Classification

Jiří Vomlel

Academy of Sciences of the Czech Republic

10th July, 2007

Classification task

- We assume objects characterized by a number of attributes $X = (X_1, \ldots, X_p)$.

Classification task

- We assume objects characterized by a number of attributes $X = (X_1, \dots, X_p)$.
- The goal is to classify objects to one of K classes - we want to predict the value of a class variable Y taking $1, \dots, K$ values.

Classification task

- We assume objects characterized by a number of attributes $X = (X_1, \dots, X_p)$.
- The goal is to classify objects to one of K classes - we want to predict the value of a class variable Y taking $1, \dots, K$ values.

Example

Predict, whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.

Classification task

- We assume objects characterized by a number of attributes $X = (X_1, \ldots, X_p)$.
- The goal is to classify objects to one of K classes - we want to predict the value of a class variable Y taking $1, \ldots, K$ values.

Example

Predict, whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.

Example

Identify the numbers in a handwritten ZIP code, from a digitized image.

Classification task

- We assume objects characterized by a number of attributes $X = (X_1, \ldots, X_p)$.
- The goal is to classify objects to one of K classes - we want to predict the value of a class variable Y taking $1, \ldots, K$ values.

Example

Predict, whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.

Example

Identify the numbers in a handwritten ZIP code, from a digitized image.

Example

Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.

Example

Whether variable may have values *Sunny, Overcast, and Rainy*.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.

Example

Boolean variable with values *TRUE* and *FALSE*.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.

Example

Climate may have values *Hot, Mild, Cold*. They can be ordered, *Mild* lays between *Hot* and *Cold*.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.

Example

An example is *Temperature* measured in degrees of centigrade.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.
- **Ratio variables** are ones for which which the measurement method inherently defines a zero point.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.
- **Ratio variables** are ones for which which the measurement method inherently defines a zero point.

Example

A distance of two objects.

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.
- **Ratio variables** are ones for which which the measurement method inherently defines a zero point.

In most applications we distinguish only between

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.
- **Ratio variables** are ones for which which the measurement method inherently defines a zero point.

In most applications we distinguish only between

- **Nominal** (also called categorical) variables and

Variables (Attributes and the Class)

- **Nominal variables** have values that are distinct symbols. The values themselves serve as a label or a symbol.
- **Ordinal variables** have values that can be ordered or ranked.
- **Interval variables** have values that can be not only ordered but also measured in fixed and equal units.
- **Ratio variables** are ones for which which the measurement method inherently defines a zero point.

In most applications we distinguish only between

- **Nominal** (also called categorical) variables and
- **Ordinal** (also called numerical) variables.

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

- precision $\pi = \frac{tp}{tp+fp}$

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

- precision $\pi = \frac{tp}{tp+fp}$
- recall (sensitivity or TP-rate) $\varrho = \frac{tp}{tp+fn}$

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

- precision $\pi = \frac{tp}{tp+fp}$
- recall (sensitivity or TP-rate) $\varrho = \frac{tp}{tp+fn}$
- specificity $\sigma = \frac{tn}{tn+fp}$

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

- precision $\pi = \frac{tp}{tp+fp}$
- recall (sensitivity or TP-rate) $\varrho = \frac{tp}{tp+fn}$
- specificity $\sigma = \frac{tn}{tn+fp}$
- accuracy $\eta = \frac{tp+tn}{tp+tn+fn+fp}$

Classifier performance measures

Typically, we use a testing set of objects and for each classifier we compute number of objects

- classified correctly to the class Y ... *tp* (true positive),
- classified incorrectly to the class Y ... *fp* (false positive),
- classified correctly out of the class Y ... *tn* (true negative), and
- classified incorrectly out of the class Y ... *fn* (false negative).

Performance measures based on these counts:

- precision $\pi = \frac{tp}{tp+fp}$
- recall (sensitivity or TP-rate) $\varrho = \frac{tp}{tp+fn}$
- specificity $\sigma = \frac{tn}{tn+fp}$
- accuracy $\eta = \frac{tp+tn}{tp+tn+fn+fp}$
- F_1-measure $F_1 = \frac{2\pi\varrho}{\pi+\varrho}$

ROC curve

- Most classifiers can be tuned so that they sacrifice precision to high recall or vice versa.

ROC curve

- Most classifiers can be tuned so that they sacrifice precision to high recall or vice versa.
- Typically, it can be done by setting a threshold for the predicted value of the object. The objects with a higher predicted value a classified as belonging to the class.

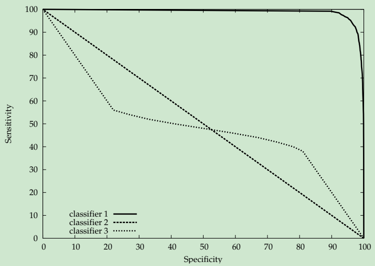
ROC curve

- Most classifiers can be tuned so that they sacrifice precision to high recall or vice versa.
- Typically, it can be done by setting a threshold for the predicted value of the object. The objects with a higher predicted value a classified as belonging to the class.
- ROC (Receiver Operating Characteristic) curve is a plot of sensitivity vs. specificity.

ROC curve

- Most classifiers can be tuned so that they sacrifice precision to high recall or vice versa.
- Typically, it can be done by setting a threshold for the predicted value of the object. The objects with a higher predicted value a classified as belonging to the class.
- ROC (Receiver Operating Characteristic) curve is a plot of sensitivity vs. specificity.

Example (ROC curves)



Cross validation

- Typically, a training dataset is used to learn classification models.

Cross validation

- Typically, a training dataset is used to learn classification models.
- The learned models may depend too much on the details of the objects in the training data set. This phenomenon is called the overfitting problem.

Cross validation

- Typically, a training dataset is used to learn classification models.
- The learned models may depend too much on the details of the objects in the training data set. This phenomenon is called the overfitting problem.
- Therefore, the models should be always tested on a different dataset - a testing dataset.

Cross validation

- Typically, a training dataset is used to learn classification models.
- The learned models may depend too much on the details of the objects in the training data set. This phenomenon is called the overfitting problem.
- Therefore, the models should be always tested on a different dataset - a testing dataset.
- There are diverse approaches to avoid overfitting models.

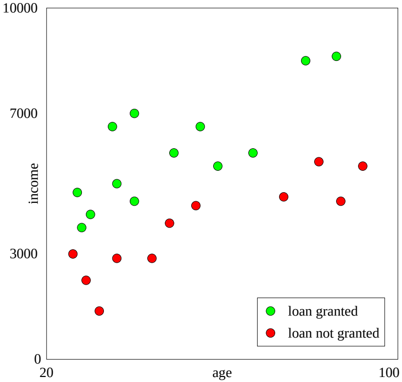
Cross validation

- Typically, a training dataset is used to learn classification models.
- The learned models may depend too much on the details of the objects in the training data set. This phenomenon is called the overfitting problem.
- Therefore, the models should be always tested on a different dataset - a testing dataset.
- There are diverse approaches to avoid overfitting models.

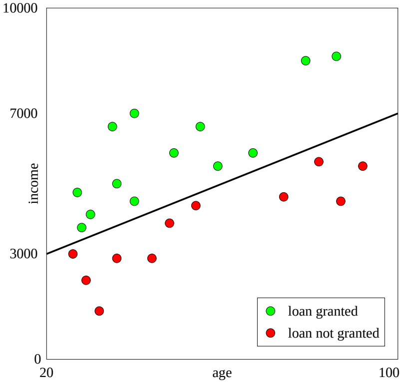
Example

Complex models performs well on training data but they get penalized for their size and simpler models that do not behave that well are selected instead.

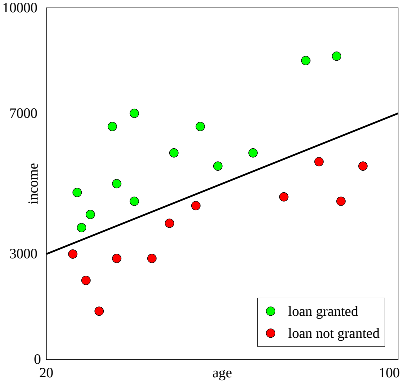
Linear regression



Linear regression



Linear regression



Loan granted if

$$
\begin{aligned}
\text{income} &\geq 3000 + \frac{7000-3000}{100-20}(\text{age}-20) \\
-2000 + \text{income} - 50 \cdot \text{age} &\geq 0
\end{aligned}
$$

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.
- Let X_2 be the age and $\beta_2 = -50$ its coefficient.

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.
- Let X_2 be the age and $\beta_2 = -50$ its coefficient.
- Y be the class variable meaning that the loan is granted if $Y \geq 0$ and not granted if $Y < 0$.

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.
- Let X_2 be the age and $\beta_2 = -50$ its coefficient.
- Y be the class variable meaning that the loan is granted if $Y \geq 0$ and not granted if $Y < 0$.
- Let X_0 be an auxiliary variable always taking value 1 with its coefficient $\beta_0 = -2000$ (intercept).

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.
- Let X_2 be the age and $\beta_2 = -50$ its coefficient.
- Y be the class variable meaning that the loan is granted if $Y \geq 0$ and not granted if $Y < 0$.
- Let X_0 be an auxiliary variable always taking value 1 with its coefficient $\beta_0 = -2000$ (intercept).

Then we can use for classification a linear model

$$\begin{aligned} Y &= \beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2 \\ &= -2000 + X_1 - 50 X_2 \end{aligned}$$

Linear regression

- Let X_1 be the income and $\beta_1 = 1$ its coefficient.
- Let X_2 be the age and $\beta_2 = -50$ its coefficient.
- Y be the class variable meaning that the loan is granted if $Y \geq 0$ and not granted if $Y < 0$.
- Let X_0 be an auxiliary variable always taking value 1 with its coefficient $\beta_0 = -2000$ (intercept).

Then we can use for classification a linear model

$$\begin{aligned} Y &= \beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2 \\ &= -2000 + X_1 - 50 X_2 \end{aligned}$$

Generally,

$$Y = \sum_{j=0}^{p} \beta_j X_j$$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x^i_1, \ldots, x^i_p)$.

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T; i = 0; n = 0;$
`while` $n < N$
  $i = i + 1; n = n + 1;$
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i;\ n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i;\ n = 0$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T; i = 0; n = 0;$

```
while n < N
  i = i + 1; n = n + 1;
  if (i > N) then i = 1
  if ((ỹ^i < 0) ∧ (y^i ≥ 0)) then
    β = β + x^i;  n = 0
  if ((ỹ^i ≥ 0) ∧ (y^i < 0)) then
    β = β − x^i;  n = 0
```

Example

$\boldsymbol{\beta} = (0, 0, 0)^T$

$i = 0$

$n = 0$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T\beta$ be the current prediction of y^i.

Definition (A learning algorithm)

$\beta = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1; n = n + 1;$
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\beta = \beta + \boldsymbol{x}^i;\ n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\beta = \beta - \boldsymbol{x}^i;\ n = 0$

Example

$\beta = (0, 0, 0)^T$
$i = 1$
$n = 1$
$\boldsymbol{x}^1 = (1, 2000, 30)^T$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x^i_1, \ldots, x^i_p)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1$; $n = n + 1$;
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i$; $n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i$; $n = 0$

Example

$\boldsymbol{\beta} = (0, 0, 0)^T$
$i = 1$
$n = 1$
$\boldsymbol{x}^1 = (1, 2000, 30)^T$
$\tilde{y}^1 = 0$
$y^1 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \beta$ be the current prediction of y^i.

Definition (A learning algorithm)

$\beta = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1$; $n = n + 1$;
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\beta = \beta + \boldsymbol{x}^i$; $n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\beta = \beta - \boldsymbol{x}^i$; $n = 0$

Example

$\beta = (-1, -2000, -30)^T$
$i = 1$
$n = 0$
$\boldsymbol{x}^1 = (1, 2000, 30)^T$
$\tilde{y}^1 = 0$
$y^1 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \beta$ be the current prediction of y^i.

Definition (A learning algorithm)

$\beta = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1; n = n + 1;$
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\beta = \beta + \boldsymbol{x}^i;\ n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\beta = \beta - \boldsymbol{x}^i;\ n = 0$

Example

$\beta = (-1, -2000, -30)^T$
$i = 2$
$n = 1$
$\boldsymbol{x}^2 = (1, 4000, 30)^T$
$\tilde{y}^1 = 0$
$y^1 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1$; $n = n + 1$;
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i$; $n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i$; $n = 0$

Example

$\boldsymbol{\beta} = (-1, -2000, -30)^T$
$i = 2$
$n = 1$
$\boldsymbol{x}^2 = (1, 4000, 30)^T$
$\tilde{y}^2 = -8000901$
$y^2 = +1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T; i = 0; n = 0;$
`while` $n < N$
  $i = i + 1; n = n + 1;$
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i;\ n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i;\ n = 0$

Example

$\boldsymbol{\beta} = (0, 2000, 0)^T$
$i = 2$
$n = 0$
$\boldsymbol{x}^2 = (1, 4000, 30)^T$
$\tilde{y}^2 = -8000901$
$y^2 = +1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x^i_1, \ldots, x^i_p)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T\boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T; i = 0; n = 0;$

`while` $n < N$

  $i = i + 1; n = n + 1;$

  `if` $(i > N)$ `then` $i = 1$

  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`

    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i;\ n = 0$

  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`

    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i;\ n = 0$

Example

$\boldsymbol{\beta} = (0, 2000, 0)^T$

$i = 3$

$n = 1$

$\boldsymbol{x}^3 = (1, 4000, 80)^T$

$\tilde{y}^2 = -8000901$

$y^2 = +1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;

```
while n < N
  i = i + 1; n = n + 1;
  if (i > N) then i = 1
  if ((ỹ^i < 0) ∧ (y^i ≥ 0)) then
    β = β + x^i;  n = 0
  if ((ỹ^i ≥ 0) ∧ (y^i < 0)) then
    β = β − x^i;  n = 0
```

Example

$\boldsymbol{\beta} = (0, 2000, 0)^T$

$i = 3$

$n = 1$

$\boldsymbol{x}^3 = (1, 4000, 80)^T$

$\tilde{y}^3 = +8000000$

$y^3 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x^i_1, \ldots, x^i_p)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \beta$ be the current prediction of y^i.

Definition (A learning algorithm)

$\beta = (0, 0, \ldots, 0)^T; i = 0; n = 0;$

`while` $n < N$

  $i = i + 1; n = n + 1;$

  `if` $(i > N)$ `then` $i = 1$

  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`

    $\beta = \beta + \boldsymbol{x}^i;\ n = 0$

  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`

    $\beta = \beta - \boldsymbol{x}^i;\ n = 0$

Example

$\beta = (-1, -2000, -80)^T$

$i = 3$

$n = 0$

$\boldsymbol{x}^3 = (1, 4000, 80)^T$

$\tilde{y}^3 = +8000000$

$y^3 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1$; $n = n + 1$;
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i$; $n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i$; $n = 0$

Example

$\boldsymbol{\beta} = (-1, -2000, -80)^T$
$i = 4$
$n = 1$
$\boldsymbol{x}^4 = (1, 5000, 40)^T$
$\tilde{y}^3 = +8000000$
$y^3 = -1$

Learning linear models

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T\boldsymbol{\beta}$ be the current prediction of y^i.

Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
`while` $n < N$
  $i = i + 1$; $n = n + 1$;
  `if` $(i > N)$ `then` $i = 1$
  `if` $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i$; $n = 0$
  `if` $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ `then`
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i$; $n = 0$

Example

$\boldsymbol{\beta} = (-1, -2000, -80)^T$
$i = 4$
$n = 1$
$\boldsymbol{x}^4 = (1, 5000, 40)^T$
$\tilde{y}^4 = -20003201$
$y^4 = +1$

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x^i_1, \ldots, x^i_p)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \beta$ be the current prediction of y^i.

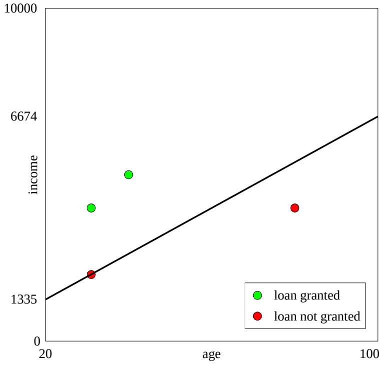
Definition (A learning algorithm)

$\boldsymbol{\beta} = (0, 0, \ldots, 0)^T$; $i = 0$; $n = 0$;
while $n < N$
  $i = i + 1$; $n = n + 1$;
  if $(i > N)$ then $i = 1$
  if $((\tilde{y}^i < 0) \wedge (y^i \geq 0))$ then
    $\boldsymbol{\beta} = \boldsymbol{\beta} + \boldsymbol{x}^i$; $n = 0$
  if $((\tilde{y}^i \geq 0) \wedge (y^i < 0))$ then
    $\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{x}^i$; $n = 0$

Example

$\boldsymbol{\beta} = (0, 3000, -40)^T$
$i = 4$
$n = 0$
$\boldsymbol{x}^4 = (1, 5000, 40)^T$
$\tilde{y}^4 = -20003201$
$y^4 = +1$

- Let $(y^i, \boldsymbol{x}^i)$ be the values of the class and the attributes of i-th object from the training dataset of N objects, where $\boldsymbol{x}^i = (1, x_1^i, \ldots, x_p^i)$.
- Let $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ be the vector of coefficients we want to learn and $\tilde{y}^i = (\boldsymbol{x}^i)^T \beta$ be the current prediction of y^i.



Example

$\beta = (-839, 1000, -66730)^T$
after 4637 iterations

Learning linear models - least squares

- The residual sum of squares is

$$RSS(\beta) = \sum_{i=1}^{N}(y^i - (\boldsymbol{x}^i)^T\beta)^2$$

Learning linear models - least squares

- The residual sum of squares is

$$RSS(\beta) = \sum_{i=1}^{N} (y^i - (\boldsymbol{x}^i)^T \beta)^2$$

- The least squares method finds β that minimizes $RSS(\beta)$.

Learning linear models - least squares

- The residual sum of squares is

$$RSS(\beta) = \sum_{i=1}^{N} (y^i - (\boldsymbol{x}^i)^T \beta)^2$$

- The least squares method finds β that minimizes $RSS(\beta)$.
- If $\boldsymbol{X}$ denotes the matrix that has vectors $\boldsymbol{x}^i, i = 1, \ldots, N$ as its rows and $\boldsymbol{y} = (y^1, \ldots, y^N)^T$ then

$$RSS(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta)$$

Learning linear models - least squares

- The residual sum of squares is

$$RSS(\beta) = \sum_{i=1}^{N} (y^i - (\boldsymbol{x}^i)^T \beta)^2$$

- The least squares method finds β that minimizes $RSS(\beta)$.
- If $\boldsymbol{X}$ denotes the matrix that has vectors $\boldsymbol{x}^i, i = 1, \ldots, N$ as its rows and $\boldsymbol{y} = (y^1, \ldots, y^N)^T$ then

$$RSS(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta)$$

- If $\boldsymbol{X}^T\boldsymbol{X}$ is nonsingular then the solution is

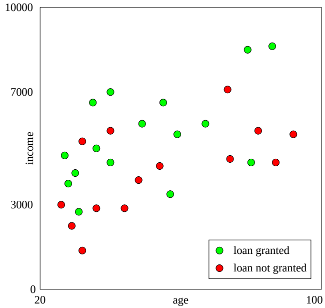
$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Learning linear models - least squares

- Note that the least squares method finds a solution also when the training dataset is not linearly separable.

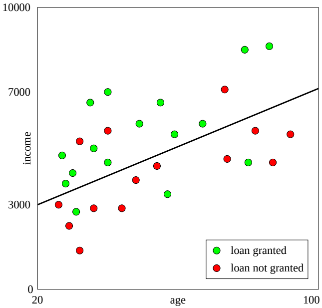
Learning linear models - least squares

- Note that the least squares method finds a solution also when the training dataset is not linearly separable.



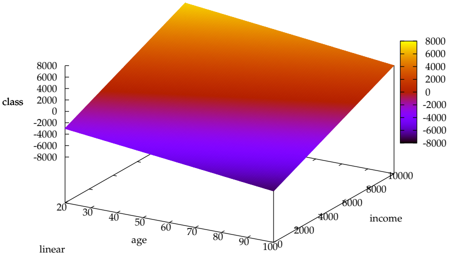
Learning linear models - least squares

- Note that the least squares method finds a solution also when the training dataset is not linearly separable.



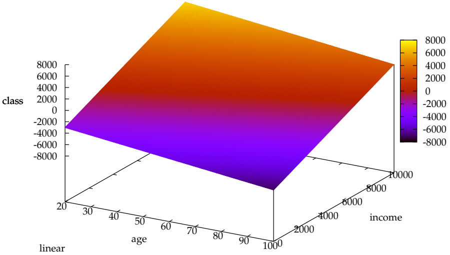
The scale of the Y variable

Linear regression
The range of Y was
$[-8000, +8000]$.

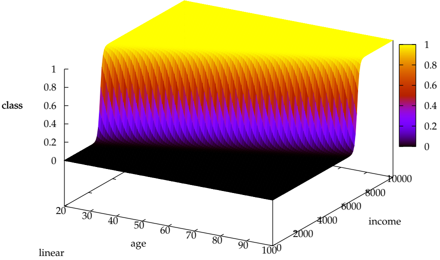


The scale of the Y variable

Linear regression
The range of Y was $[-8000, +8000]$.



Logistic regression
The range of Y is $[0, 1]$.

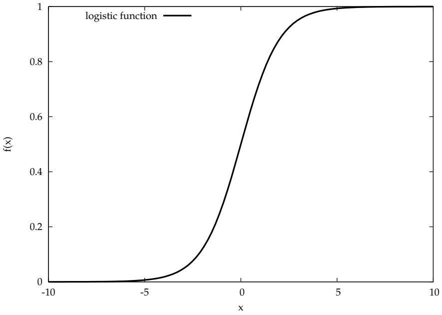


Logistic function

$$\sigma(x) = \frac{exp(x)}{1 + exp(x)}$$

Logistic function

$$\sigma(x) = \frac{exp(x)}{1 + exp(x)}$$



Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\beta^T \boldsymbol{x})$$

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) \; = \; P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) \;\; = \;\; \sigma(\beta^T \boldsymbol{x})$$

- we get the **logistic regression model** for a binary class variable.

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\beta^T \boldsymbol{x})$$

- we get the **logistic regression model** for a binary class variable.
- Note that

$$P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) = 1 - P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$$

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\beta^T \boldsymbol{x})$$

- we get the **logistic regression model** for a binary class variable.
- Note that

$$P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) = 1 - P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$$

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) = P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \sigma(\beta^T \boldsymbol{x})$$

- we get the **logistic regression model** for a binary class variable.
- Note that

$$\begin{aligned} P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) &= 1 - P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) \\ &= 1 - \sigma(\beta^T \boldsymbol{x}) \end{aligned}$$

Logistic regression

- Let $P(Y = 1|\boldsymbol{X} = \boldsymbol{x})$ be the probability that the object described by the vector of attributes $\boldsymbol{x}$ belongs to class 1.
- If we define

$$p(\beta, \boldsymbol{x}) \; = \; P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) \quad = \quad \sigma(\beta^T \boldsymbol{x})$$

- we get the **logistic regression model** for a binary class variable.
- Note that

$$\begin{aligned} P(Y = 0|\boldsymbol{X} = \boldsymbol{x}) &= 1 - P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) \\ &= 1 - \sigma(\beta^T \boldsymbol{x}) \\ &= \frac{1}{1 + exp(\beta^T \boldsymbol{x})} \end{aligned}$$

Learning logistic regression models

- Let $D = \{(y^i, \boldsymbol{x}^i), i = 1, \ldots, N\}$ be the training dataset.

Learning logistic regression models

- Let $D = \{(y^i, \boldsymbol{x}^i), i = 1, \ldots, N\}$ be the training dataset.
- The probability of data D being generated from the logistic regression model is

$$P(D|\beta) = \prod_{i=1}^{N} P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i) \cdot P(\boldsymbol{X} = \boldsymbol{x}^i)$$

Learning logistic regression models

- Let $D = \{(y^i, \boldsymbol{x}^i), i = 1, \ldots, N\}$ be the training dataset.
- The probability of data D being generated from the logistic regression model is

$$P(D|\beta) = \prod_{i=1}^{N} P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i) \cdot P(\boldsymbol{X} = \boldsymbol{x}^i)$$

- The maximum conditional likelihood estimate $\hat{\beta}$ of β maximizes the loglikelihood

$$\ell(\beta) = \sum_{i=1}^{N} \log P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i)$$

Learning logistic regression models

- Let $D = \{(y^i, \boldsymbol{x}^i), i = 1, \ldots, N\}$ be the training dataset.
- The probability of data D being generated from the logistic regression model is

$$P(D|\beta) = \prod_{i=1}^{N} P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i) \cdot P(\boldsymbol{X} = \boldsymbol{x}^i)$$

- The maximum conditional likelihood estimate $\hat{\beta}$ of β maximizes the loglikelihood

$$\ell(\beta) = \sum_{i=1}^{N} \log P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i)$$

Learning logistic regression models

- Let $D = \{(y^i, \boldsymbol{x}^i), i = 1, \ldots, N\}$ be the training dataset.
- The probability of data D being generated from the logistic regression model is

$$P(D|\beta) = \prod_{i=1}^{N} P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i) \cdot P(\boldsymbol{X} = \boldsymbol{x}^i)$$

- The maximum conditional likelihood estimate $\hat{\beta}$ of β maximizes the loglikelihood

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{N} \log P(Y = y^i | \boldsymbol{X} = \boldsymbol{x}^i) \\ &= \sum_{i=1}^{N} y^i \cdot \log p(\beta, \boldsymbol{x}^i) + (1 - y^i) \cdot \log(1 - p(\beta, \boldsymbol{x}^i))\end{aligned}$$

Learning logistic regression models

- To maximize the conditional likelihood we require partial derivatives to be zero.

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \left(y^i \boldsymbol{x}^i - \frac{\boldsymbol{x}^i \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)} \right)$$

$$= \sum_{i=1}^{N} \boldsymbol{x}^i (y^i - p(\boldsymbol{\beta}, \boldsymbol{x}^i)) = 0$$

Learning logistic regression models

- To maximize the conditional likelihood we require partial derivatives to be zero.

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N} \left(y^i \boldsymbol{x}^i - \frac{\boldsymbol{x}^i \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)} \right) \\ &= \sum_{i=1}^{N} \boldsymbol{x}^i (y^i - p(\boldsymbol{\beta}, \boldsymbol{x}^i)) \quad = \quad 0\end{aligned}$$

- which is a system of $p + 1$ nonlinear equations.

Learning logistic regression models

- To maximize the conditional likelihood we require partial derivatives to be zero.

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N} \left(y^i \boldsymbol{x}^i - \frac{\boldsymbol{x}^i \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)} \right) \\ &= \sum_{i=1}^{N} \boldsymbol{x}^i (y^i - p(\boldsymbol{\beta}, \boldsymbol{x}^i)) = 0\end{aligned}$$

- which is a system of $p + 1$ nonlinear equations.
- To solve the system we can use Newton-Raphson numerical method.

Learning logistic regression models

- To maximize the conditional likelihood we require partial derivatives to be zero.

$$
\begin{aligned}
\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N} \left(y^i \boldsymbol{x}^i - \frac{\boldsymbol{x}^i \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)} \right) \\
&= \sum_{i=1}^{N} \boldsymbol{x}^i (y^i - p(\boldsymbol{\beta}, \boldsymbol{x}^i)) \quad = \quad 0
\end{aligned}
$$

- which is a system of $p + 1$ nonlinear equations.
- To solve the system we can use Newton-Raphson numerical method.
- Let $\boldsymbol{\beta}^{old}$ be the value of $\boldsymbol{\beta}$ from previous iteration.

Learning logistic regression models

- To maximize the conditional likelihood we require partial derivatives to be zero.

$$\begin{aligned}\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N} \left(y^i \boldsymbol{x}^i - \frac{\boldsymbol{x}^i \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)}{1 + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^i)} \right) \\ &= \sum_{i=1}^{N} \boldsymbol{x}^i (y^i - p(\boldsymbol{\beta}, \boldsymbol{x}^i)) \;=\; 0\end{aligned}$$

- which is a system of $p+1$ nonlinear equations.
- To solve the system we can use Newton-Raphson numerical method.
- Let $\boldsymbol{\beta}^{old}$ be the value of $\boldsymbol{\beta}$ from previous iteration.
- Then the new value

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \left(\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$$

Learning logistic regression models

We use matrix notation:

Learning logistic regression models

We use matrix notation:

$$\boldsymbol{y} = (y_1, \ldots, y_N)^T$$

Learning logistic regression models

We use matrix notation:

$$\boldsymbol{y} = (y_1, \ldots, y_N)^T$$

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_p^1 \\ 1 & x_1^2 & \ldots & x_p^2 \\ \ldots & & & \\ 1 & x_1^N & \ldots & x_p^N \end{pmatrix}$$

Learning logistic regression models

We use matrix notation:

$$\boldsymbol{y} = (y_1, \ldots, y_N)^T$$

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_p^1 \\ 1 & x_1^2 & \ldots & x_p^2 \\ \ldots & & & \\ 1 & x_1^N & \ldots & x_p^N \end{pmatrix}$$

$$\boldsymbol{p} = (p(\beta^{old}, \boldsymbol{x}^1), \ldots, p(\beta^{old}, \boldsymbol{x}_N))^T$$

Learning logistic regression models

We use matrix notation:

$$\boldsymbol{y} = (y_1, \ldots, y_N)^T$$

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_p^1 \\ 1 & x_1^2 & \ldots & x_p^2 \\ \ldots & & & \\ 1 & x_1^N & \ldots & x_p^N \end{pmatrix}$$

$$\boldsymbol{p} = (p(\beta^{old}, \boldsymbol{x}^1), \ldots, p(\beta^{old}, \boldsymbol{x}_N))^T$$

$$\boldsymbol{W} = \begin{pmatrix} p(\beta^{old}, \boldsymbol{x}^1)(1 - p(\beta^{old}, \boldsymbol{x}^1)) & 0 & \ldots & 0 \\ 0 & p(\beta^{old}, \boldsymbol{x}^2)(1 - p(\beta^{old}, \boldsymbol{x}^2)) & 0 & \ldots \\ \ldots & & & \\ 0 & \ldots & 0 & p(\beta^{old}, \boldsymbol{x}^N)(1 - p(\beta^{old}, \boldsymbol{x}^N)) \end{pmatrix}$$

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- One step of the Newton-Raphson algorithm is

$$\beta^{new} = \beta^{old} - (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{p})$$

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- One step of the Newton-Raphson algorithm is

$$\beta^{new} = \beta^{old} - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{p})$$

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- One step of the Newton-Raphson algorithm is

$$\begin{aligned}\beta^{new} &= \beta^{old} - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{p}) \\ &= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p}))\end{aligned}$$

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- One step of the Newton-Raphson algorithm is

$$\begin{aligned}
\beta^{new} &= \beta^{old} - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{p}) \\
&= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})) \\
&= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{z}
\end{aligned}$$

- This formulation of one step of the Newton-Raphson algorithm corresponds to one step of **weighted least squares** since one step of the algorithm is

$$\beta^{new} = \arg\min_{\beta}(\boldsymbol{z} - \boldsymbol{X}\beta)^T\boldsymbol{W}(\boldsymbol{z} - \boldsymbol{X}\beta)$$

Learning logistic regression models

- Define $\boldsymbol{z} = \boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- One step of the Newton-Raphson algorithm is

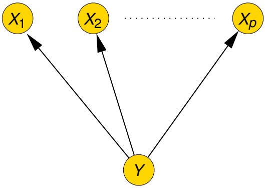
$$\begin{aligned}
\beta^{new} &= \beta^{old} - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{p}) \\
&= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}(\boldsymbol{X}\beta^{old} + \boldsymbol{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})) \\
&= (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{z}
\end{aligned}$$

- This formulation of one step of the Newton-Raphson algorithm corresponds to one step of **weighted least squares** since one step of the algorithm is

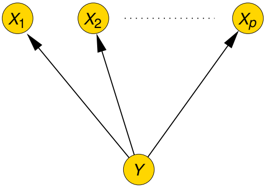
$$\beta^{new} = \arg\min_{\beta}(\boldsymbol{z} - \boldsymbol{X}\beta)^T\boldsymbol{W}(\boldsymbol{z} - \boldsymbol{X}\beta)$$

- The whole algorithm thus corresponds to **iteratively reweighted least squares (IRLS)**.

Naïve Bayes classifier

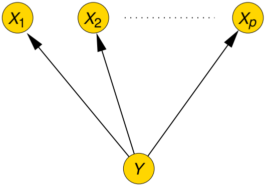


Naïve Bayes classifier



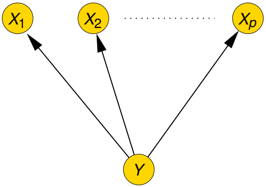
$$P(Y|X_1,\ldots,X_p) = \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)}$$

Naïve Bayes classifier



$$P(Y|X_1,\ldots,X_p) = \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} = P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}$$

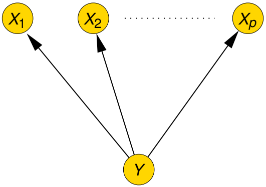
Naïve Bayes classifier



$$\begin{aligned} P(Y|X_1,\ldots,X_p) &= \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} \\ &= P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)} \end{aligned}$$

The learning algorithm is just the computation of relative frequencies.

Naïve Bayes classifier

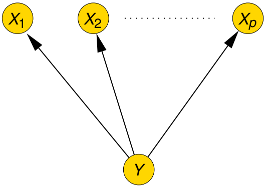


$$P(Y|X_1,\ldots,X_p) = \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} = P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}$$

The learning algorithm is just the computation of relative frequencies.

Let $\delta(y,x) = 1$ if $y = x$ and 0 otherwise.

Naïve Bayes classifier

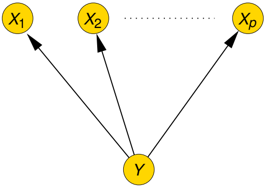


$$P(Y|X_1,\ldots,X_p) = \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} = P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}$$

The learning algorithm is just the computation of relative frequencies. Let $\delta(y,x) = 1$ if $y = x$ and 0 otherwise.

$$P(Y = y) = \frac{1}{N}\sum_{i=1}^{N}\delta(y^i, y)$$

Naïve Bayes classifier

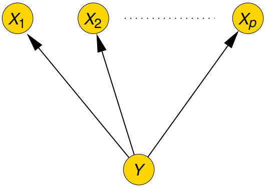


$$
\begin{aligned}
P(Y|X_1,\ldots,X_p) &= \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} \\
&= P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}
\end{aligned}
$$

The learning algorithm is just the computation of relative frequencies.
Let $\delta(y,x)=1$ if $y=x$ and 0 otherwise.

$$
P(Y=y) = \frac{1}{N}\sum_{i=1}^{N}\delta(y^i,y) \qquad P(X_j=x_j) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_j^i,x)
$$

Naïve Bayes classifier



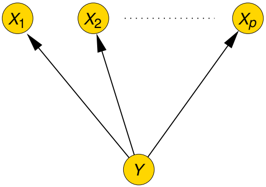
$$
\begin{aligned}
P(Y|X_1,\ldots,X_p) &= \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} \\
&= P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}
\end{aligned}
$$

The learning algorithm is just the computation of relative frequencies.
Let $\delta(y,x) = 1$ if $y = x$ and 0 otherwise.

$$
P(Y=y) = \frac{1}{N}\sum_{i=1}^{N}\delta(y^i,y) \qquad P(X_j=x_j) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_j^i,x)
$$

$$
P(X_j=x_j, Y=y) = \frac{1}{N}\sum_{i=1}^{N}\delta(y^i,y)\cdot\delta(x_j^i,x)
$$

Naïve Bayes classifier



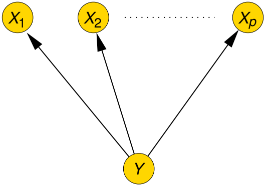
$$
\begin{aligned}
P(Y|X_1,\ldots,X_p) &= \frac{P(Y,X_1,\ldots,X_p)}{P(X_1,\ldots,X_p)} \\
&= P(Y)\prod_{j=1}^{p}\frac{P(X_j|Y)}{P(X_j)}
\end{aligned}
$$

The learning algorithm is just the computation of relative frequencies.
Let $\delta(y,x) = 1$ if $y = x$ and 0 otherwise.

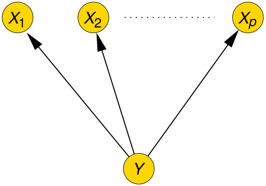
$$
P(Y=y) = \frac{1}{N}\sum_{i=1}^{N}\delta(y^i,y) \qquad P(X_j=x_j) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_j^i,x)
$$

$$
\begin{aligned}
P(X_j=x_j, Y=y) &= \frac{1}{N}\sum_{i=1}^{N}\delta(y^i,y)\cdot\delta(x_j^i,x) \\
P(X_j=x_j|Y=y) &= \frac{P(X_j=x_j, Y=y)}{P(Y=y)}
\end{aligned}
$$

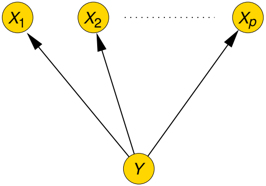
Independence vs. Dependence of Attributes



Independence vs. Dependence of Attributes

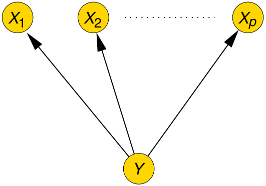


What independence relations are assumed in the Naïve Bayes classifier?



What independence relations are assumed in the Naïve Bayes classifier?
$(X_j \perp X_k | Y)$ for $j, k \in \{1, \ldots, p\}, j \neq k$.
It seems unrealistic in many applications.

Independence vs. Dependence of Attributes

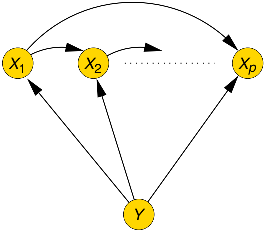


What independence relations are assumed in the Naïve Bayes classifier?
$(X_j \perp X_k | Y)$ for $j, k \in \{1, \ldots, p\}, j \neq k$.
It seems unrealistic in many applications.

Example

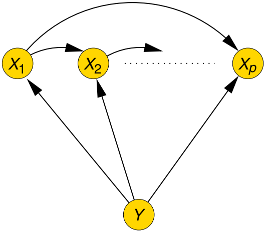
Consider a classifier for assessing the risk in loan applications: it seems counterintuitive to ignore the correlations between age, education level, and income.

Independence vs. Dependence of Attributes



In Tree Augmented Naïve Bayes (TAN) classifier the correlation are represented by a tree structure over the attributes, where all edges are outwards from a selected root node.

Independence vs. Dependence of Attributes



In Tree Augmented Naïve Bayes (TAN) classifier the correlation are represented by a tree structure over the attributes, where all edges are outwards from a selected root node.

It is no longer assumed that $X_j \perp X_k | Y$ for all $j \neq k \in \{1, \ldots, p\}$.

Learning Tree Augmented Naïve Bayes

We will abbreviate $P(\boldsymbol{X} = \boldsymbol{x})$ as $P(\boldsymbol{x})$.

Learning Tree Augmented Naïve Bayes

We will abbreviate $P(\boldsymbol{X} = \boldsymbol{x})$ as $P(\boldsymbol{x})$.

Definition (Conditional mutual information)

$$I_P(X_j, X_k | Y) = \sum_{x_j, x_k, y} P(x_j, x_k, y) \log \frac{P(x_j, x_k | y)}{P(x_j | y) \cdot P(x_k | y)}$$

Learning Tree Augmented Naïve Bayes

Definition (TAN Learning Algorithm)

Learning Tree Augmented Naïve Bayes

Definition (TAN Learning Algorithm)

- Compute $I_P(X_j, X_k | Y)$ between each pair of attributes $X_j, X_k, j \neq k$.

Learning Tree Augmented Naïve Bayes

Definition (TAN Learning Algorithm)

- Compute $I_P(X_j, X_k | Y)$ between each pair of attributes $X_j, X_k, j \neq k$.
- Build a complete undirected graph in which the nodes are the attributes $X_1, \ldots, X_n$. Annotate the weight of an edge connecting X_j to X_k by $I_P(X_j, X_k | Y)$.

Learning Tree Augmented Naïve Bayes

Definition (TAN Learning Algorithm)

- Compute $I_P(X_j, X_k | Y)$ between each pair of attributes $X_j, X_k, j \neq k$.
- Build a complete undirected graph in which the nodes are the attributes $X_1, \ldots, X_n$. Annotate the weight of an edge connecting X_j to X_k by $I_P(X_j, X_k | Y)$.
- Build a maximum weighted spanning tree.

Learning Tree Augmented Naïve Bayes

Definition (TAN Learning Algorithm)

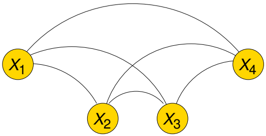
- Compute $I_P(X_j, X_k | Y)$ between each pair of attributes $X_j, X_k, j \neq k$.
- Build a complete undirected graph in which the nodes are the attributes $X_1, \ldots, X_n$. Annotate the weight of an edge connecting X_j to X_k by $I_P(X_j, X_k | Y)$.
- Build a maximum weighted spanning tree.
- Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

Learning Tree Augmented Naïve Bayes

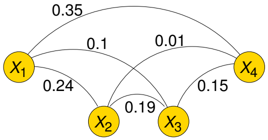
Definition (TAN Learning Algorithm)

- Compute $I_P(X_j, X_k | Y)$ between each pair of attributes $X_j, X_k, j \neq k$.
- Build a complete undirected graph in which the nodes are the attributes $X_1, \ldots, X_n$. Annotate the weight of an edge connecting X_j to X_k by $I_P(X_j, X_k | Y)$.
- Build a maximum weighted spanning tree.
- Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
- Construct a TAN model by adding a vertex labeled by Y and adding an edge from Y to each X_i.

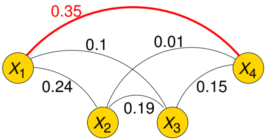
Learning Tree Augmented Naïve Bayes



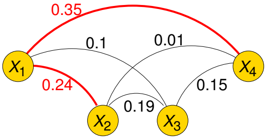
Learning Tree Augmented Naïve Bayes



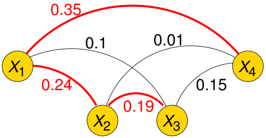
Learning Tree Augmented Naïve Bayes



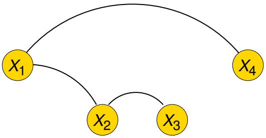
Learning Tree Augmented Naïve Bayes



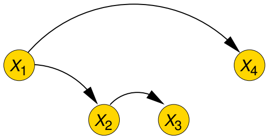
Learning Tree Augmented Naïve Bayes



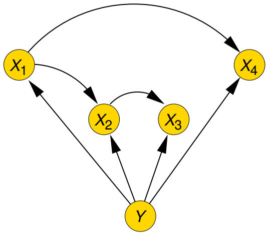
Learning Tree Augmented Naïve Bayes



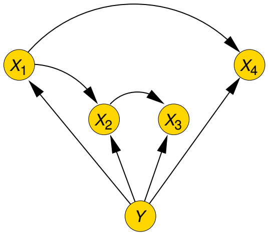
Learning Tree Augmented Naïve Bayes



Learning Tree Augmented Naïve Bayes



Learning Tree Augmented Naïve Bayes



Theorem

Let D be a collection of N instances of $Y, X_1, \ldots, X_p$. The TAN Learning Algorithm builds a TAN that maximizes loglikelihood given data D and has time complexity $O(p^2 \cdot N)$.

Other classification methods

- Decision trees and rule-based systems

Other classification methods

- Decision trees and rule-based systems
- Support vector machines

Other classification methods

- Decision trees and rule-based systems
- Support vector machines
- Neural networks

Other classification methods

- Decision trees and rule-based systems
- Support vector machines
- Neural networks
- k-nearest neighbor

Other classification methods

- Decision trees and rule-based systems
- Support vector machines
- Neural networks
- *k*-nearest neighbor
- Unrestricted Bayesian networks

Other classification methods

- Decision trees and rule-based systems
- Support vector machines
- Neural networks
- k-nearest neighbor
- Unrestricted Bayesian networks
- Bayesian networks with a local structure (e.g., noisy-or)

References

Textbooks

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.
- Ian H. Witten and Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005.

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.
- Ian H. Witten and Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005.

Journal article

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.
- Ian H. Witten and Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005.

Journal article

- Nir Friedman, Dan Geiger, and Moises Goldszmidt: Bayesian Network Classifiers. Machine Learning, Vol. 29, pages 131–163, 1997.

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.
- Ian H. Witten and Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005.

Journal article

- Nir Friedman, Dan Geiger, and Moises Goldszmidt: Bayesian Network Classifiers. Machine Learning, Vol. 29, pages 131–163, 1997.

Software

References

Textbooks

- T. Hastie, R. Tibshirani, and J. Friedman: The elements of statistical learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2003.
- Ian H. Witten and Eibe Frank: Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Morgan Kaufmann, 2005.

Journal article

- Nir Friedman, Dan Geiger, and Moises Goldszmidt: Bayesian Network Classifiers. Machine Learning, Vol. 29, pages 131–163, 1997.

Software

- Weka - a collection of machine learning algorithms for data mining tasks. `http://www.cs.waikato.ac.nz/ml/weka/`