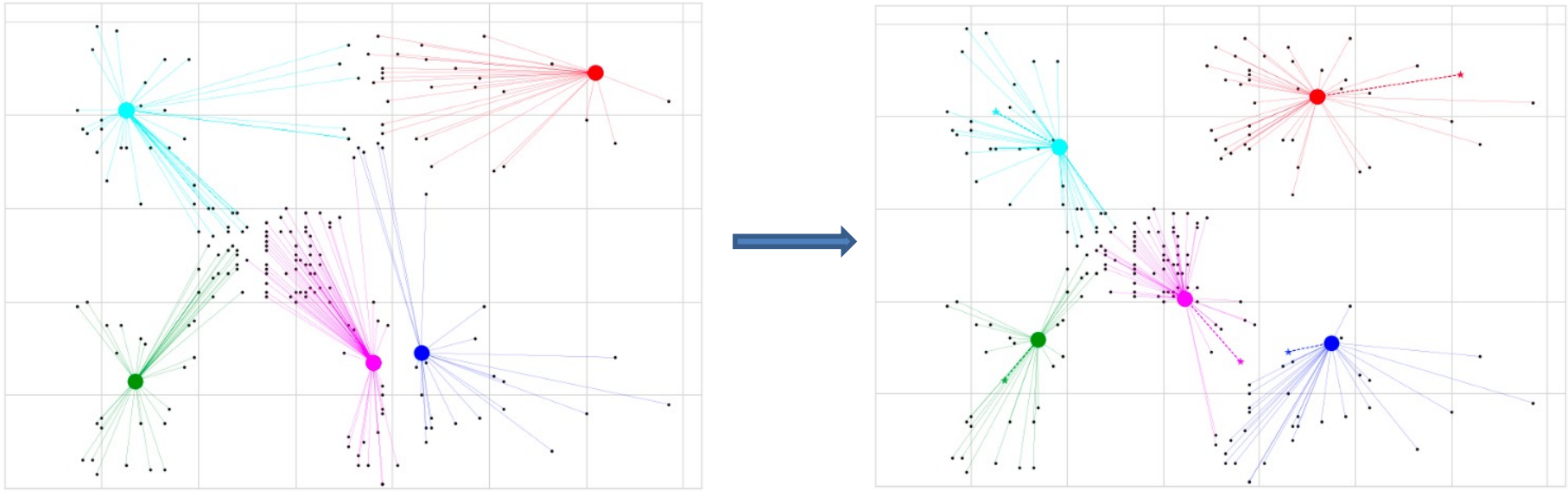
- For each point:
  - Calculate the distance (Euclidean) to each centroid
  - Find which centroid has the lowest distance



Cluster assignment

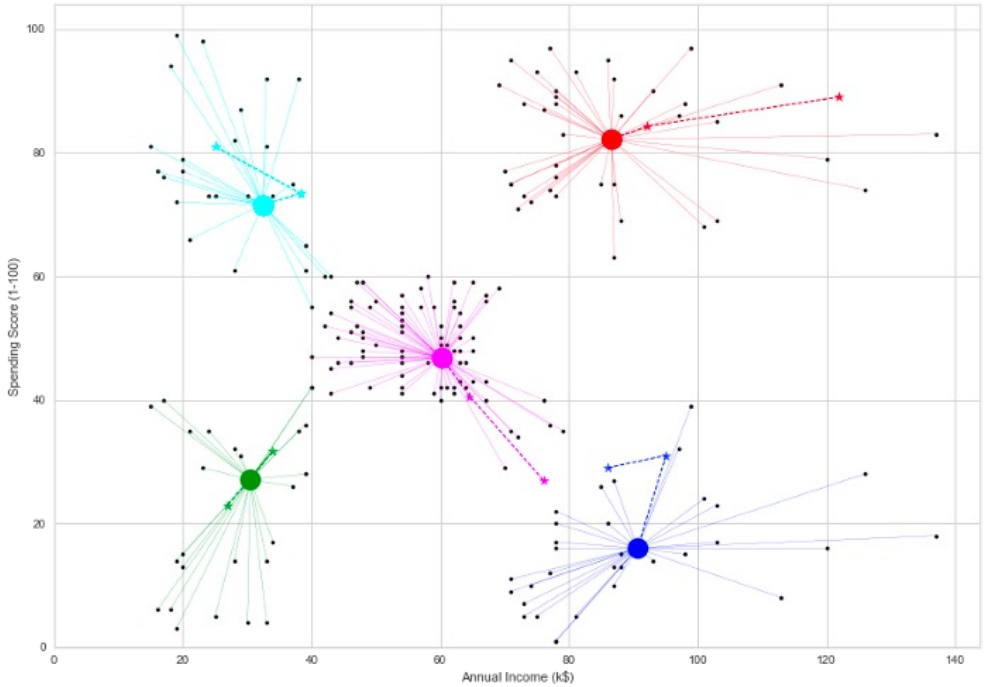
Step 4: Compute the new centroid of each cluster and recompute cluster assignment

- Current centroids might not be really the ``means'':
  - Data points in different clusters might be similar than those from the same cluster → violate clustering constraint
- Need to compute new centroid:
  - Average of all data points in a cluster
  - Recompute cluster assignment as in step 3



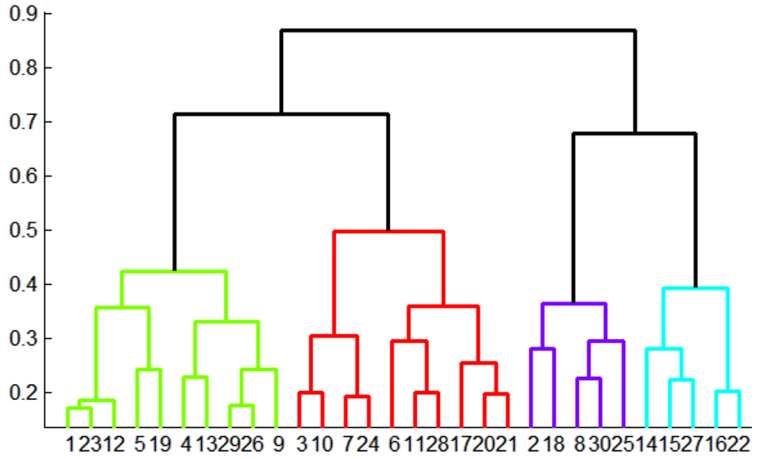
Step 5: Repeat step 4 until no centroid change

- If current centroid is exactly the ``mean'' of its cluster:
  - Stop and output the final clusters
- Otherwise:
  - Repeat step 4



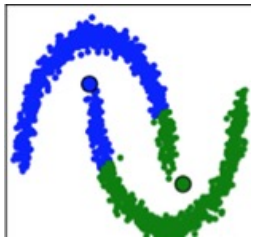
Hierarchical Clustering

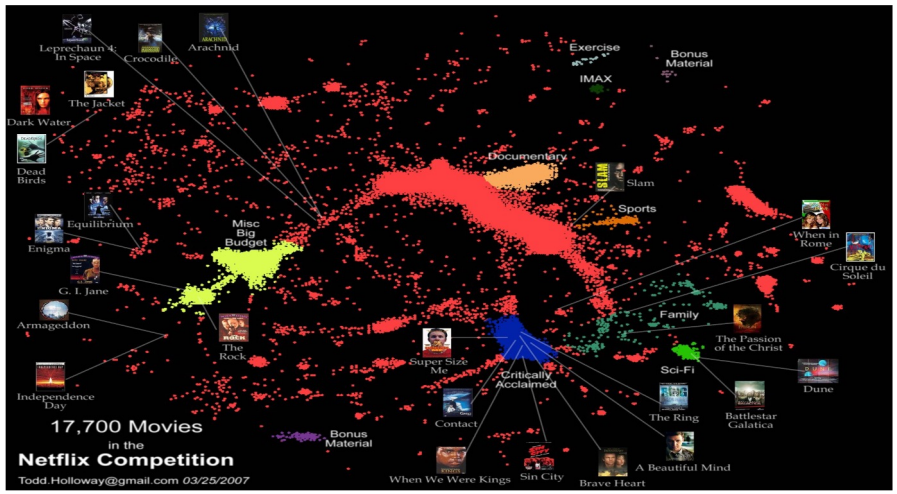
- Agglomerative (bottom up):
  - Initially, each point is a cluster
  - Repeatedly combine the two "nearest" clusters into one
- Divisive (top down):
  - Start with one cluster and recursively split it
- Does not require the number of clusters k in advance
- But needs a cut-off threshold to return clusters



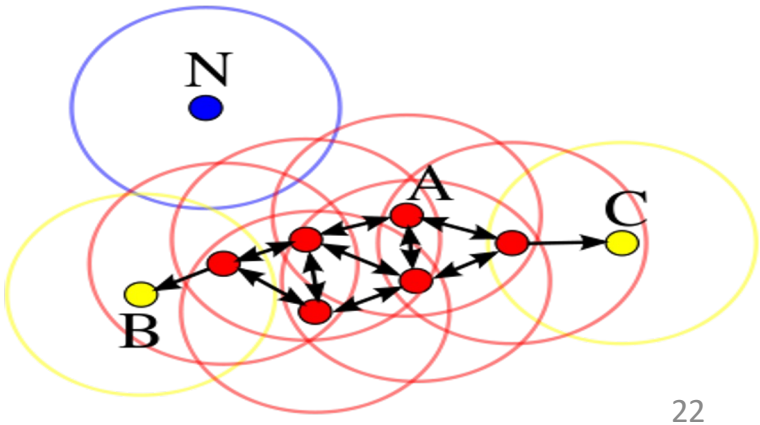
Density-based clustering: DBScan

- **Motivation:** Centroid-based clustering methods like k-means favor clusters that are spherical, and have great difficulty with anything else, e.g.





- **How it works:** follows the shape of dense neighborhoods of points.
  - Core points have at least minPts neighbors in a sphere of diameter ε around them.
  - The red points here are core points with at least minPts = 3 neighbors in an ε-sphere around them.



DBScan: Problem Statement

- DBScan: density-based clustering
- Problem statement: construct a set of clusters such that
  - Each cluster satisfies
    - Maximality: if $q \in C$ is a core point, and p is density reachable from q, then also $p \in C$
    - Connectivity: any two points in C must be density connected

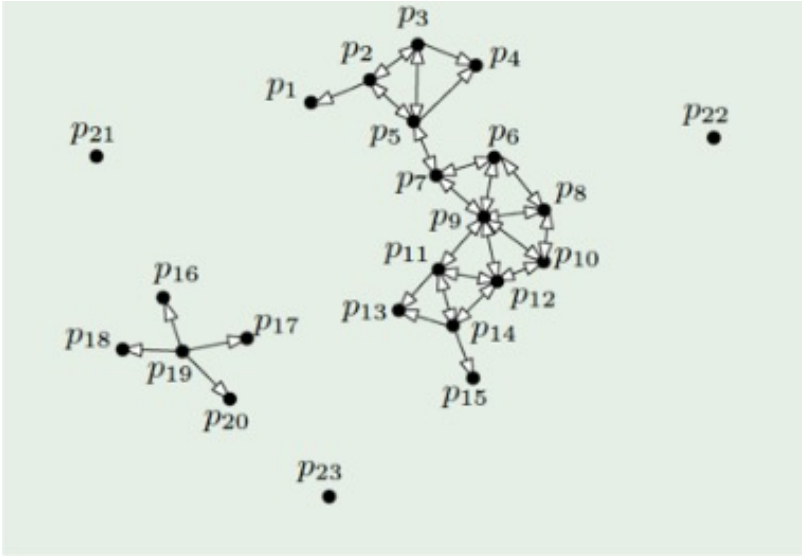
      →A cluster contains at least one core point
  - The set of clusters is unique
  - Clusters are not necessarily disjoint

DBScan Algorithm (Optional)

- Construct a directed graph G using direct density-reachability
- Initialize
  - V_{core}=set of core points
  - P=set of all points
  - set of clusters $C = \{\}$
- While V_{core} not empy
  - Select a point p from V_{core} and construct $S(p)$, the set of all points density-reachable from p: Breadth-first search on G starting from p
  - $C = C \cup \{S(p)\}$
  - $P = P \setminus S(p)$
  - $V_{core} = V_{core} \setminus S_{core}(p)$
    - where $S_{core}(p)$=core points in $S(p)$
- Mark remaining points in P as unclustered

DBScan - Example

- First cluster: choose p_2 and compute $S(p_2) = \{p_1, \dots, p_{15}\}$
- Then, $V_{core} = \{p_{19}\}$
- Second cluster: $\{p_{16}, \dots, p_{20}\}$
- $p_{21}, \dots, p_{23}$ are outliers



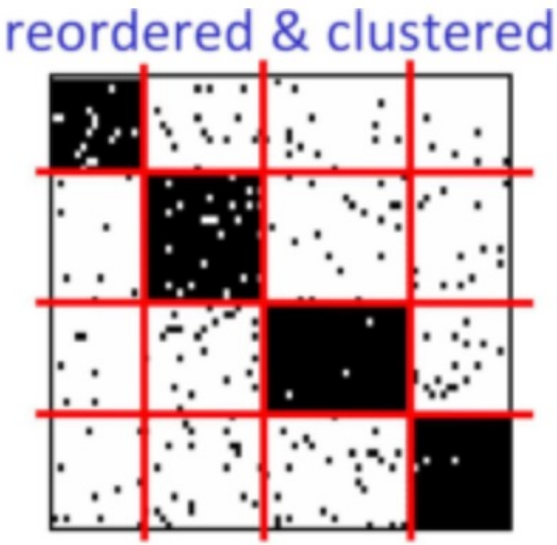
Source: Yufei Tao, Chinese University of Hong Kong

Co-clustering

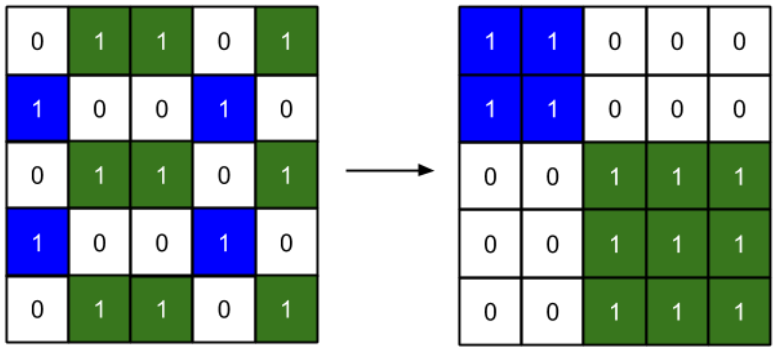
- Real world data is often bimodal
  - Created by a joint interaction between two variables
  - E.g. a customer buying a product
- Traditional clustering (1 variable):
  - Customer clustering: group customers buying the same products, regardless of product clusters
  - Product clustering: group products being bought by the same customers, regardless of customer clusters
- Co-clustering (2 variables):
  - Group customers and products simultaneously
    - Two customers are similar if they buy similar products
    - Two products are similar if they are bought by similar customers

Co-clustering: Example

- Clustering two variables simultaneously



A simple case: similarity values are only 1s and 0s.



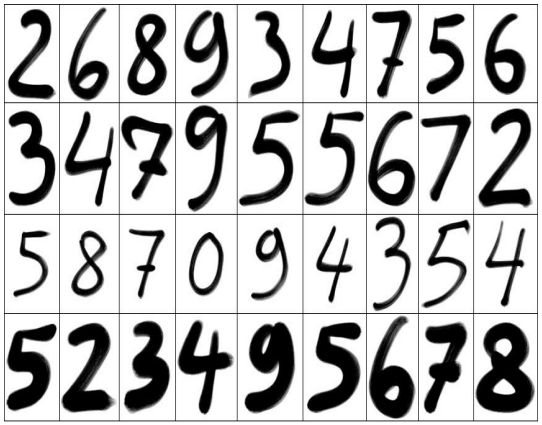
Vlachos, Michail et al. "Improving Co-Cluster Quality with Application to Product Recommendations." CIKM (2014).

II. Classification

- **Task:** predict the label/class of data
- **How it works** (supervised learning):
  - Learn from a collection of pre-labeled data (training set)
    - Each record contains a set of attributes or features
    - Each record is assigned with a label/class
  - Build a model for calculating the class as a function of the attributes.
  - Evaluate the model on previously unseen records (test set)
- **Goal:** increase the accuracy of test set
- **Applications:**
  - Pattern Recognition
  - Time Series Prediction
  - Signal Processing
  - Anomaly Detection

Classification: Other Applications

- Handwritten digit recognition:



Classifier

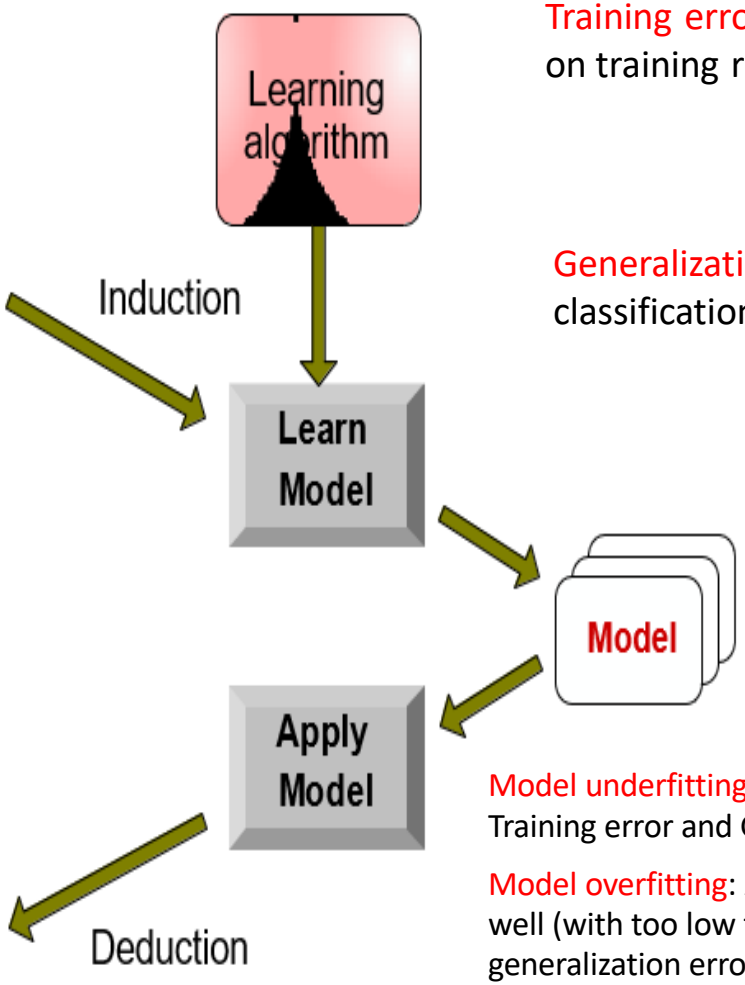
2	6	8	9	3	4	7	5	6
3	4	7	9	5	5	6	7	2
5	8	7	0	9	4	3	5	4
5	2	3	4	9	5	6	7	8

- Email spam recognition:

Classification Pipeline

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Training errors: number of classification errors on training records.

Generalization (test) errors: number of classification errors on test records.

Model underfitting: model is too simple such that both Training error and Generalization error are high.

Model overfitting: A model that fits the training data too well (with too low training errors) may have a higher generalization error than a model with higher training error.

A good model should have low errors of both types.

Classification: Methods

1. K Nearest Neighbors (k-NN)
2. Decision Tree
3. Naïve Bayes
4. Linear Classifier: Support vector machine
5. Non-linear Classifiers
6. Supervised neural networks
7. Ensemble method

1. k-NN: Example

Find the label for a query item
from k closest matches
in a labeled dataset

With k=3, majority vote
should be “cat”

k-NN

- **Training Time:** memorize all data and labels
- **Testing Time:** predict the label of a new data by take majority vote from K closest points
  - Can also use a weighted vote of the neighbors.
  - Can use different similarity metrics:
    - Euclidean distance: simplest, fast to compute, geometric space

$$d(x, y) = ||x - y||$$

    - Cosine distance: good for documents, images

$$d(x, y) = 1 - \frac{x.y}{||x||.||y||}$$

    - Jaccard distance: for set data

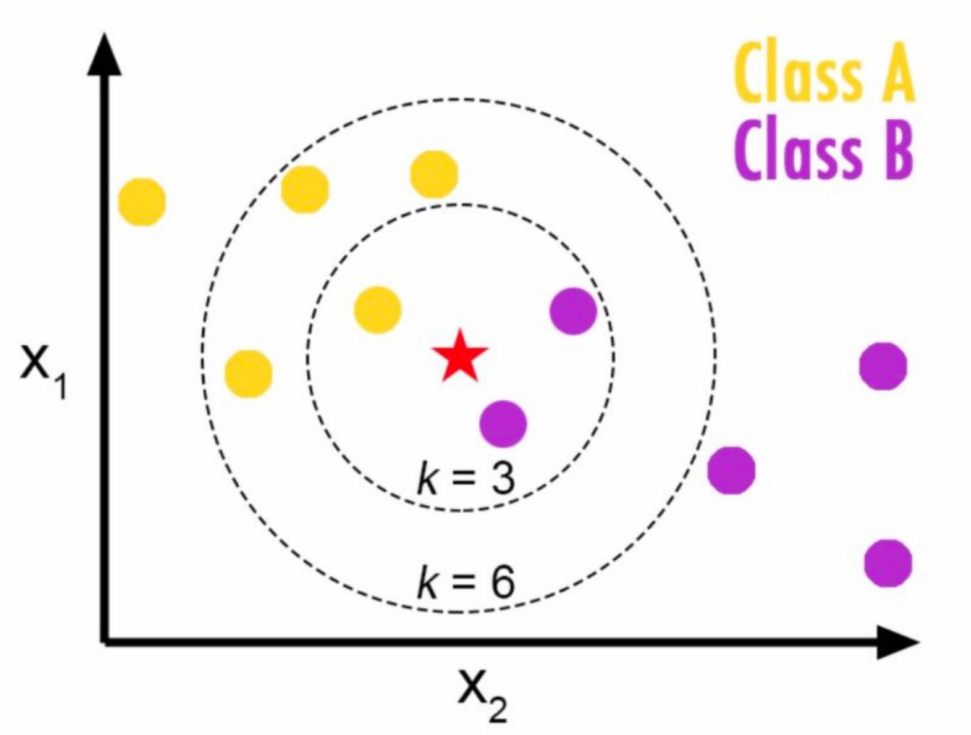
$$d(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

    - Hamming distance: for string data

$$d(x, y) = \sum_{i=1}^{n} 1_{x_i \neq y_i}$$

k-NN

- Choosing a K will affect what class a new point is assigned to:



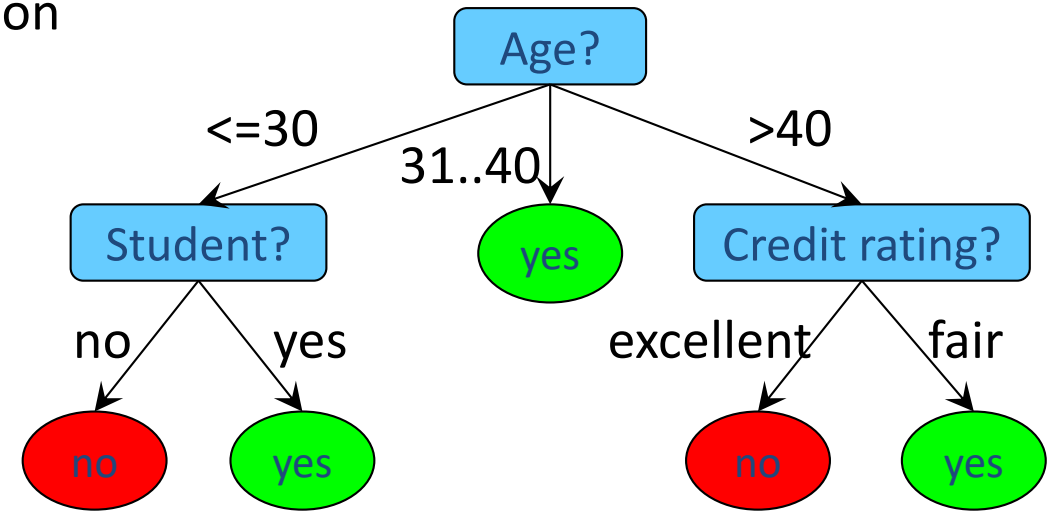
K-NN: pros and cons

- **Pros:**
  - Very simple
  - Training is trivial
  - Works with any number of classes
  - Easy to add more data
  - Few parameters: k, distance metric
- **Cons:**
  - High computation cost for testing (worse for large data sets)
  - Not good with high dimensional data
    - The curse of dimensionality: similar data in low-dimensional space might be far away from each other in high-dimensional space
  - Categorical features don't work well

2. Decision Tree

- **Example:**
- **Model:** flow-chart-like tree structures
  - Nodes are tests on a single attribute
  - Branches are attribute values
  - Leaves are marked with class labels
- **Training time:** optimize classification accuracy (decision tree induction)
  - NP-hard
  - Heuristic: top-down tree construction + pruning

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



Decision Tree: Pros and Cons

- **Pros:**
  - Easy to interpret visually
  - Work well with qualitative (categorical) features
- **Cons:**
  - Prone to over-fitting
  - Non-incremental: need to be re-trained from scratch if new training data becomes available

3. Naïve Bayes Classifier

- Given a new instance, estimate for each possible value of the class attribute its probability.
  - Choose the value with the maximum probability.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

1. Estimate conditional probabilities
P1= P(cheat=yes| Ref=No, Ms=Married, Income=80)
P2= P(cheat=No| Ref=No, Ms=Married, Income=80)

If P1>P2, classify as cheat, otherwise no cheat

How to estimate the probabilities?

Naïve Bayes Classifier

- **Solution:** use Bayes theorem $P(C|A) = \frac{P(A|C)P(C)}{P(A)}$
  - No need to estimate P(A) for comparisons of $P(C = c_1|A)$ and $P(C = c_2|A)$
  - $P(C = c_k)$ can be estimated as the fraction of training records taking class value c_k

$$P(c_k) = \frac{N_k}{N}$$

  - Estimate $P(A|c_k)$ is hard: assume the features $a_1, \ldots, a_n$ are independent given class C.

$$P(A|c_k) = \prod_{i=1}^{n} P(a_i|c_k)$$

$\frac{N_{ik}}{N_k}$

N: Total number of data

N_k: Total number of records in class k

N_{ik}: Total number of records with value a_i for feature i in class k

Zero conditional probability (OPTIONAL)

- If no sample contains the feature value
  - We face a zero conditional probability problem during testing:

    $\prod_{i=1}^{N} P(a_i|c_k) = 0$ if any $P(a_i|c_k) = 0$
  - Thus, we have to re-estimate class conditional probabilities as:

$$P(a_i|c_k) = \frac{N_{ik} + mp}{N_k + m}$$

    - $m \geq 1$: weight to prior (number of "virtual" examples, usually up to 1% number of training samples with class c_k)
    - p: prior estimate (usually, $p = 1/t$ for t possible values of a_i)

Naïve Bayes Classifier: Example

- E.g. compute
  $P(C|No, Married, 80K) \sim P(No, Married, 80K|C)\, P(C)$
  - P(C=Yes) = 3/10, P(C=No) = 7/10
  - P(No|C=Yes) = 3/3, P(No|C=No) = 4/7
  - P(Married|C=Yes) = 0, P(Married|C=No) = 3/7
    → Have to use zero conditional probability
    - P(Married | C=Yes) = (0+1∗1/3)(3+1)=1/12
  - P(80K | C=Yes) = 0, P(80K | C=No) = 0
    → Have to use zero conditional probability
    - P(80K | C=Yes) = (0+1*1/10)/(3+1) = 1/40
    - P(80K | C=No) = (0+1*1/10)/(7+1) = 1/80
- Put it all:
  - P(C =Yes|No, Married, 80) = 3/10 * 3/3 * 1/12 * 1/40 = 0.000625
  - P(C =No|No, Married, 80) = 7/10 * 4/7 * 3/7 * 1/80 = 0.00214
  - So the new sample is not likely to cheat

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Naïve Bayes Classifier: Pros and Cons

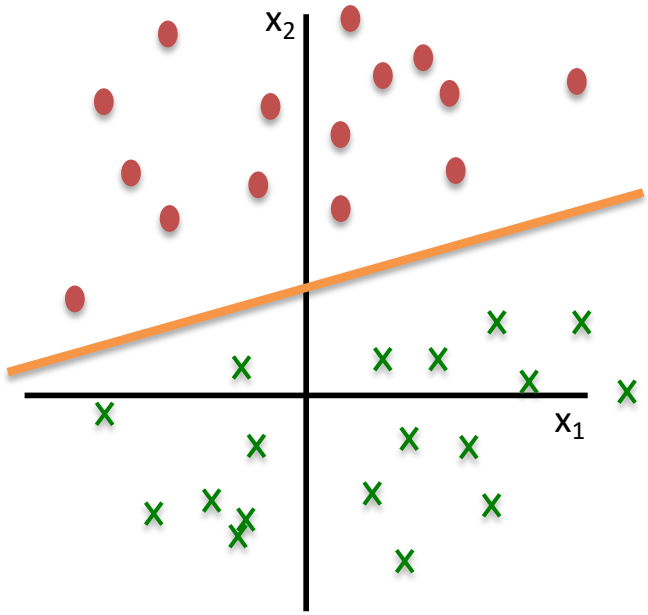
- **Pros:**
  - Computationally fast
  - Simple to implement
  - Works well with high dimensions
- **Cons:**
  - Relies on independence assumption
  - Not work well with categorical inputs (zero conditional probability issue)

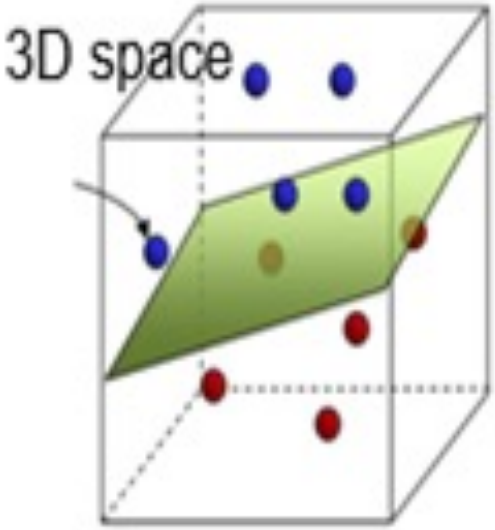
4. Linear Classification: Support Vector Machine

- We want to draw a line to separate two classes

Linearly separable case!

An Ideal case



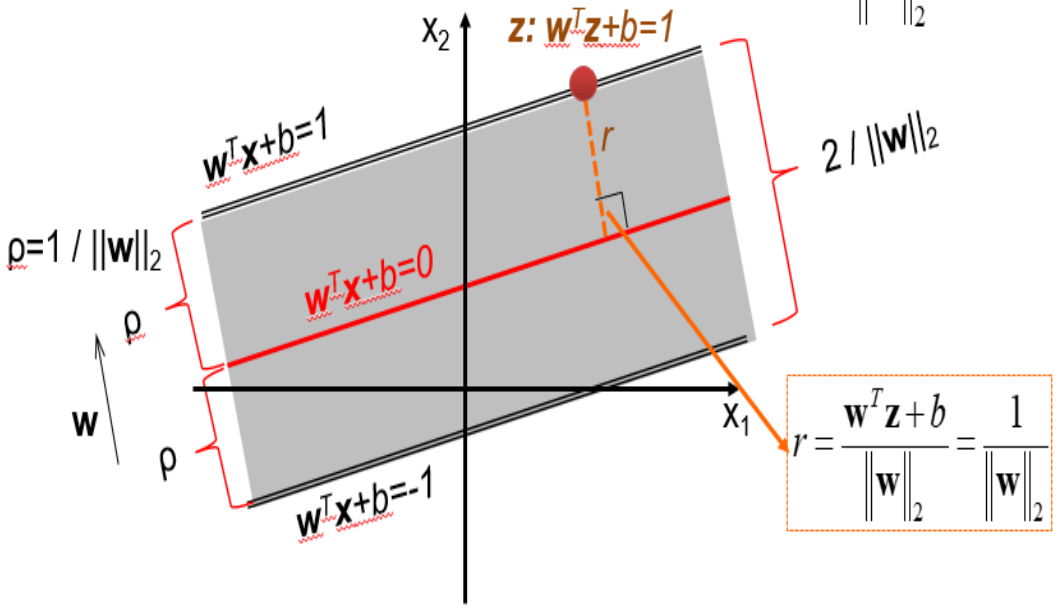


SVM: Core Idea

- **Model:** we focus on two parallel hyperplanes:

$$\begin{cases} \mathbf{w}^T\mathbf{x}+b=1 \\ \mathbf{w}^T\mathbf{x}+b=-1 \end{cases}$$

Geometrically, the distance between these two planes is $\frac{2}{\|\mathbf{w}\|_2}$



SVM: Objective function

- **Goal:** The aim of SVM is simply to find an optimal hyperplane to separate two-classes of data points with the widest margin.

- This results in the following constrained optimisation:

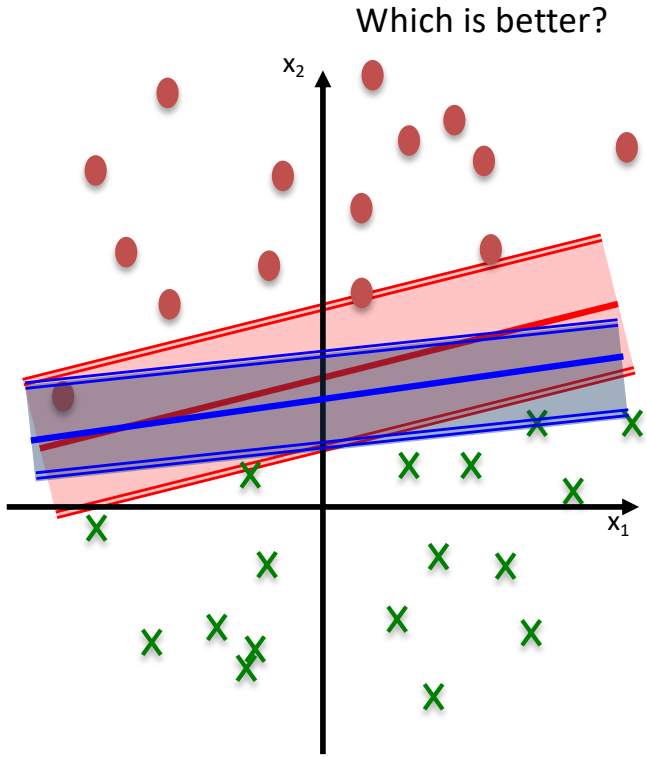
$$\text{margin:} \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$$

Margin maximisation

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{s.t.} \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\}$$

Stopping training samples from falling into the margin.



SVM - Training (Optional)

- **SVM training:** the process of solving the following constrained optimisation problem (aka hard-margin SVM):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i \in \{1,\ldots,N\} \end{aligned}$$

Lagrange →

$$\text{Dual problem} \begin{cases} \max\limits_{\lambda \in \Re^N} \sum\limits_{i=1}^{N} \lambda_i - \frac{1}{2}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \\ \lambda_i \geq 0 \end{cases}$$

- **Solution:** quadratic programming https://en.wikipedia.org/wiki/Quadratic_programming
  - Following calculus and optimisation theory, we use Lagrange multipliers $\lambda_i \geq 0$ and formulate the Lagrangean function:

$$L\left(b,\mathbf{w},\lambda\right) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \lambda_i \left[1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right]$$

→

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

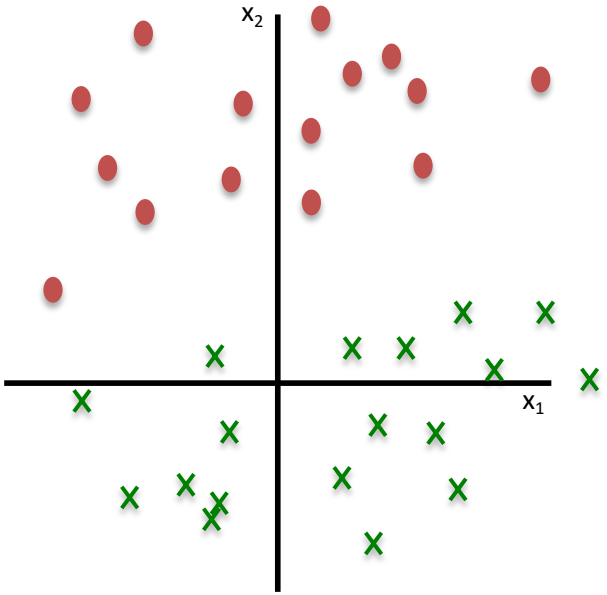
$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = \mathbf{w}$$

  - By differentiating L() and setting the gradient to zero, we can express the Lagrangean function using only the multipliers (called a dual problem).

$$L\left(\lambda\right) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

SVM: Soft Margin

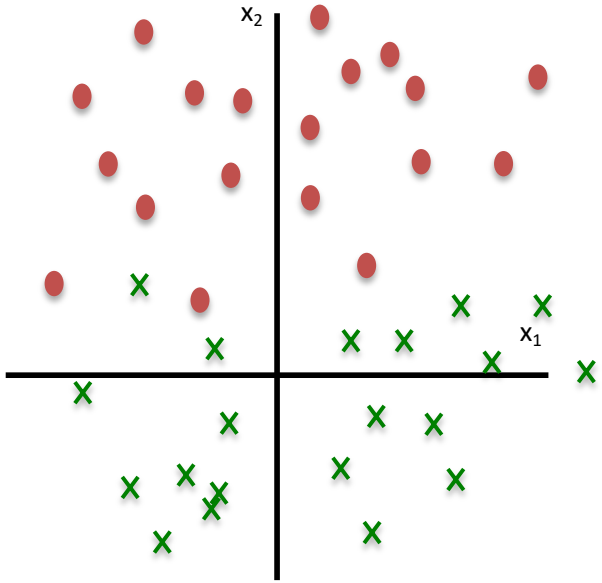
So far, we work on simple cases like this:



separable data patterns

In practice, no datasets are ideally linearly separable. This means that some data points are bound to be misclassified with a linear hyperplane.

What if the data points look like this?



non-separable data patterns

SVM with softmax margins

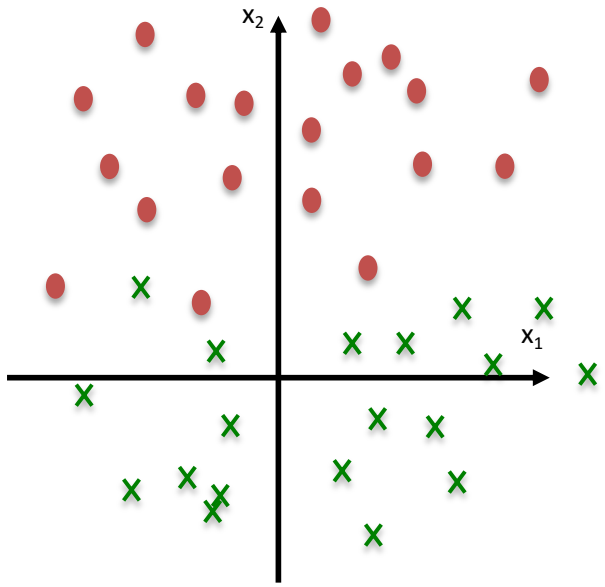
$$\min_{(\mathbf{w},b)\in\Re^{d+1}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \left.\begin{array}{l} y_i\left(\mathbf{w}^T\mathbf{x}+b\right) \geq 1-\xi_i \\ \xi_i \geq 0 \end{array}\right\} \forall i \in \{1,\ldots,N\}$$

C≥0 is a user defined parameter, which controls the **regularisation**. This is the trade-off between complexity and non-separable patterns.

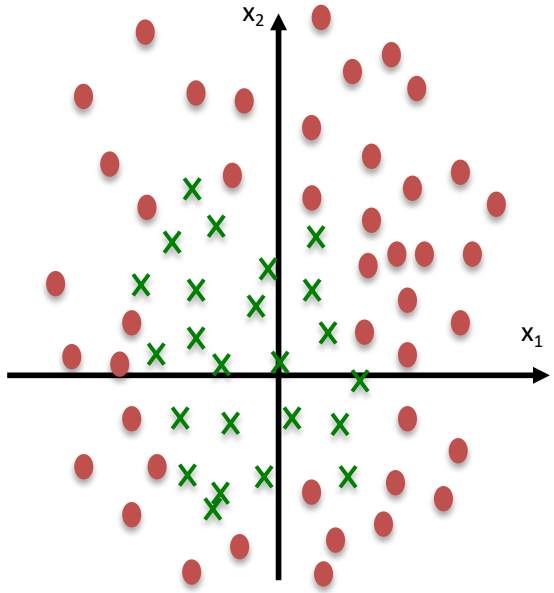
5. Non-linear Classifier

So far, we can handle linear cases like this:



linear data patterns

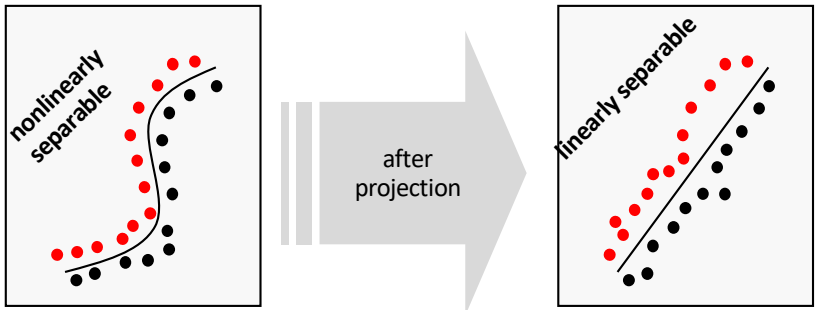
What if the data points look like this?



non-linear data patterns

SVM - Kernel Methods

- A method to handle nonlinear data patterns.
- Similar to linear basis function model: we will **project each data point to a new feature space** in order to make the patterns linearly separable in that space.



Kernel SVM demos:
https://www.youtube.com/watch?v=3liCbRZPrZA
https://www.youtube.com/watch?v=ndNE8he7Nnk

- Commonly used Kernel functions

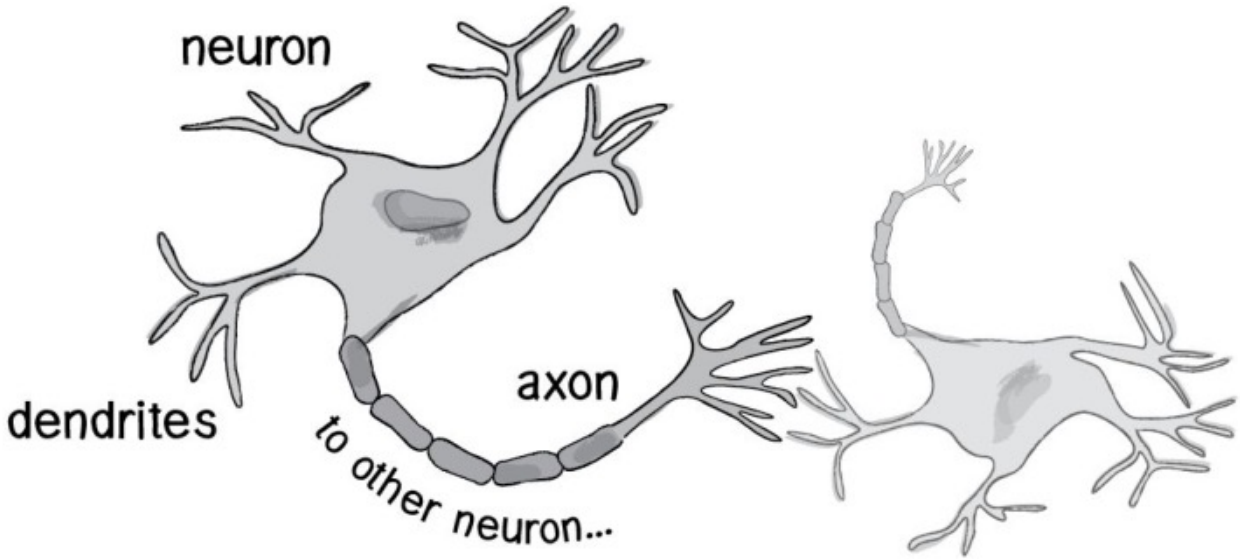
Kernel Name	Expression K(**x**,**y**)	Comments
Linear	$\mathbf{x}^T\mathbf{y}$	No parameter
Polynomial	$\left(\mathbf{x}^T\mathbf{y}+1\right)^p$	p is a user defined parameter
Gaussian, also called radial basis function (RBF)	$\exp\left(-\left\|\mathbf{x}-\mathbf{y}\right\|_2^2/\left(2\sigma^2\right)\right)$	σ is the user defined width parameter

SVM: Pros and Cons

- **Pros:**
  - Handle high dimensional data well
  - Work with both linear and non-linear boundary depending on the kernel used
- **Cons:**
  - Susceptible to over-fitting depending on kernel → Different kernels have different uses (No clear winner)

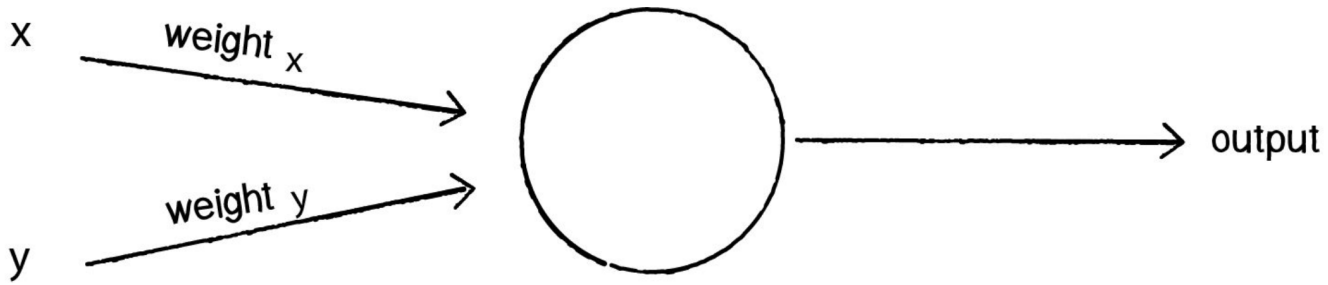
6. Neural Networks

- Neural networks are modeled after biological neural networks and attempt to allow computers to learn in a similar manner to humans
- The human brain has interconnected neurons with dendrites that receive inputs, and then based on those inputs, produce an electrical signal output through the axon



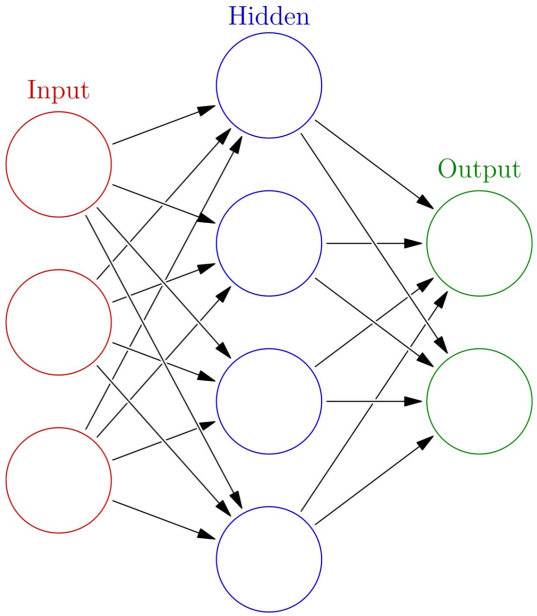
6. Neural Networks

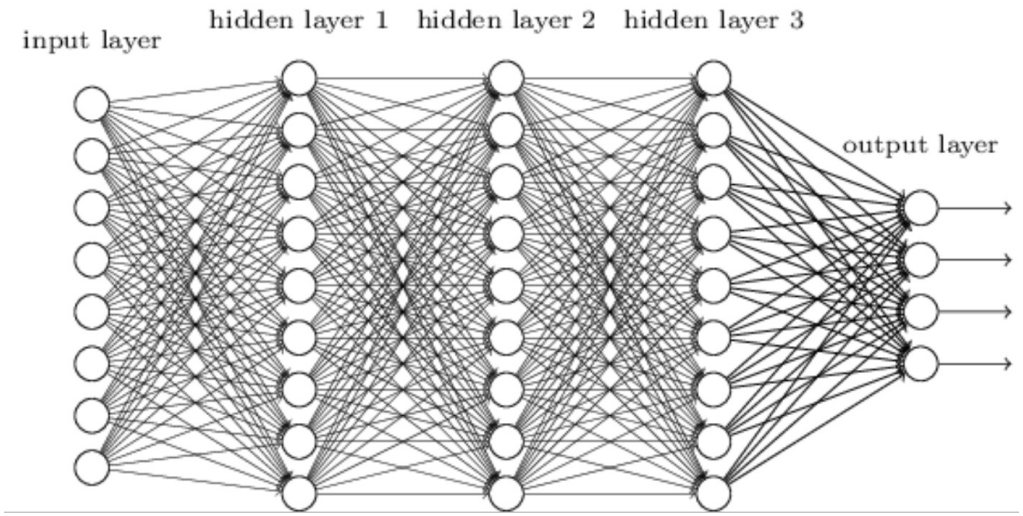
- **Perceptron** (the simplest neural network): follows the "feed-forward" model, meaning inputs are sent to the neuron, are processed, are result in an output.
  1. Receive inputs
  2. Weight inputs
  3. Sum Inputs
  4. Generate Output
- E.g. f(x,y) = weight_x * x + weight_y * y
- **Learning** = find the "best" weights for a set of inputs



Neural networks

- In general neural networks, there are 3 types of layers:
  - Input layer
  - Hidden layer
  - Output layer
- Where does the term "Deep Learning" come from?
  - A neural network with many hidden layers
  - Microsoft won vision recognition contest with 152 layers





Neural networks

- **General learning process:**
  1. Provide the inputs for which there is a known answer
  2. Ask the network to guess an answer
  3. Compute the error (how far off the correct answer?)
  4. Adjust all the weights according to the error
  5. Return to step 1 and repeat until we are satisfied with the error
- **More details:** see Deep Learning courses

Neural networks: pros and cons

- **Pros:**
  - State-of-the-art model with very high accuracy
  - Learning process can be parallelized and incremental
- **Cons:**
  - Computationally expensive (need special hardware such as GPU)
  - Hard to choose parameters (e.g. number of layers, types of layers, width of layers)

7. Classification: Ensemble Method

Are like **crowdsourced machine learning algorithms**:

- Take a collection of simple or *weak* learners
- Combine their results to make a single, better learner

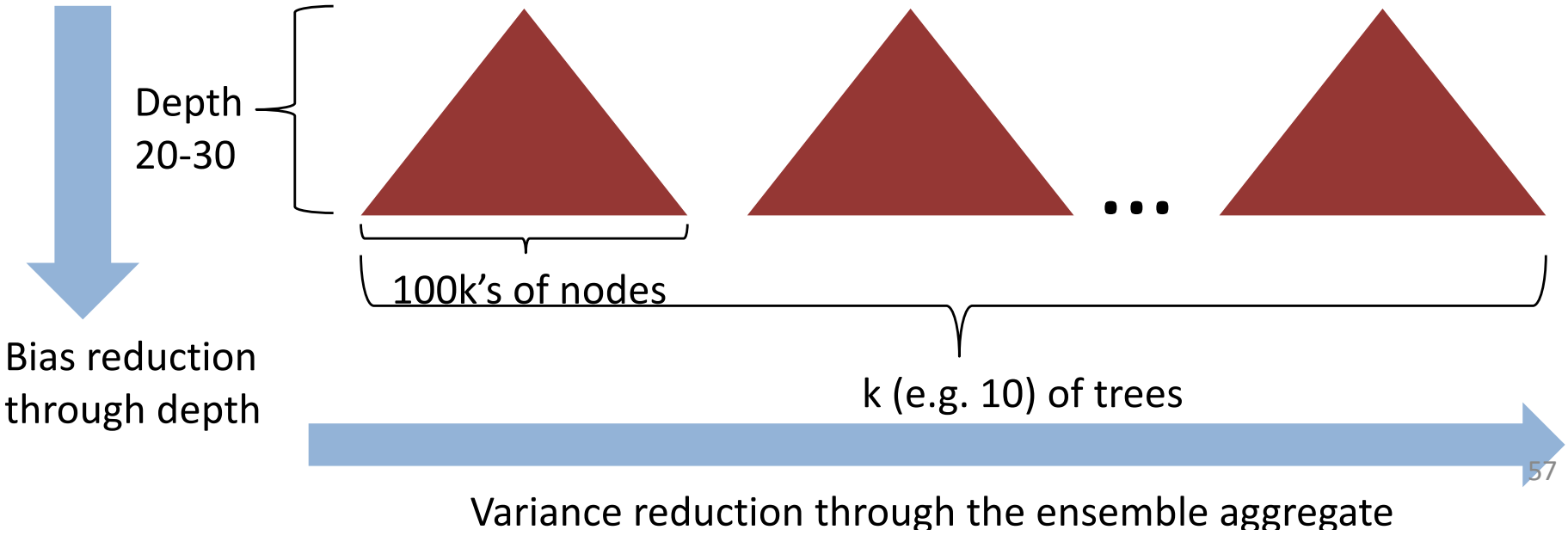
Types:

- **Bagging:** train learners in parallel on different samples of the data, then combine by voting (discrete output) or by averaging (continuous output).
- **Stacking:** combine model outputs using a second-stage learner like linear regression.
- **Boosting:** train learners on the filtered/weighted/... output of other learners.

Ensemble Method: Random Forest (Optional)

Combine multiple decision trees:

- Draw K bootstrap samples of size N from original dataset
- Grow a decision tree for each sample, by selecting a random set of m out of p features at each node, and choosing the best feature to split on. (typically m = sqrt(p) but can be smaller)
- Aggregate the predictions of the trees (most popular vote) to produce the final class.

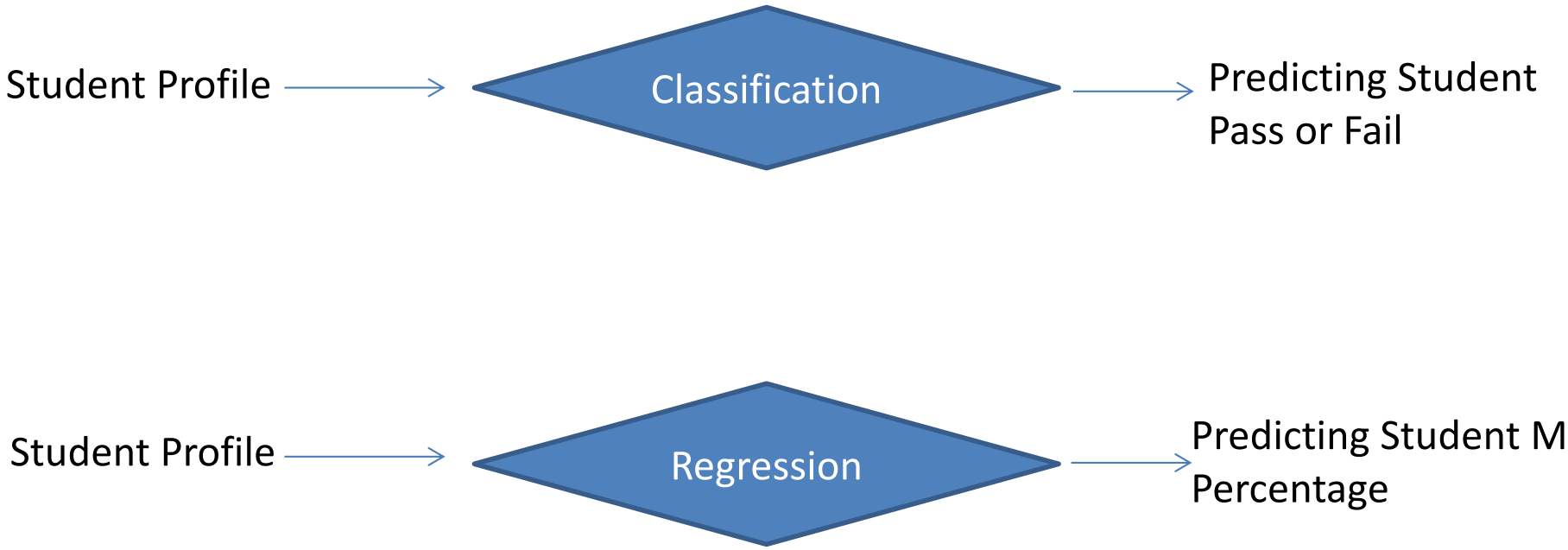


Ensemble Method: Pros and Cons

- **Pros:**
  - Effective for dense data (a few thousand features)
  - Parallelizes easily
- **Cons:**
  - Not easy to implement (how many classifiers are enough?)
  - Needs many passes over the data – at least the max depth of the trees.

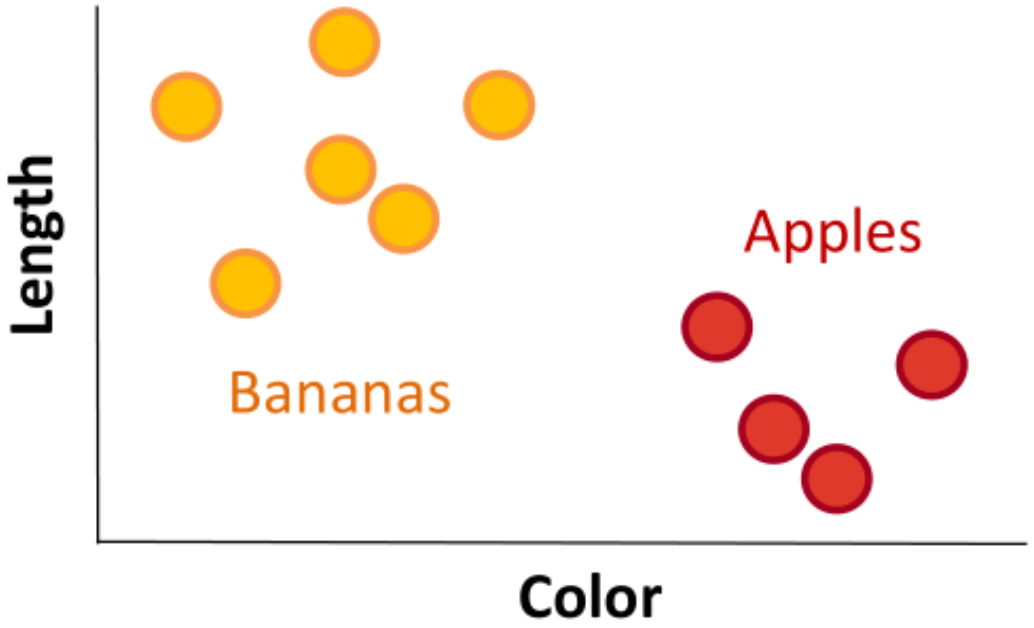
III. Regression

❖ **Classification vs. Regression:**



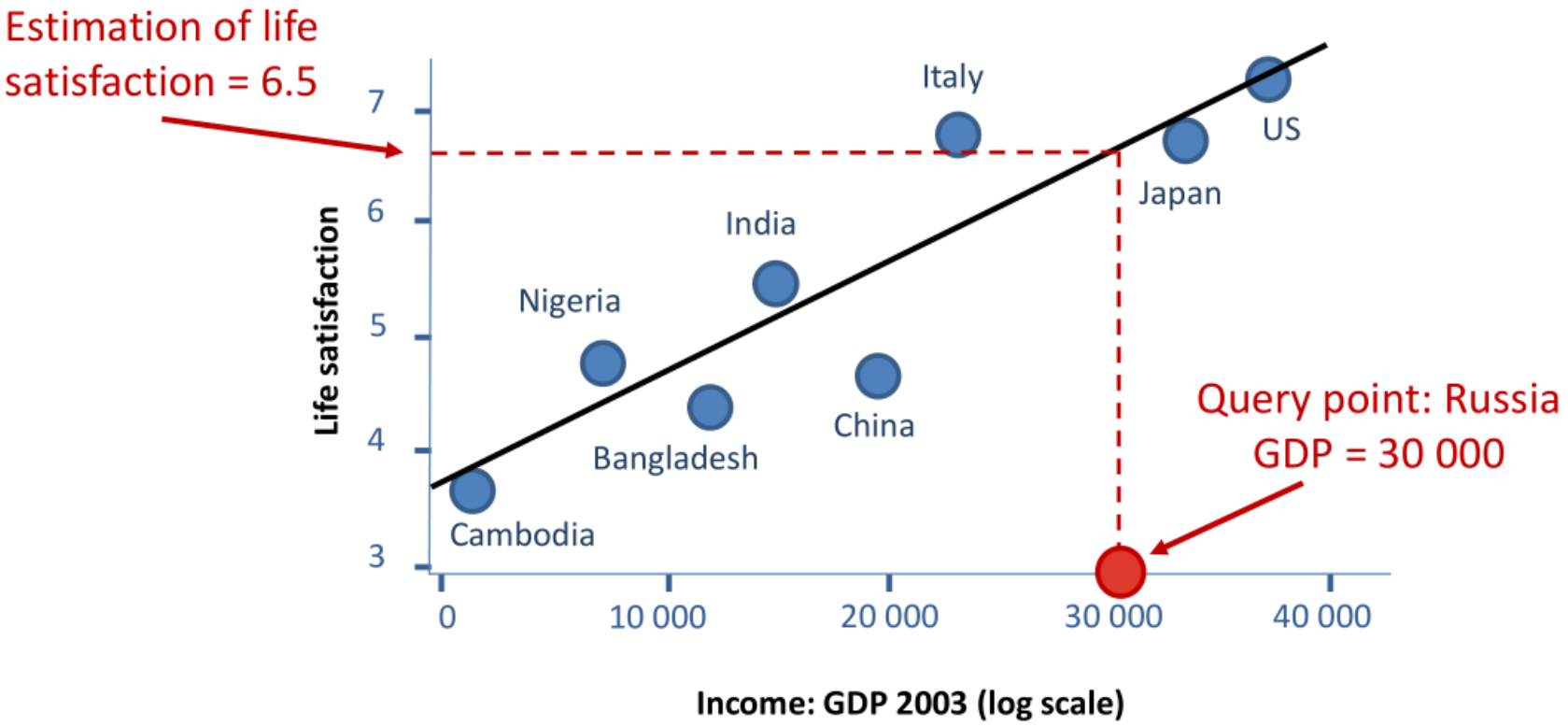
Classification: Recap

- Maps N-dimensional input $x \in R^N$ to discrete values y
- e.g. x = [Length, Color], y = {“Banana”, “Apple”}



Regression: Definition

- Maps N-dimensional input $x \in R^N$ to **continuous** values $y \in R$
- e.g. x = [Income (GDP)], y = Continuous value of life satisfaction



Regression: Applications

- **Predictive analytics:** forecasting future opportunities and risks
  - Forecast the number of shoppers who will pass in front of a particular billboard
  - Use that model to estimate the price for an advertisement
- **Operation Efficiency:** optimize business processes
  - Call center: correlation between wait times of callers vs. number of complaints
  - Use that model to reduce the number of complaints to some degree
- **Support Decisions:** understand customer behavior to make informed business decisions
- **Correcting Errors:** verify assumptions and adjust business strategies
- **New Insights:** Uncover patterns that were previously unnoticed
  - E.g. Market demand increases on certain days of the week or certain times of the year

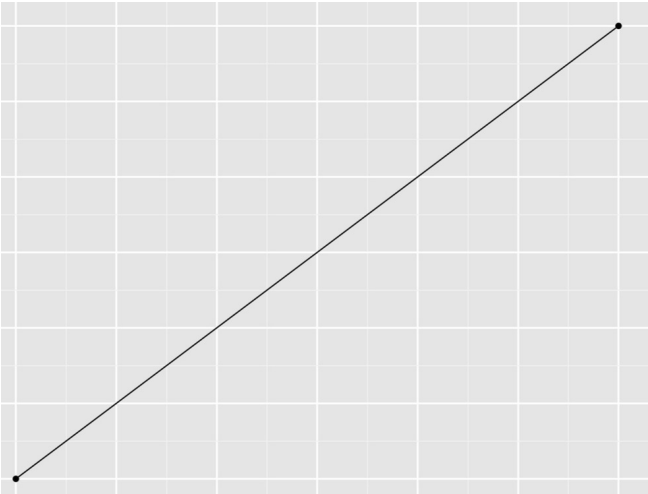
https://www.newgenapps.com/blog/business-applications-uses-regression-analysis-advantages

Regression Models

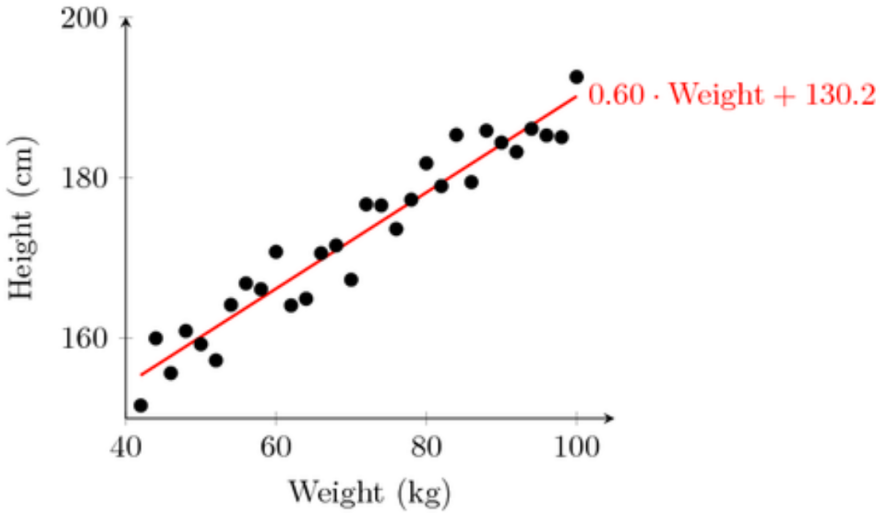
1. Linear Regression
2. Weighted Linear Regression
3. Non-Linear Regression
4. Logistic Regression (Optional)

1. Linear Regression

- Fit a line to a data set of observations
- Use this line to predict unobserved values
- **Goal:** calculate a regression line that's as close to every dot as possible.



2 points: simple

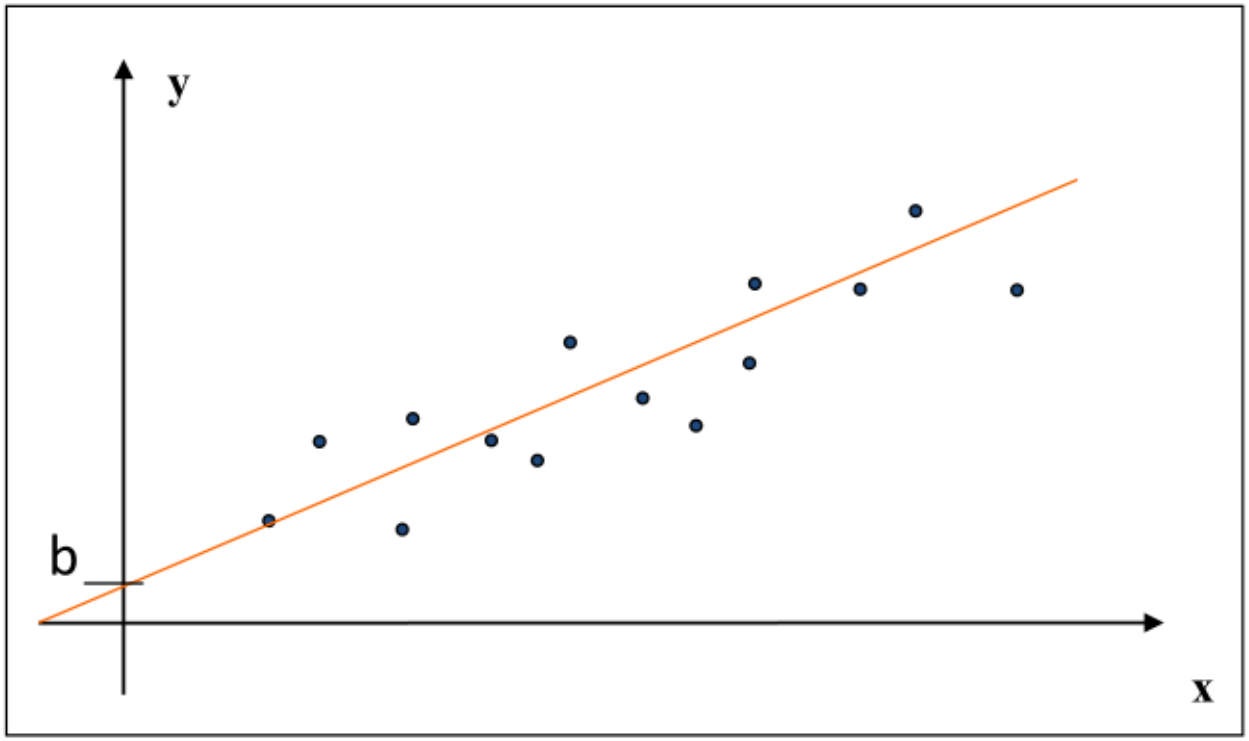


many data points: not simple

Linear Regression

- Linear regression searches a linear mapping between input x and output y, parametrized by the slope vector w and intercept b.

$$y = f(x; w, b) = w^T x + b$$



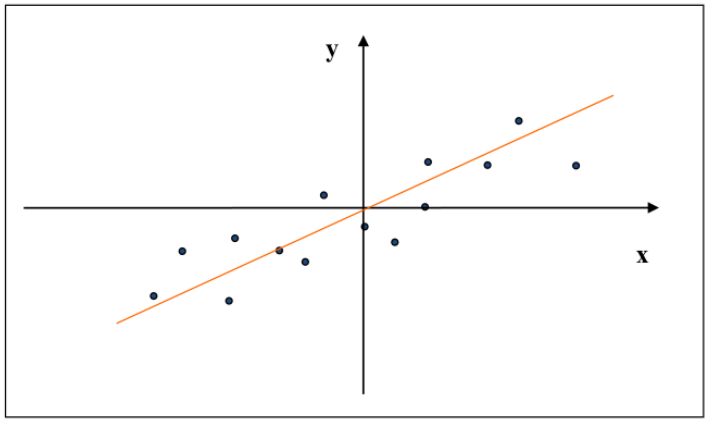
Linear Regression: Optimization

- **Input:** Pair of M training points $X = [x^1, \dots, x^M]$ and $y = [y^1, \dots, y^M]^T$
  - $x^i \in R^N, y^i \in R$
- **Optimization Criteria:** least-square error

$$w^* = \min_w \left(\sum_{i=1}^{M} \frac{1}{2} \left(w^T x^i + b - y^i \right)^2 \right)$$

  - For convenience, omit the intercept b by centering the data:
    - $y' = y - \bar{y}, x' = x - \bar{x}, b' = b + w^T \bar{x} - \bar{y}$

    ➔ $y' = w^T x'$



For brevity sake, we write $y = w^T x$ with the assumption that the data has been centered

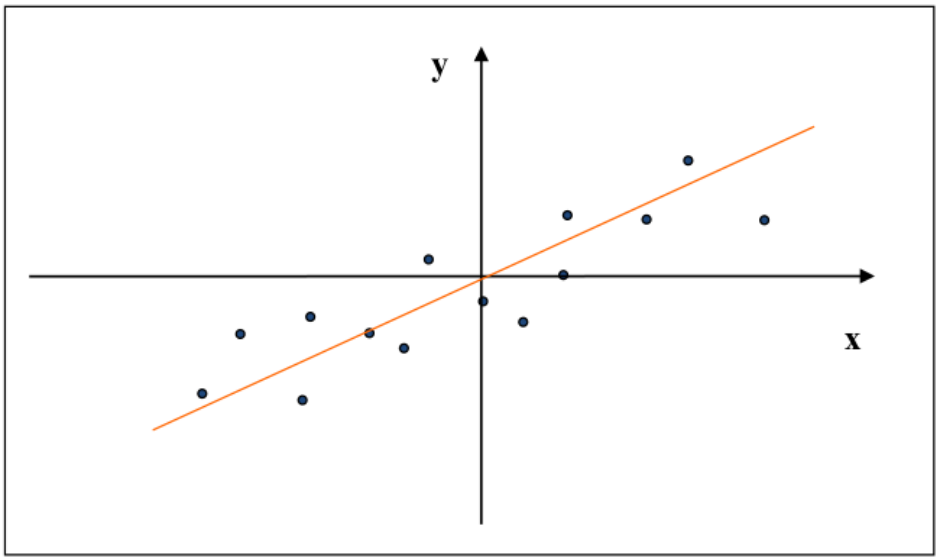
Linear Regression: Optimization

- **Optimization Criteria:** least-square error
  - Find the optimal parameter w to minimize least-square error:

$$w^* = \min_w \left(\sum_{i=1}^{M} \frac{1}{2} \left(w^T x^i - y^i \right)^2 \right)$$

  - This can be solved analytically through partial differentiation:

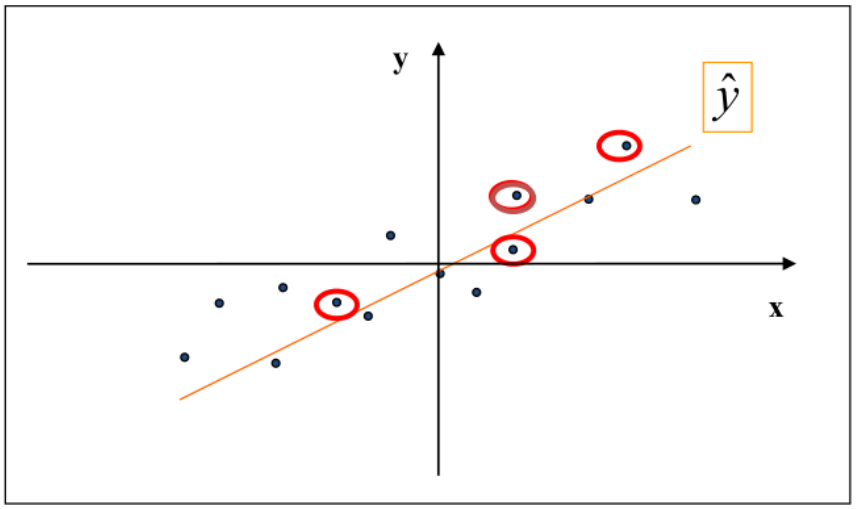
$$w^* = (XX^T)^{-1} Xy$$



2. Weighted Linear Regression

- Sometimes we want to prioritize some important points
  - Assume a set of weights β_i for all data points, $\{\beta_1, \dots, \beta_M\}$, $\beta_i \in [0,1]$
  - We set B a diagonal matrix with entries β_i:

$$B = \begin{pmatrix} \beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_M \end{pmatrix}$$



Points in red have large weights.

Weighted Linear Regression

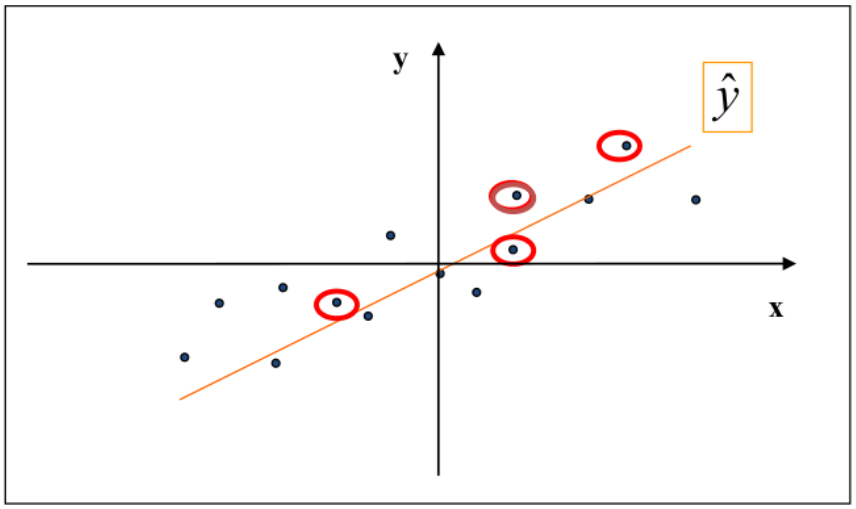
- **Optimization problem:** weighted least square

$$w^* = \min_w \left(\sum_{i=1}^{M} \frac{1}{2} \beta_i \left(w^T x^i - y^i \right)^2 \right)$$

- **Solution:**

$$w^* = (ZZ^T)^{-1} Zv$$

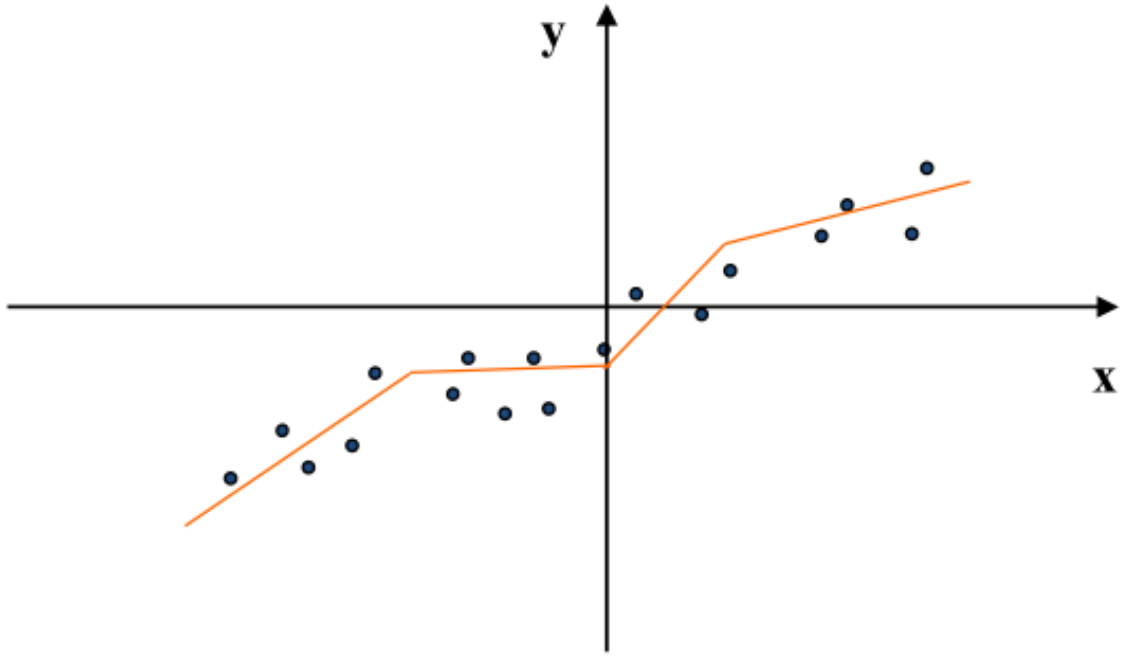
  - where $Z = XB^{1/2}$ and $v = B^{1/2} y$



Points in red have large weights.

Locally Weighted Linear Regression

- **Limitations** of (weighted) linear regression
  - Set (constant) weights by ourselves is difficult
  - Not useful for datasets with local dependencies (i.e. difference input ranges have different linear dependency)

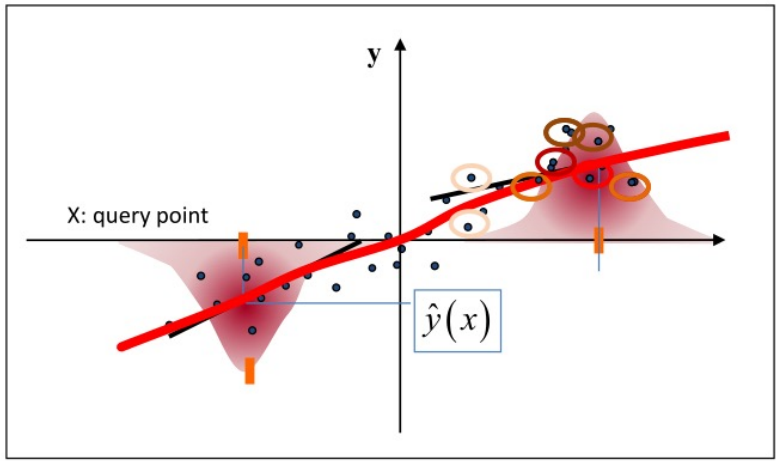


Locally Weighted Linear Regression

- **New formulation:** estimate is determined through local influence of each group of data points

$$\hat{y}(x) = \sum_{i=1}^{M} \beta_i(x) y^i \Big/ \sum_{j=1}^{M} \beta_j(x)$$

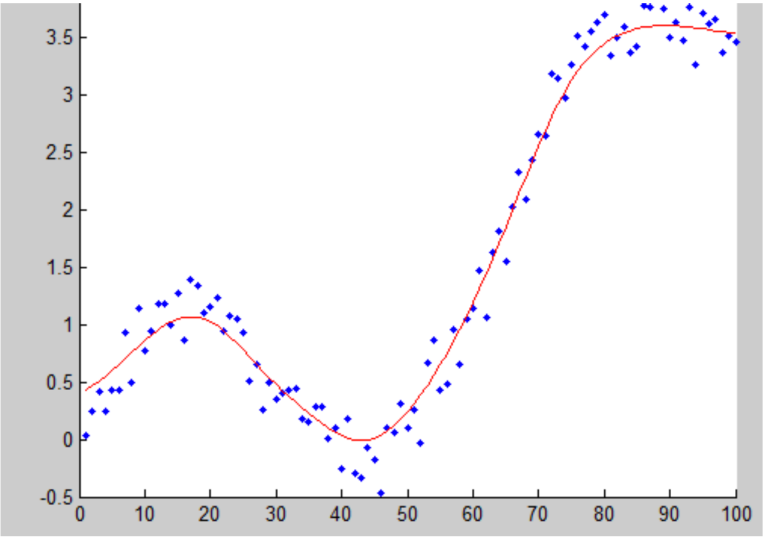
  - $\beta_i(x) \in R$: weights function of x
  - Now we can generate a smooth function $\hat{y}(x)$ by freely choosing a weight kernel $\beta_i(x)$



$$\beta_i(x) = e^{-||x^i - x||}$$

3. Non-linear Regression

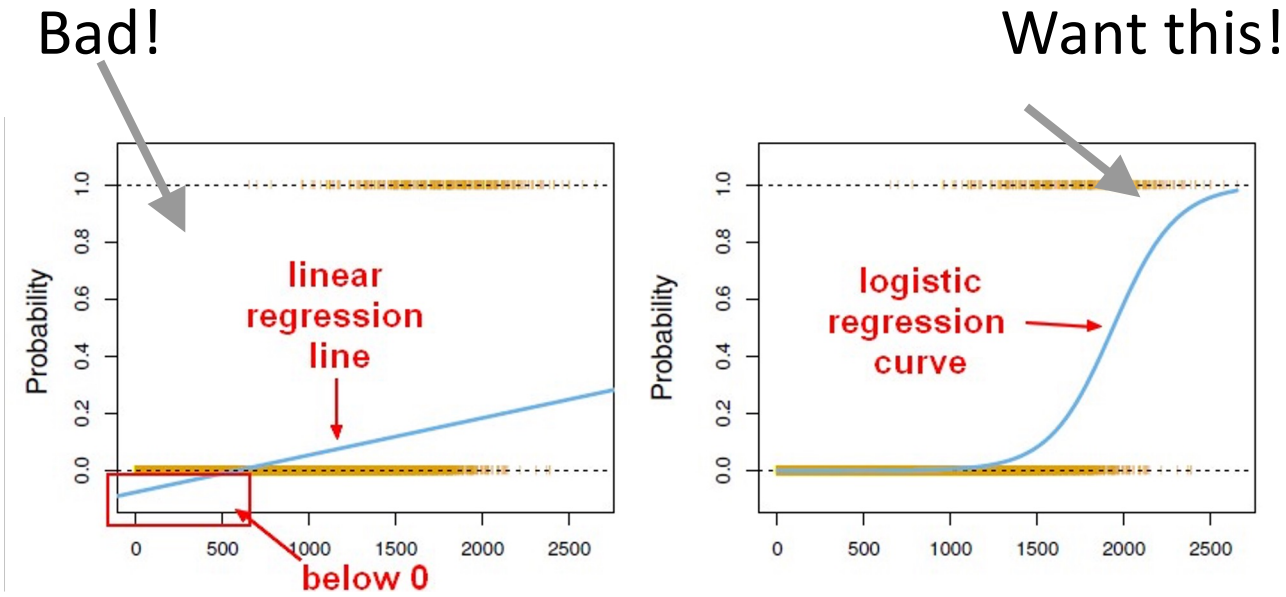
- Why limit ourselves to straight lines (first order)?
- Not all relationships are linear
  - E.g. "Second order" polynomial: $y = ax^2 + bx + c$
  - Third order: $y = ax^3 + bx^2 + cx + d$
  - Higher orders produce more complex curves



4. Logistic Regression

- Sometimes we want to see a degree of classification instead of yes/no answer
  - Easy for human in investigate false positives and false negatives
- **Example:**
  - The level of spam of an email
  - The default level of a loan
  - Disease diagnos
- **Idea**: linear regression with output is a probability

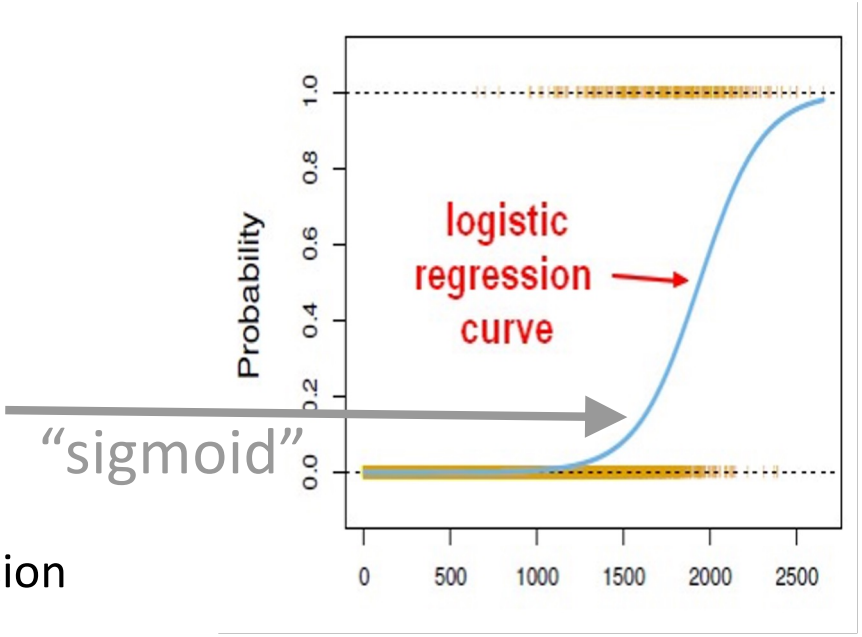
Logistic Regression



- Trick: don't deal with probabilities, which range from 0 to 1, but with log odds, which range from -inf to +inf
- Probability y ⇔ odds y/(1-y) ⇔ log odds log[y/(1-y)]
- Model log odds as a linear function of X

Logistic regression

- Model log odds as a linear function of X
- $\beta^T X = \log[y/(1-y)]$
- Solve for y: $y = 1 / (1 + \exp(-\beta^T X))$
- Finding best model β:
  - Don't use square loss as in linear regression
  - Use cross-entropy loss instead



Logistic Regression: Pros and Cons

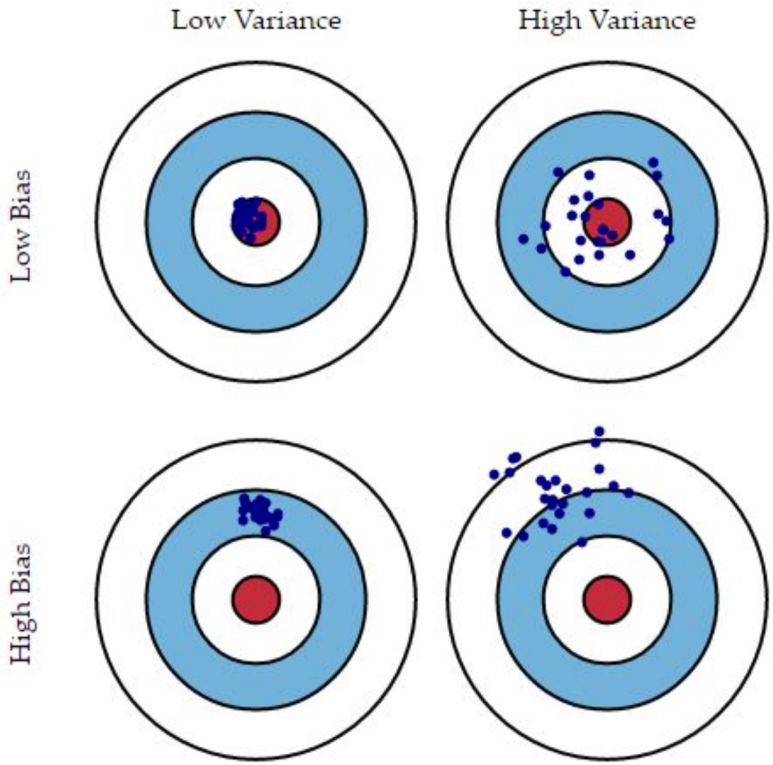
- **Pros:**
  - Low variance
  - Provide probabilities for outcomes
  - Perform similarly to SVM when linear separation
- **Cons:**
  - High bias

Multivariate Regression

- What if more than one variable influences the one you're interested in?
- Example: predicting a price for a car based on its many attributes (body style, brand, mileage, etc.)
- Model:
  - Price = a + b_1*mileage + b_2*age + b_3*doors
  - Still use least-squares optimization
  - Assumption: different factors are not dependent on each other

IV. Model Evaluation

- **Bias and Variance:**
  - Bias is how far the mean of your predicted values is from the “real” answer
  - Variance is how scattered your predicted values are from the “real” answer



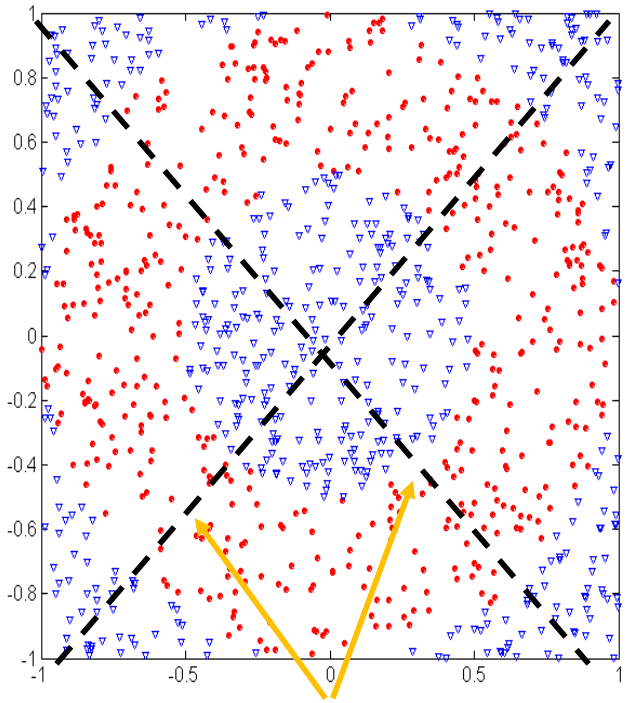
(assuming the center is the correct result)

Underfitting vs. Overfitting

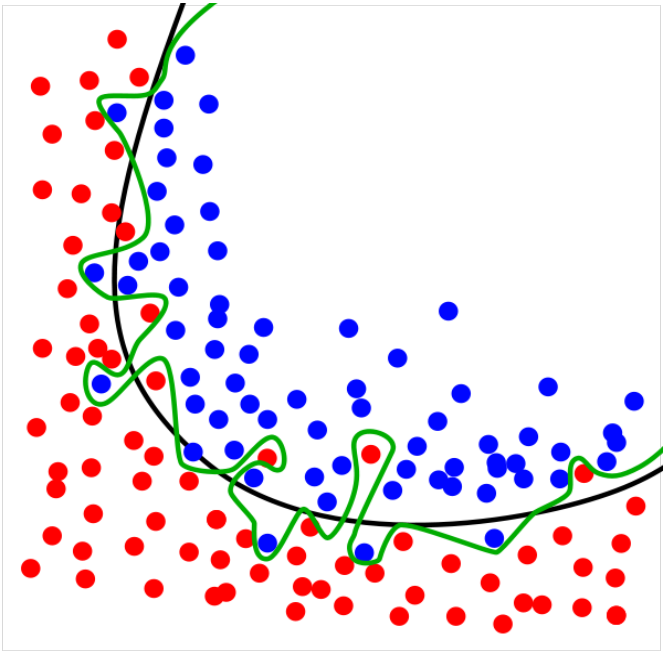
Underfitting

E.g. Regression

Overfitting



Models are too simple!



- Green line is overfitting (tailored too much to the training data)
- Black line is what we want

Low variance, high bias

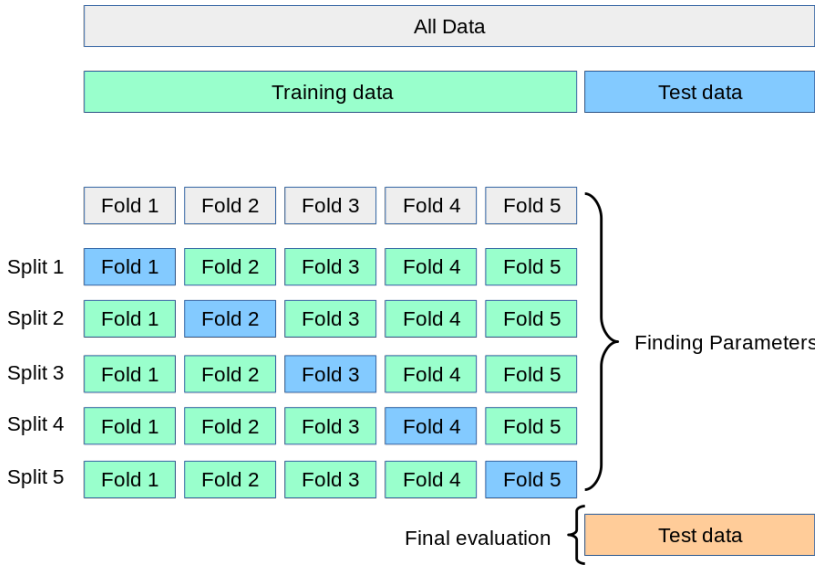
High variance, low bias

Underfitting vs. Overfitting

- **Deal with underfitting:**
  - Increase model complexity (e.g. use polynomial instead of linear)
  - Use more training data
- **Deal with overfitting:**
  - The more features the better?
    - NO!
    - More features mean less bias, but more variance
  - Carefully selected features can improve model accuracy
    - E.g., keep features that correlate with the label y
    - Forward/backward feature selection
    - Regularization (e.g., penalize norm of weight vector)
  - K-fold cross validation

K-Fold Cross validation

- Prevent you from over-fitting a single train/test split
- General process:
  - Split your training data into K randomly-assigned folds
  - Reserve one segment as your validation data
  - Train on each of the remaining K-1 folds to find model parameters
  - Take the best parameters based on validation accuracy
- You can do the same process for different train/test splits and take the average result



Classification: Performance Issues

- Predictive performance (accuracy, AUC/ROC, precision, recall, F1-score, etc.)
- Speed and scalability
  - Time to build the model
  - Time to use the model
  - In memory vs. on disk processing
  - Communication cost
- Robustness
  - Handling noise, outliers, missing values
- Interpretability
  - Understanding the model and its decisions (black box vs. white box)
- Compactness of the model
  - Mobile and embedded devices

Classification Performance Measures

- Classification accuracy can be unreliable when assessing unbalanced data.
  - For example, if there are 95 samples from class A and only 5 from class B in the data set, a particular classifier might classify all the observations as class A.
- **Confusion matrix** is a table with two rows and two columns that reports the number of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). In multi-class classification, a confusion matrix can be computed for each class.
- Sensitivity (recall): $$\frac{TP}{\text{the number of real positives}} = \frac{TP}{TP+FN}$$
- Specificity: $$\frac{TN}{\text{the number of real negatives}} = \frac{TN}{TN+FP}$$
- Precision: $$\frac{TP}{TP+FP}$$
- F1-score: $$\frac{2}{\frac{1}{\text{precision}}+\frac{1}{\text{recall}}} = \frac{2\text{precision} \times \text{recall}}{\text{precision}+\text{recall}}$$
- Accuracy:

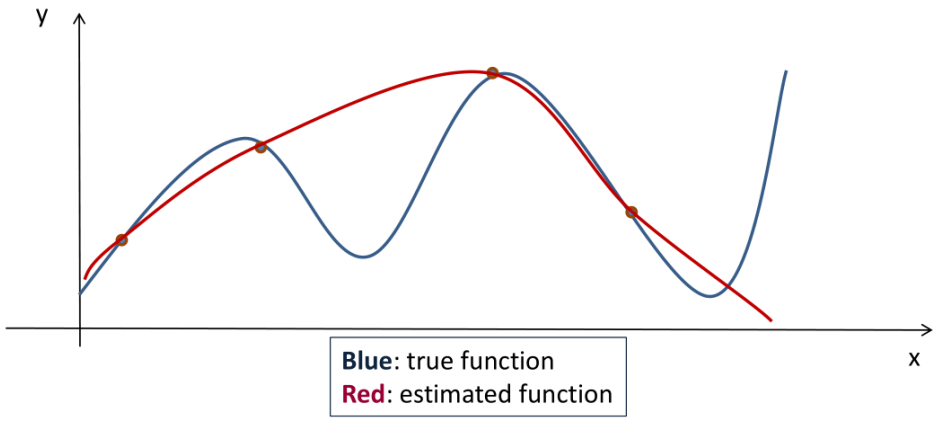
		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	5	2	0
	Dog	3	3	2
	Rabbit	0	1	11

confusion matrix for "cat" class

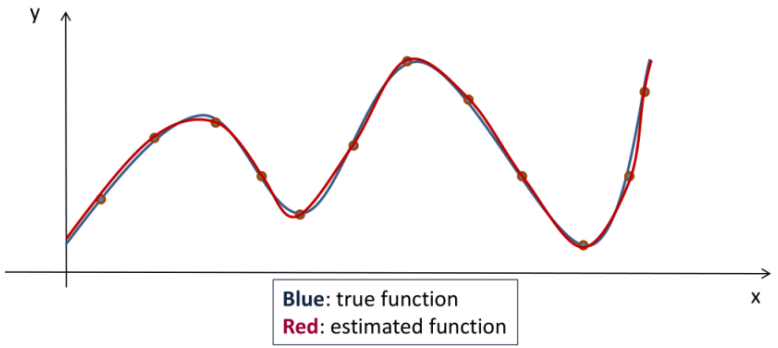
		Actual class	
		Cat	Non-cat
Predicted class	Cat	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

Regression: Performance Issues

- Good prediction depends on the choice of datapoints
- The more datapoints, the better the fit
- Computational costs increase dramatically with number of datapoitns



Less data points



More data points

References

[1] https://blog.exploratory.io/find-correlation-or-similarity-among-categories-or-variables-4813130f53c0

[2] https://blog.exploratory.io/an-introduction-to-regression-analysis-in-exploratory-9422237c0ff8

[3] https://www.wiley.com/en-us/Statistical+Data+Analysis+Explained%3A+Applied+Environmental+Statistics+with+R-p-9780470985816

[4] https://www.bigskyassociates.com/blog/bid/356764/5-Most-Important-Methods-For-Statistical-Data-Analysis

[5] https://www.infogix.com/blog/differentiating-between-cluster-analysis-and-factor-analysis/

[6] https://en.wikipedia.org/wiki/Correlation_clustering

[7] http://dataaspirant.com/2014/09/27/classification-and-prediction/

[8] http://syllabus.cs.manchester.ac.uk/ugt/2017/COMP24111/lectures.php

[9] http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

[10] https://github.com/ctufts/Cheat_Sheets/wiki/Classification-Model-Pros-and-Cons