Normal distributions and Naive Bayes classifier

1 Introduction

The objective of this session is to practice with Normal distributions and Naive Bayes classifiers, 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 includes a beginning level of computer science (Linux, Python programming with `Sklearn`, `Numpy`, `Matplotlib`, `Opencv` and `Pytorch` (for sessions on Deep Learning)) and applied mathematics.

2 Normal distributions

The general equation for a d dimensions normal distribution, denoted $N(\boldsymbol{\mu}, \Sigma)$ is:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{1}$$

with :

- $\boldsymbol{\mu}$: mean vector
$$\boldsymbol{\mu} = E[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x}) d\mathbf{x}$$
- Σ : covariance matrix
$$\Sigma = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})']$$

For a 2 dimensional distribution, the shape of the resulting curve is related to the parameters of the distribution: the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ. There are several specific normal distributions, listed below

2.1 Independent normal distribution with same variance along each component

This is a special case in which each component of the ramdom vector independant and with the same variance. The Covariance matrix is written by:

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 & . & 0 \\ 0 & \sigma^2 & . & 0 \\ . & . & . & 0 \\ 0 & . & 0 & \sigma^2 \end{pmatrix}$$

The python script `ML_p11.py`, presented on listing 1 page **??** displays a 2D normal distribution from a given mean vector and covariance matrix.

You have to modify this script to show the shape of the normal distribution for the following set of parameters:

- $\boldsymbol{\mu} = (0.5, 0.5)^T$ and $\sigma^2 = 0.2$
- $\boldsymbol{\mu} = (0, 0)^T$ and $\sigma^2 = 0.4$
- $\boldsymbol{\mu} = (-0.5, 0.5)^T$ and $\sigma^2 = 0.6$

2.2 Independent normal distribution with different variance on each component

For an independent normal distribution (diagonal covariance matrix) with different variance on each component, the covariance matrix is written:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & . & 0 \\ 0 & \sigma_2^2 & . & 0 \\ . & . & . & 0 \\ 0 & . & 0 & \sigma_d^2 \end{pmatrix}$$

You should modify `ML_p11.py` to visualize the shape of the normal distribution for a mean value $\boldsymbol{\mu} = (0.5, 0.5)^T$ and:

1. $$\Sigma = \begin{pmatrix} 0.3 & 0 \\ 0 & 0.1 \end{pmatrix}$$
2. $$\Sigma = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.3 \end{pmatrix}$$

2.3 The generic normal distribution

The generic expression of the parameters associated to a normal distribution is:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ . \\ \mu_d \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12}^2 & . & \sigma_{1d}^2 \\ \sigma_{21}^2 & \sigma_2^2 & . & \sigma_{2d}^2 \\ . & . & . & \sigma_{(d-1)d}^2 \\ \sigma_{d1}^2 & . & \sigma_{d(d-1)}^2 & \sigma_d^2 \end{pmatrix}$$

You should modify `ML_p11.py` to visualize the shape of the normal distribution for a mean value $\boldsymbol{\mu} = (0.5, 0.5)^T$ and:

1. $$\Sigma = \begin{pmatrix} 1 & 0.6 \\ 0.6 & 0.7 \end{pmatrix}$$
2. $$\Sigma = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

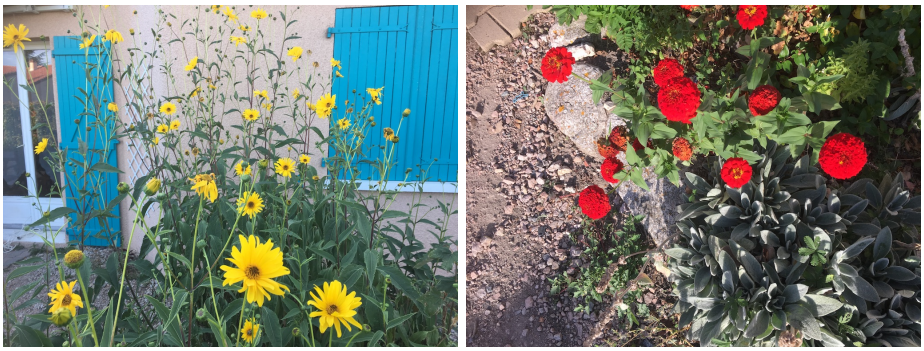

Figure 1: samples of images for the two classes of flowers that should be classified.

3 Gaussian distribution and Naive Bayes Classifier: application to flower classification

This section illustrates, on a toy example (flower classification) how to build gaussian distribution from real datas and classify an unknown sample using a Naive Bayes classifier.

3.1 Training the classifier

Figure 1 illustrates the two classes of flowers and figure 2 shows an example of extracted flowers from acquired images. Both a training and testing dataset have been created. Each class has 10 samples in each dataset. The learning process is divided into two steps:

1. Feature extraction: a feature vector should be created for each sample. There exists two categories of features: 1) hand-crafted features and 2) Deep Neural Network based features. Here we focus on the first category. It seems that colour is a good way to classify each class. We use a 2D feature vector composed by the mean red and green components of the image.

2. Estimate parameters of the gaussian pdf distribution associated to each class (likelihood): as we have written before, the parameters are given by the mean vector and the covariance matrix, computed from the training set.

   The script `ML_p12.py` loads samples, computes features from a dataset and displays both a feature space representation of the training set and a 3D representation of the normal distributions associated to the two classes of the training set.

   You should complete this script in order to compute the covariance matrices and the mean vector from the feature vectors **X1** for class 1 and **X2** for class 2.

3.2 Testing the naive Bayes classifier

The naive bayes classifier is a simple but powerfull framework. It needs a parametric representation of probability density functions (here we use a gaussian model). Using this parametric representation, the Bayes rule can be applied:

$$p(\omega_i|\mathbf{X}) = \frac{p(\mathbf{X}|\omega_i)p(\omega_i)}{p(\mathbf{X})}$$

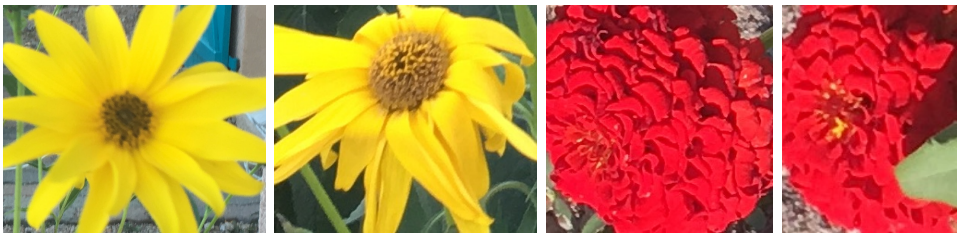

Figure 2: samples of flowers extracted from the original images: the two left images for class 1 and the two right images for class 2

with $\{\omega_1, \omega_2\}$ the two classes, $p(\mathbf{X}|\omega_i)$ the likelihood function (here the gaussian pdf distribution computed before) and $p(\omega_i)$ the prior.

The script `ML_p13.py` loads samples, computes features from a dataset, applies a Gaussian Naive Bayes classifier (with the library `sklearn`) and displays a 2D plot in the feature space of the samples and the decision regions. Moreover, a 3D decision function curve is displayed.

1. You should run this script and conclude on the performance of the classifier.
2. Now we will work on a second dataset (please change both `training_images_path` and `testing_images_path`) to `dataset2`. Run the script and conclude on the performances of the classifier. What is the problem?
3. In order to improve the classifier, let convert the RGB colour space to HSV (Hue, Saturation, Value) space and keep only a 2D mean (H, S) feature vector. The simplest way is to change the flag `cv2.COLOR_BGR2RGB` in the command `imrgb = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)` by the flag `cv2.COLOR_BGR2HSV`. Run the script and compare the performances of the new features with the original ones.

4 Additional questions

1. Dataset 2 has 3 classes. Try to build a script that creates a Gaussian Naive Bayes classifier for three classes and to compute performance (using the testing dataset) of the classifier. Graphical display is not needed here.
2. Try to improve the feature extraction step (you may add components such high order moments (variances) into the feature vector)

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

#Parameters to set
mu_x = 0
variance_x = 3

mu_y = 0
variance_y = 15

#Create grid and multivariate normal
x = np.linspace(-10,10,500)
y = np.linspace(-10,10,500)
X, Y = np.meshgrid(x,y)
pos = np.empty(X.shape + (2,))
pos[:, :, 0] = X; pos[:, :, 1] = Y
rv = multivariate_normal([mu_x, mu_y], [[variance_x, 0], [0, variance_y]])

#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')
plt.show()
```