COMP336 — Big Data

Week 11 Lecture 1: Large-Scale Machine Learning (I)

Diego Mollá

Department of Computer Science
Macquarie University

COMP348 2018H1

Programme

Reading

- Leskovec, Rajaraman, Ullman (2014): Mining of Massive Datasets, Chapter 12. `http://www.mmds.org/`

Programme

What is Statistical Learning?

- Statistical learning is about inferring rules based on sample data.
- Possible uses of statistical learning are:
  - **Analysis:** Process a data set with the goal to achieve a better understanding of its characteristics.
    - We have seen one example in a previous lecture: learning association rules.
  - **Prediction:** Learn rules that allow us to predict outcomes.

Example of ML for Analysis

Analysis

We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.

Example of ML for Prediction

Prediction

We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

Types of Machine Learning

Supervised machine learning

- In supervised machine learning, we have a training set where we know the prediction.
- This training set has been annotated, usually manually.
- The training set is used to learn a model.
- The model is then used to make predictions on unseen data.
  - Unseen data is data that is not part of the training data.

Unsupervised machine learning

- In unsupervised machine learning, there is no training set.
- We process a data set with the aim to extract useful information from it.
- An example is mining association rules.
- Another example is clustering data.

Supervised Learning

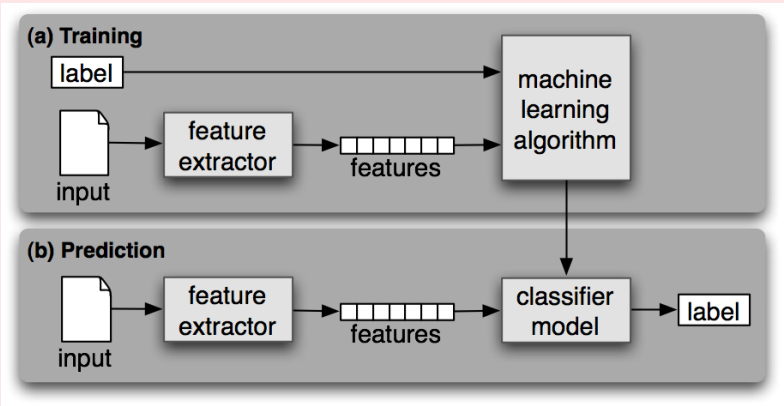
Given

Training data annotated with class information.

Goal

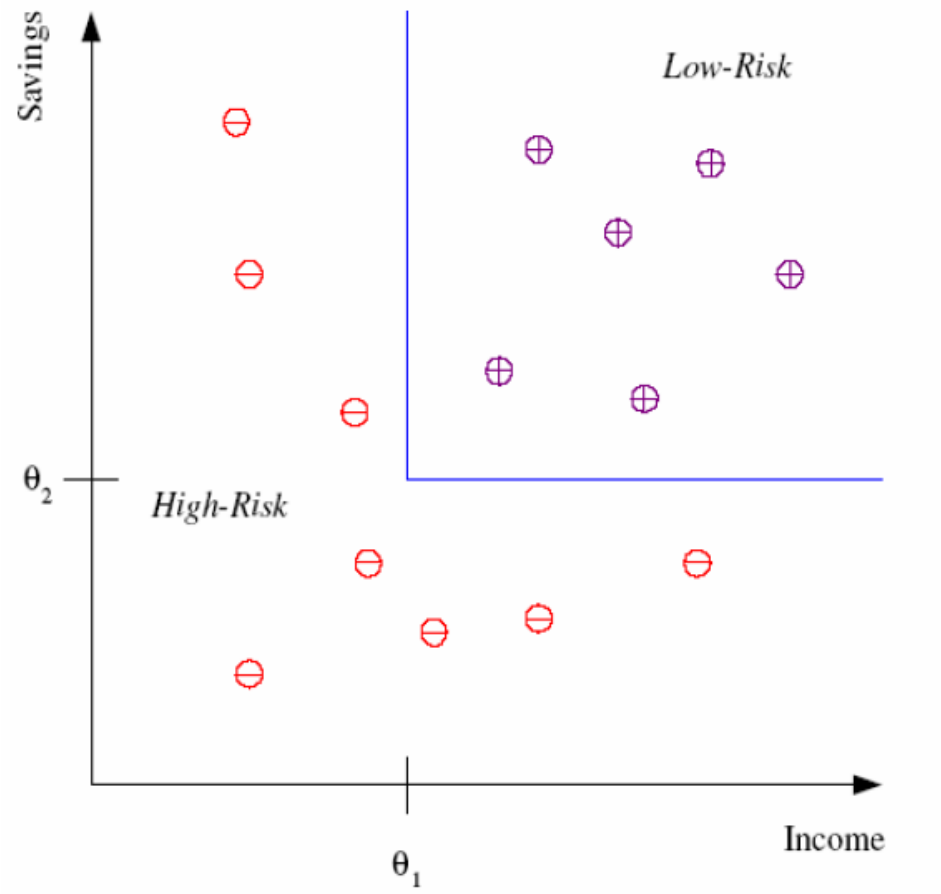
Build a model which will allow classification of new data.

Method



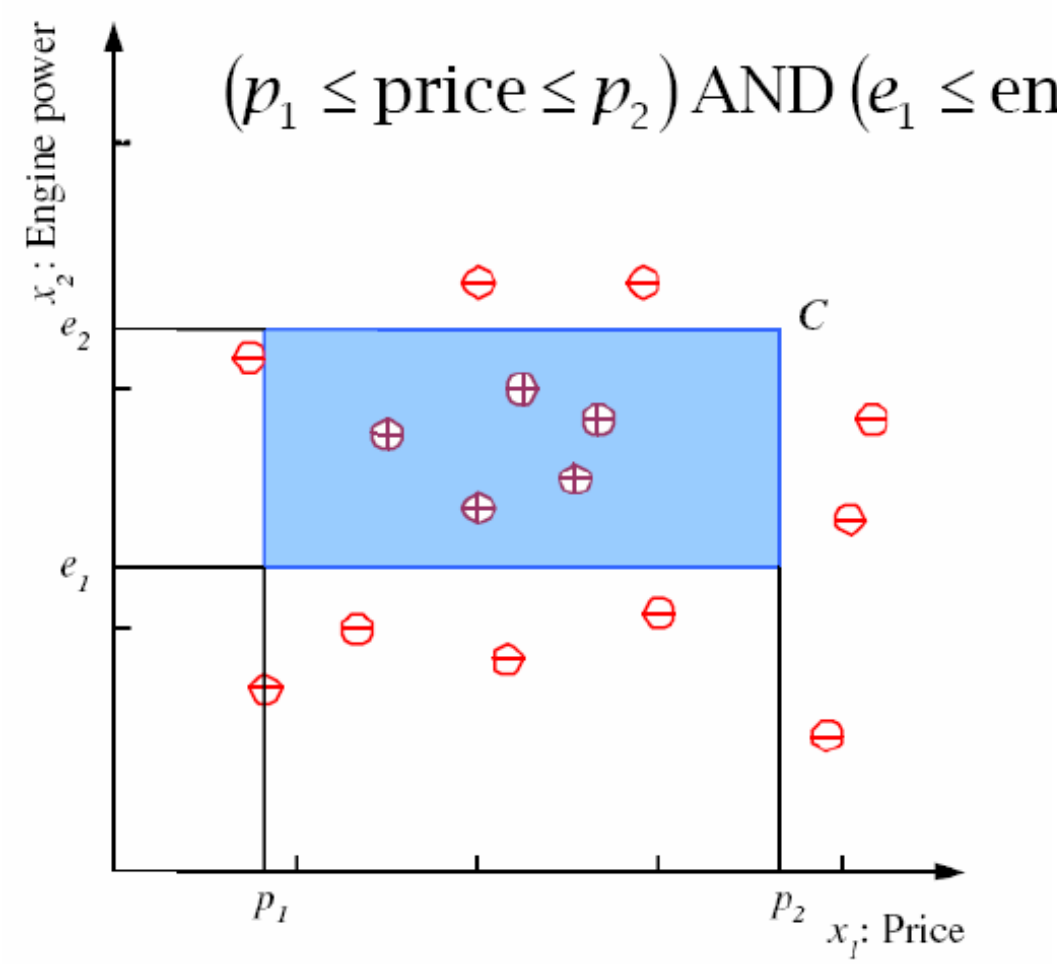
1. Feature extraction: Convert samples into vectors.
2. Training: Automatically learn a model.
3. Classification: Apply the model on new data.

Supervised Learning Example: Bank Customers



(from Alpaydin (2004))

Supervised Learning Example: Family Cars



(from Alpaydin (2004))

Programme

Programme

Basic Idea

- For a given record to be classified, identify nearby records.
- "Near" means records with similar predictor values $X_1, X_2, \cdots X_p$.
- Classify the record based on the nearby records (the "neighbours").

  Regression If the target value is a number: average the values of all neighbours.

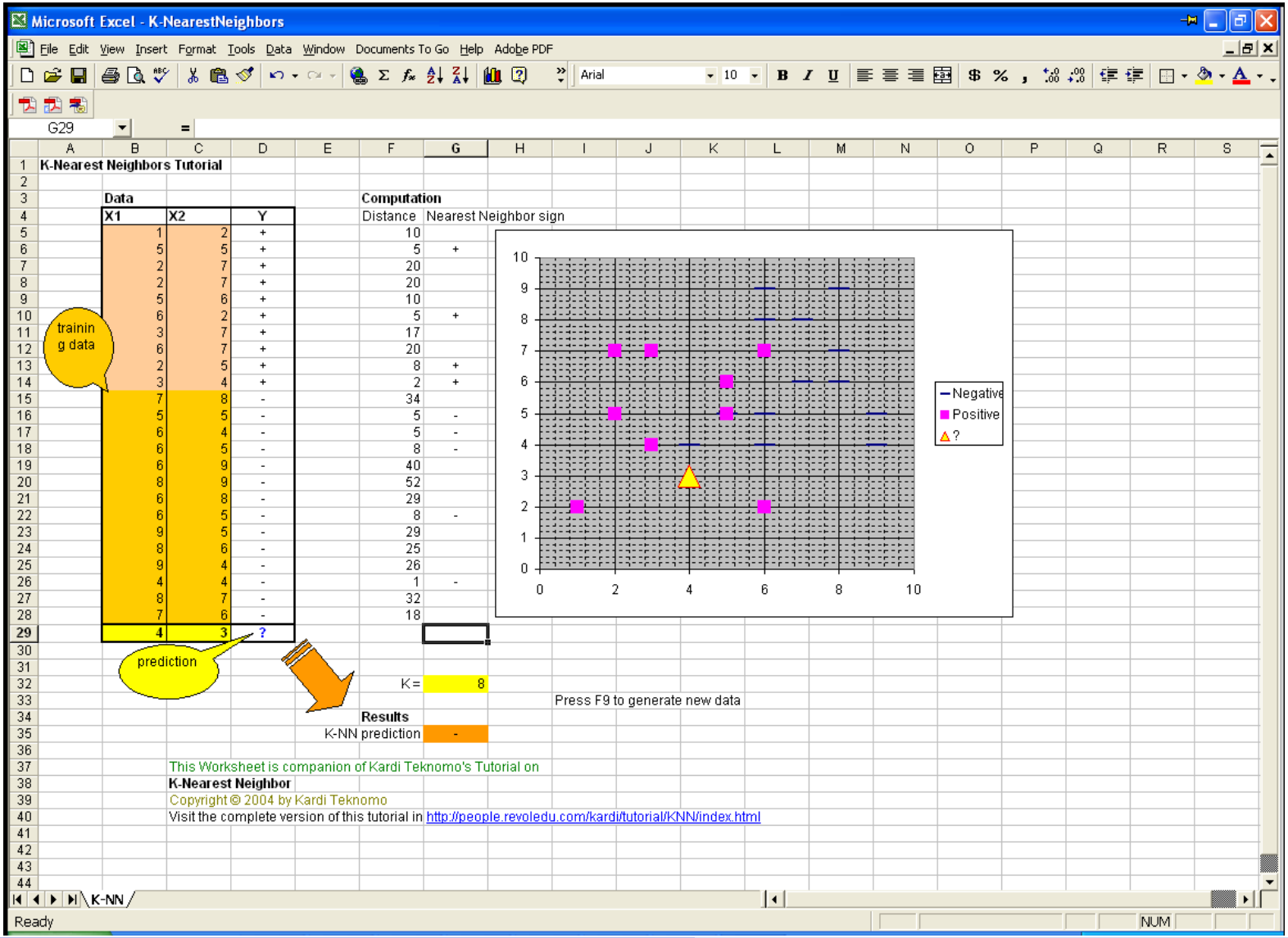
  Classification If the target value is a label: Choose the most frequent label in the neighbours.

How to Measure "nearby"?

The most popular distance measure is the Euclidean distance. Given two points in p-dimensional space: $x = (x_1, x_2, \ldots, x_p)$ and $y = (y_1, y_2, \ldots, y_p)$:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_p - y_p)^2}$$

Example with an Artificially Constructed Set



Characteristics of K-NN

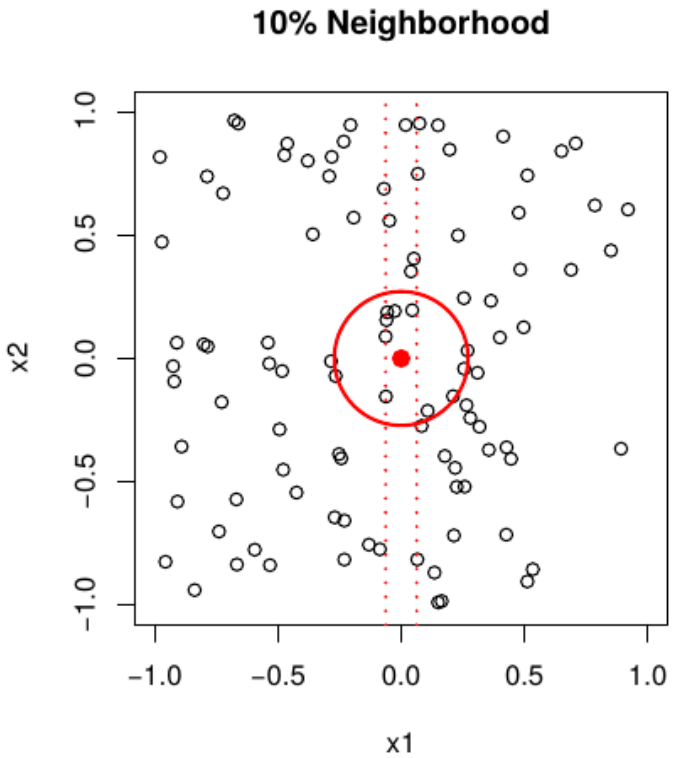
- Data-driven, not model-driven.
- Makes no assumptions about the data.
  - This is a non-parametric method.
- Lazy classifier: delay all computation until the classification stage.
  - Very fast training.
  - Classification speed depends on size of training data.

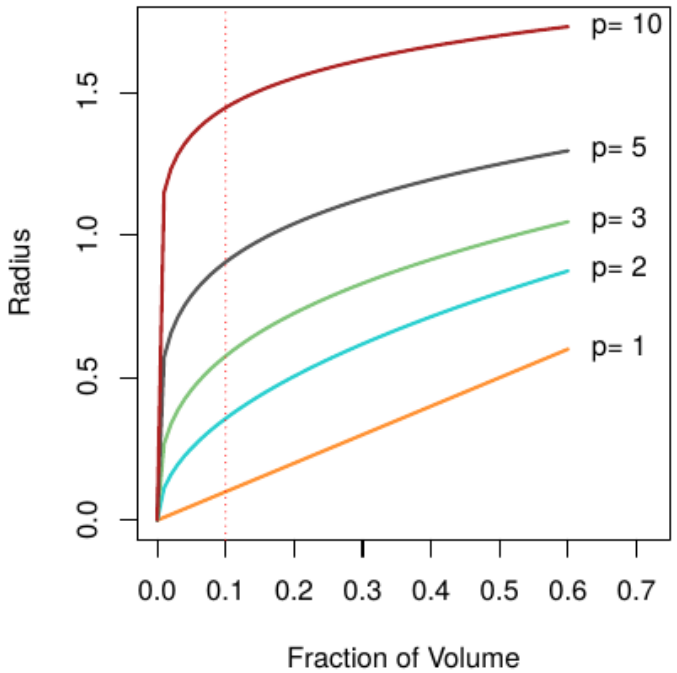
Programme

K-Nearest Neighbours on Big Data

- KNN needs to store the entire training data.
  - We need to use efficient methods to store the training data.
- At run time, KNN needs to search on the training data.
  - We need to find efficient methods to find the nearest neighbours.
- KNN can also suffer from the Curse of Dimensionality.
  - Using many predictors produces a decrease in performance.

The Curse of Dimensionality





Dealing with the Curse of Dimensionality

VA Files

- With high-dimensional data we will end up searching a large portion of the data.
- Using VA files we can quickly scan the entire file of training data in a two-stage manner.
  1. For each record, use a quick approximation of the values of the record to build a list of candidates. For example, use a fraction of the high-order bits of the representation of the numbers.
  2. Use the list of candidates to find the nearest neighbours.

Dimensionality Reduction

- Reduce the number of dimensions by applying a standard method, e.g. Principal Components Analysis (PCA) — see textbook Chapter 11.

Dealing with the Curse of Dimensionality

VA Files

- With high-dimensional data we will end up searching a large portion of the data.
- Using VA files we can quickly scan the entire file of training data in a two-stage manner.
  1. For each record, use a quick approximation of the values of the record to build a list of candidates. For example, use a fraction of the high-order bits of the representation of the numbers.
  2. Use the list of candidates to find the nearest neighbours.

Dimensionality Reduction

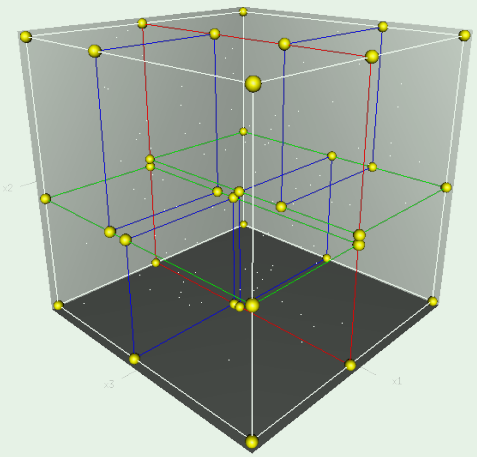
- Reduce the number of dimensions by applying a standard method, e.g. Principal Components Analysis (PCA) — see textbook Chapter 11.

Efficient Search on Large Data

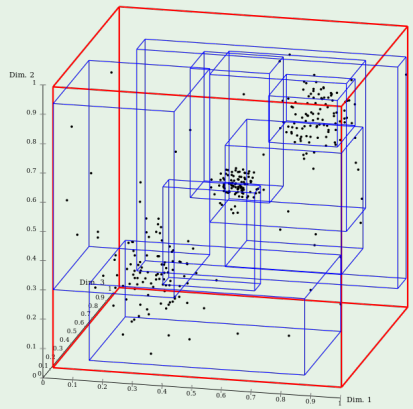
Multidimensional index structures

There are several data structures that have been developed for finding near neighbours when the number of dimensions grows and the training set is large.

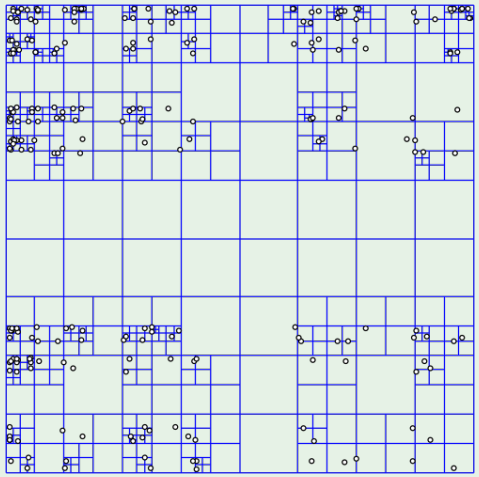
kd-Trees



R-Trees



Quad Trees



Locality Sensitive Hashing

- Recall that LSH was used to find near neighbours of text documents.
- LSH can be used as part of KNN for text documents.
- The trick is to focus only on those documents that are candidate neighbours to our sample document.
- We only need to add our sample document to the list of documents, and check the documents that hash to the same bucket in each band.
- This approach may yield false negatives but might suffice.

Using LSH for KNN

	S_1	S_2	S_3	S_4	S_q
r_1	1	0	1	1	1
r_2	2	1	0	2	2
r_3	1	2	2	0	1
r_4	0	1	1	2	1
r_5	3	0	3	1	0
r_6	1	0	1	0	0

The near neighbours candidates for S_q are S_1, S_2 and S_4

LSH for Other Distances

- LHS as we have seen it so far is designed for Jaccard similarity.
- But there are variants of LHS for other kinds of similarities.
- See textbook, sections 3.5 to 3.7 for details.
- Here we will show LHS for Euclidean distances.

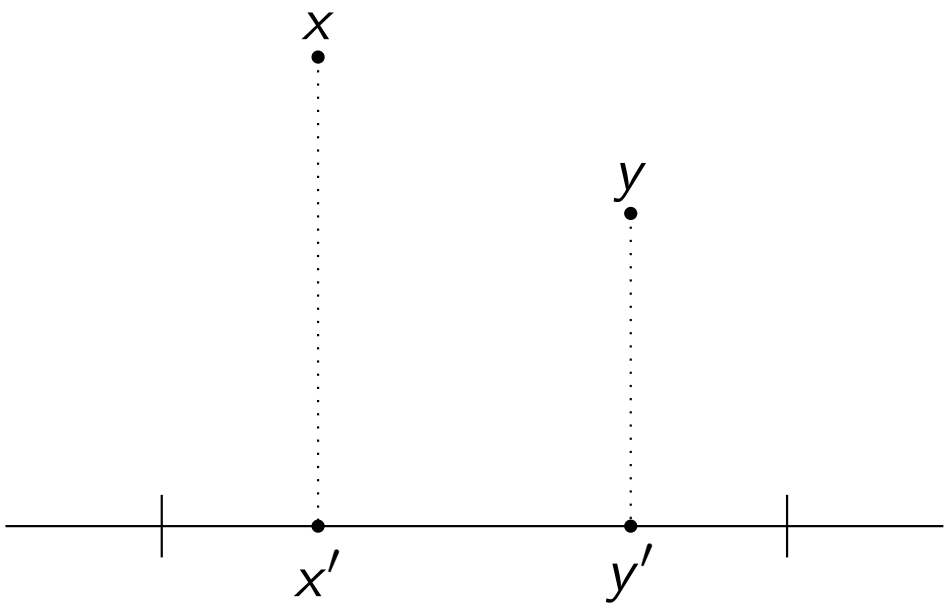
The Euclidean distance

$$d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

LSH for Euclidean Distance (intuition)

- We need to define a family of hash functions so that, if two points hash to the same bucket in one of the hashes, we determine that their Euclidean distance is small.
- Each hash function is associated with a randomly chosen line.
- The line is divided into segments of equal size a. Each segment represents a bucket.
- A point is hashed by projecting the point to the line. The segment where the projection falls is the bucket.
  - The projection of a point p to a line l is the point p' in l with smallest Euclidean distance to p.
- Two points with a small Euclidean distance will be more likely to hash to the same bucket in at least one of the hash functions.

LSH for Euclidean Distance (graphically)



Programme

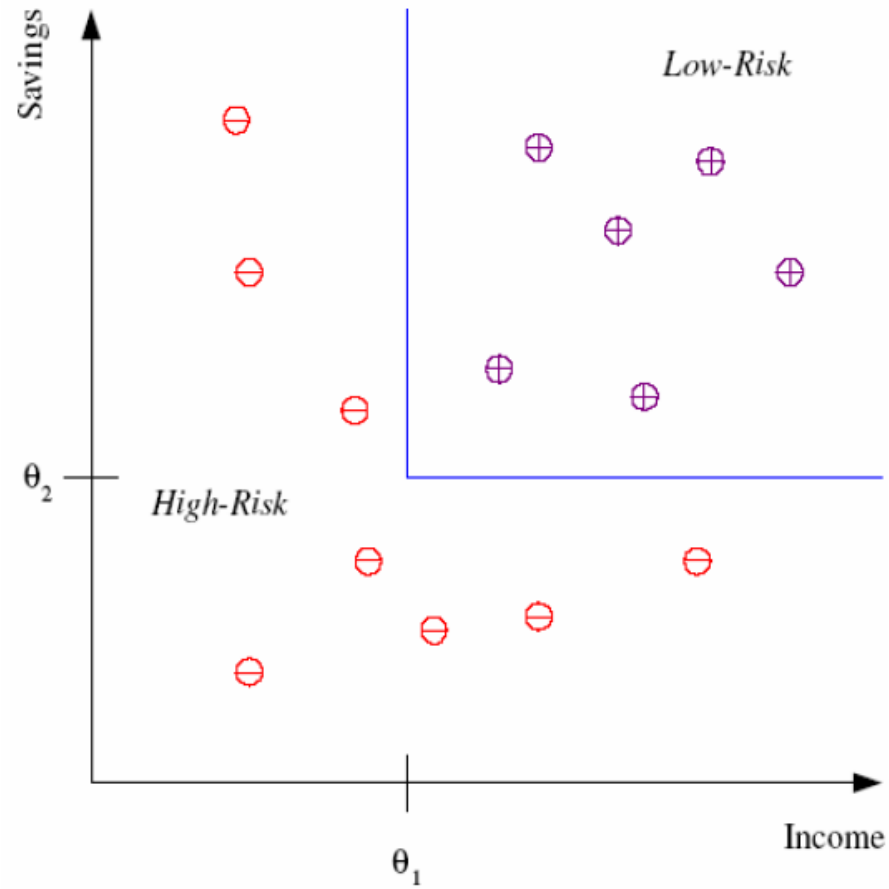
The Regression Function

- Supervised machine learning is about inferring a function given some sample data.

$$y = f(x) + \epsilon$$

  - ϵ is the irreducible error which we treat as random noise that we cannot predict.
- The regression function is the ideal $f(x)$.
- Nearest neighbours is a non-parametric approach to machine learning.
  - We don't assume anything about the shape of the regression function.
- Most supervised machine learning approaches are parametric.
- In a parametric approach, we assume the regression function has a particular shape and we try to learn its parameters.

Example: Bank Customers



Assumed regression function

IF income $> \theta_1$

AND savings $> \theta_2$

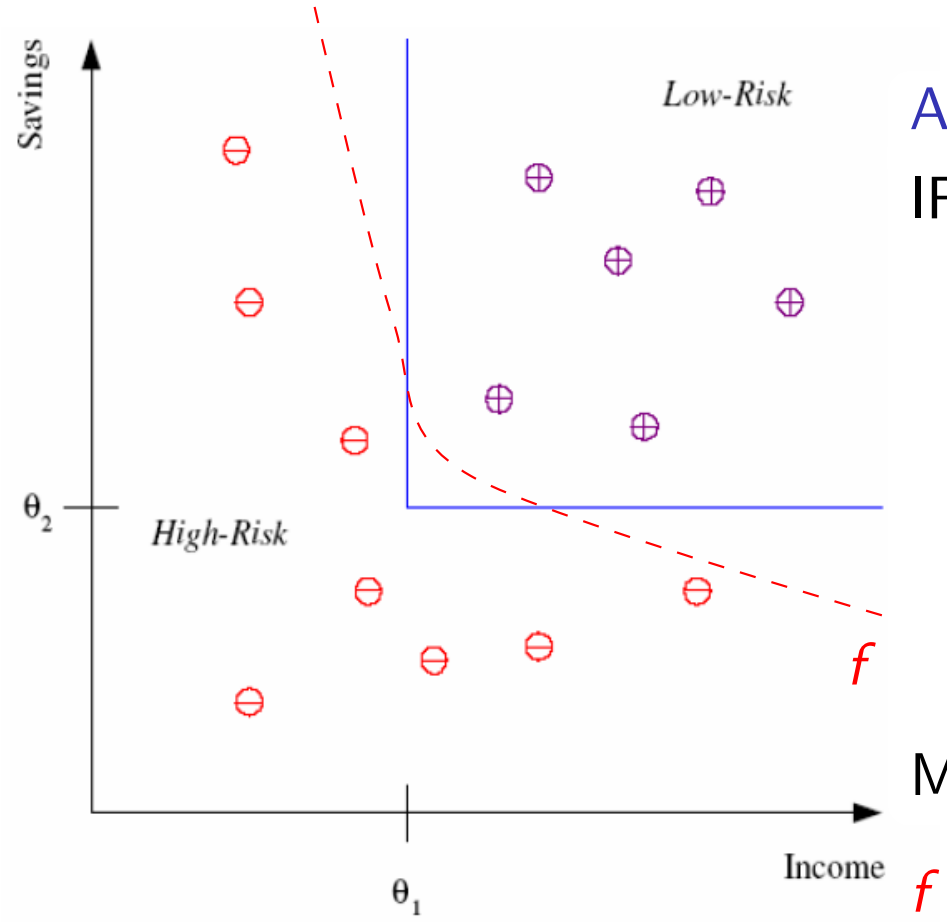
THEN $+$

ELSE $-$

Model parameters: θ_1, θ_2

f = Real regression function

Example: Bank Customers



Assumed regression function

IF income $> \theta_1$

AND savings $> \theta_2$

THEN $+$

ELSE $-$

Model parameters: θ_1, θ_2

f = Real regression function

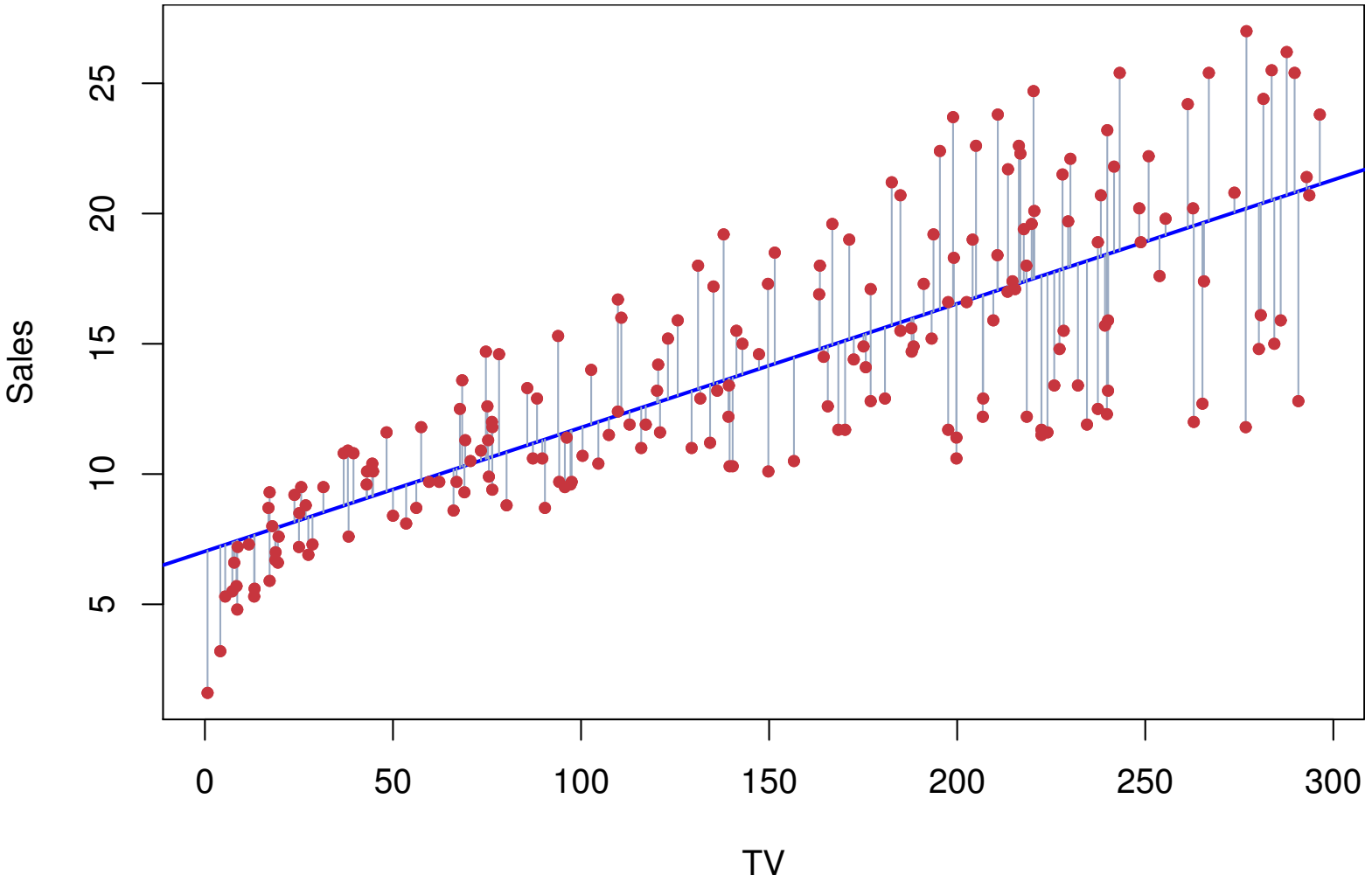
Example: Linear Regression

- Linear regression assumes that the regression function is a linear function.

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p + \epsilon$$

- Often the assumption is wrong but we can still get reasonably good results.
- The training data consists of $X = x^{(1)}, x^{(2)}, \cdots, x^{(n)}$ and $Y = y^{(1)}, y^{(2)}, \ldots, y^{(n)}$ where
  - $x^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)})$ is input sample i, and
  - $y^{(i)}$ is the corresponding annotated prediction.
- The learning approach will try to find $\Theta = \theta_0, \theta_1, \ldots$ that minimises the prediction error on the training data.
  - Usually called the loss $L(X, Y, \Theta)$.

Example: Linear Regression



Supervised Machine Learning as an Optimisation Problem

- The machine learning approach will attempt to learn the parameters of the learning function that minimise the loss (prediction error) in the training data.

$$\Theta = \text{argmin}_{\Theta} L(X, Y, \theta)$$

- In linear regression:
  - $f(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_p x_p^{(i)}$
  - $L(X, Y, \Theta) = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2$
    This loss is the mean squared error.

Optimisation Problems in Other Approaches

Logistic Regression

Logistic regression is commonly used for classification

- $f(x^{(i)}) = \frac{1}{1+e^{-\theta_0-\theta_1 x_1^{(i)}-\cdots-\theta_p x_p^{(i)}}}$
- $L(X, Y, \Theta) =$
  $\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \times \log f(x^{(i)}) + (1 - y^{(i)}) \times \log(1 - f(x^{(i)}))$

Support Vector Machines

Initially, SVM was formulated differently (see the textbook for details) but it can also be seen as:

- $f(x^{(i)}) = \text{sign} p(x^{(i)})$
  $p(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_p x_p^{(i)}$
- $L(X, Y, \Theta) = \frac{1}{n}\max\{0, 1 - y^{(i)} \times p(x^{(i)})\}$
  This is called the hinge loss.

Solving the Optimisation Problem

- A common approach to find the minimum of the loss function is to find the value where the gradient of the loss function is zero.
- This results in a system of equations that can be solved.

System of equations in linear regression

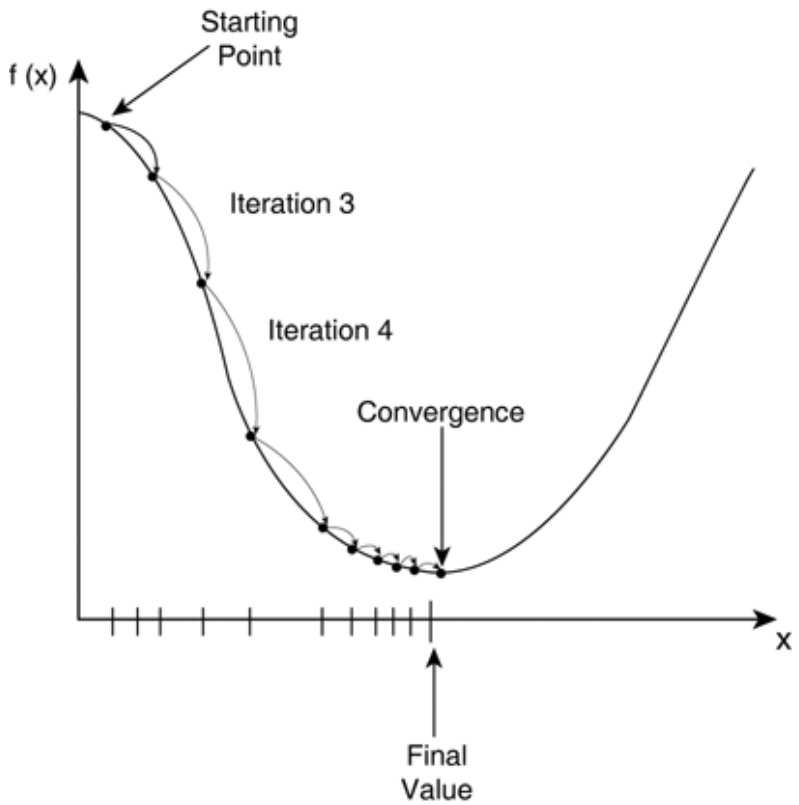
$$\frac{\partial}{\partial \theta_0} 1/n \sum_{i=1}^{n} (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \cdots - \theta_p x_p^{(i)})^2 = 0$$
$$\frac{\partial}{\partial \theta_1} 1/n \sum_{i=1}^{n} (y^{(i)} - \theta_0 - \theta_1 x_1^{(i)} - \cdots - \theta_p x_p^{(i)})^2 = 0$$
$$\cdots$$
$$\frac{\partial}{\partial \theta_p} 1/n \sum_{i=1}^{n} (y^{(i)} - \theta_0 - \theta_1 x_p^{(i)} - \cdots - \theta_p x_p^{(i)})^2 = 0$$

Gradient Descent

- Solving the system of equations $\frac{\partial L}{\partial \theta_0} L(X, Y, \Theta) = 0, \frac{\partial}{\partial \theta_1} L(X, Y, \Theta) = 0, \ldots$ can be too time-consuming.
- e.g. in linear regression, the complexity of solving the equations is $O(n^3)$.
- Some loss functions are very complex (e.g. in deep learning approaches) and it is not practical to attempt to solve the equations at all.
- Gradient descent is an iterative approach that finds the minimum of the loss function.

Gradient Descent Algorithm

1. $\theta_0 = 0, \ldots, \theta_p = 0$
2. Repeat until convergence:
   $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X, Y, \Theta)$

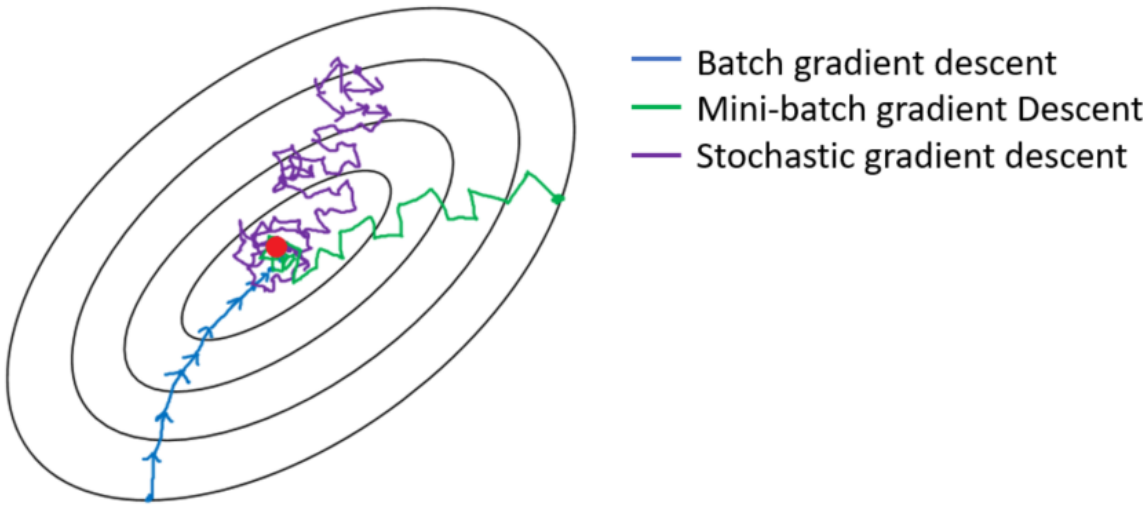


Mini-Batch Gradient Descent

- There are automated methods to compute the derivatives of many complex loss functions.
  - This made it possible to develop the current deep learning approaches.
- Note, however, that every step of the gradient descent algorithm requires to process the entire training data.
- This is what is called mini-batch gradient descent.
  - If the batch size is 1, it is usually called stochastic gradient descent.
- In mini-batch gradient descent, only part of the training data is processed.
- The entire data set is partitioned into small batches, and at each step of the gradient descent iterations, only one batch is processed.

Mini-Batch Gradient Descent Algorithm

1. $\theta_0 = 0, \ldots, \theta_p = 0$
2. Repeat until (near) convergence:
   1. Shuffle (X, Y) and split it into mini-batches.
   2. For every mini-batch (X', Y'):
      1. $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(X', Y', \Theta)$



https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Batch vs. Mini-Batch Gradient Descent

Batch Gradient Descent

- At each iteration step we take the most direct path towards reaching a minimum.
- The algorithm converges in a relatively small number of steps.
- Each step may take long to compute (if the training data is large).

Mini-batch Gradient Descent

- At each iteration step there's some random noise introduced and we take a path roughly in the direction of the minimum.
- The algorithm reaches near convergence in a larger number of steps.
- Each step is very quick to compute.

Online Gradient Descent and Streamed Data

- The online gradient descent is used used for streamed data.
- Like stochastic gradient descent, in online gradient descent the size of the mini-batch is 1.
- The difference is that the online gradient descent will use the latest training sample available from the stream.
- At every iteration of the gradient descent, the most recent item available from the stream is used.
- This allows the gradient descent mechanism model data that changes over time.

Take-home Messages

- What is machine learning?
- K-nearest neighbours.
- The curse of dimensionality.
- locality sensitive hashing for KNN.
- Parametric vs. non-parametric models.
- Supervised ML as an optimisation problem.
- Batch gradient descent.
- Mini-batch gradient descent.
- Stochastic gradient descent.
- Online gradient descent.

What's Next

Week 12

- Large-Scale Machine Learning (II)
- Submission deadline for assignment 3.