Lecture Pattern Analysis

Part 02: Non-Parametric Density Estimation

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
April 19, 2021

Introduction

- Density Estimation = create a PDF from a set of samples
- The lecture Pattern Recognition introduces parametric density estimation:
  - Here, a parametric model (e.g., a Gaussian) is fitted to the data
  - Maximum Likelihood (ML) estimator:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \boldsymbol{\theta}) \tag{1}$$

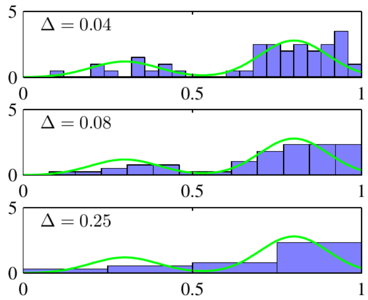
  - Maximum a Posteriori (MAP) estimator:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta} | \mathbf{x}_1, \ldots, \mathbf{x}_N) \overset{\text{Bayes}}{=} \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathbf{x}_1, \ldots, \mathbf{x}_N)} \tag{2}$$

  - Browse the PR slides if you like to know more
- Parametric density estimators require a good function representation
- Non-parametric density estimators can operate on arbitrary distributions

Non-Parametric Density Estimation: Histograms

- Non-parametric estimators do not use functions with a limited set of parameters
- A **simple non-parametric baseline** is to create a histogram of samples[1]
  - The number of bins is important to obtain a good fit



  - Pro: Good for a quick visualization
  - Pro: "Cheap" for many samples in low-dimensional space
  - Con: Discontinuities at bin boundaries
  - Con: Scales poorly to high dimensions (cf. curse of dimensionality later)

[1] See introduction of Bishop Sec. 2.5

FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Improving on the Histogram Approach

- A kernel-based method and a nearest-neighbor method are slightly better
- Both variants share their mathematical framework:
  - Let $p(\mathbf{x})$ be a PDF in D-dim. space, and R a small region around $\mathbf{x}$
    $\rightarrow$ The probability mass in R is $p = \int_R p(\mathbf{x})\,\mathrm{d}\mathbf{x}$
  - Assumption 1: in R are many points $\rightarrow$ p is a relative frequency,

$$p = \frac{\#\text{ points in } R}{\text{total } \#\text{ of points}} = \frac{K}{N} \tag{3}$$

  - Assumption 2: R is small enough s.t. $p(\mathbf{x})$ is approximately constant,

$$p = \int_R p(\mathbf{x})\,\mathrm{d}\mathbf{x} = p(\mathbf{x}) \int_R \mathrm{d}\mathbf{x} = p(\mathbf{x}) \cdot V \tag{4}$$

  - Both assumptions together are slightly contradictory, but they yield

$$p(\mathbf{x}) = \frac{K}{N \cdot V} = \frac{\#\text{ points in } R}{\text{total } \#\text{ of points} \cdot \text{Volume of } R} \tag{5}$$

Kernel-based DE: Parzen Window Estimator (1/2)

- The Parzen window estimator fixes V and leaves K/N variable[2]
- D-dimensional Parzen window kernel function (a.k.a. “box kernel”):

$$k(\mathbf{u}) = \begin{cases} 1 & \text{if } |u_i| \leq \frac{1}{2} \quad \forall i = 1, \ldots, D \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

- Calculate K with this kernel function:

$$K(\mathbf{x}) = \sum_{i=1}^{N} k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{7}$$

  where h is a scaling factor that adjusts the box size
- Hence, the whole density is

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{8}$$

[2] See Bishop Sec. 2.5.1

Kernel-based DE: Parzen Window Estimator (2/2)

- The kernel removes much of the discretization error at histogram bins, but is still “blocky”
- Using a Gaussian kernel instead of a box kernel further smooths the result:

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\pi h^2}^{D/2} \cdot \exp\left(-\frac{\mathbf{x} - \mathbf{x}_i}{2h^2}\right) \tag{9}$$

  where h^2 is the standard deviation of the Gaussian
- Mathematically, also any other kernel is possible if these conditions hold:

$$k(\mathbf{u}) \geq 0 \tag{10}$$

$$\int k(\mathbf{u})\, \mathrm{d}\mathbf{u} = 1 \tag{11}$$

K-Nearest Neighbors (k-NN) Density Estimation

- Recall our derived equation for estimating the density

$$p(\mathbf{x}) = \frac{K}{N \cdot V} = \frac{\text{\# points in } R}{\text{total \# of points} \cdot \text{Volume of } R} \tag{12}$$

- The Parzen window estimator fixes V, and K varies
- The k-Nearest Neighbors estimator fixes K, and V varies
- k-NN calculates V from the distance of the K nearest neighbors[3]
- Note that both the Parzen window estimator and the k-NN estimator are "non-parametric", but they are not free of parameters
  - The kernel scaling h and the number of neighbors k are **hyper-parameters**, i.e., prior knowledge to guide the model creation
  - The model parameters are the samples themselves. Both estimators need to store all samples, which is why they are also called **memory methods**

[3] See Bishop Sec. 5.2.2

First Glance at the Model Selection Problem

- Optimizing the hyperparameters is also called **Model Selection Problem**
- Supervised methods use cross validation (CV) via Maximum Likelihood (ML) $\rightarrow$ PR lecture
- We can use CV to optimize the DE hyperparameters by using the **prediction of held-out samples** as objective function:
  - Split the data into J folds:
    $S^j_{\text{train}} = S \setminus \{x_{\lfloor \frac{N}{J} \rfloor \cdot j}, \ldots x_{\lfloor \frac{N}{J} \rfloor \cdot (j+1)-1}\}$,
    $S^j_{\text{test}} = S \setminus S^j_{\text{train}}$
  - Let $\boldsymbol{\alpha}$ be the unknown hyperparameters, and let $p_j(\mathbf{x}|\boldsymbol{\alpha})$ be the density estimate for samples S^j_{train} on hyperparams $\boldsymbol{\alpha}$
  - Then, the ML estimate is

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \prod_{j=1}^{J} \prod_{\mathbf{x} \in S^j_{\text{test}}} p_j(\mathbf{x}|\boldsymbol{\alpha}) \tag{13}$$

  - In practice, take the logarithm ("log likelihood") to mitigate numerical issues $\rightarrow$ the product becomes a sum