Programa de Verão FGV EMAp 2019

Introduction to Machine Learning with Python

CLASSIFICATION TECHNIQUES

Prof. Luis Gustavo Nonato

University of São Paulo - São Carlos - SP

Lots of classification techniques:

Lots of classification techniques:

- Naive Bayes Classifier
- Logistic Regression
- SVM
- Neural Networks
- $\vdots$

Lots of classification techniques:

- Naive Bayes Classifier
- Logistic Regression
- SVM
- Neural Networks
- ⋮

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

i.e., the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

i.e., the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Therefore,

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\, p(y) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

i.e., the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Therefore,

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\, p(y) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

The class y to be assigned to $\mathbf{x}$ is the one satisfying

$$\arg\max_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

x_i	age	income	job status	gender	class
1	20-30	medium	manager	male	ok
2	40-50	high	engineer	female	rich
3	20-30	medium	student	male	ok
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	60-70	medium	retired	male	poor

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

x_i	age	income	job status	gender	class
1	20-30	medium	manager	male	ok
2	40-50	high	engineer	female	rich
3	20-30	medium	student	male	ok
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	60-70	medium	retired	male	poor

$p(poor) = \frac{\#poor}{n}, p(ok) = \frac{\#ok}{n}, p(rich) = \frac{\#rich}{n}$

$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$

$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$

$\vdots$

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

x_i	age	income	job status	gender	class
1	20-30	medium	manager	male	ok
2	40-50	high	engineer	female	rich
3	20-30	medium	student	male	ok
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	60-70	medium	retired	male	poor

$p(poor) = \frac{\#poor}{n}, p(ok) = \frac{\#ok}{n}, p(rich) = \frac{\#rich}{n}$

$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$

$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$

$\vdots$

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\, p(y) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Gaussian

Naive Bayes Classifier: Gaussian

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Gaussian

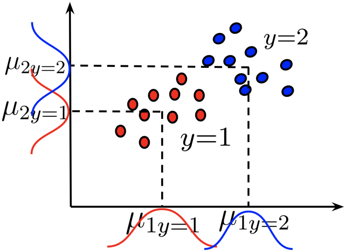
$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

$$p(x_j|y) = N(\mu_{jy}, \sigma_{jy}^2)$$

Naive Bayes Classifier: Gaussian

$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

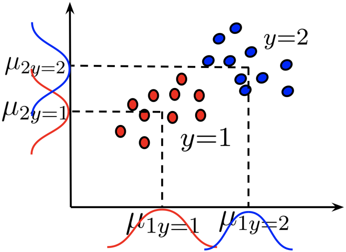
$$p(x_j|y) = N(\mu_{jy}, \sigma_{jy}^2)$$



Naive Bayes Classifier: Gaussian

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

$$p(x_j|y) = N(\mu_{jy}, \sigma_{jy}^2)$$



The set of parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ can be obtained by maximizing the likelihood function,

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{y} p(y) \prod_{\mathbf{x} \in y} \prod_{j=1}^{d} p(x_j|y)$$

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, with the same Σ for all classes

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, with the same Σ for all classes

The decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, with the same Σ for all classes

The decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1} \mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2} \boldsymbol{\mu}_2^\top \Sigma^{-1} \boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, with the same Σ for all classes

The decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

$$\mathbf{w}^\top \mathbf{x} + w_o$$

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{1, 2\}$ (two classes), and assuming

- $p(y)$ is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, with the same Σ for all classes

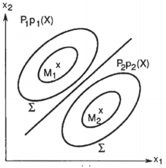
The decision boundary is given by:

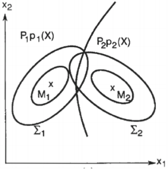
$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

$$\mathbf{w}^\top \mathbf{x} + w_o$$

If Σ fixed then the decision boundary is linear





Logistic Regression

Logistic Regression

Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalently $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalently $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Logistic regression, assumes a density distribution to $p(y|\mathbf{x})$ (*discriminative model*).

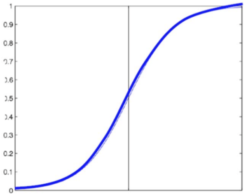
Logistic Regression

Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalently $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Logistic regression, assumes a density distribution to $p(y|\mathbf{x})$ (*discriminative model*).

Considering two classes $y \in \{0, 1\}$, the logistic regression assumption is:

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta}))}$$



$\beta_0 + \mathbf{x}^\top \boldsymbol{\beta} \geq 0$ means $\mathbf{x}$ to class 1 and to class 0 otherwise (linear decision boundary).

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta}))}$$

Parameters β_0 and $\boldsymbol{\beta}$ can be obtained by Maximum Likelihood Estimation.

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta}))}$$

Parameters β_0 and $\boldsymbol{\beta}$ can be obtained by Maximum Likelihood Estimation.

The likelihood function is given by:

$$L(\beta_0, \boldsymbol{\beta}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i)$$

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta}))}$$

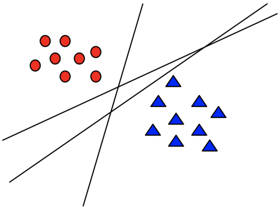
Parameters β_0 and $\boldsymbol{\beta}$ can be obtained by Maximum Likelihood Estimation.

The likelihood function is given by:

$$L(\beta_0, \boldsymbol{\beta}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i)$$

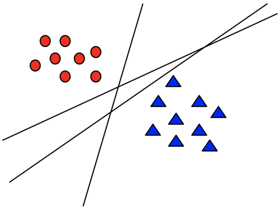
The maximization has no analytical formula and a gradient descent is typically applied to find the parameters.

Support Vector Machine

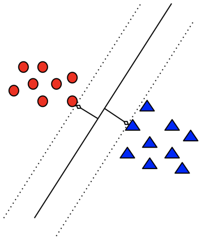


Which plane separate the classes better?

Support Vector Machine

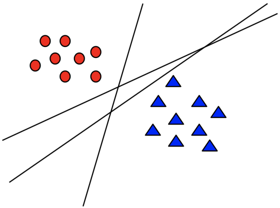


Which plane separate the classes better?

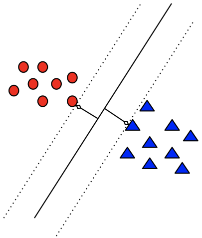


The one farther away from the samples.

Support Vector Machine



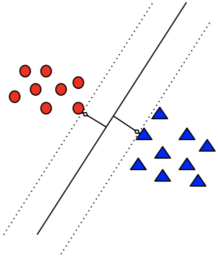
Which plane separate the classes better?



The one farther away from the samples.

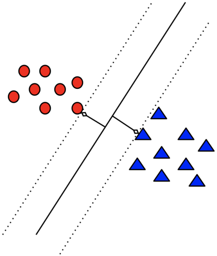
- Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed.
- Support vectors are the critical elements of the training set.

Support Vector Machine



The one farther away from the samples.

Support Vector Machine

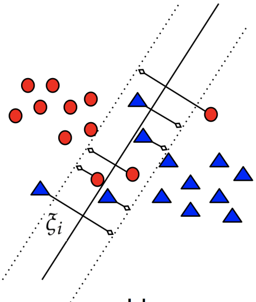


The one farther away from the samples.

When the classes are linearly separable, the farther plane can be found by solving:

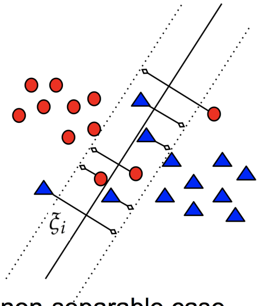
$$\min \mathbf{w}^\top \mathbf{w}$$

$$\textit{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$



The non-separable case.

Support Vector Machine



The non-separable case.

When the classes are not linearly separable, the farther plane can be found by solving:

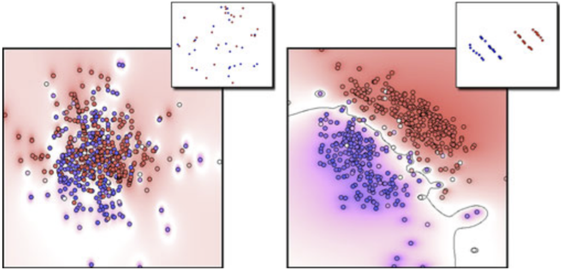
$$\min \mathbf{w}^\top \mathbf{w} + s \sum \xi_i$$

$$\textit{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \text{where } \xi_i \geq 0$$

- The solution is computed via the dual optimization problem.

- The solution is computed via the dual optimization problem.
- $\xi_i = 0$ for instances not contributing to the margin.

- The solution is computed via the dual optimization problem.
- $\xi_i = 0$ for instances not contributing to the margin.
- Non-linear boundaries can be obtained when using kernels.



(a) (b)

Multiple Classes

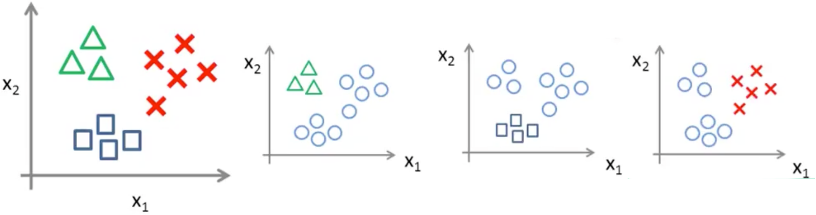
Multiple Classes

Most schemes build upon binary classification:

Multiple Classes

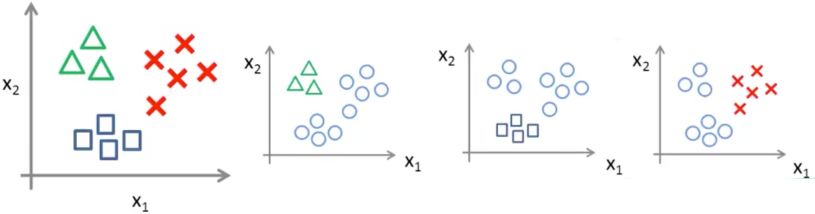
Most schemes build upon binary classification:

- One-vs-All: $k-1$ classifiers $\boxed{y = \arg\max_i f_i(x)}$



Most schemes build upon binary classification:

- One-vs-All: $k-1$ classifiers $\boxed{y = \arg\max_i f_i(x)}$



- One-vs-One: $k(k-1)/2$ classifiers $\boxed{y = \arg\max_i (\sum_j f_{ij}(x))}$