Lecture 13: Classification

COMP90049
Knowledge Technologies

Hasti Samadi & Sarah Erfani & Karin Verspoor, CIS

Semester 2, 2019

What is Classification?

- **Classification** is a supervised learning approach in which we build a probabilistic model of the training data, and then use that to predict the class labels of the test data.

What is Classification?

- **Classification** is a supervised learning approach in which we build a probabilistic model of the training data, and then use that to predict the class labels of the test data.

ID	*Outl*	*Temp*	*Humi*	*Wind*	PLAY
TRAINING INSTANCES					
A	s	h	h	F	N
B	s	h	h	T	N
C	o	h	h	F	Y
D	r	m	h	F	Y
E	r	c	n	F	Y
F	r	c	n	T	N
TEST INSTANCES					
G	o	c	n	T	?
H	s	m	h	F	?

Example: Supervised Learning

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
overcast	cool	high	true	no

What (do you think) is the class of sunny, cool, normal, false?

Example: Supervised Learning

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
sunny	cool	normal	false	yes
sunny	mild	high	false	no
overcast	cool	high	true	no

What (do you think) is the class of rainy, hot, normal, true?

Example: Supervised Learning

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
overcast	mild	normal	true	yes
sunny	mild	normal	false	yes
overcast	hot	high	true	yes
sunny	cool	high	false	yes
rainy	cool	normal	true	no
overcast	hot	normal	true	no
sunny	hot	normal	false	no
sunny	mild	normal	true	no
rainy	cool	high	true	no

What (do you think) is the class of overcast, mild, high, false?

What are (Supervised) Classifiers?

Given:

1. A fixed representation of attributes
2. A fixed set of pre-classified training instances
3. A fixed set of classes, C
4. A learner algorithm which can identify patterns in the training instances

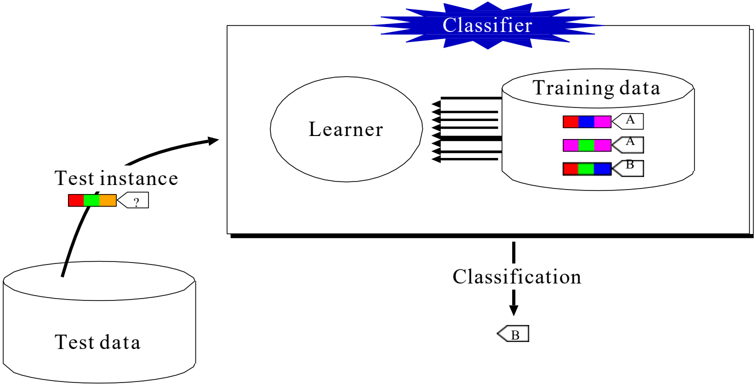
Predicts (estimates):

- *the category of a novel input x* : $c(x) \in C$

Model:

- *discover the function that predicts the label* $c(x)$ *given a previously unseen x*

Supervised classification paradigm



The goal of learning from examples is not to **memorise** but rather to **generalise**, e.g., predict.

Naïve Bayes Classifier

- Learning and classification methods based on probability theory
- Given the train instances and their labels (***C***), which class $\hat{c}$ is most likely for the test instance ***T***?

$$\hat{c} = \arg\max_{c \in C} P(c|T)$$

Bayesian Methods

- Learning and classification methods based on probability theory
- Given the train instances and their labels ***C***, which class $\hat{c}$ is <u>most likely</u> for the test instance ***T***?

$$\hat{c} = \arg\max_{c \in C} P(c|T)$$

- Bayes' Rules:

$$P(C,X) = P(C|X)P(X) = P(X|C)P(C)$$

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Bayesian Methods

- Based on the prior knowledge (training data), the learner calculates the (posterior) probability of label $c_j \in C$ for the test instance T $\Longrightarrow$ $P(c_j|T)$
- After calculating the probabilities of $P(c_j|T)$ for *all* the class labels in C, the model will choose the most probable class as the prediction.

- Based on the prior knowledge (training data), the learner calculates the (posterior) probability of label $c_j \in C$ for the test instance $T \Longrightarrow P(c_j|T)$
- After calculating the probabilities of $P(c_j|T)$ for *all* the class labels in C, the model will choose the most probable class as the prediction.

$$\begin{aligned}\hat{c} &= \arg\max_{c_j \in C} P(c_j|T) \\ &= \arg\max_{c_j \in C} \frac{P(T|c_j)P(c_j)}{P(T)}\end{aligned}$$

Bayesian Methods

$$\hat{c} = \arg\max_{c_j \in C} P(c_j | T)$$

$$
\begin{aligned}
\hat{c} &= \arg\max_{c_j \in C} P(c_j | T) \\
&= \arg\max_{c_j \in C} \frac{P(T|c_j)P(c_j)}{P(T)}
\end{aligned}
$$

$$
\begin{aligned}
\hat{c} &= \underset{c_j \in C}{\arg\max}\, P(c_j | T) \\
&= \underset{c_j \in C}{\arg\max}\, \frac{P(T|c_j)P(c_j)}{P(T)}
\end{aligned}
$$

Since $P(T)$ is constant for all classes, only $P(T|c_j)P(c_j)$ needs to be maximised.

$$\hat{c} = \arg\max_{c_j \in C} P(c_j | T)$$

$$= \arg\max_{c_j \in C} \frac{P(T|c_j)P(c_j)}{P(T)}$$

Since $P(T)$ is constant for all classes, only $P(T|c_j)P(c_j)$ needs to be maximised.

$$\hat{c} \cong \arg\max_{c_j \in C} P(T|c_j)P(c_j)$$

Bayesian Methods

- To classify an instance T = $(x_1, x_2, \ldots, x_n)$ according to one of the classes $c_j \in C$

$$\begin{aligned}
\hat{c} &= \arg\max_{c_j \in C} P(c_j | x_1, x_2, \ldots, x_n) \\
&= \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n | c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)} \\
&\cong \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n | c_j) P(c_j)
\end{aligned}$$

The "Naïve" part

- To get around this problem, we do something stupid 🤪:

$$\begin{aligned} P(x_1, x_2, \ldots, x_n | c_j) &\approx P(x_1|c_j)P(x_2|c_j)\ldots P(x_n|c_j) \\ &= \prod_i P(x_i|c_j) \end{aligned}$$

The "Naïve" part

- To get around this problem, we do something stupid 🤪:

$$
\begin{aligned}
P(x_1, x_2, \ldots, x_n | c_j) &\approx P(x_1|c_j)P(x_2|c_j)\ldots P(x_n|c_j) \\
&= \prod_i P(x_i|c_j)
\end{aligned}
$$

- This is a **conditional independence assumption**, and makes Naive Bayes a tractable method.
- It is also demonstrably untrue in almost every dataset. However, Naïve Bayes (kinda) works anyway! 😉

Naïve Bayes Classifier

- To classify an instance T = $(x_1, x_2, \ldots, x_n)$ according to one of the classes $c_j \in C$

$$\hat{c} = \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n | c_j) P(c_j)$$

Naïve Bayes Classifier

- To classify an instance T = $(x_1, x_2, \ldots, x_n)$ according to one of the classes $c_j \in C$

$$\begin{aligned}\hat{c} &= \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n | c_j) P(c_j) \\ &= \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)\end{aligned}$$

Naïve Bayes Classifier

- To classify an instance $T = (x_1, x_2, \ldots, x_n)$ according to one of the classes $c_j \in C$

$$\hat{c} = \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n | c_j) P(c_j)$$

$$= \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)$$

Other assumption:

- **$P(c_j)$** can be estimated from the frequency of classes in the training examples ***[maximum likelihood estimate]***
- The distribution of data in the training instances is (roughly) the same as the distribution in the test instances

- Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Naive Bayes Example

- Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

- We need $P(c_j)$, $P(x_i|c_j)$: for every x_i, c_j

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	**Flu**
mild	no	normal	no	Cold
severe	severe	normal	yes	**Flu**

$P(Flu) = 3/5$ $\qquad$ $P(Cold) = 2/5$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$P(Flu) = 3/5$ $P(Cold) = 2/5$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5 \qquad P(Cold) = 2/5$$

$$P(Headache = severe \mid Flu) = 2/3$$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$

$$P(Headache = severe \mid Flu) = 2/3$$

$$P(Cold) = 2/5$$

$$P(Headache = severe \mid Cold) = 0/2$$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$
$$P(Headache = severe \mid Flu) = 2/3$$
$$P(Headache = mild \mid Flu) = 1/3$$

$$P(Cold) = 2/5$$
$$P(Headache = severe \mid Cold) = 0/2$$
$$P(Headache = mild \mid Cold) = 1/2$$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$

$$P(Headache = severe \mid Flu) = 2/3$$

$$P(Headache = mild \mid Flu) = 1/3$$

$$P(Headache = no \mid Flu) = 0/3$$

$$P(Cold) = 2/5$$

$$P(Headache = severe \mid Cold) = 0/2$$

$$P(Headache = mild \mid Cold) = 1/2$$

$$P(Headache = no \mid Cold) = 1/2$$

Naive Bayes Example

Headache	**Sore**	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$
$$P(Headache = severe \mid Flu) = 2/3$$
$$P(Headache = mild \mid Flu) = 1/3$$
$$P(Headache = no \mid Flu) = 0/3$$

$$P(Sore = severe \mid Flu) = 1/3$$
$$P(Sore = mild \mid Flu) = 2/3$$
$$P(Sore = no \mid Flu) = 0/3$$

$$P(Cold) = 2/5$$
$$P(Headache = severe \mid Cold) = 0/2$$
$$P(Headache = mild \mid Cold) = 1/2$$
$$P(Headache = no \mid Cold) = 1/2$$

$$P(Sore = severe \mid Cold) = 1/2$$
$$P(Sore = mild \mid Cold) = 0/2$$
$$P(Sore = no \mid Cold) = 1/2$$

Naive Bayes Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$
$$P(Headache = severe \mid Flu) = 2/3$$
$$P(Headache = mild \mid Flu) = 1/3$$
$$P(Headache = no \mid Flu) = 0/3$$
$$P(Sore = severe \mid Flu) = 1/3$$
$$P(Sore = mild \mid Flu) = 2/3$$
$$P(Sore = no \mid Flu) = 0/3$$

$$P(Temp = high \mid Flu) = 1/3$$
$$P(Temp = normal \mid Flu) = 2/3$$

$$P(Cold) = 2/5$$
$$P(Headache = severe \mid Cold) = 0/2$$
$$P(Headache = mild \mid Cold) = 1/2$$
$$P(Headache = no \mid Cold) = 1/2$$
$$P(Sore = severe \mid Cold) = 1/2$$
$$P(Sore = mild \mid Cold) = 0/2$$
$$P(Sore = no \mid Cold) = 1/2$$

$$P(Temp = high \mid Cold) = 0/2$$
$$P(Temp = normal \mid Cold) = 2/2$$

Naive Bayes Example

Headache	Sore	Temperature	**Cough**	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$P(Flu) = 3/5$

$P(Headache = severe|Flu) = 2/3$

$P(Headache = mild|Flu) = 1/3$

$P(Headache = no|Flu) = 0/3$

$P(Sore = severe|Flu) = 1/3$

$P(Sore = mild|Flu) = 2/3$

$P(Sore = no|Flu) = 0/3$

$P(Temp = high|Flu) = 1/3$

$P(Temp = normal|Flu) = 2/3$

$P(Cough = yes|Flu) = 3/3$

$P(Cough = no|Flu) = 0/3$

$P(Cold) = 2/5$

$P(Headache = severe|Cold) = 0/2$

$P(Headache = mild|Cold) = 1/2$

$P(Headache = no|Cold) = 1/2$

$P(Sore = severe|Cold) = 1/2$

$P(Sore = mild|Cold) = 0/2$

$P(Sore = no|Cold) = 1/2$

$P(Temp = high|Cold) = 0/2$

$P(Temp = normal|Cold) = 2/2$

$P(Cough = yes|Cold) = 1/2$

$P(Cough = no|Cold) = 1/2$

Naive Bayes Example

- Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a cold, or the flu?

- Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a cold, or the flu?

Cold:

$$P(C) \times P(H=m|C)P(S=s|C)P(T=n|C)P(C=n|C)$$
$$\frac{2}{5} \times (\frac{1}{2})(\frac{1}{2})(\frac{2}{2})(\frac{1}{2}) = 0.05$$

Flu:

$$P(F) \times P(H=m|F)P(S=s|F)P(T=n|F)P(C=n|F)$$
$$\frac{3}{5} \times (\frac{1}{3})(\frac{1}{3})(\frac{2}{3})(\frac{0}{3}) = 0$$

Naive Bayes Example

- Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Naive Bayes Example

- Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccc} P(C) & \times & P(H=s|C)P(S=m|C)P(T=h|C)P(C=n|C) \\ \frac{2}{5} & \times & (\frac{0}{2})(\frac{0}{2})(\frac{0}{2})(\frac{1}{2}) = 0 \end{array}$$

Flu:

$$\begin{array}{ccc} P(F) & \times & P(H=s|F)P(S=m|F)P(T=h|F)P(C=n|F) \\ \frac{3}{5} & \times & (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\frac{0}{3}) = 0 \end{array}$$

Probabilistic Smoothing

- Notice that this is a product, so if any $P(x_i|c_j) = 0$, then the final value is 0
- This is bad for two reasons:
  - To make plausible predictions, we still need to see every possible attribute value–class pair ... and so, we still require lots and lots of data
  - Unseen events mean that we discard a whole lot of otherwise useful information
- Solution: no event is impossible (every probability > 0)
- To maintain a **probability distribution**, we need to reduce the probability of seen events

Probabilistic Smoothing

- The (conceptually) simplest approach:
- If we calculate $P(x_i|c_j) = 0$, we replace zero with a trivially small non-zero number, typically called ε
- ε is a constant, which needs to be less (and preferably substantially less) than $\frac{1}{n}$, for n instances
- Effectively reduces most comparisons to the cardinality of ε (fewest εs wins)
- Assume that $(1 + \varepsilon) = 1$, so that we don't need to do anything extra with the non-zero probabilities

- Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{aligned} P(C) &\times P(H=s|C)P(S=m|C)P(T=h|C)P(C=n|C) \\ \frac{2}{5} &\times (\epsilon)(\epsilon)(\epsilon)(\frac{1}{2}) = \frac{\epsilon^3}{5} \end{aligned}$$

Flu:

$$\begin{aligned} P(F) &\times P(H=s|F)P(S=m|F)P(T=h|F)P(C=n|F) \\ \frac{3}{5} &\times (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\epsilon) = \frac{4\epsilon}{45} \end{aligned}$$

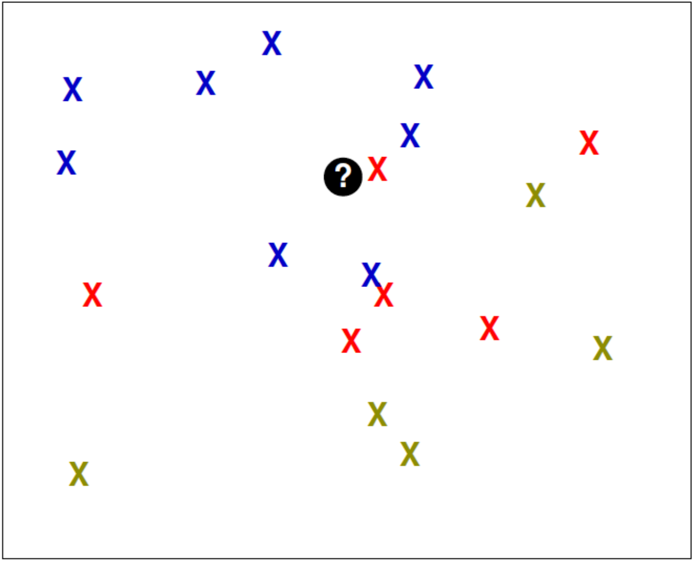
Naive Bayes, analysis

Naive Bayes (NB) Classifier is very simple to build, extremely fast to make decisions, and easy to change the probabilities when the new data becomes available.

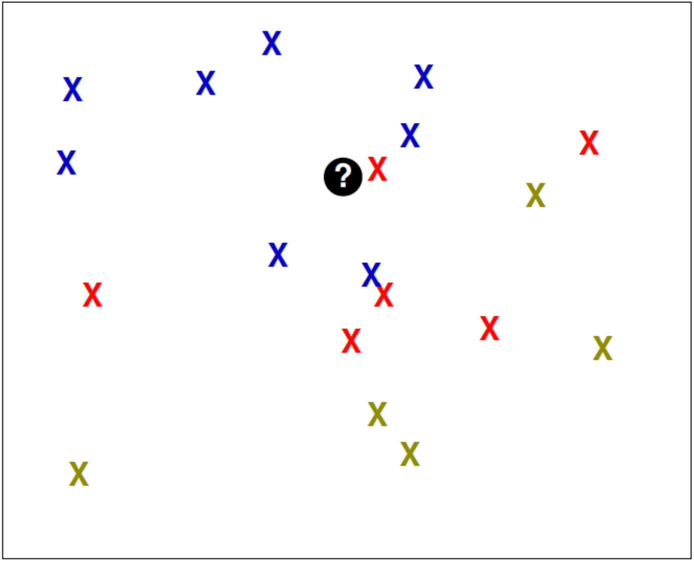
- Works well in many application areas.
- Scales easily for large number of dimensions (100s) and data sizes.
- Easy to explain the reason for the decision made.
- One should apply NB first before launching into more sophisticated classification techniques.

k-Nearest Neighbour Classifier

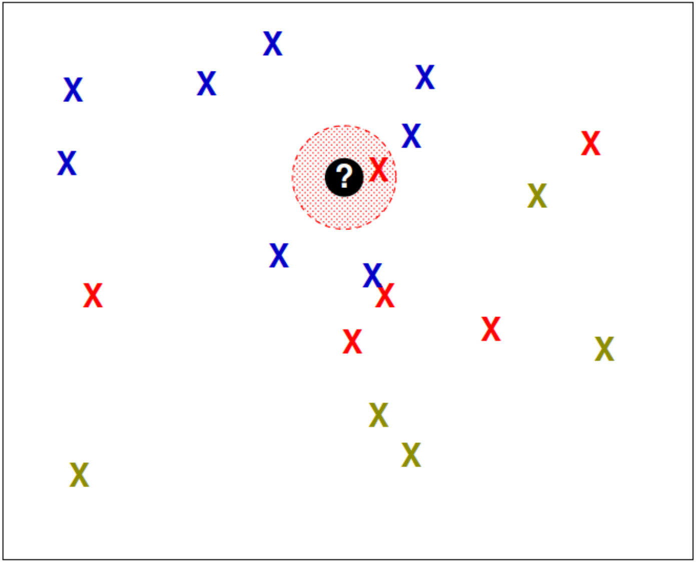
k-Nearest Neighbour Classifier



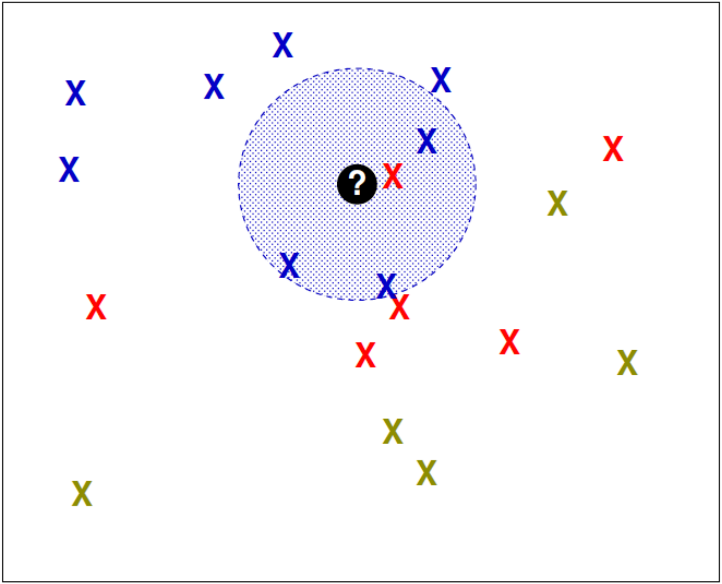
k-Nearest Neighbour Classifier



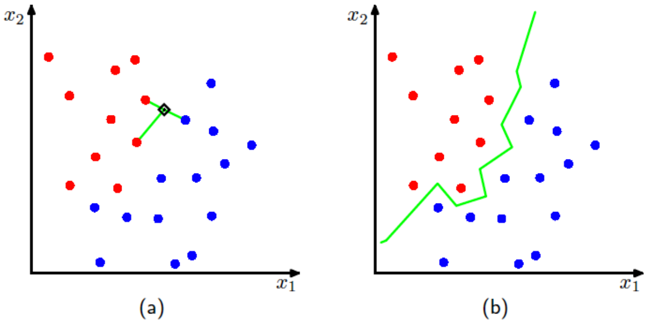
k-Nearest Neighbour Classifier



k-Nearest Neighbour Classifier



k−Nearest Neighbour methods in Classification



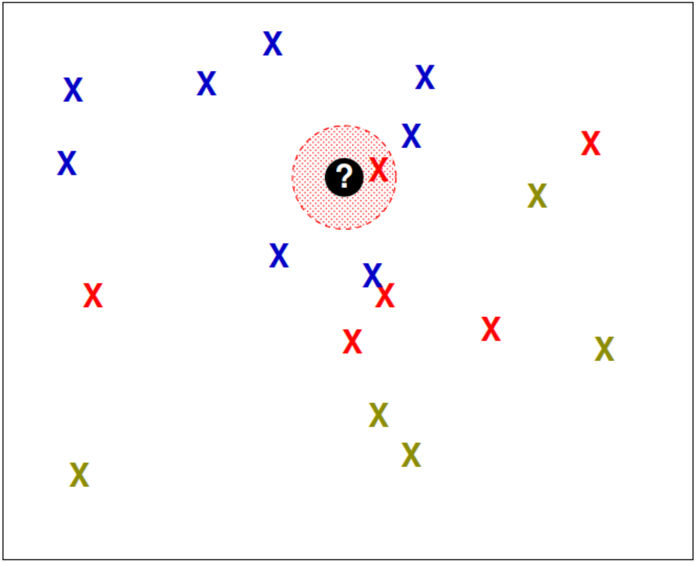
Given class assignments for existing data points, classify a new point (black).

- Consider the class membership of the K closest data points.
- For $k = 1$, the induced decision boundary. (figure b)

See: Charles Elkan, UCSD, 2011 lecture notes (posted on LMS)

[1-NN]: Classify the test input according to the class of the closest training instance.

1-Nearest Neighbour

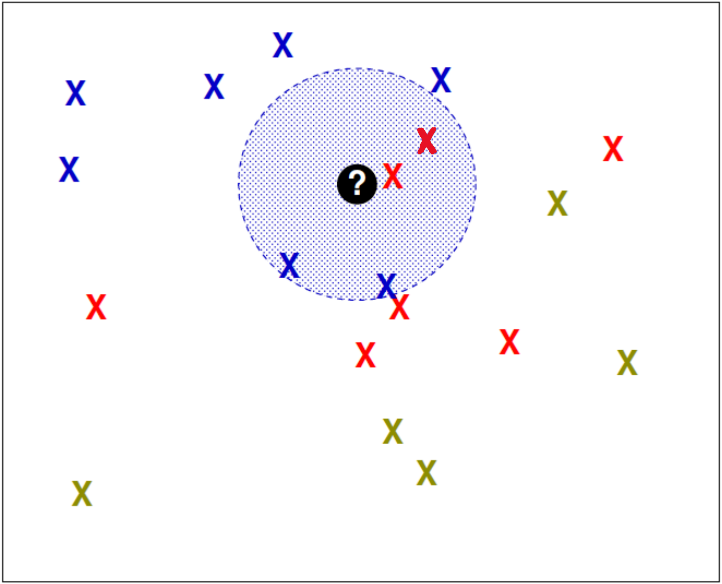


$k-$Nearest Neighbour classification strategies

[1-NN]: Classify the test input according to the class of the closest training instance.

[*k* -NN]: Classify the test input according to the majority class of the *k* nearest training instances.

k-Nearest Neighbour



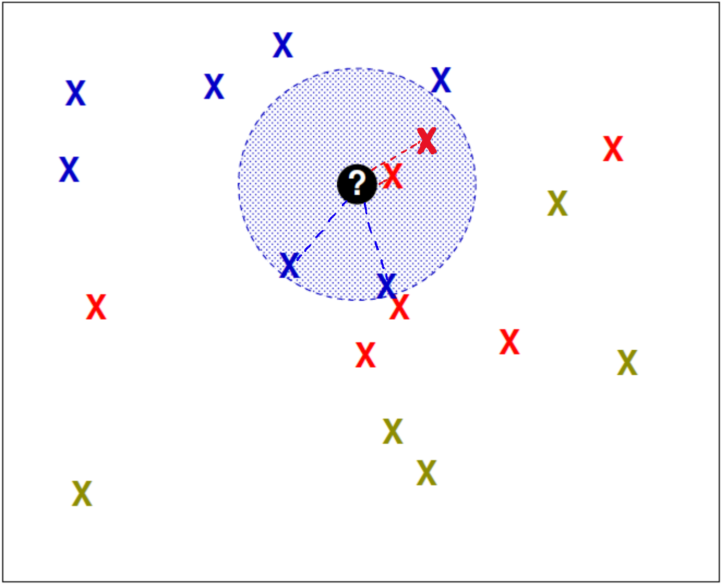
k−Nearest Neighbour classification strategies

[1-NN]: Classify the test input according to the class of the closest training instance.

[k -NN]: Classify the test input according to the majority class of the k nearest training instances.

[weighted k -NN]: Classify the test input according to the weighted accumulative class of the k nearest training instances, where weights are based on similarity of the input to each of the k neighbours.

Weighted *k*-Nearest Neighbour



$k-$Nearest Neighbour classification implementation

The most naive neighbour search implementation involves the brute-force computation of distances between all pairs of points in the dataset.

For N samples in D dimensions, this approach scales as $O(DN^2)$.

- Efficient brute-force neighbours searches can be very competitive for small data samples.
- However, as the number of samples N grows, the brute-force approach quickly becomes infeasible.

Alternative: tree-based data structures

- These structures attempt to reduce the required number of distance calculations by efficiently encoding aggregate distance information for the sample.
- The basic idea is that if point A is very distant from point B, and point B is very close to point C, then we know that points A and C are very distant, without having to explicitly calculate their distance.
- In this way, the computational cost of a nearest neighbours search can be reduced to $O(DN \log(N))$ or better.

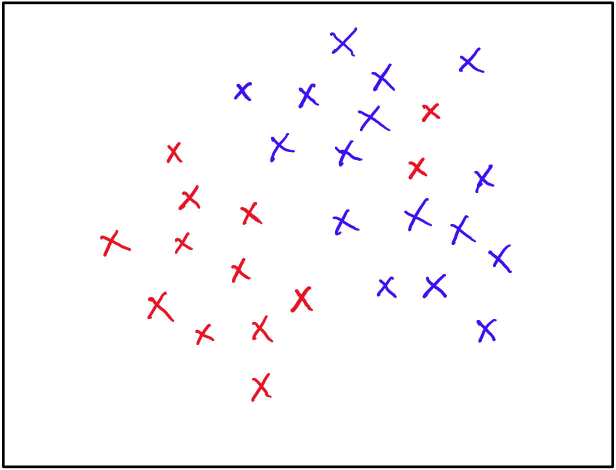
Strengths

- Simple
- Can handle arbitrarily many classes (multi-class and multi-label)

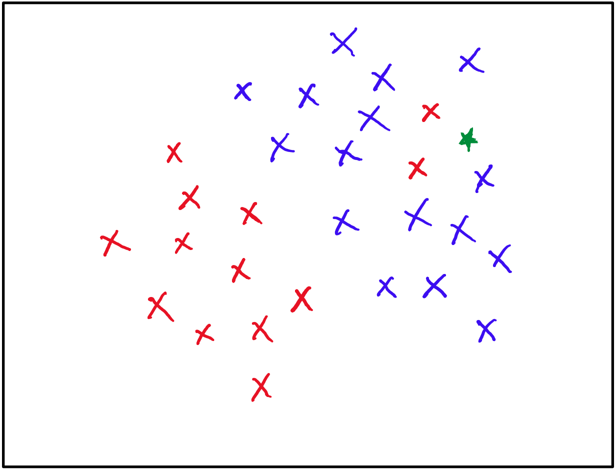
Weaknesses

- We need a useful distance function, which may not be obvious to design for some sets.
- We need some sort of averaging or voting function for combining the labels of multiple training examples, which may also not be obvious to design.
- Expensive (in terms of index accesses)
- Everything is done at run time (lazy Learner)
- Arbitrary k value
- Prone to bias

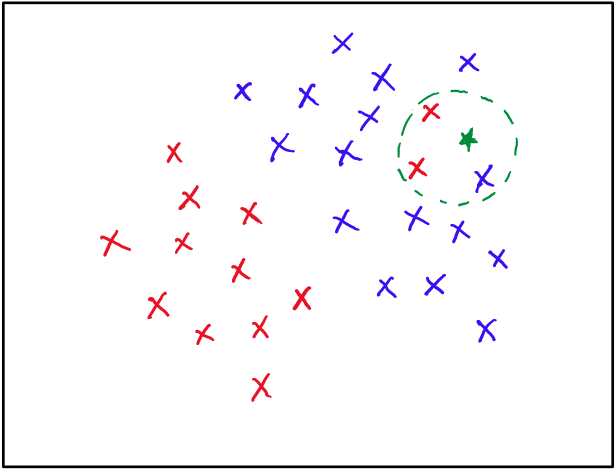
Bias issue in K-NN methods



Bias issue in K-NN methods



Bias issue in K-NN methods



So far we have studied Naive Bayes and K-NN classifiers:

- The training data consists of input-output pairs.
- **Input**: categorical or numeric,
- **Output**: categorical (e.g. Yes/No, Blue/Red/Green).

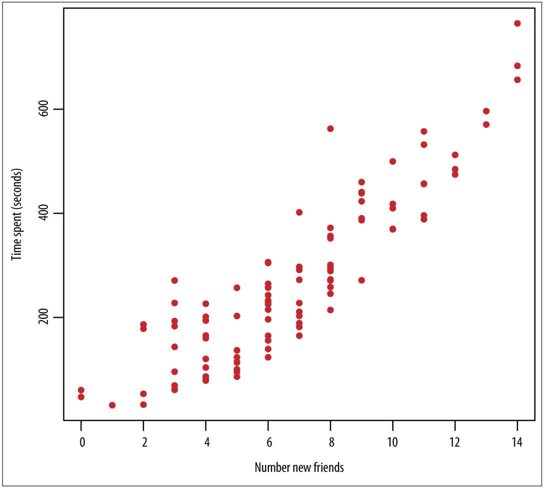
- What if the output/class is continuous?

Linear Regression

Regression is an important type of *Supervised Learner* where the output is continuous (e.g. is a real number) with many applications:

- Predict wind farm energy output from weather data
- Predict the number of customers for a shop from date/weather/holidays
- Predict the price of a product (e.g. gold/stocks) in future (for economic planning).

Explore the relationship



From Schutt & O'Neil, *Doing Data Science*

Linear regression, mathematically

Linear regression captures a relationship between the attribute values and the numeric output c.

It makes the assumption that there is a *linear* relationship between the two variables.

1. A predictor (aka independent variable, explanatory variable, or feature)
2. An outcome variable (aka response variable, dependent variable, or label)

Linear regression, mathematically

At its most basic, the relationship can be expressed as a *line* (the correlation between one variable and the output).

$$y = f(x)$$

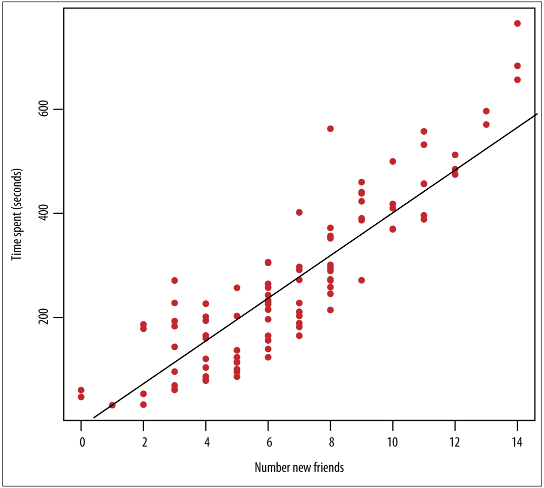
$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$$

$$y = \beta \,.\, X \quad (\text{given } x_0 = 1)$$

- Linear functions are less descriptive than non-linear functions, but permit simpler (mathematical) strategies.
- They capture changes in one variable that correlate linearly with changes in another variable.
- For some variables, this makes sense. For example: The more umbrellas you sell, the more money you make. How much money you make is directly proportional to how many umbrellas you sell.

Explore the relationship

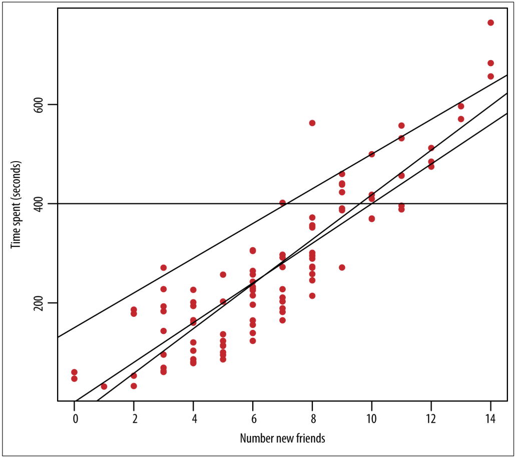
We derive a linear model by estimating it from training examples.



From Schutt & O'Neil, *Doing Data Science*

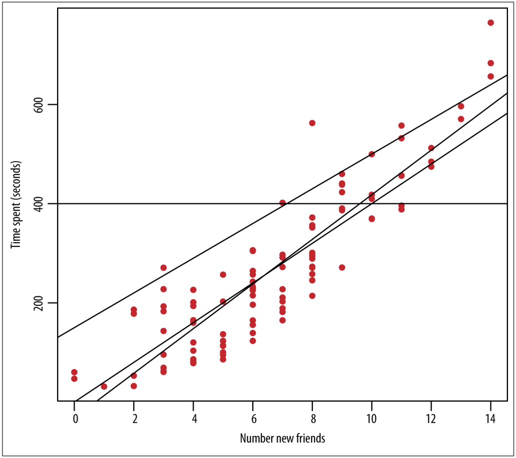
Explore the relationship

Given examples (x_1, y_1), (x_2, y_2),...,(x_N; y_N), we determine the optimal $\beta_0, \beta_1, \ldots, \beta_N$



Explore the relationship

Given examples $(x_1, y_1), (x_2, y_2), \ldots, (x_N; y_N)$, we determine the <u>optimal</u> $\beta_0, \beta_1, \ldots, \beta_N$



To find the optimal β, operationally we are looking for the line that minimises the ***distance*** between all points and the line.

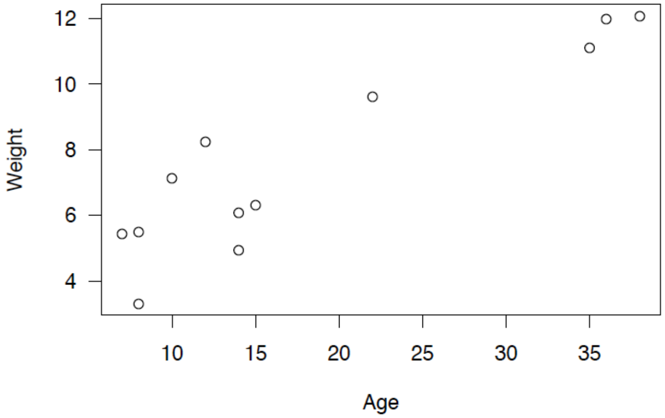
Least squares estimation:

- Find the line that minimises the *sum of the squares* of the vertical distances between approximated/predicted $\hat{y}_i$ and observed y_i.

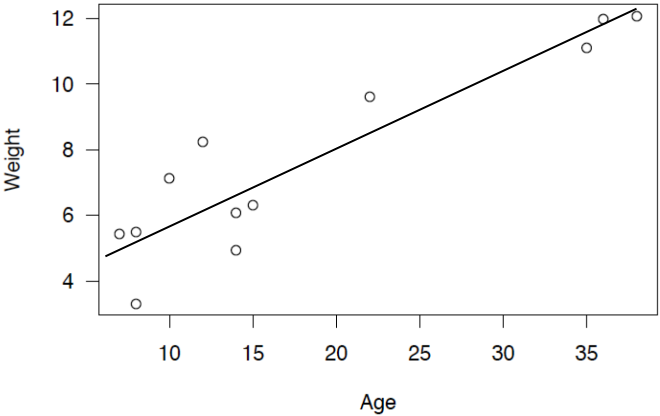
$$RSS(\beta) = \sum_i (y_i - \hat{y}_i)^2$$

- Put another way, we want to find the β that produces $\hat{y}_i$ for each x_i that is closest to the known y_i.

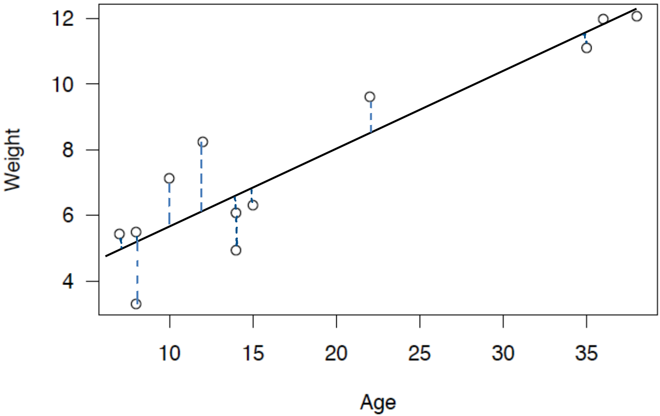
Fitting the model



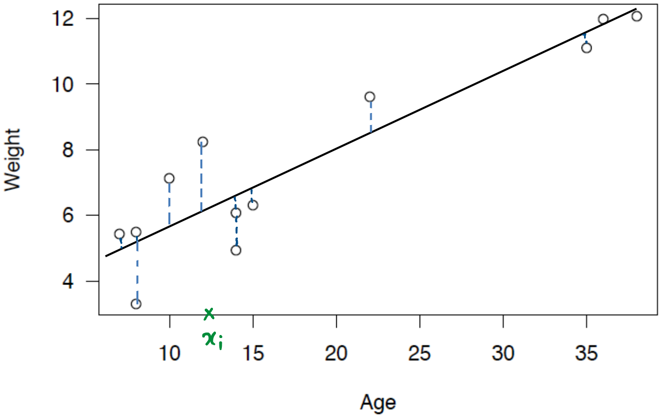
Fitting the model



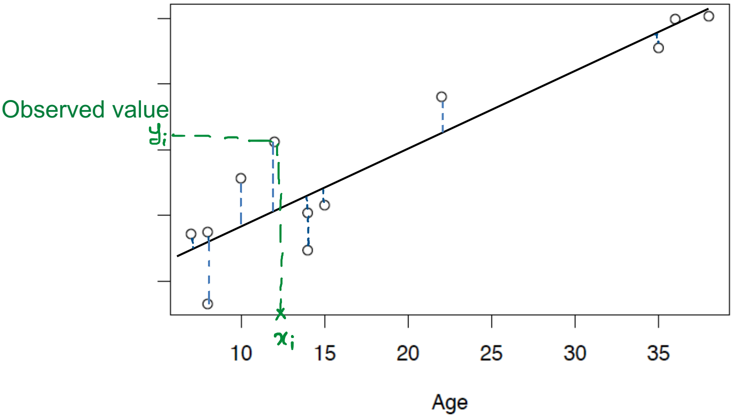
Fitting the model



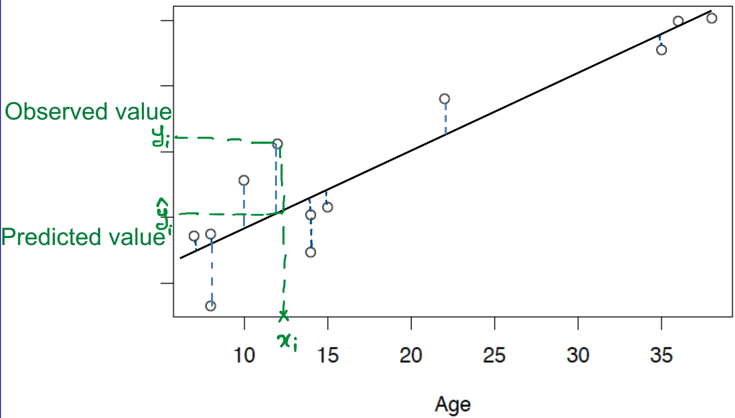
Fitting the model

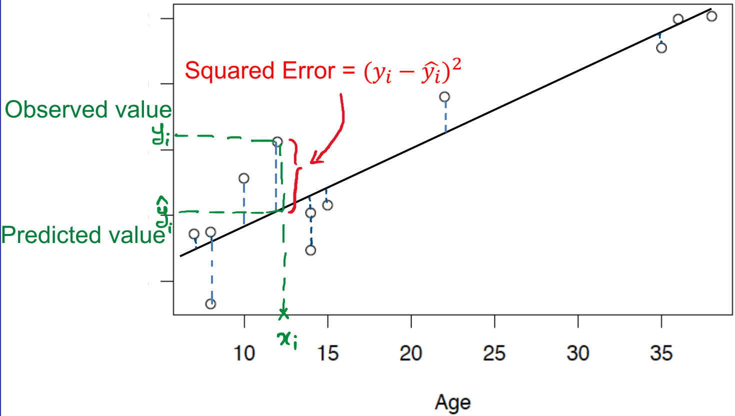


Fitting the model



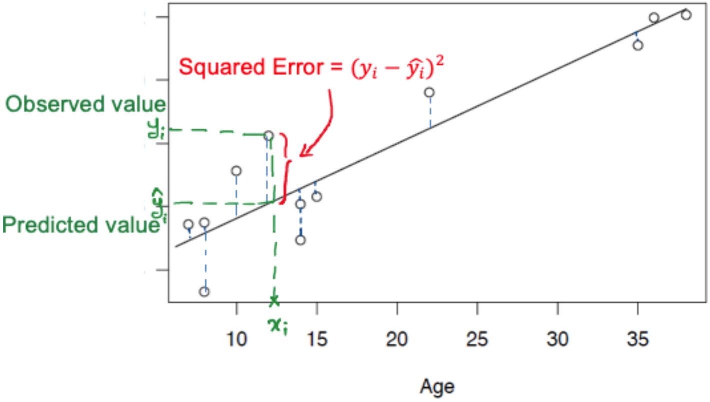
Fitting the model




Squared Error = $(y_i - \hat{y}_i)^2$
Observed value
y_i
$\hat{y}_i$
Predicted value
x_i
10
15
20
25
30
35
Age

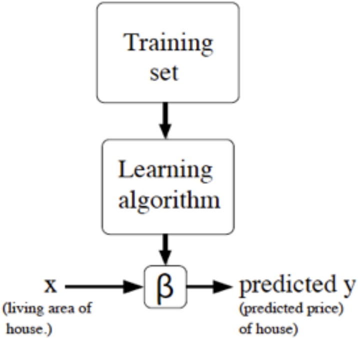
Fitting the model

$$
\begin{aligned}
RSS(\beta) &= \sum_i (y_i - \hat{y}_i)^2 \\
&= \sum_i (y_i - \beta \cdot x_i)^2
\end{aligned}
$$



Minimizing the *Residual Sum of Squares (RSS)* or *Sum of Square Errors (SSE)*

$$\hat{\beta} = \arg\min \mathrm{RSS}(\beta; \{\mathbf{X}, Y\})$$



Predicting with Linear Regression

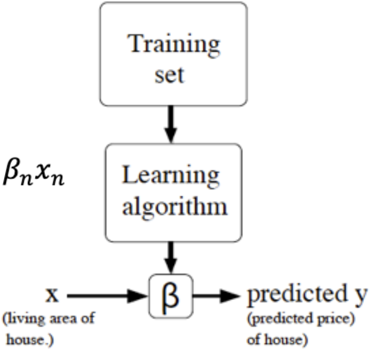
- Armed with a linear model $y = \beta_0 + \beta \,.\, X$, we can straightforwardly predict a continuous valued output for y given a value of x .

- To classify a test instance X = $(x_1, x_2, \ldots, x_n)$ using our linear regression model, we will have:

$$y = \beta_0 + \beta.X$$

$$y = \beta_0 + \beta.(x_1, x_2, \ldots, x_n)$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



Summary

- How does the Naive Bayes algorithm work? What assumptions are required to make the computation tractable?
- How does the k-nearest neighbour method operate, and what are some of the variants on the original algorithm?
- What is linear regression, and how does it operate?

Resources

Charles Elkan, UCSD, lecture notes
http://cseweb.ucsd.edu/~elkan/250Bwinter2010/nearestn.pdf

Tan et al. (2006), Introduction to Data Mining. p. 227–238

Witten, Frank, Hall (2011) Data Mining. Chapter 4. (*kD – tree*, ball tree)