AI6102: Machine Learning Methodologies & Applications

L7: Bayesian Classifiers & K-NN Classifiers

Sinno Jialin Pan

Nanyang Technological University, Singapore

Homepage: http://www.ntu.edu.sg/home/sinnopan

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

- Bayesian Classifiers
  - Naïve Bayes classifiers
- K Nearest neighbors (K-NN) classifiers
  - A lazy classifier

Supervised Learning: Recall

In mathematics

- Given: a set of $\{\boldsymbol{x}_i, y_i\}$ for $i = 1, \ldots, N$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{im}]$ is m-dimensional vector of numerical values, and y_i is a scalar
- Aim to learn a mapping $f: \boldsymbol{x} \rightarrow y$ by requiring $f(\boldsymbol{x}_i) = y_i$
- The learned mapping f is expected to make precise predictions on any unseen $\boldsymbol{x}^*$ as $f(\boldsymbol{x}^*)$

In Probability Point of View

- The mapping $f: \boldsymbol{x} \to y$ can be considered as a conditional probability $P(y|\boldsymbol{x})$
- Given a test data instance $\boldsymbol{x}^*$

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y = c|\boldsymbol{x}^*), c \in \{0, \dots, C-1\}$$

- In logistic regression, the conditional probabilities of different classes are assumed to be expressed as specific forms in terms of parameters $\boldsymbol{w}$, e.g., for binary classification (0 or 1)

$$P(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})} \qquad P(y = 0|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}^T\boldsymbol{x})}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

- Today, we introduce another way to estimate $P(y|\boldsymbol{x})$

Probability Review

- Let A be a random variable (a feature / a label in machine learning)
- Marginal probability

$0 \leq P(A = a) \leq 1$

$$P(A = a)$$

refers to the probability that variable $A = a$

$$\sum_{a_i} P(A = a_i) = 1$$

Probability Review (cont.)

- Let A and B be a pair of random variables (features/labels in machine learning).
- Their joint probability

$$P(A = a, B = b)$$

refers to the probability that variable $A = a$, and at the same time variable $B = b$

Probability Review (cont.)

- Conditional probability

$$P(B = b|A = a)$$

refers to the probability that variable B will take on the value b, given that the variable A is observed to have the value a

$$\sum_{b_i} P(B = b_i|A = a) = 1$$

Sum Rule

- The connection between joint probability of A and B and marginal probability of A:

$$P(A=a)=\sum_{b_i}P(A=a,B=b_i) \quad \textbf{OR} \quad P(A)=\sum_{B}P(A,B)$$

$$P(A=a)=\sum_{c_j}\sum_{b_i}P(A=a,B=b_i,C=c_j)$$

OR

$$P(A)=\sum_{C}\sum_{B}P(A,B,C)$$

Product Rule

- The connections between marginal, joint and conditional probabilities of A and B:

$$P(A = a, B = b) = P(B = b|A = a) \times P(A = a)$$

$$= P(A = a|B = b) \times P(B = b)$$

OR

$$P(A, B) = P(B|A) \times P(A)$$

$$= P(A|B) \times P(B)$$

Bayes Rule / Bayes Theorem

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- The 1st and the 2nd equations are both based on product rule

$$P(A,B) = P(A|B) \times P(B) = P(B|A) \times P(A)$$

- Can be generalized to the case when **A** and **B** are a set of variables

$$P\left(A_1 \ldots A_k \middle| B_1 \ldots B_p\right) = \frac{P\left(B_1 \ldots B_p \middle| A_1 \ldots A_k\right) P(A_1 \ldots A_k)}{P(B_1 \ldots B_p)}$$

Bayesian Classifiers

- To estimate $P(y|\boldsymbol{x})$ from the training set $\{\boldsymbol{x}_i, y_i\}$ $i = 1, \dots, N$, we can use the Bayes Rule

$$P(y|\boldsymbol{x}) = \frac{P(y, \boldsymbol{x})}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

- Recall that to make a prediction on $\boldsymbol{x}^*$

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y = c|\boldsymbol{x}^*), c \in \{0, \dots, C-1\}$$

$$= \arg\max_c \frac{P(\boldsymbol{x}^*|y = c)P(y = c)}{P(\boldsymbol{x}^*)}$$

$P(\boldsymbol{x}^*)$ is a constant w.r.t. different c

$$= \arg\max_c P(\boldsymbol{x}^*|y = c)P(y = c)$$

An Example

- Suppose we aim to predict the class label (Repay = Yes or No) of the following data instance

Fixed Assets	Occupation	Income	Repay
Yes	Manager	125K	?

- That is we need to compute

$$P(\text{Rapy}=\text{Yes} \mid \text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K})$$

$$= \frac{P(\text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K} \mid \text{Rapy}=\text{Yes})P(\text{Rapy}=\text{Yes})}{\cancel{P(\text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K})}}$$

v.s.

$$P(\text{Rapy}=\text{No} \mid \text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K})$$

$$= \frac{P(\text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K} \mid \text{Rapy}=\text{No})P(\text{Rapy}=\text{No})}{\cancel{P(\text{Assets}=\text{Yes},\text{Occ.}=\text{Manager},\text{Income}=125\text{K})}}$$

Bayesian Classifiers (cont.)

$$P(y|\boldsymbol{x}) = \frac{P(y, \boldsymbol{x})}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y = c|\boldsymbol{x}^*)\,, c \in \{0, \dots, C-1\}$$

$$= \arg\max_c \frac{P(\boldsymbol{x}^*|y = c)P(y = c)}{P(\boldsymbol{x}^*)}$$

$P(\boldsymbol{x}^*)$ is a constant w.r.t. different c

$$= \arg\max_c \boxed{P(\boldsymbol{x}^*|y = c)P(y = c)}$$

Estimate these two types of probabilities from training data

Bayesian Classifiers (cont.)

$$P(y|\boldsymbol{x}) \propto \boxed{P(\boldsymbol{x}|y)}\boxed{P(y)}$$

Class probabilities from training data, easy to estimate

In general, difficult to estimate as all the possible combinations need to be considered in training

- Consider the risk estimation task

$$P(\text{Assets=Yes,Occ.=Manager,}\boxed{\text{Income=125K}} \mid \text{Rapy=Yes})$$

$$P(\text{Assets=No,Occ.=Manager,Income=125K} \mid \text{Rapy=Yes})$$

$$P(\text{Assets=Yes,Occ.=Engineer,Income=125K} \mid \text{Rapy=Yes})$$

$$P(\text{Assets=Yes,Occ.=Lawyer,Income=125K} \mid \text{Rapy=Yes})$$

...

Bayesian Classifiers (cont.)

- In theory, the estimation of $P(\boldsymbol{x}|y)$ is computationally expensive
  - Need to consider all possible value combination of $\boldsymbol{x}$ and y
  - How to make the estimation of $P(\boldsymbol{x}|y)$ computationally tractable?
- Two implementations of Bayesian classification methods
  - Naïve Bayes classifier
    - Based on a strong conditional independence assumption
  - Bayesian belief network
    - Based on a graph of dependence among variables

Not covered

Naïve Bayes Classifier

- Assume that the features are <u>conditionally independent</u> given the class label:

$$P(\boldsymbol{x}|y=c)=\prod_{i=1}^{d}P(x_i|y=c) \quad \text{where } \boldsymbol{x}=[x_1,x_2,\dots,x_d]$$

$$P(x_1,x_2,..,x_d|y=c)=\prod_{i=1}^{d}\boxed{P(x_i|y=c)}$$

Only need to different combinations of x_i and y, no need to consider all combinations of $[x_1,\dots,x_d]$ and y

For example:

P(Assets=Yes,Occ.=Manager,Income=125K | Rapy=Yes)

= P(Assets=Yes | Rapy=Yes)P(Occ.=Manager | Rapy=Yes)P(Income=125K | Rapy=Yes)

Independence

- Let A and B be two random variables
- A is said to be independent of B, if the following condition holds:

$$P(A|B) = P(A)$$

$$P(A,B) = P(A|B) \times P(B) = P(A) \times P(B)$$

- This can be generalized to the setting where **A** and **B** are two sets of random variables
- The variables in **A** are said to be independent of **B**, if the following condition holds:

$$P(\mathbf{A},\mathbf{B}) = P(\mathbf{A}|\mathbf{B}) \times P(\mathbf{B}) = P(\mathbf{A}) \times P(\mathbf{B})$$

Conditional Independence

- Let **A**, **B**, and **C** be three <u>sets</u> of random variables
- The variables in **A** are said to be <u>conditionally independent</u> of **B**, given **C**, if the following condition holds:

$$P(\mathbf{A}|\mathbf{B},\mathbf{C}) = P(\mathbf{A}|\mathbf{C})$$

$$\Downarrow$$

$$P(\boldsymbol{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

Conditional Independence (cont.)

- The conditional independence between **A** and **B** given **C** can also be written as follows

$$P(\mathbf{A},\mathbf{B}|\mathbf{C}) = \frac{P(\mathbf{A},\mathbf{B},\mathbf{C})}{P(\mathbf{C})}$$

Product rule: $P(\mathbf{A},\mathbf{B}|\mathbf{C})P(\mathbf{C}) = P(\mathbf{A},\mathbf{B},\mathbf{C})$

$$= \frac{P(\mathbf{A},\mathbf{B},\mathbf{C})}{\boxed{P(\mathbf{B},\mathbf{C})}} \times \frac{\boxed{P(\mathbf{B},\mathbf{C})}}{P(\mathbf{C})}$$

Product rule:
$P(\mathbf{A}|\mathbf{B},\mathbf{C})P(\mathbf{B},\mathbf{C}) = P(\mathbf{A},\mathbf{B},\mathbf{C})$
$P(\mathbf{B}|\mathbf{C})P(\mathbf{C}) = P(\mathbf{B},\mathbf{C})$

$$= P(\mathbf{A}|\mathbf{B},\mathbf{C}) \times P(\mathbf{B}|\mathbf{C})$$

Conditional independence:
$P(\mathbf{A}|\mathbf{B},\mathbf{C}) = P(\mathbf{A}|\mathbf{C})$

$$= P(\mathbf{A}|\mathbf{C}) \times P(\mathbf{B}|\mathbf{C})$$

Naïve Bayes Classifier (cont.)

- The set of varilables **A** and **B** are said to be independent given **C** if $P(\mathbf{A}, \mathbf{B}|\mathbf{C}) = P(\mathbf{A}|\mathbf{C}) \times P(\mathbf{B}|\mathbf{C})$
- Recall that naïve Bayes classifier assumes that the features are conditionally independent given the class label

$$\mathbf{A} = \{x_1, \dots, x_{d-1}\}, \mathbf{B} = \{x_d\}, \mathbf{C} = \{y = c\}$$

$$P(x_1, x_2, .., x_d|y = c) = \boxed{P(x_1, \dots, x_{d-1}|y = c)} P(x_d|y = c)$$

$$\Downarrow$$

$$P(x_1, .., x_{d-1}|y = c) = \boxed{P(x_1, \dots, x_{d-2}|y = c)} P(x_{d-1}|y = c)$$

$$\Downarrow$$

$$\dots$$

$$P(x_1, x_2, .., x_d|y = c)$$
$$= P(x_1|y = c)P(x_2|y = c) \dots P(x_d|y = c) = \prod_{i=1}^{d} P(x_i|y = c)$$

Naïve Bayes Classifier (cont.)

- For any test data instance $\boldsymbol{x}^*$

$$c^* = \arg\max_c P(y = c|\boldsymbol{x}^*)$$

$$= \arg\max_c \frac{P(\boldsymbol{x}^*|y = c)P(y = c)}{P(\boldsymbol{x}^*)}$$

$$= \arg\max_c P(\boldsymbol{x}^*|y = c)P(y = c)$$

$$= \arg\max_c P(y = c) \prod_{i=1}^{d} P(x_i^*|y = c)$$

- In training, we need to estimate $P(y)$ for different classes, and for each class c and feature x_i, $P(x_i|y = c)$ for different possible values of x_i

Credit Risk Estimation

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

Testing

ID	Assets	Occupation	Income	Repay
11	No	Engineer	85K	?

Training

$P(x_i|y = c)$

P(Assets=Yes | Repay=No)
P(Assets=No | Repay=No)
P(Assets=Yes | Repay=Yes)
P(Assets=No | Repay=Yes)

P(Occ.=Manager | Repay=No)
P(Occ.=Engineer | Repay=No)
P(Occ.=Lawyer | Repay=No)
P(Occ.=Manager | Repay=Yes)
P(Occ.=Engineer | Repay=Yes)
P(Occ.=Lawyer | Repay=Yes)

P(Income=v | Repay=Yes)
P(Income=v | Repay=No)
where $v \geq 0$

$P(y = c)$

P(Repay=Yes)
P(Repay=No)

P(No)P(Assets=No | No)P(Occu.=Engineer | No)P(Income=85K | No)

v.s.

P(Yes)P(Assets=No | Yes)P(Occu.=Engineer | Yes)P(Income=85K | Yes)

Margin Probability of Class

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

Number of data instances of class *c*

$$P(y=c) = \frac{\boxed{|y=c|}}{\boxed{N}}$$

Total number of training data instances

$$P(\text{Repay=Yes}) = \frac{7}{10}$$

$$P(\text{Repay=No}) = \frac{3}{10}$$

Conditional Probability on Discrete Features

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

$$P(\text{Assets} = \text{Yes} \mid \text{Repay} = \text{Yes})$$

$$= \frac{\#(\text{Assets} = \text{Yes} \wedge \text{Repay} = \text{Yes})}{\#(\text{Repay} = \text{Yes})} = \frac{3}{7}$$

$$P(\text{Occ.} = \text{Manager} \mid \text{Repay} = \text{No})$$

$$= \frac{\#(\text{Occ.} = \text{Manager} \wedge \text{Repay} = \text{No})}{\#(\text{Repay} = \text{No})} = \frac{2}{3}$$

Number of data instances of class c, whose values of the i-th feature are z

$$P(x_i = z | y = c) = \frac{\boxed{|(x_i = z) \wedge (y = c)|}}{\boxed{|y = c|}}$$

Value of the i-th feature equals to z

Number of data instances of class c

Conditional Probability on Continuous Features

- Assume the values of a specific feature x_i given a specific class c follow a Guassian distribution, i.e., $P(x_i|y = c)$ is a Guassian distribution

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

  - Use training data of class c to estimate parameters of the Gaussian distribution, i.e., mean μ and variance σ^2
  - Once the parameters are estimated, the Guassian distribution is known, and we can use it to compute conditional probability
- Note: more methods will be introduced when introducing density estimation

Conditional Probability on Continuous Features (cont.)

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

{Income, Repay=Yes}, Gaussian distribution:

$$P(\text{Inc.}|\text{Yes}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\text{Inc.}-\mu)^2}{2\sigma^2}}$$

μ and σ^2 are the mean and variance of the income of the data instances whose labels are Yes (Repay=Yes)

$$\mu_x = \frac{1}{N}\sum_{k=1}^{N} x_k \qquad \sigma_x = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(x_i - \mu_x)^2}$$

$$\mu_{\{\text{inc., Yes}\}} = 110$$

$$\sigma^2_{\{\text{Inc., Yes}\}} = 2975$$

$$\sigma_{\{\text{Inc., Yes}\}} = 54.54$$

$$P(\text{Inc.}|\text{Yes}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(\text{Inc.}-110)^2}{2\times 2975}}$$

Conditional Probability on Continuous Features (cont.)

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

{Income, Repay=No}, Gaussian distribution:

$$P(\text{Inc.}|\text{No}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\text{Inc.}-\mu)^2}{2\sigma^2}}$$

μ and σ^2 are the mean and variance of the income of the data instances whose labels are No (Repay=No)

$$\mu_{\{\text{inc., No}\}} = 90$$

$$\sigma^2_{\{\text{Inc.,No}\}} = 25$$

$$\sigma_{\{\text{Inc.,No}\}} = 5$$

$$P(\text{Inc.}|\text{No}) = \frac{1}{\sqrt{2\pi} \times 5} e^{-\frac{(\text{Inc.}-90)^2}{2\times 25}}$$

Conditional Probability on Continuous Features (cont.)

$$P(\text{Inc.}|\text{No}) = \frac{1}{\sqrt{2\pi} \times 5} e^{-\frac{(\text{Inc.}-90)^2}{2\times 25}}$$

$$P(\text{Inc.}|\text{Yes}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(\text{Inc.}-110)^2}{2\times 2975}}$$

ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

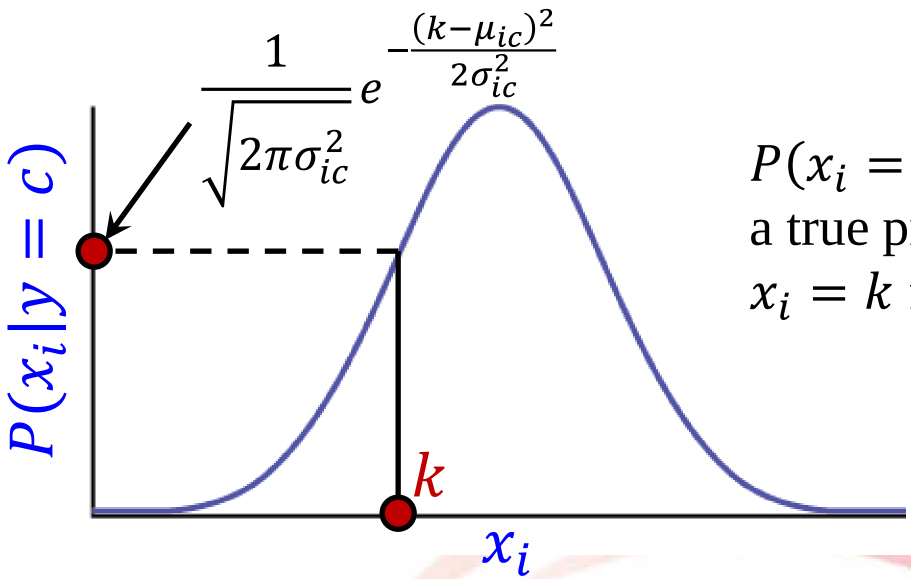
$$P(\text{Income=85}|\text{No}) = \frac{1}{\sqrt{2\pi} \times 5} e^{-\frac{(85-90)^2}{2\times 25}} = 0.048$$

$$P(\text{Income=85}|\text{Yes}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(85-110)^2}{2\times 2975}} = 0.007$$

Additional Notes

Probability density function $P(x_i|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}}$

- The probability density function is continuous, the probability is defined as the area under the curve of the probability density function



$P(x_i = k|y = c)$ is not a true probability that $x_i = k$ for class c

Additional Notes (cont.)

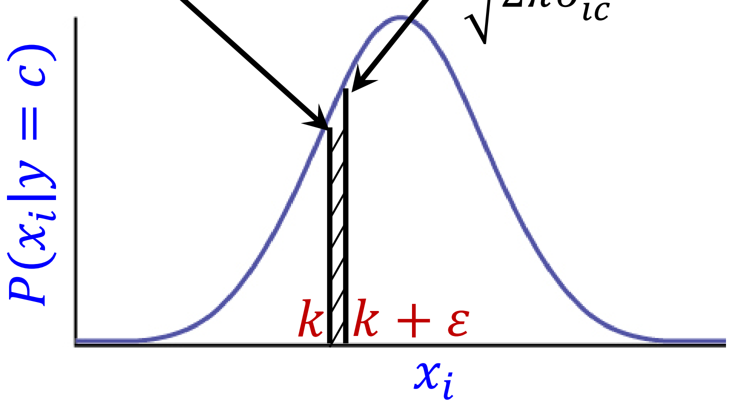
- Instead, we should compute

Small positive constant

$$P(k \le x_i \le k + \varepsilon | y = c) = \int_{k}^{k+\varepsilon} \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}} dx_i$$

$$\frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

$$\frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k+\varepsilon-\mu_{ic})^2}{2\sigma_{ic}^2}}$$



Assume $\frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k+\varepsilon-\mu_{ic})^2}{2\sigma_{ic}^2}}$

$$\approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \times \varepsilon$$

Additional Notes (cont.)

- Since ε appears as a constant multiplicative factor for each class, it cancels out when comparing posterior probabilities $P(y = c|\boldsymbol{x})$ for each class
- E.g., consider binary classification and instance is represented by a single feature of continues values

$$P(y = 0|x = k) \quad \textit{vs.} \quad P(y = 1|x = k)$$

$$\Downarrow$$

$$P(x = k|y = 0)P(y = 0) \quad \textit{vs.} \quad P(x = k|y = 1)P(y = 1)$$

$$\Downarrow$$

$$\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(k-\mu_0)^2}{2\sigma_0^2}} \times \cancel{\varepsilon} \times P(y = 0) \quad \textit{vs.} \quad \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(k-\mu_1)^2}{2\sigma_1^2}} \times \cancel{\varepsilon} \times P(y = 1)$$

Additional Notes (cont.)

- Therefore, we can still apply the following equation to approximate the probability of $x_i = k$ for class c

$$P(x_i = k | y = c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

Naïve Bayes Classifier: An Example

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

$$\arg\max_c P(y=c)\prod_{i=1}^{d} P(x_i^*|y=c)$$

$P(\text{Assets=Yes} \mid \text{Repay=No}) = 0/3$
$P(\text{Assets=No} \mid \text{Repay=No}) = 3/3$
$P(\text{Assets=Yes} \mid \text{Repay=Yes}) = 3/7$
$P(\text{Assets=No} \mid \text{Repay=Yes}) = 4/7$

$P(\text{Occ.=Manager} \mid \text{Repay=No}) = 2/3$
$P(\text{Occ.=Engineer} \mid \text{Repay=No}) = 0/3$
$P(\text{Occ.=Lawyer} \mid \text{Repay=No}) = 1/3$
$P(\text{Occ.=Manager} \mid \text{Repay=Yes}) = 2/7$
$P(\text{Occ.=Engineer} \mid \text{Repay=Yes}) = 4/7$
$P(\text{Occ.=Lawyer} \mid \text{Repay=Yes}) = 1/7$

$P(\text{Income} \mid \text{Repay=Yes})$
$\mu_{\{\text{inc., Yes}\}} = 110,\ \sigma^2_{\{\text{Inc.,Yes}\}} = 2975$
$P(\text{Income} \mid \text{Repay=No})$
$\mu_{\{\text{inc., No}\}} = 90,\ \sigma^2_{\{\text{Inc.,No}\}} = 25$

$P(\text{Repay=Yes}) = 7/10$
$P(\text{Repay=No}) = 3/10$

ID	Fixed Assets	Occupation	Income	Repay
11	No	Engineer	85k	?

$P(\text{Assets=Yes} \mid \text{Repay=No}) = 0/3$
$P(\text{Assets=No} \mid \text{Repay=No}) = 3/3$
$P(\text{Assets=Yes} \mid \text{Repay=Yes}) = 3/7$
$P(\text{Assets=No} \mid \text{Repay=Yes}) = 4/7$

$P(\text{Occ.=Manager} \mid \text{Repay=No}) = 2/3$
$\boxed{P(\text{Occ.=Engineer} \mid \text{Repay=No}) = 0/3}$
$P(\text{Occ.=Lawyer} \mid \text{Repay=No}) = 1/3$
$P(\text{Occ.=Manager} \mid \text{Repay=Yes}) = 2/7$
$P(\text{Occ.=Engineer} \mid \text{Repay=Yes}) = 4/7$
$P(\text{Occ.=Lawyer} \mid \text{Repay=Yes}) = 1/7$

$P(\text{Income} \mid \text{Repay=Yes})$
$\mu_{\{\text{inc., Yes}\}} = 110,\ \sigma^2_{\{\text{Inc.,Yes}\}} = 2975$
$P(\text{Income} \mid \text{Repay=No})$
$\mu_{\{\text{inc., No}\}} = 90,\ \sigma^2_{\{\text{Inc.,No}\}} = 25$

$P(\text{Repay=Yes}) = 7/10$
$P(\text{Repay=No}) = 3/10$

$$P(\boldsymbol{x}^* \mid \text{No}) = P(\text{Assets=No} \mid \text{No}) \times P(\text{Occ.=Engineer} \mid \text{No}) \times P(\text{Income=85} \mid \text{No})$$
$$\boxed{= 1 \times 0 \times 0.048 = 0}$$

one of the conditional probabilities is 0, the entire expression is 0

$$P(\boldsymbol{x}^* \mid \text{Yes}) = P(\text{Assets=No} \mid \text{Yes}) \times P(\text{Occ.=Engineer} \mid \text{Yes}) \times P(\text{Income=85} \mid \text{Yes})$$
$$= 4/7 \times 4/7 \times 0.007 = 0.0023$$

$$P(\boldsymbol{x}^* \mid \text{No}) \times P(\text{No}) = 0 \times 0.3 = 0$$

$$P(\boldsymbol{x}^* \mid \text{Yes}) \times P(\text{Yes}) = 0.0023 \times 0.7 = 0.0016$$

predict Repay=Yes

Laplace Estimate or Smoothing

Original: $P(x_i = z|y = c) = \frac{|(x_i = z) \wedge (y = c)|}{|y = c|}$

Laplace: $P(x_i = z|y = c) = \frac{|(x_i = z) \wedge (y = c)| + 1}{|y = c| + \boxed{n_i}}$

Number of possible values of x_i

$$P(\text{Engineer}|\text{No}) = \frac{\#(\text{Engineer} \wedge \text{No})}{\#(\text{No})} = \frac{0}{3}$$

$$P(\text{Engineer}|\text{No}) = \frac{\#(\text{Engineer} \wedge \text{No}) + 1}{\#(\text{No}) + 3} = \frac{1}{6}$$

The same to $P(\text{Manager}|\text{No})$ and $P(\text{Lawyer}|\text{No})$

Extreme case - no training data:

$$P(\text{Manager}|\text{No}) = P(\text{Engineer}|\text{No}) = P(\text{Lawyer}|\text{No}) = \frac{1}{3}$$

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

More General Form

$\alpha > 0$ is a smoothing parameter

Laplace: $$P(x_i = z|y = c) = \frac{|(x_i = z) \land (y = c)| + \boxed{\alpha}}{|y = c| + \alpha n_i}$$

For example, $\alpha = 0.1$

$$P(\text{Engineer}|\text{No}) = \frac{\#(\text{Engineer} \land \text{No}) + 0.1}{\#(\text{No}) + 0.3} = \frac{1}{33}$$

For example, $\alpha = 10$

$$P(\text{Engineer}|\text{No}) = \frac{\#(\text{Engineer} \land \text{No}) + 10}{\#(\text{No}) + 30} = \frac{10}{33}$$

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	Engineer	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	No	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

Practice

Use Laplace smoothing with $\alpha = 1$ to re-estimate $P(\text{Assets}|\text{Repay})$ and $P(\text{Occ.}|\text{Repay})$

$P(\text{Assets=Yes} \mid \text{Repay=No}) = 0/3$
$P(\text{Assets=No} \mid \text{Repay=No}) = 3/3$
$P(\text{Assets=Yes} \mid \text{Repay=Yes}) = 3/7$
$P(\text{Assets=No} \mid \text{Repay=Yes}) = 4/7$

$P(\text{Occ.=Manager} \mid \text{Repay=No}) = 2/3$
$P(\text{Occ.=Engineer} \mid \text{Repay=No}) = 0/3$
$P(\text{Occ.=Lawyer} \mid \text{Repay=No}) = 1/3$
$P(\text{Occ.=Manager} \mid \text{Repay=Yes}) = 2/7$
$P(\text{Occ.=Engineer} \mid \text{Repay=Yes}) = 4/7$
$P(\text{Occ.=Lawyer} \mid \text{Repay=Yes}) = 1/7$

$P(\text{Income} \mid \text{Repay=Yes})$
$\mu_{\{\text{inc., Yes}\}} = 110,\ \sigma^2_{\{\text{Inc.,Yes}\}} = 2975$
$P(\text{Income} \mid \text{Repay=No})$
$\mu_{\{\text{inc., No}\}} = 90,\ \sigma^2_{\{\text{Inc.,No}\}} = 25$

$P(\text{Repay=Yes}) = 7/10$
$P(\text{Repay=No}) = 3/10$

$P(\text{Assets=Yes} \mid \text{Repay=No}) = ?$
$P(\text{Assets=No} \mid \text{Repay=No}) = ?$
$P(\text{Assets=Yes} \mid \text{Repay=Yes}) = ?$
$P(\text{Assets=No} \mid \text{Repay=Yes}) = ?$

$P(\text{Occ.=Manager} \mid \text{Repay=No}) = ?$
$P(\text{Occ.=Engineer} \mid \text{Repay=No}) = ?$
$P(\text{Occ.=Lawyer} \mid \text{Repay=No}) = ?$
$P(\text{Occ.=Manager} \mid \text{Repay=Yes}) = ?$
$P(\text{Occ.=Engineer} \mid \text{Repay=Yes}) = ?$
$P(\text{Occ.=Lawyer} \mid \text{Repay=Yes}) = ?$

$P(\text{Income} \mid \text{Repay=Yes})$
$\mu_{\{\text{inc., Yes}\}} = 110,\ \sigma^2_{\{\text{Inc.,Yes}\}} = 2975$
$P(\text{Income} \mid \text{Repay=No})$
$\mu_{\{\text{inc., No}\}} = 90,\ \sigma^2_{\{\text{Inc.,No}\}} = 25$

$P(\text{Repay=Yes}) = 7/10$
$P(\text{Repay=No}) = 3/10$

$$P(x_i = z | y = c) = \frac{|(x_i = z) \wedge (y = c)| + \alpha}{|y = c| + \alpha n_i} \qquad \alpha = 1$$

P(Assets=Yes | Repay=No) = 0/3
P(Assets=No | Repay=No) = 3/3
P(Assets=Yes | Repay=Yes) = 3/7
P(Assets=No | Repay=Yes) = 4/7

P(Occ.=Manager | Repay=No) = 2/3
P(Occ.=Engineer | Repay=No) = 0/3
P(Occ.=Lawyer | Repay=No) = 1/3
P(Occ.=Manager | Repay=Yes) = 2/7
P(Occ.=Engineer | Repay=Yes) = 4/7
P(Occ.=Lawyer | Repay=Yes) = 1/7

$$P(\text{Assets=Yes} \mid \text{Repay=No}) = \frac{0+1}{3+2} = \frac{1}{5}$$
$$P(\text{Assets=No} \mid \text{Repay=No}) = \frac{3+1}{3+2} = \frac{4}{5}$$
$$P(\text{Assets=Yes} \mid \text{Repay=Yes}) = \frac{3+1}{7+2} = \frac{4}{9}$$
$$P(\text{Assets=No} \mid \text{Repay=Yes}) = \frac{4+1}{7+2} = \frac{5}{9}$$

$$P(\text{Occ.=Manager} \mid \text{Repay=No}) = \frac{2+1}{3+3} = \frac{1}{2}$$
$$P(\text{Occ.=Engineer} \mid \text{Repay=No}) = \frac{0+1}{3+3} = \frac{1}{6}$$
$$P(\text{Occ.=Lawyer} \mid \text{Repay=No}) = \frac{1+1}{3+3} = \frac{1}{3}$$
$$P(\text{Occ.=Manager} \mid \text{Repay=Yes}) = \frac{2+1}{7+3} = \frac{3}{10}$$
$$P(\text{Occ.=Engineer} \mid \text{Repay=Yes}) = \frac{4+1}{7+3} = \frac{1}{2}$$
$$P(\text{Occ.=Lawyer} \mid \text{Repay=Yes}) = \frac{1+1}{7+3} = \frac{1}{5}$$

Naïve Bayes vs. Logistic Regression

- Both are probabilistic models for classification
- Use different ways to estimate $P(y|\boldsymbol{x})$
  - Naïve Bayes:

Generative model

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, y)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

  - Logistic Regression:

Discriminative model

$$P(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

Binary classification

$$P(y = 0|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}^T\boldsymbol{x})}{1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naïve Bayes, Andrew Ng and Michael Jordon, NIPS 2001

Deal with Missing Values

- In training, we only need to compute $P(x_i = z|y = c)$ for each feature independently
  - Ignore the missing value, e.g., when we compute $P(\text{Occ.} = z|\text{Repay} = \text{Yes})$ and $P(\text{Occ.} = z|\text{Repay} = \text{No})$, where $z \in \{\text{Manager,Engineer, Laywer}\}$, we only consider the data instances without missing values of Occ.
  - No need to remove whole data instances or features from the training dataset

ID	Assets	Occupation	Income	Repay
1	Yes	Manager	125K	Yes
2	No	?	100K	Yes
3	No	Manager	70K	Yes
4	Yes	Engineer	120K	Yes
5	?	Lawyer	95K	No
6	No	Engineer	60K	Yes
7	Yes	Lawyer	220K	Yes
8	No	Manager	85K	No
9	No	Engineer	75K	Yes
10	No	Manager	90K	No

- In testing,

ID	Assets	Occupation	Income	Repay
11	?	Engineer	85k	?

$$\text{v.s.} \begin{cases} \boxed{P(\text{No} \mid \text{Occ.=Engineer,Income=85})} \\ P(\text{Yes} \mid \text{Occ.=Engineer,Income=85}) \end{cases}$$

$$P(\text{No} \mid \text{Occ.=Engineer,Income=85}) \propto P(\text{Occ.=Engineer,Income=85} \mid \text{No})P(\text{No})$$

$$= P(\text{Occ.=Engineer,Income=85,No})$$

By using the sum rule $\sum_B P(\boldsymbol{A}, B) = P(\boldsymbol{A})$

$$= P(\text{Assets=No,Occ.=Eng.,Income=85,No}) + P(\text{Assets=Yes,Occ.=Eng.,Income=85,No})$$

$$= P(\text{Assets=No} \mid \text{No}) \times P(\text{Occ.=Engineer} \mid \text{No}) \times P(\text{Income=85} \mid \text{No}) \times P(\text{No})$$
$$+P(\text{Assets=Yes} \mid \text{No}) \times P(\text{Occ.=Engineer} \mid \text{No}) \times P(\text{Income=85} \mid \text{No}) \times P(\text{No})$$

$$= \underbrace{\boxed{\big(P(\text{Assets=No} \mid \text{No}) + P(\text{Assets=Yes} \mid \text{No})\big)}}_{=1}$$
$$\times P(\text{Occ.=Engineer} \mid \text{No}) \times P(\text{Income=85} \mid \text{No}) \times P(\text{No})$$

$$= P(\text{Occ.=Engineer} \mid \text{No}) \times P(\text{Income=85} \mid \text{No}) \times P(\text{No})$$

$$P(\text{Yes} \mid \text{Occ.=Engineer,Income=85}) \propto P(\text{Occ.=Engineer} \mid \text{Yes}) \times P(\text{Income=85} \mid \text{Yes}) \times P(\text{Yes})$$

Summary

- Computationally efficient
- Computational efficiency is obtained based on a very strong assumption of conditional independence
  - The assumption may not hold in practice (most of the time)
  - That is why we call it “naïve”
  - It was widely used for text classification in the past

Implementation using scikit-learn

- API: sklearn.naive_bayes: Naive Bayes
  https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes

`sklearn.naive_bayes`: Naive Bayes

The `sklearn.naive_bayes` module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

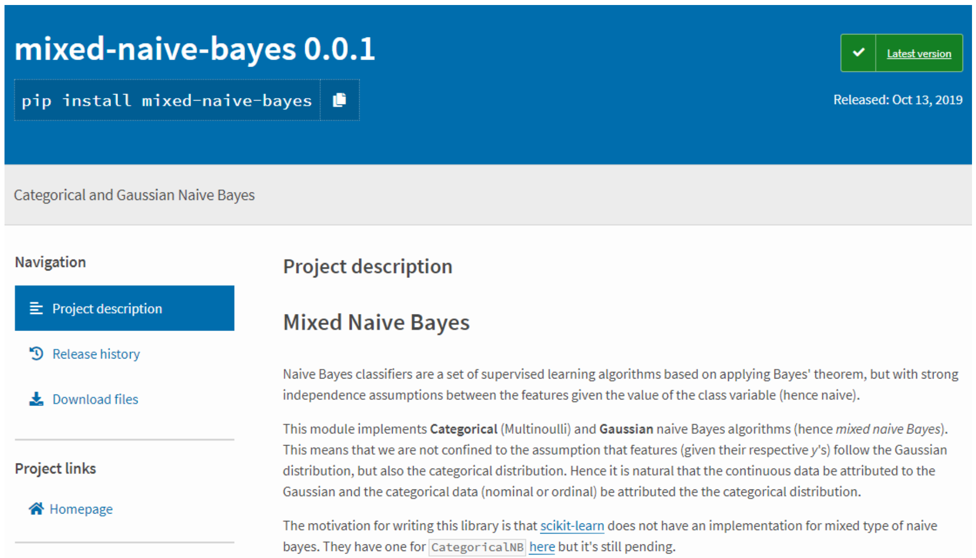
User guide: See the Naive Bayes section for further details.

`naive_bayes.BernoulliNB(*[, alpha, ...])`	Naive Bayes classifier for multivariate Bernoulli models.
`naive_bayes.CategoricalNB(*[, alpha, ...])`	Naive Bayes classifier for categorical features
`naive_bayes.ComplementNB(*[, alpha, ...])`	The Complement Naive Bayes classifier described in Rennie et al.
`naive_bayes.GaussianNB(*[, priors, ...])`	Gaussian Naive Bayes (GaussianNB)
`naive_bayes.MultinomialNB(*[, alpha, ...])`	Naive Bayes classifier for multinomial models

Documentation: https://scikit-learn.org/stable/modules/naive_bayes.html

Mixed Naïve Bayes Implementation

https://pypi.org/project/mixed-naive-bayes/#installation


mixed-naive-bayes 0.0.1

pip install mixed-naive-bayes

Latest version

Released: Oct 13, 2019

Categorical and Gaussian Naive Bayes

Navigation

Project description

Release history

Download files

Project links

Homepage

Project description

Mixed Naive Bayes

Naive Bayes classifiers are a set of supervised learning algorithms based on applying Bayes' theorem, but with strong independence assumptions between the features given the value of the class variable (hence naive).

This module implements **Categorical** (Multinoulli) and **Gaussian** naive Bayes algorithms (hence *mixed naive Bayes*). This means that we are not confined to the assumption that features (given their respective *y*'s) follow the Gaussian distribution, but also the categorical distribution. Hence it is natural that the continuous data be attributed to the Gaussian and the categorical data (nominal or ordinal) be attributed the the categorical distribution.

The motivation for writing this library is that scikit-learn does not have an implementation for mixed type of naive bayes. They have one for `CategoricalNB` here but it's still pending.

```
>>> from mixed_naive_bayes import MixedNB
```

```
>>> nbC = MixedNB(categorical_features=[0,1,3])
>>> nbC.fit(X, y)
>>> nbC.predict(X)
```

Specify which columns are categorical features

Outline

- Bayesian Classifiers
  - Naïve Bayes classifiers
- *K* Nearest neighbors (*K*-NN) classifiers
  - A lazy classifier

Typical Learning Procedure

Inductive Learning

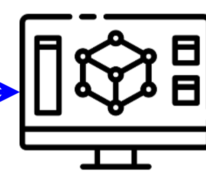
Labeled training data $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$

ID	Gender	Profession	Income	Saving	Repay
1	F	Engineer	60k	200k	Yes
2	M	Student	10k	20k	Yes
…	…	…	…	…	…
10	M	Student	8k	5k	No

Training phase

Test data $\boldsymbol{x}^*$

ID	Gender	Profession	Income	Saving
11	F	Lawyer	70k	100k



Test phase

Repay
Yes or No

Model

$f: \boldsymbol{x} \rightarrow y$

Lazy Learning Procedure

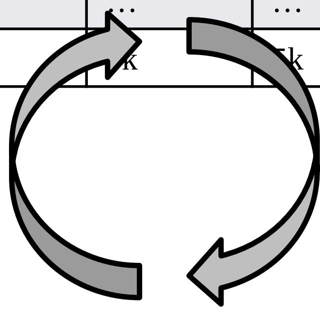
Lazy Learning

Labeled training data $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$

ID	Gender	Profession	Income	Saving	Repay
1	F	Engineer	60k	200k	Yes
2	M	Student	10k	20k	Yes
…	…	…	…	…	…
10	M	Student	[illegible]k	[illegible]k	No

Training phase

- A model is not learned during "training phase"
- Instead, hashing table or indexing can be built

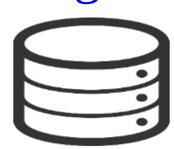


Test phase

- Retrieve similar training instances

Test data $\boldsymbol{x}^*$

ID	Gender	Profession	Income	Saving
11	F	Lawyer	70k	100k



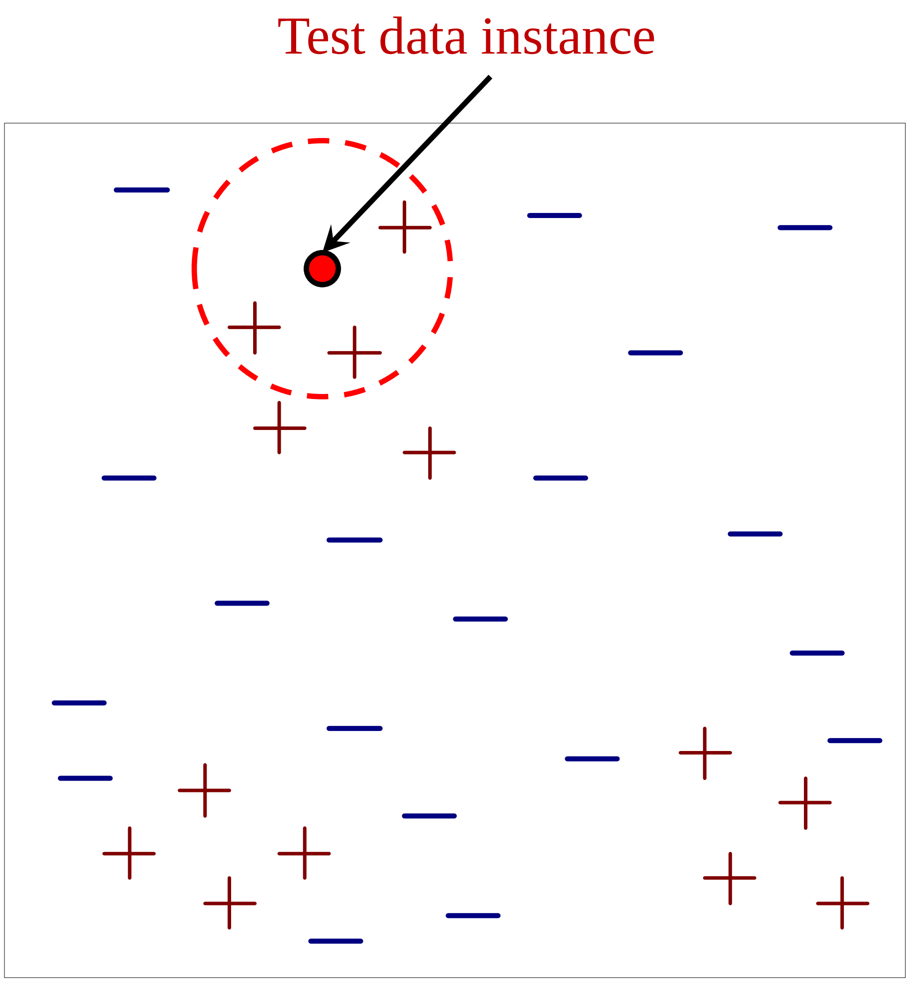
Repay
Yes or No

Based on the labels of the retrieved data instances

K-Nearest Neighbors Classifiers

- Algorithm:
  - For each test instance $\boldsymbol{x}^*$, retrieve K training data instances that are the most similar to $\boldsymbol{x}^*$ (K nearest neighbors) from the training set
  - Based on the class labels of the K nearest neighbors to make a prediction on $\boldsymbol{x}^*$

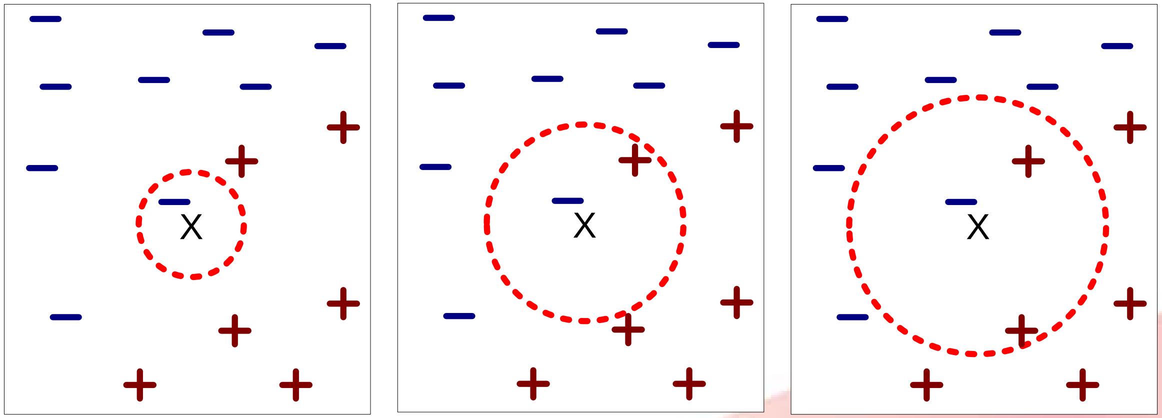
Illustration



- Requirements:
  - A set of stored labeled training instances
  - Distance measure to compute distance between instances
  - The value of K, the number of nearest neighbors to retrieve
- To classify a test instance:
  - Compute distance to all the training instances
  - Identify K nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of the test instance (e.g., using the majority class)

Distance & Nearest Neighbors

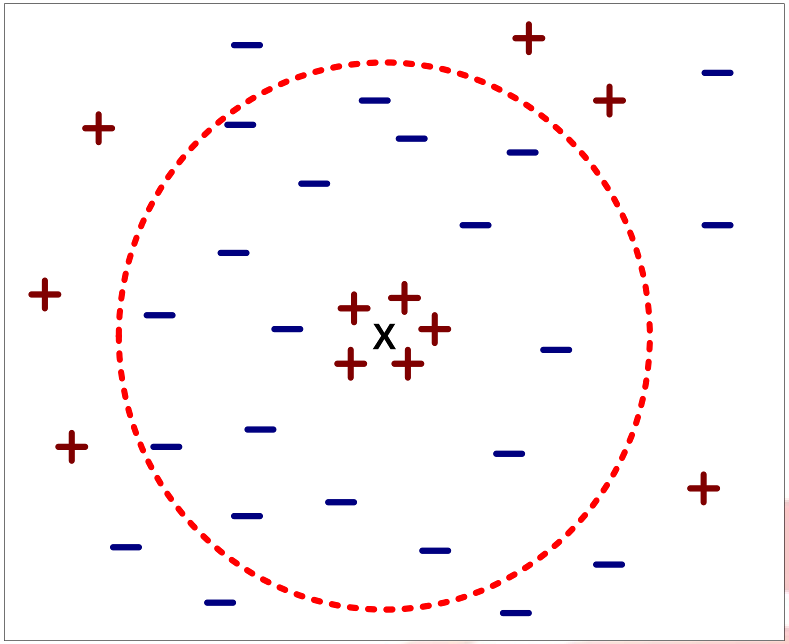
- Euclidean distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{m}(x_{ik} - x_{jk})^2}$
  - The smaller is the value, the more similar are two data instances
  - K-nearest neighbors of an instance $\boldsymbol{x}$ are data instances that have the K smallest distance to $\boldsymbol{x}$



(a) 1-nearest neighbor
(b) 2-nearest neighbor
(c) 3-nearest neighbor

Value of K

- K is a hyper-parameter:
  - If K is too small, sensitive to noise points
  - If K is too large, neighborhood may include points from other classes



Determine Class Label

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the K-nearest neighbors
- For majority voting:

Indicator function that returns 1 if its input is true, otherwise 0

$$y^* = \arg\max_c \sum_{\boxed{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_{\boldsymbol{x}^*}}} \boxed{I(c = y_i)}$$

Nearest neighbors of the test instance $\boldsymbol{x}^*$

  - Every neighbor has the same impact on the classification
  - This makes the algorithm more sensitive to the choice of K

Weighted Voting

- Alternative scheme: distance-weight voting
  - Weight the influence of each nearest neighbor $\boldsymbol{x}_i$ according to its distance to the test data

$$w_i = \frac{1}{d(\boldsymbol{x}^*, \boldsymbol{x}_i)^2}$$

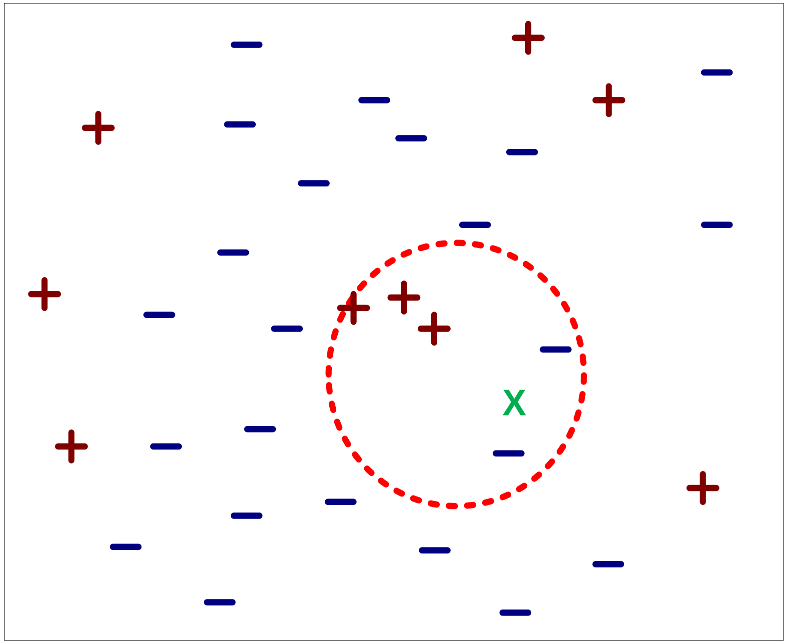
  - That is

$$y^* = \arg\max_c \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_{\boldsymbol{x}^*}} w_i I(c = y_i)$$

The larger is the distance to the test data, the smaller influence of the corresponding nearest neighbor to the vote

An Example

Consider a binary classification problem, and a 5-NN classifier



Instance ID	Class	Squared distance to test data
1	+	9
2	+	12.25
3	+	16
4	−	2.25
5	−	4

- Majority voting:

$$+: 3 > -: 2$$

- Distance-weight voting:

Distance−Weight votes for +:

$$\frac{1}{9} + \frac{1}{12.25} + \frac{1}{16} = 0.2552$$

$$<$$

Distance−Weight votes for −:

$$\frac{1}{4} + \frac{1}{2.25} = 0.6944$$

Potential Issues

- Features may have very different scales, e.g.,
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 3kg to 200kg
  - income of a person may vary from $10K to $1M
- Features need to be rescaled to prevent distance from being dominated by some features
- Solution: normalization on features of different scales to the same scale
  - Min-Max normalization
  - Standardization (*z*-score normalization)

Summary

- The *K*NN classifiers are a lazy learner
  - A classification model is not built explicitly
  - "Training" is very efficient
  - Classifying test instances is relatively expensive

Implementation using scikit-learn

- API: sklearn.neighbors.KNeighborsClassifier
  https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

sklearn.neighbors.KNeighborsClassifier

class `sklearn.neighbors.`**`KNeighborsClassifier`**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters:

n_neighbors : *int, default=5*
Number of neighbors to use by default for `kneighbors` queries.

weights : *{'uniform', 'distance'} or callable, default='uniform'*
weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'*
Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

Example

```
>>> from sklearn.neighbors import KNeighborsClassifier
>>> import numpy as np

>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.integers(2, n_samples)
>>> X = rng.randn(n_samples, n_features)

>>> knnC = KNeighborsClassifier(n_neighbors=3 )
>>> knnC.fit(X, y)
>>> pred= knnC.predict(X)
```

set number of neighbors

Build indices s.t. it is more efficient
when making predictions on test data

Thank you!