non-parametric methods

Thierry Chateau

Pascal Institute

2017

Plan

Introduction

- Bayes is an optimal classifier (if several assumptions are true) which provides analytical solutions if $P(\omega_i)$ and $p(\mathbf{x}|\omega_i)$ can be formalized.
- In this framework, learn $P(\omega_i)$ and $p(\mathbf{x}|\omega_i)$ is a essential operation.
- We suppose that an annotated dataset is available (feature vector and associated class)
- The aim of the learning step is to estimate both the prior $P(\omega_i)$ and the likelihood $p(\mathbf{x}|\omega_i)$ from a training dataset.
- Several methods exist to do that.

Introduction

- Parametric methods:
  - $p(\mathbf{x}|\omega_i)$ is modelled by a parametric function $f_i(\mathbf{x};\boldsymbol{\theta}_i)$ if a parameter vector $\boldsymbol{\theta}_i$). (example: Gaussian function)
  - Learning consists of estimating the parameter vector $\boldsymbol{\theta}_i$) from a training dataset.
- Non parametric methods:
  - In this case, the pdf. is modelled directly from the training step:

$$p(\mathbf{x}|\omega_i) = f(\mathbf{x}, \text{samples})$$

Parametric models

- We assume that the parametric model (not parameters) of pdf $p(\mathbf{x}|\omega_i)$ known :
  - normal law,
  - Gamma law,
  - ...
- Several solutions exist to estimate the unknowns parameter vector $\boldsymbol{\theta}_i$:
  - Maximum likelihood,
  - Bayesien estimation,
  - ...

Parametric models

- **Rq :** Here, only the maximum likelihood method is presented (and for supervised learning).

Maximum likelihood

- Given $\mathcal{E}_\infty, \mathcal{E}_\in, \mathcal{E}_\rfloor$, c sets of samples that represents the c classes. We assume that
  - the samples are independent
  - sets are representative from $p(\mathbf{x}|\omega_i)$
- **Hypothesis:** The paramtric shape of $p(\mathbf{x}|\omega_i)$ is known, and determined by a parameter vectot $\boldsymbol{\theta}_\mathbf{i}$.
- Example: $p(\mathbf{x}|\omega_i)$ can be approximated by a normal distribution $N(\boldsymbol{\mu}_i, \Sigma_i)$. In this case: $\theta_i = \{\boldsymbol{\mu}_i, \Sigma_i\}$.
- If $\mathbf{x} \in \boldsymbol{R}^d$:

$$\theta_i = \{\mu_1, \mu_2, ..\mu_d, \sigma_{22}, \sigma_{23}, .., \sigma_{2d}, \sigma_{33}, \sigma_{34}, .., \sigma_{3d}, ..., \sigma_{dd}\}$$

Maximum likelihood

- **Notation**: $p(\mathbf{x}|\omega_i;\boldsymbol{\theta}_\mathbf{i})$ stands for $p(\mathbf{x}|\omega_i)$ depends on $\boldsymbol{\theta}_\mathbf{i}$
- **Goal**: Estimating the parameter vector $\boldsymbol{\theta}_\mathbf{i}$
- since the classes are independent, it is possible de have a separate study for each class:

$$\mathcal{E} = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, .., \mathbf{X}_n\}$$

Maximum likelihood

- The probability to draw $\mathcal{E}$, given a class follows a random law defined by the parameter vector $\boldsymbol{\theta}$):

$$p(\mathcal{E}, \boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{X}_k | \boldsymbol{\theta})$$

- We find the an estimation of $\boldsymbol{\theta}$, defined $\hat{\boldsymbol{\theta}}$, which maximizes $p(\mathcal{E}, \boldsymbol{\theta})$

Maximum likelihood

- si $\boldsymbol{\theta} = \{\theta_1, \theta_2, .., \theta_p\}$:

$$\nabla_{\boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \\ \frac{\partial}{\partial \theta_2} \\ . \\ \frac{\partial}{\partial \theta_p} \end{pmatrix}$$

- We define the expression $l(\boldsymbol{\theta}) = \log[p(\mathcal{E}|\boldsymbol{\theta})]$:

$$l(\boldsymbol{\theta}) = \sum_{k=1}^{n} \log[p(\mathbf{X}_k|\boldsymbol{\theta})]$$

Maximum likelihood

- Then,

$$\nabla_{\boldsymbol{\theta}} l = \nabla_{\boldsymbol{\theta}} \sum_{k=1}^{n} \log[P(\mathbf{X}_k|\boldsymbol{\theta})]$$

- $p(\mathcal{E}|\theta)$ will be maximum if:

$$\nabla_{\boldsymbol{\theta}} l = \mathbf{0}$$

Non parametric models

- Parametric model $\rightarrow$ we need to fix the analytic model $p(\mathbf{x}|\omega_i)$:
  - Most of models are unimodal (exept.: mixture of Gaussian)
  - Strong assumption (if not verified, some results can be wrong)
- Parametric models deal directly with samples and their spatial repartition. Les méthodes non paramétriques prennent en compte les échantillons et leur répartition spatiale dans l'espace des paramétres.
- la conséquence et une estimation de $p(\mathbf{x}|\omega_i)$ plus proche de la réalité

Non parametric models

The global framework

- Estimation of $p(\mathbf{x}|\omega_i)$ or $p(\omega_i|\mathbf{x})$
- Given $\mathcal{D}$ a domain of the features space ($\mathcal{D} \subset \boldsymbol{R}^d$) which can be considered as a neighbourhood of the feature vector $\mathbf{x}$ for which we want to estimate $p(\mathbf{x}|\omega_i)$.
- Hypothesis: $p(\mathbf{x}|\omega_i) \approx$ const. on $\mathcal{D}$
- then:

$$p(\mathbf{x} \in \mathcal{D}) = \int_{\mathcal{D}} p(\mathbf{x}'|\omega_i) d\mathbf{x'} \approx p(\mathbf{x}|\omega_i) \int_{\mathcal{D}} d\mathbf{x}'$$

Non Parametric Models

- If $V(\mathcal{D})$ is the hyper-volume of $\mathcal{D}$:

$$p(\mathbf{x} \in \mathcal{D}) \approx p(\mathbf{x}|\omega_i)V(\mathcal{D})$$

- then:

$$\forall \mathbf{x} \in \mathcal{D} p(\mathbf{x}|\omega_i) \approx \frac{p(\mathbf{x} \in \mathcal{D})}{V(\mathcal{D})}$$

- If t samples, on n total ones belong in the domain $\mathcal{D}$, then:

$$p(\mathbf{x} \in \mathcal{D}) \approx \frac{t}{n}$$

Non Parametric Models

- and: $\boxed{p(\mathbf{x}|\omega_i) \approx \frac{\frac{t}{n}}{V(\mathcal{D})}}$
- Given $\mathbf{x}_0$ a feature and $D(\mathbf{x}_0)$ a domain around the deature $\mathbf{x}_0$; we want to build an estimator $\hat{P}(\mathbf{x}_0|\omega)$. So,

$$\hat{P}(\mathbf{x}_0|\omega) \approx \frac{\frac{t}{n}}{V(\mathcal{D}(\mathbf{x}_0))}$$

Non Parametric Models

- Good estimator $\rightarrow$ converging estimator:

$$\lim_{n\rightarrow\infty} \hat{P}(\mathbf{x}_0|\omega) = P(\mathbf{x}_0|\omega)$$

$$\lim_{n\rightarrow\infty} \frac{\frac{t}{n}}{V(\mathcal{D}(\mathbf{x}_0))} = P(\mathbf{x}_0|\omega)$$

- The estimator smooths the probability on the neighbourhood of $\mathbf{x}_0$
- The lower $\mathcal{D}(\mathbf{x}_0)$ is, the better the estimator is (closer to the true value)

Non Parametric Models

- If $\mathcal{D}(\mathbf{x}_0)$ is too small, many domains won't have any sample, resulting to $p(\mathbf{x}|\omega_i) = 0$.
- It is necessary to link the number of samples to the size of the domain. (n and $\mathcal{D}(\mathbf{x}_0)$, will follow the notation $\mathcal{D}_n(\mathbf{x}_0)$) :

$$\hat{P}(\mathbf{x}_0|\omega) = \frac{\frac{t_n}{n}}{V_n(\mathcal{D}(\mathbf{x}_0))}$$

Non Parametric Models

There are 3 necessary conditions for a converging estimator:

$$\hat{p}_n(\mathbf{x}_0|\omega) \to p(\mathbf{x}_0|\omega) \text{ si :}$$

1. $\lim_{n\to\infty} V(\mathcal{D}_n(\mathbf{x}_0)) = 0$
2. $\lim_{n\to\infty} t_n = +\infty$
3. $\lim_{n\to\infty} \dfrac{t_n}{n} = 0$

Non Parametric Models

There are two main categories of non parametric models:

1. link $V(\mathcal{D}_n(\mathbf{x}_0))$ to n : this is kernel based models (Kernel Density Estimation also called Parzen windows)
2. link t_n according to n adjusting the domain $V(\mathcal{D}_n(\mathbf{x}_0))$ until k samples belong to it: this is the k nearest neighbours models

KDE models

also called Parzen Windows

- Main idea: working with the domain function $\mathcal{D}_n(\mathbf{x}_0)$
- example with a hypercube of side h_n :

$$V(\mathcal{D}_n(\mathbf{x}_0)) = (h_n)^d$$

- the features dimension is $\boldsymbol{R}^d$
- We define a function $\varphi(\mathbf{u})$, egal to 1 inside the unit and centered on the origin hypercube:

KDE models

$$\begin{cases} \varphi(\mathbf{u}) = 1 \text{ si } |u_j| \leq 0.5 \; j = 1, .., d \\ \varphi(\mathbf{u}) = 0 \text{ sinon} \end{cases}$$

If $\mathcal{D}_n(\mathbf{x}_0)$ is a hypercube of side h_n, then:

$$\mathbf{x} \in \mathcal{D}_n(\mathbf{x}_0) \Leftrightarrow \varphi\left(\frac{\mathbf{x}_0 - \mathbf{x}}{h_n}\right) = 1$$

KDE models

The number of samples t_n that belongs to the domain $\mathcal{D}_n(\mathbf{x}_0)$ is computed by the following equation:

$$t_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}_0 - \mathbf{x}_i}{h_n}\right)$$

where $\mathbf{x}_i$ is the sample i.

$$\hat{P}_n(\mathbf{x}_0|\omega) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V(\mathcal{D}_n(\mathbf{x}_0))}\,\varphi\left(\frac{\mathbf{x}_0 - \mathbf{x}_i}{h_n}\right)$$

The function φ is called **kernel of the estimator**

Kernels

- The kernel must be normalized:

$$\begin{cases} \varphi(\mathbf{x}) \geq 0 \ \forall \mathbf{x} \in \boldsymbol{R}^d \\ \int_{\boldsymbol{R}^d} \varphi(\mathbf{x}) d\mathbf{x} = 1 \end{cases}$$

- Samples of kernels
  - cubic kernel,
  - triangular kernel,
  - normal kernel,
  - exponential kernel,
  - ...

KNN models

Knn models are widely used

- Main idea: fit the domain size according to the neighbourhood of $\mathbf{x}_0$
- A fix number t_n must belongs to the domain (it ensures that we cannot have 0 samples into the domain resulting to non-zeros value for $\hat{p}_n(\mathbf{x}_0)$)
- Given $\mathcal{D}_r(\mathbf{x}_0)$ a unit-volume and $\mathbf{x}_0$ centered domain:

$$V[\mathcal{D}_r(\mathbf{x}_0)] = 1$$

- Given $\mathcal{D}(\mathbf{x}_0, \alpha)$ the homothetic domain of $\mathcal{D}_r(\mathbf{x}_0)$ centered on $\mathbf{x}_0$ with the ratio α.

KNN models

- Then :

$$V[\mathcal{D}(\mathbf{x}_0,\alpha)] = \alpha^d$$

- The KNN model consists of increasing α until $\mathcal{D}(\mathbf{x}_0,\alpha)$ includes t_n samples.
- The probability density function estimator is computed by the following function:

$$\hat{p}_n(\mathbf{x}_0) = \frac{\frac{t_n}{n}}{V[\mathcal{D}(\mathbf{x}_0,\alpha)]}$$

KNN models

- This estimator converges toward the true value of $p_n(\mathbf{x}_0)$ if:

$$t_n = t_0 * \sqrt{n} \text{ ou } t_n = t_0 * logn$$

- with t_0 : a parameter (to adjust)

KNN models for posterior estimation

- In non-parametric models used for classification into a Bayesian framework, the goal is to estimate the posterior distribution $p(\mathbf{x}|\omega_i)$ from the samples.
- Samples encodes:
  - Prior probabilities $P(\omega_i)$,
  - likelihoods.
- **Demonstration:**
- Given n samples drawing from the classes. K_i samples are drawn from the class ω_i. So:

$$n = \sum_{i=1}^{c} K_i$$

Posterior distribution estimation

- Given V, a volume around $\mathbf{x}$, and k samples which belong to this volume with k_i samples associated to the class ω_i. We can write:

$$P(\omega_i) = \frac{K_i}{n}$$

$$p(\mathbf{x}|\omega_i) = \frac{\frac{k_i}{K_i}}{V}$$

So:

$$p(\mathbf{x}|\omega_i).P(\omega_i) = \frac{\frac{k_i}{n}}{V}$$

Estimation of posterior distribution

- Let apply the Bayes rule:

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i).P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j)} \approx \frac{\frac{\frac{k_i}{n}}{V}}{\sum_{j=1}^{c} \frac{\frac{k_i}{n}}{V}}$$

- Finally, the posterior probability is estimated by the ratio between the number of samples associated to the class ω_i by the number of samples into the volume.

$$p(\omega_i|\mathbf{x}) \approx \frac{k_i}{k}$$

Classification rule based on samples

- Bayesian models assume the estimation of the likelihood $p(\mathbf{x}|\omega_i)$ follow by the Bayes rule.
- Using KNN, it is possible to define decision rules from the samples:
  - Decision rule of the nearest neighbour,
  - Decision rule of the k nearest neighbours.

Decision rule of the nearest neighbour

- This method assumes that a distance measure between the features is possible and can be defined.
- The unknown feature is then classified with the same class than the nearest feature.

Decision rule of the k nearest neighbours

- Just an extension of the the nearest neighbour decision rule,
- Given an unknown feature $\mathbf{x}$:
- Given a distance measure between $\mathbf{x}$ and all the samples of a supervised dataset,
- We sort feature according to their distances to $\mathbf{x}$ and keep only the q samples associated to the q smallest distances.
- $\mathbf{x}$ is associated to the majority class inside the subset build by the q selected samples

Conclusion

- KDE is a very useful way to estimate probability density functions.
- KNN is very popular for simple classification problems.
- Both models are $O(N^2)$ and complexity must be handle for huge datasets (eg: kdtrees of fast sort methods)