Bayes decision rule

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

Costs

Classification

- supervised learning
- $y \in \{1, 2, ...C\}$ takes finite discrete set of values

- λ_{yf} is the cost of predicting true class y with forecasted class f.
- Examples with costs: diagnosis prediction, fraud detection, spam filtering, intrusion detection.

Costs

- Matrix of outcomes:

	$f=1$	$f=2$	$\cdots$	$f=C$
$y=1$	λ_{11}	λ_{12}	$\cdots$	λ_{1C}
$y=2$	λ_{21}	λ_{22}	$\cdots$	λ_{2C}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$y=C$	λ_{C1}	λ_{C2}	$\cdots$	λ_{CC}

- Expected cost of solution $\widehat{y}(x)=f$:

$$\mathcal{L}(f)=\sum_{y}p(y|x)\lambda_{yf}$$

Decision rule

- Which best prediction $\widehat{y}(x)$ for object x to select?

Decision rule

- Which best prediction $\widehat{y}(x)$ for object x to select?

Bayes minimum risk decision rule

Assign class, yielding minimum expected cost:

$$\widehat{y}(x) = \arg\min_{f} \mathcal{L}(f) \tag{1}$$

Decision rule

- Which best prediction $\widehat{y}(x)$ for object x to select?

Bayes minimum risk decision rule

Assign class, yielding minimum expected cost:

$$\widehat{y}(x) = \arg\min_f \mathcal{L}(f) \quad (1)$$

- This rule minimizes expected cost among all rules (if $p(y|x)$ are correct).

Simplifications

- $\lambda_{yf} \equiv \lambda_y \mathbb{I}[y \neq f]$: constant within class cost of misclassification.

Simplifications

- $\lambda_{yf} \equiv \lambda_y \mathbb{I}[y \neq f]$: constant within class cost of misclassification.

Matrix of outcomes:

	$f=1$	$f=2$	$\cdots$	$f=C$
$y=1$	0	λ_1	$\cdots$	λ_1
$y=2$	λ_2	0	$\cdots$	λ_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$y=C$	λ_C	λ_C	$\cdots$	0

Simplifications

- $\lambda_{yf} \equiv \lambda_y \mathbb{I}[y \neq f]$: constant within class cost of misclassification.

Matrix of outcomes:

	$f=1$	$f=2$	$\cdots$	$f=C$
$y=1$	0	λ_1	$\cdots$	λ_1
$y=2$	λ_2	0	$\cdots$	λ_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$y=C$	λ_C	λ_C	$\cdots$	0

- Expected cost of solution $\widehat{y}(x) = f$:
  $\mathcal{L}(f) = \sum_y p(y|x)\lambda_y \mathbb{I}[f \neq y]$

Equal misclassification costs

- Then cost of prediction equals:

$$\mathcal{L}(f) = \sum_y p(y|x)\lambda_y \mathbb{I}[f \neq y] = \overbrace{\sum_y p(y|x)\lambda_y}^{const(f)} - p(f|x)\lambda_f$$

- So (1) becomes:

$$\widehat{y}(x) = \arg\min_f \mathcal{L}(f) = \arg\max_f \lambda_f p(f|x) \tag{2}$$

- Suppose further $\lambda_y \equiv \lambda \, \forall y$, then

$$\widehat{y}(x) = \arg\max_f p(f|x)$$

- This is termed **maximum posterior probability rule** or **Bayes minimum error rule** because it yields minimum probability of misclassification among all decision rules (given that $p(f|x)$ is correct)

Equal misclassification costs

- This rule minimizes expected error rate.
    - if $p(y|x)$ are known

Equal misclassification costs

- This rule minimizes expected error rate.
  - if $p(y|x)$ are known
- If x and y are independent, then (2) reduces to

$$\widehat{y}(x) = \arg\max_f p(f|x) = \arg\max_f p(f)$$

Table of Contents

Gaussian classifier

- In Gaussian classifier
$$p(x|y) = \frac{1}{(2\pi)^{D/2}|\Sigma_y|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y)\right\}$$

Gaussian classifier

- In Gaussian classifier

$$p(x|y) = \frac{1}{(2\pi)^{D/2}|\Sigma_y|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y)\right\}$$

- It follows that

$$\begin{aligned}\log p(y|x) &= \log p(x|y) + \log p(y) - \log p(x) \\ &= -\frac{1}{2}(x-\mu_y^T)\Sigma_y^{-1}(x-\mu_y) - \frac{1}{2}\log|\Sigma_y| \\ &\quad -\frac{D}{2}\log(2\pi) + \log p(y) - \log p(x)\end{aligned}$$

Gaussian classifier

- In Gaussian classifier

$$p(x|y) = \frac{1}{(2\pi)^{D/2}|\Sigma_y|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y)\right\}$$

- It follows that

$$\begin{aligned}\log p(y|x) &= \log p(x|y) + \log p(y) - \log p(x) \\ &= -\frac{1}{2}(x-\mu_y^T)\Sigma_y^{-1}(x-\mu_y) - \frac{1}{2}\log|\Sigma_y| \\ &\quad -\frac{D}{2}\log(2\pi) + \log p(y) - \log p(x)\end{aligned}$$

- Removing common additive terms, we obtain discriminant functions:

$$g_y(x) = \log p(y) - \frac{1}{2}\log|\Sigma_y| - \frac{1}{2}(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y) \quad (3)$$

Practical application

- In practice we replace theoretical terms μ_y, Σ_y with their sample estimates $\widehat{\mu}_y$, $\widehat{\Sigma}_y$.
- $\widehat{p}(y) = \frac{N_y}{N}$.

$$g_y(x) = \log \widehat{p}(y) - \frac{1}{2}\log|\widehat{\Sigma}_y| - \frac{1}{2}(x-\widehat{\mu}_y)^T \widehat{\Sigma}_y^{-1}(x-\widehat{\mu}_y)$$

- Analysis:
  - depends on normality assumptions (in particular - on unimodality)
  - needs to specify:
    - CD parameters to estimate $\widehat{\mu}_y$, $y = 1, 2, ... C$.
    - $CD(D+1)/2$ parameters to estimate $\widehat{\Sigma}_y$, $j = 1, 2, ... C$.

Simplifying assumptions

- $CD(D+3)/2$ may be too large for multidimensional tasks with small training sets.
- Simplifying assumptions:
  - **Naive Bayes**: assume that $\Sigma_1, \Sigma_2, ...\Sigma_C$ are diagonal.
  - **Project data onto a subspace**: for example on first few principal components.
  - **Proportional covariance matrices**: assume that $\Sigma_1 = \alpha_1\Sigma,\ \Sigma_2 = \alpha_2\Sigma,\ ...\ \Sigma_C = \alpha_C\Sigma$.
  - **Fisher's linear discriminant analysis**: assume that $\Sigma_1 = \Sigma_2 = ... = \Sigma_C$.

QDA vs. LDA

Gaussian classifier is called:

- *Quadratic discriminant analysis* (QDA) when $\Sigma_1, \Sigma_2, ...\Sigma_C$ are arbitrary.
    - class boundaries are quadratic[1]
- *Linear discriminant analysis* (LDA) when $\Sigma_1 = \Sigma_2 = ... = \Sigma_C$
    - class boundaries are linear[2]

[1] prove this
[2] prove this

Table of Contents

High dimensional problem

$p(x^1, x^2, ...x^D) = p(x^1)p(x^2|x^1)...p(x^D|x^1, x^2, ...x^{D-1})$

Problem: exponential to D number of observations needed for estimation.

High dimensional problem

$p(x^1, x^2, ...x^D) = p(x^1)p(x^2|x^1)...p(x^D|x^1, x^2, ...x^{D-1})$

Problem: exponential to D number of observations needed for estimation.

Solution: make simplifying assumptions.

High dimensional problem

$p(x^1, x^2, ...x^D) = p(x^1)p(x^2|x^1)...p(x^D|x^1, x^2, ...x^{D-1})$

Problem: exponential to D number of observations needed for estimation.

Solution: make simplifying assumptions.

Independence assumption

Individual features are independent: $p(x) = p(x^1)p(x^2)...p(x^D)$

High dimensional problem

$p(x^1, x^2, ...x^D) = p(x^1)p(x^2|x^1)...p(x^D|x^1, x^2, ...x^{D-1})$

Problem: exponential to D number of observations needed for estimation.

Solution: make simplifying assumptions.

Independence assumption

Individual features are independent: $p(x) = p(x^1)p(x^2)...p(x^D)$

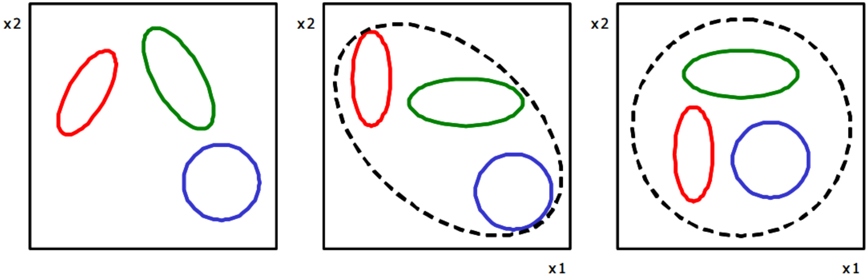
Naive Bayes assumption in classification

Individual features are **class conditionally** independent:
$p(x|y) = p(x^1|y)p(x^2|y)...p(x^D|y)$

Under Naive Bayes assumption max-posterior probability rule becomes:

$$\widehat{y}(x) = \arg\max_y p(y)p(x^1|y)p(x^2|y)...p(x^D|y)$$

Conditional independence visualization



$p(x|\omega_i) \neq \prod_{d=1}^{D} p(x_d|\omega_i)$

$p(x|\omega_i) = \prod_{d=1}^{D} p(x_d|\omega_i)$
$p(x) \neq \prod_{d=1}^{D} p(x_d)$

$p(x|\omega_i) = \prod_{d=1}^{D} p(x_d|\omega_i)$
$p(x) \cong \prod_{d=1}^{D} p(x_d)$

Table of Contents

Text models

- Restrict attention to D words $w_1, w_2, ...w_D$
    - all unique words
    - possibly with stop words removal
    - possibly only most frequent words
    - or only words relevant to the topic of study
- Two major models:
    - Bernoulli
    - Multinomial

Bernoulli model[5]

- Document is represented with feature vector $x \in \mathbb{R}^D$
- $x^i = \mathbb{I}[w_i \text{ appeared in the document}]$
- $\theta_y^d = p(x^d = 1|y)$
- $p(x|y) = \prod_{d=1}^{D} \left(\theta_y^d\right)^{x^d} \left(1 - \theta_y^d\right)^{1-x^d}$
- $p(y) = \frac{N_y}{N}$
- $\theta_y^d = \frac{N_{yx^d}}{N_y}$
- Smoothed variant[3][4]: $\theta_y^d = \frac{N_{yx^d}+\alpha}{N_y+2\alpha}$

[3] interpret this in terms of adding artificial observations
[4] modify for smoothing towards uncoditional word distribution
[5] is it linear classifier?

Multinomial model[8]

- Document is represented with feature vector $x \in \mathbb{R}^D$
- x^d =number of times w_d appeared in the document
- θ_y^d =probability of w_d on word position
- $p(x|y) = \frac{\left(\sum_d x^d\right)!}{\prod_d \left(x^d\right)!} \prod_{d=1}^{D} \left(\theta_y^d\right)^{x^d}$
- $p(y) = \frac{N_y}{N}$
- $\theta_y^d = \frac{n_{yd}}{n_y}$ where
  - n_{yd} - number of times word w_d appeared in documents$\in y$
  - n_y - number of words in documents$\in y$
- Smoothed version[6][7]: $\theta_y^d = \frac{n_{yd+\alpha}}{n_y+\alpha D}$

[6] interpret this in terms of adding artificial observations
[7] modify for smoothing towards uncoditional word distribution
[8] is it linear classifier?