pdf. estimation (non parametric models)

1 Introduction

The objective of this session is to practice with probability density function estimation and classification using non parametric models, following the Machine Learning course. You can get more information about lectures and associated practical sessions on my website: `http://chateaut.fr`. Essential needed knowledges include a beginning level of computer science (Linux, Python programming with `Sklearn`, `Numpy`, `Matplotlib`, `Opencv` and `Pytorch` (for sessions on Deep Learning)) and applied mathematics. this practical uses the same datasets (flowers) than practical one.

Given $\mathbf{x}_0$ a feature and $D(\mathbf{x}_0)$ a domain around the feature $\mathbf{x}_0$; we want to build an estimator $\hat{P}(\mathbf{x}_0|\omega)$. So,

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

t is the number of samples which belongs to the domain $\mathcal{D}$ and n is the total number of samples. 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 (KNN)

2 Kernel Density Estimation Model

2.1 Estimating likelihood with KDE

The script `ML_p21.py` should be used for this section.

Here, we want to estimate the probability density function associated to the two classes of flowers using a kernel density estimation model. The kernel density estimator is defined by:

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

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

The KDE model is defined on `sklearn` python library by the function:
`sklearn.neighbors.KernelDensity`

1. complete the python script `ML_p21.py` that loads datasets, computes features, estimates KDE and displays it. The options `kernel='gaussian'` and `bandwidth=0.5` will be used.

2. The option kernel defines the kernel to use. Valid kernels are ['gaussian'|'tophat'|'epanechnikov'|'exponential'|'
Default is 'gaussian'. Select at least three kernels, give their expression, and compare it.

3. The option bandwidth selects the bandwidth of the kernel. Try several values from 0.01 to 5 and conclude on the influence of this parameter.

2.2 Bayesian Classification using KDE

Since KDE model can estimate likelihood function, it can be used into a Bayes rule to estimate posterior distribution for a classification problem.

1. run the script `ML_p21C.py` and analyse the KDE decision function figure for $banwidth = 0.25$.

2. change the bandwidth of the estimator to 0.1, 0.5 and 1 and compare the resulting decision function.

3 KNN Model

3.1 Estimating likelihood with KNN

The probability density function estimator for KNN is computed by the following function:

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

with t_n : number of samples into the domain $\mathcal{D}$, n total number of samples and α a scale factor. The main idea of KNN model is to adjust α in order to include t_n samples into the domain $\mathcal{D}$.

1. complete the python script `ML_p22.py` that loads datasets, computes features, estimates KNN pdf and displays it. the option `n_neighbors=x` defines the number of neighbours t_n. Use $t_n = 1$

2. Change t_n to 2, 4 and 6 and compare the resulting pdf.

3.2 Classification with KNN

KNN is widely used for classification. The posterior distribution with the Bayes Rule can be written using samples:

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

and can be simplify by:

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

1. run the python script `ML_p22C.py` that loads datasets, computes features, estimates KNN pdf and displays the posterior for class1.
2. change the number of neighbour and compare the resulting posterior

4 Additional questions

1. Modify the script `ML_p22C.py` to compute the classification error on the test database.

Listing 1: Python script to display a 2d normal distribution in a 3D view

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 11 07:56:48 2018
Practical 1 on Machine Learning Course, part 1:
python script to display in a 3D view a 2-d normal law
@author: thierrychateau
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from mpl_toolkits.mplot3d import Axes3D

# plot a covariance 2D law in 3D
def plotGaus(mu,C, min_x=-2, max_x=2, min_y=-2,max_y=2):
    #Create grid and multivariate normal
    x = np.linspace(min_x,max_x,200)
    y = np.linspace(min_y,max_y,200)
    X, Y = np.meshgrid(x,y)
    pos = np.empty(X.shape + (2,))
    pos[:, :, 0] = X; pos[:, :, 1] = Y
    rv = multivariate_normal(mu, C)
    #Make a 3D plot
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    ax.plot_surface(X, Y, rv.pdf(pos),cmap='viridis',linewidth=0)
    ax.set_xlabel('X axis')
    ax.set_ylabel('Y axis')
    ax.set_zlabel('Z axis')


def main():
    #Parameters to set
    mu_x = 0
    mu_y = 0
```

```
    mu = np.array([mu_x, mu_y])
    C = np.array([[1 ,  1], \
                  [1 ,  2]])
    print("cov matrix = ", C)
    plotGaus(mu,C)
    plt.show()


if __name__ == "__main__":
    # execute only if run as a script
    main()
```