"Generative" models: Discriminant Analysis, Naïve Bayes

Nicolas Keriven
CNRS, IRISA, Rennes

ENSTA 2025

Reminder on classification problem

Variable terminology

- observed data referred to as *input* variables, *predictors* or *features* $\leftarrow$ usually denoted as x
- data to predict referred to as *output* variables, or *responses* $\leftarrow$ usually denoted as y

Reminder on classification problem

Variable terminology

- observed data referred to as *input* variables, *predictors* or *features* $\leftarrow$ usually denoted as x
- data to predict referred to as *output* variables, or *responses* $\leftarrow$ usually denoted as y

Classification task

y are *categorical* data (discrete qualitative variables) that take values in a discrete set $\mathcal{Y}$, e.g.

- $\texttt{email} \in \{\texttt{spam}, \texttt{ham}\}$,
- $\texttt{handwritten digits} \in \{0, \ldots, 9\}$

Given a feature vector $x \in \mathbb{R}^d$, build a function $f(x)$ that takes as input the feature vector x and predicts its value for $y \in \mathcal{Y}$

☞ Try to minimize the misclassification rate $\mathcal{R}(f) = \mathbb{P}(f(x) \neq y) = \mathbb{E}1_{f(x) \neq y}$ (aka expected risk for the $0 - 1$ loss)

Table of Contents

Bayes classifier

Definition

The Bayes classification rule f^* is defined as

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k | X = x)$$

Bayes classifier

Definition

The Bayes classification rule f^* is defined as

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k | X = x)$$

Theorem

The Bayes classification rule f^* is optimal in the misclassification rate sense where $\mathcal{R}(f) = \mathbb{P}(f(X) \neq Y)$:

$$\text{for any rule } f,\ \mathcal{R}(f) \geq \mathcal{R}(f^*),$$

Bayes classifier

Definition

The Bayes classification rule f^* is defined as

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k | X = x)$$

Theorem

The Bayes classification rule f^* is optimal in the misclassification rate sense where $\mathcal{R}(f) = \mathbb{P}(f(X) \neq Y)$:

$$\text{for any rule } f,\ \mathcal{R}(f) \geq \mathcal{R}(f^*),$$

Remarks

- In practice, the distribution of (x, y) is unknown $\Rightarrow$ no analytical expression of $f^*(x)$. But useful reference on academic examples.

Estimation of $f^*(X)$

Bayes rule :

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(Y = k)p(x|Y = k)}{p(x)}$$

Estimation of $f^*(X)$

Bayes rule :

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(Y = k)p(x|Y = k)}{p(x)}$$

Hence

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k|X = x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k)p(x|Y = k)$$

Estimation of $f^*(X)$

Bayes rule :

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(Y = k)p(x|Y = k)}{p(x)}$$

Hence

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k|X = x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k)p(x|Y = k)$$

1. **Discriminative approaches** : direct learning of $\mathbb{P}(Y|X)$
   - e.g. logistic regression : $\mathbb{P}(Y = k|x) \approx \text{sigmoid}_k(f(x)) = \frac{e^{f(x)_k}}{\sum_\ell e^{f(x)_\ell}}$ where f is a linear function, a neural net...
   - Very powerful, but not very interpretable

Estimation of $f^*(X)$

Bayes rule :

$$\mathbb{P}(Y = k|X = x) = \frac{\mathbb{P}(Y = k)p(x|Y = k)}{p(x)}$$

Hence

$$f^*(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k|X = x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k)p(x|Y = k)$$

1. **Discriminative approaches** : direct learning of $\mathbb{P}(Y|X)$
   - e.g. logistic regression : $\mathbb{P}(Y = k|x) \approx \text{sigmoid}_k(f(x)) = \frac{e^{f(x)_k}}{\sum_\ell e^{f(x)_\ell}}$ where f is a linear function, a neural net...
   - Very powerful, but not very interpretable
2. **Generative models** : learning of the joint distribution $p(X, Y)$

$$\mathbb{P}(X = x, Y = k) = p(x|Y = k)\mathbb{P}(Y = k),$$

   - $p_k(x) = p(x|Y = k)$ is the data distribution of class k
   - $\pi_k = \mathbb{P}(Y = k)$ is the weight (proportion) of class k
   - linear/quadratic discriminant analysis, Naïve Bayes
   - Interpretable, but requires good generative models (difficult)
   - Can generate new data !

Maximum likelihood estimation (MLE)

- How to learn a generative model ?

Maximum likelihood estimation (MLE)

- How to learn a generative model ?
- MLE is a general methodology which consists in maximizing the probability of observing the training data with respect to some parametric distribution p_θ.
  The likelihood function is :

$$\mathcal{L}(\theta) = p_\theta((x_1, y_1), \dots, (x_n, y_n))$$

Maximum likelihood estimation (MLE)

- How to learn a generative model ?
- MLE is a general methodology which consists in maximizing the probability of observing the training data with respect to some parametric distribution p_θ. The likelihood function is :

$$\mathcal{L}(\theta) = p_\theta((x_1, y_1), \ldots, (x_n, y_n))$$

- Generally (x_i, y_i) are supposed i.i.d., and

$$\mathcal{L}(\theta) = \prod_i p_\theta(x_i, y_i)$$

Maximum likelihood estimation (MLE)

- How to learn a generative model ?
- MLE is a general methodology which consists in maximizing the probability of observing the training data with respect to some parametric distribution p_θ.
  The likelihood function is :

$$\mathcal{L}(\theta) = p_\theta((x_1, y_1), \ldots, (x_n, y_n))$$

- Generally (x_i, y_i) are supposed i.i.d., and

$$\mathcal{L}(\theta) = \prod_i p_\theta(x_i, y_i)$$

- Products are complicated, hence one always work with the log-likelihood :

$$\ell(\theta) = \log \mathcal{L}(\theta) = \sum_i \log p_\theta(x_i, y_i) = \sum_i (\log \pi_{y_i} + \log p_{y_i}(x_i))$$

  - $p_k(x) \equiv p_\theta(x|Y = k)$ is the *density* for X in class k
  - $\pi_k \equiv p_\theta(Y = k)$ is the *weight*, or *prior* probability of class k

Maximum likelihood estimation (MLE)

- How to learn a generative model ?
- MLE is a general methodology which consists in maximizing the probability of observing the training data with respect to some parametric distribution p_θ. The likelihood function is :

$$\mathcal{L}(\theta) = p_\theta((x_1, y_1), \ldots, (x_n, y_n))$$

- Generally (x_i, y_i) are supposed i.i.d., and

$$\mathcal{L}(\theta) = \prod_i p_\theta(x_i, y_i)$$

- Products are complicated, hence one always work with the log-likelihood :

$$\ell(\theta) = \log \mathcal{L}(\theta) = \sum_i \log p_\theta(x_i, y_i) = \sum_i (\log \pi_{y_i} + \log p_{y_i}(x_i))$$

  - $p_k(x) \equiv p_\theta(x|Y = k)$ is the *density* for X in class k
  - $\pi_k \equiv p_\theta(Y = k)$ is the *weight*, or *prior* probability of class k
- MLE is a classic, interpretable source of loss functions ! The log-likelihood is directly the empirical risk ! (up to $1/n$)

Table of Contents

Discriminant Analysis

Two kinds of Discriminant Analysis : Linear and Quadratic. In both cases, the key assumption is that, within each class, the input variables X_i are assumed to be normally distributed.

Supplementary materials

- Wikipedia page (quite complete and detailed)
  https://en.wikipedia.org/wiki/Linear_discriminant_analysis
- short and simple Scikit-learn documentation (with examples)
  https://scikit-learn.org/stable/modules/lda_qda.html

Table of Contents

Quadratic Discriminant Analysis (QDA)

Supervised classification assumptions

- $x \in \mathbb{R}^d$, $y \in \mathcal{Y} = \{1, \ldots, K\}$,
- sized n training set $(x_1, y_1), \ldots (x_n, y_n)$

QDA Assumptions

The input variables x, given a class $y = k$, are distributed according to a Gaussian distribution :

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k) \Leftrightarrow p_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

Quadratic Discriminant Analysis (QDA)

Supervised classification assumptions

- $x \in \mathbb{R}^d$, $y \in \mathcal{Y} = \{1, \dots, K\}$,
- sized n training set $(x_1, y_1), \dots (x_n, y_n)$

QDA Assumptions

The input variables x, given a class $y = k$, are distributed according to a Gaussian distribution :

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k) \Leftrightarrow p_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

The Gaussian parameters are, for each class $k = 1, \dots, K$

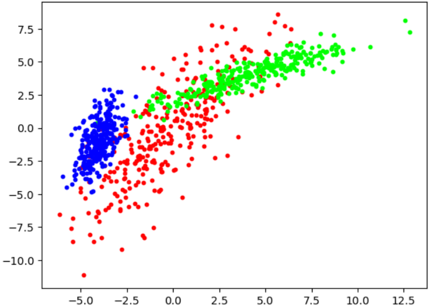
- mean vectors $\mu_k \in \mathbb{R}^d$,
- covariance matrices $\Sigma_k \in \mathbb{R}^{d \times d}$,

☞ set of parameters $\theta_k = \{\mu_k, \Sigma_k\}$, plus the weights π_k, for $k = 1, \dots, K$.

Example

Mixture of $K = 3$ Gaussians

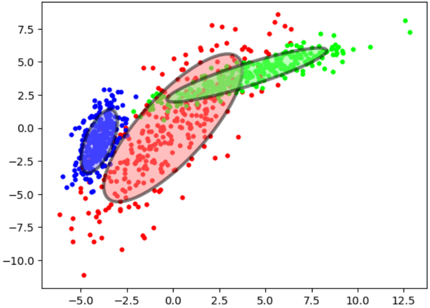
- $\mathcal{Y} = \{1, 2, 3\}$
- $d = 2$



Example

Mixture of $K = 3$ Gaussians

- $\mathcal{Y} = \{1, 2, 3\}$
- $d = 2$



True mean μ_k and covariance Σ_k parameters, for $k = 1, 2, 3$

QDA parameter estimation

Log-likelihood

The log-likelihood with Gaussians is :

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{n} \log \pi_{y_i} + \log p(x_i, \theta_{y_i}) \\ &= \sum_{i=1}^{n} \log \pi_{y_i} - \frac{1}{2}\log|\Sigma_{y_i}| - \frac{1}{2}(x_i - \mu_{y_i})^T \Sigma_{y_i}^{-1}(x_i - \mu_{y_i})\end{aligned}$$

Remark : $\pi_K = 1 - \sum_{j=1}^{K-1} \pi_j$ so there is one less parameter.

QDA parameter estimation (Cont'd)

Maximizing $\ell(\theta)$ by setting its gradient to 0, we obtain

- $\hat{\pi}_k = \frac{n_k}{n}$ where $n_k = \sharp\{y_i = k\}$. Sample proportion, valid for any model p_k.

QDA parameter estimation (Cont'd)

Maximizing $\ell(\theta)$ by setting its gradient to 0, we obtain

- $\hat{\pi}_k = \frac{n_k}{n}$ where $n_k = \sharp\{y_i = k\}$. Sample proportion, valid for any model p_k.
- $\hat{\mu}_k = \frac{\sum_{y_i=k} x_i}{n_k}$: empirical mean, a classic quantity. Easy derivation.

QDA parameter estimation (Cont'd)

Maximizing $\ell(\theta)$ by setting its gradient to 0, we obtain

- $\hat{\pi}_k = \frac{n_k}{n}$ where $n_k = \sharp\{y_i = k\}$. Sample proportion, valid for any model p_k.
- $\hat{\mu}_k = \frac{\sum_{y_i=k} x_i}{n_k}$: empirical mean, a classic quantity. Easy derivation.
- $\hat{\Sigma}_k = \frac{1}{n_k} \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$. Empirical covariance, again classic. Zeroing the gradient is a bit harder! hint : derive wrt Σ^{-1} and not Σ, use the chain rule.
  - Unlike $\hat{\mu}$, $\hat{\Sigma}$ is biased. An unbiased version is $\frac{n_k}{n_k-1}\hat{\Sigma}_k = \frac{1}{n_k-1} \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$

see `https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter13.pdf`

QDA decision rule

Since we have estimated $p(x, y)$, we can derive a classification rule. Starting from the expression of the Bayes estimator :

$$\begin{aligned} f(x) &= \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k) p(x|Y = k) \\ &\approx \arg\max_{k \in \mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k) = \arg\max_{k \in \mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k) \end{aligned}$$

QDA decision rule

Since we have estimated $p(x, y)$, we can derive a classification rule. Starting from the expression of the Bayes estimator :

$$
\begin{aligned}
f(x) &= \arg\max_{k\in\mathcal{Y}} \mathbb{P}(Y=k)p(x|Y=k) \\
&\approx \arg\max_{k\in\mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k) = \arg\max_{k\in\mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k)
\end{aligned}
$$

Taking the logarithm, which does not change the argmax : $f(x) = \arg\max_k \delta_k(x)$ where

$$
\delta_k(x) = -\frac{1}{2}\log\left|\hat{\Sigma}_k\right| - \frac{1}{2}(x-\hat{\mu}_k)^T\hat{\Sigma}_k^{-1}(x-\hat{\mu}_k) + \log\hat{\pi}_k + \cancel{\mathfrak{Cst}},
$$

is the discriminant function

QDA decision rule

Since we have estimated $p(x, y)$, we can derive a classification rule. Starting from the expression of the Bayes estimator :

$$f(x) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k) p(x|Y = k)$$
$$\approx \arg\max_{k \in \mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k) = \arg\max_{k \in \mathcal{Y}} \hat{\pi}_k p(x|\hat{\theta}_k)$$

Taking the logarithm, which does not change the argmax : $f(x) = \arg\max_k \delta_k(x)$ where

$$\delta_k(x) = -\frac{1}{2} \log \left|\hat{\Sigma}_k\right| - \frac{1}{2}(x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1}(x - \hat{\mu}_k) + \log \hat{\pi}_k + \cancel{\mathrm{Cst}},$$

is the discriminant function

Remarks

If the Gaussian model is correct :

1. this is an estimation of the Bayes classifier with θ replaced by $\hat{\theta}$ (and π replaced by $\hat{\pi}$)
2. when $n \gg d$, $\hat{\theta} \to \theta$ (and $\hat{\pi} \to \pi$) : convergence to the Bayes classifier

QDA decision boundary

The boundary between two classes k and l is described by the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x + x^T Q_{k,l}^T x = 0,$$

where

- $C_{k,l} = -\frac{1}{2} \log \frac{|\hat{\Sigma}_k|}{|\hat{\Sigma}_l|} + \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}_k^{-1} \hat{\mu}_k + \frac{1}{2} \hat{\mu}_l^T \hat{\Sigma}_l^{-1} \hat{\mu}_l, \quad \leftarrow$ scalar
- $L_{k,l} = \hat{\Sigma}_k^{-1} \hat{\mu}_k - \hat{\Sigma}_l^{-1} \hat{\mu}_l, \quad \leftarrow$ vector in $\mathbb{R}^d$
- $Q_{k,l} = \frac{1}{2} \left(-\hat{\Sigma}_k^{-1} + \hat{\Sigma}_l^{-1} \right), \quad \leftarrow$ matrix in $\mathbb{R}^{d \times d}$

QDA decision boundary

The boundary between two classes k and l is described by the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x + x^T Q_{k,l}^T x = 0,$$

where

- $C_{k,l} = -\frac{1}{2} \log \frac{|\hat{\Sigma}_k|}{|\hat{\Sigma}_l|} + \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}_k^{-1} \hat{\mu}_k + \frac{1}{2} \hat{\mu}_l^T \hat{\Sigma}_l^{-1} \hat{\mu}_l, \quad \leftarrow$ scalar
- $L_{k,l} = \hat{\Sigma}_k^{-1} \hat{\mu}_k - \hat{\Sigma}_l^{-1} \hat{\mu}_l, \quad \leftarrow$ vector in $\mathbb{R}^d$
- $Q_{k,l} = \frac{1}{2} \left(-\hat{\Sigma}_k^{-1} + \hat{\Sigma}_l^{-1} \right), \quad \leftarrow$ matrix in $\mathbb{R}^{d \times d}$

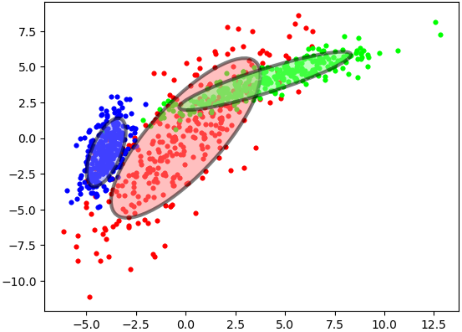
- This is a quadratic equation, which defines an ellipsoid

☞ hence Quadratic discriminant analysis

QDA example

Mixture of $K = 3$ Gaussians

- Estimation of the parameters $\hat{\mu}_k$, $\hat{\Sigma}_k$ and $\hat{\pi}_k$, for $k = 1, 2, 3$

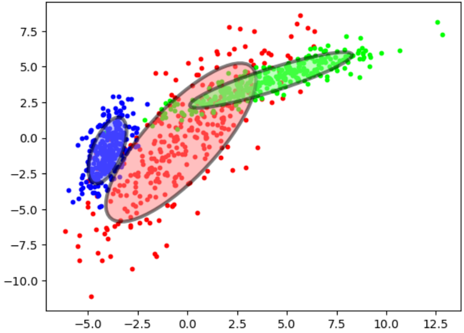


true parameters

QDA example

Mixture of $K = 3$ Gaussians

- Estimation of the parameters $\hat{\mu}_k$, $\hat{\Sigma}_k$ and $\hat{\pi}_k$, for $k = 1, 2, 3$



estimated parameters

QDA example

Mixture of $K = 3$ Gaussians

- Classification rule : $\arg\max_{k=1,2,3} \delta_k(x)$
- Quadratic boundaries $\{x; \delta_k(x) = \delta_l(x)\}$

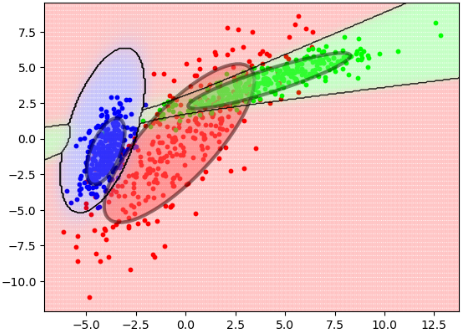


Table of Contents

LDA principle

LDA Assumptions

Additional simplifying assumption w.r.t. QDA : all the class covariance matrices are identical ("homoscedasticity"), i.e. $\Sigma_k = \Sigma$, for $k = 1, \ldots, K$

Maximum likelihood estimators (MLE)

- $\hat{\pi}_k$ and $\hat{\mu}_k$ are unchanged,
- $\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{y_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$ pooled covariance
  - Again, this is a biased estimator. The unbiased version is $\frac{n}{n-K}\hat{\Sigma}$.

LDA discriminant function

$$\delta_k(x) = -\frac{1}{2} \log \left|\hat{\Sigma}\right| - \frac{1}{2}(x - \hat{\mu}_k)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_k) + \log \hat{\pi}_k + \cancel{\mathrm{Cst}},$$

LDA decision boundary

The boundary between two classes k and l reduces to the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x = 0$$

where

- $C_{k,l} = \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2}\hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + \frac{1}{2}\hat{\mu}_l^T \hat{\Sigma}^{-1} \hat{\mu}_l, \quad \leftarrow$ scalar

- $L_{k,l} = \hat{\Sigma}^{-1} (\hat{\mu}_k - \hat{\mu}_l), \quad \leftarrow$ vector in $\mathbb{R}^d$

LDA decision boundary

The boundary between two classes k and l reduces to the equation

$$\delta_k(x) = \delta_l(x) \Leftrightarrow C_{k,l} + L_{k,l}^T x = 0$$

where

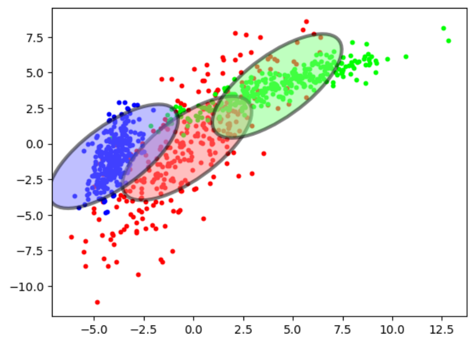
- $C_{k,l} = \log \frac{\hat{\pi}_k}{\hat{\pi}_l} - \frac{1}{2}\hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + \frac{1}{2}\hat{\mu}_l^T \hat{\Sigma}^{-1} \hat{\mu}_l, \quad \leftarrow$ scalar
- $L_{k,l} = \hat{\Sigma}^{-1} \left(\hat{\mu}_k - \hat{\mu}_l\right), \quad \leftarrow$ vector in $\mathbb{R}^d$
- This is a linear equation

☞ Linear discriminant analysis

Linear Discriminant Analysis (LDA)

Mixture of $K = 3$ Gaussians

- Estimation of the parameters $\hat{\mu}_k$, $\hat{\pi}_k$, for $k = 1, 2, 3$, and $\hat{\Sigma}$

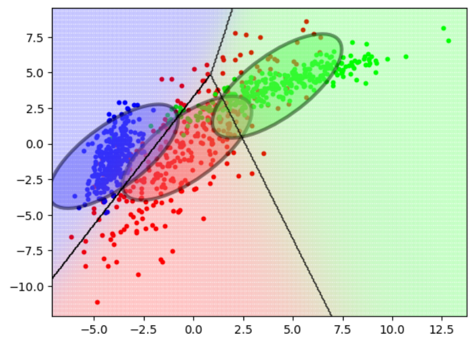


estimated parameters

Linear Discriminant Analysis (LDA)

Mixture of $K = 3$ Gaussians

- Classification rule : $\arg\max_{k=1,2,3} \delta_k(x)$
- linear boundaries $\{x; \delta_k(x) = \delta_l(x)\}$



Complexity of discriminant analysis methods

Effective number of parameters

- LDA : $K - 1 + Kd + \frac{d(d+1)}{2} = O(Kd + d^2)$
- QDA : $K - 1 + Kd + K\frac{d(d+1)}{2} = O(Kd^2)$

Complexity of discriminant analysis methods

Effective number of parameters

- LDA : $K - 1 + Kd + \frac{d(d+1)}{2} = O(Kd + d^2)$
- QDA : $K - 1 + Kd + K\frac{d(d+1)}{2} = O(Kd^2)$

Remarks

- in high dimension, i.e. $d \approx n$ or $d > n$, LDA is more stable than QDA which is more prone to overfitting,
- both methods appear however to be robust on a large number of real-word datasets
- LDA can be viewed in some cases as a least squares regression method
- LDA performs a dimension reduction to a subspace of dimension $\leq K - 1$ generated by the vectors $z_k = \hat{\Sigma}^{-1}\hat{\mu}_k \leftarrow$ dimension reduction from p to $K - 1$! (same for QDA, but more rarely used)

Table of Contents

Non-parametric modelling

Non-parametric estimation of $p_k(x) = p(x|Y = k)$: density estimation. Then $f(\hat{x}) = \arg\max_k \hat{\pi}_k \hat{p}_k(x)$ as usual.

Non-parametric modelling

Non-parametric estimation of $p_k(x) = p(x|Y = k)$: density estimation. Then $\hat{f(x)} = \arg\max_k \hat{\pi}_k \hat{p}_k(x)$ as usual.

Parzen kernel approach

To locally estimate the density, take a weighted average of the number of points in the neighborhood of the desired location :

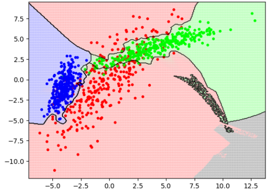
$$\hat{p}_k(x) = \frac{1}{n_k} \sum_{y_i = k} k_\lambda(x, x_i)$$

for a kernel function k_λ. Usually λ is a bandwidth, and $k_\lambda(x, x') = \frac{1}{\lambda^d} k(\frac{x-x'}{\lambda})$, with $\int k = 1$. Classic choice includes :

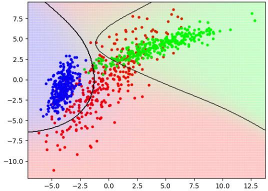
- 0-1 kernel : $k(x, x') = 1/V_d$ if $\|x - x_i\| \leq 1$, 0 otherwise, where V_d is the volume of the d-sphere. True unweighted average of the number of points in a fixed-radius neighborhood.
- Gaussian kernel : $k(x, x') = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|x-x'\|^2}$. Classic choice.

- Same problem than k-NN : in high-dimension, the space is mostly empty !

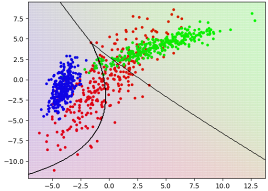
KDE example



$\lambda = 0.25$



$\lambda = 2$



$\lambda = 5$

Complexity parameter λ (kernel bandwidth)

- large λ w.r.t. to the dispersion of X $\rightarrow$ under-fitting
- small λ w.r.t. to the dispersion of X $\rightarrow$ over-fitting

Table of Contents

Naïve Bayes (NB)

NB classifiers

Family of "probabilistic classifiers" based on applying Bayes' theorem on a generative model, with strong (naïve) independence assumptions between the features. Particularly useful for high-dimensional data (avoids quadratic cost d^2).

Naïve Bayes (NB)

NB classifiers

Family of "probabilistic classifiers" based on applying Bayes' theorem on a generative model, with strong (naïve) independence assumptions between the features. Particularly useful for high-dimensional data (avoids quadratic cost d^2). Can be coupled with

- parametric models (Gaussian, Bernoulli, Multinomial,...) with maximum likelihood estimation
- or non-parametric models with kernel density estimation

Supplementary materials

- Wikipedia page (quite detailed) `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`
- short and simple Scikit-learn documentation `https://scikit-learn.org/stable/modules/naive_bayes.html`

Naïve Bayes (NB)

- $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$, $y \in \mathcal{Y} = \{1, \ldots, K\}$

Naive Bayes Assumption

Simplifying assumption : given Y, the components $x^1, \ldots, x^d$ are assumed to be independent :

$$p(x|Y = k) = p_k(x) = \prod_{j=1}^{d} p_{k,j}(x^j).$$

Naïve Bayes (NB)

- $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$, $y \in \mathcal{Y} = \{1, \ldots, K\}$

Naive Bayes Assumption

Simplifying assumption : given Y, the components $x^1, \ldots, x^d$ are assumed to be independent :

$$p(x|Y=k) = p_k(x) = \prod_{j=1}^{d} p_{k,j}(x^j).$$

Remarks

- independence reduces one estimation problem in d dimensions to d much simpler 1D estimation problems $\leftarrow$ prevent from curse of dimensionality
- independence assumption is naïve, i.e. not realistic in practice... but yields efficient/stable/robust approaches especially in high dimension !

Naïve Bayes for parametric estimation

Gaussian model

- NB + QDA : $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$, where the Σ_k are diagonal
  - $\hat{\mu}_k$ don't change
  - $(\hat{\Sigma}_k)_{jj} = \frac{1}{n_k - 1} \sum_{y_i = k} (x_i^j - \mu_k^j)^2$
- NB + LDA : $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$, where Σ is diagonal.
  - $\hat{\Sigma}_{jj} = \frac{1}{n - K} \sum_k \sum_{y_i = k} (x_i^j - \mu_k^j)^2$

Naïve Bayes for parametric estimation

Gaussian model

- NB + QDA : $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$, where the Σ_k are diagonal
  - $\hat{\mu}_k$ don't change
  - $(\hat{\Sigma}_k)_{jj} = \frac{1}{n_k - 1} \sum_{y_i = k} (x_i^j - \mu_k^j)^2$
- NB + LDA : $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$, where Σ is diagonal.
  - $\hat{\Sigma}_{jj} = \frac{1}{n - K} \sum_k \sum_{y_i = k} (x_i^j - \mu_k^j)^2$

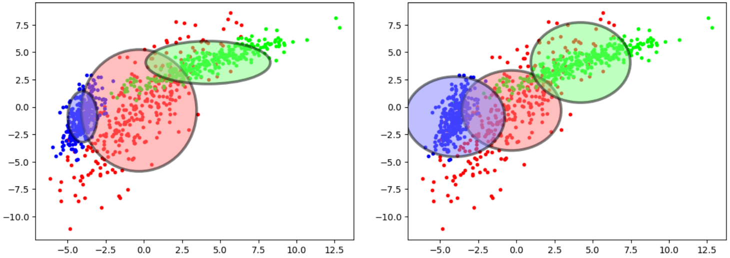
Other classical parametric models

- Bernoulli NB for binary events models (e.g., word occurence vectors in text processing)
- Multinomial NB for multiple events models (e.g., word count vectors in text processing)
- Mixed models (e.g. Gaussian and Multinomial) for mixed quantitative/qualitative features
- ...

NB + QDA example

Mixture of $K = 3$ Gaussians

- Gaussian model : $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$ with $\hat{\Sigma}_k = \begin{pmatrix} (\hat{\Sigma}_k)_{11} & 0 \\ 0 & (\hat{\Sigma}_k)_{22} \end{pmatrix}$

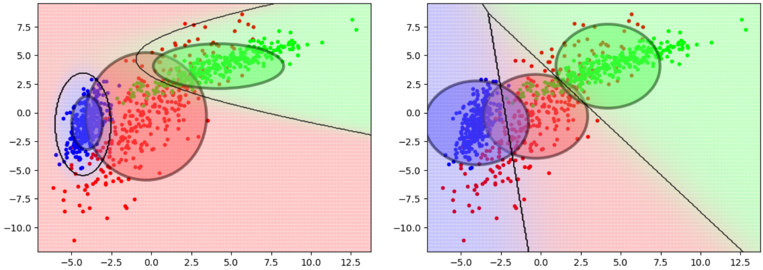


Naive Bayes QDA (left), LDA (right). The Gaussians are "axis-aligned"

Naïve Bayes (NB)

Mixture of $K = 3$ Gaussians

- Classification rule : $\arg\max_{k=1,2,3} \delta_k(x)$
- quadratic boundaries $\{x; \delta_k(x) = \delta_l(x)\}$



Naïve Bayes for non-parametric estimation

Non-parametric estimation of $p_{k,j}(x^j) = p(x^j | Y = k)$, where x^j is the jth component of x : univariate density estimation. Then $\hat{p}_k(x) = \prod_j p_{k,j}(x^j)$, and $\hat{f}(x) = \arg\max_k \hat{\pi}_k$.

Parzen kernel approach

Apply Parzen window to each component, with a univariate kernel (not necessarily the same for each component) :

$$\hat{p}_{k,j}(x^j) = \frac{1}{n_k \lambda} \sum_{y_i = k} k^j\left(\frac{x_j - x_{j,i}}{\lambda}\right)$$

- Avoids the curse of dimensionality, to the price of simplification
- Note : when $n \to \infty$, regular KDE with classic Gaussian kernel is already Naive Bayes ! Since $\frac{1}{(2\pi)^{d/2}} e^{-\|x - x_i\|^2} = \prod_j \frac{1}{\sqrt{2\pi}} e^{-(x^j - x_i^j)^2}$. The only difference is how we compute their empirical versions.

Table of Contents

Conclusions

Generative models

- learning/estimation of $p(X, Y) = p(X|Y)\Pr(Y)$,
- derivation of $\Pr(Y|X)$ from Bayes rule,

Different assumptions on the class densities $p_k(x) = p(X = x|Y = k)$

- QDA/LDA : Gaussian parametric model
  - performs well on many real-word datasets
  - LDA is especially useful when n is small
- Parzen window (aka KDE) : non-parametric
  - more flexible, necessitates a lot of data, poor performance in high-dimension
- Naive Bayes : independence of the feature X components given Y
  - useful when d is very large (high dimension)

Incoming...

Discriminative approaches : direct learning of $\Pr(Y|X)$