Samy Labsir

AI introduction

TIn-324, 3rd year SET, IPSA

Project (12 H)

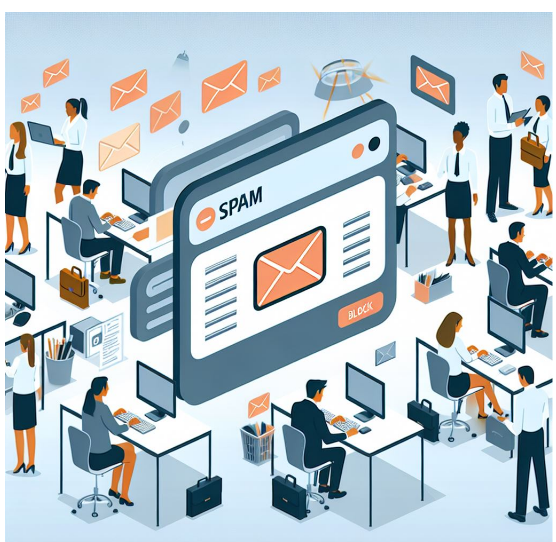
This tutorial aims to study the different methods covered in the course: the KNN algorithm and the Bayesian classification rule.

The goal of this project is to implement, test, and compare the machine learning algorithms studied in class, on different datasets. The project will be divided into two parts:

- *KNN and statistical classifier on SPAM Dataset: in the first part, K-Nearest Neighbors (KNN), Naive Bayes and LDA will be developed and tested on a SPAM dataset.*
- *Comparison with Logistic Regression: In the second part, more complex data will be utilized, and the previously developed algorithms will be compared to a pioneering neural network method: logistic regression.*

ChatGPT/Copilot/MistralAI/DeepSeek, etc... can be used ONLY to obtain help/assistance for misunderstandings. Using any of these AIs to generate Python code or write the report will result in a zero score !

1 First part: SPAM detection



In the first part, we aim to address the problem of detecting the presence of spam o in received emails. To solve this issue, we will develop and test classification algorithms specifically designed to classify a message in a spam or ham. To address this, we propose to use a dataset provided by the Irvine university at the following adress: `https://archive.ics.uci.edu/dataset/228/sms+spam+collection`*.*

1.1 Pre-processing

1) Open the file "PartI.py": this is the Python file that you have to complete.

   In the header, a code snippet for loading the dataset is provided. The dataset contains two types of data: messages classified as SPAM (class 0) and trusted messages, also known as HAM (class 1).

2) Understand and explain the different steps of the data loading process.

   To process the SPAM data, it is necessary to extract the information from each message and convert it into a numerical format.

   For instance if the data is constituted with two sentences "It is false", "Is is true", the vocabulary is $\{$"it, "is", "$false$", "$true$"$\}$. *We want to transform each sentence into a row containing the number of times each word from the vocabulary appears.*
   "It is false" $\Rightarrow [1, 1, 1, 0]$ "It is true" $\Rightarrow [1, 1, 0, 1]$

3) Use the **Vectorizer** object to transform the SPAM data into a set of values $\{\boldsymbol{x}_i\}_{i=1}^N$, where each value represents the token counts for each data point. What is the dimension P of $\mathbf{x}_i$?

4) Use the "train-test-split" function to divide the dataset into training and test sets, allocating **80% of the data for training and 20% for testing**.

1.2 Algorithms implementation

Now, we propose to classify the processed data using various classification algorithms. Given the training dataset $\{\mathbf{x}_i\}_{i=1}^N$ *with known classes* $\{y_i\}_{i=1}^N \in \{\underbrace{0}_{ham}, \underbrace{1}_{spam}\}^N$*, we aim to predict the class of the test data* $\mathbf{x}^\star$*.*

1.2.1 KNN

To begin, we want to implement the K-Nearest-Neighbor (KNN) algorithm, which finds the nearest training data neighbor of test data according to a specified distance metric. To achieve this, a class called KNN has been pre-created and consists of three methods.

5) Identify the class "KNN" in the code "PartI.py". Complete the method "predict" which for any test data implement the KNN by predicting the class **spam** or **ham**, based on a Euclidean distance

6) By using the function “accuracy-score”, compute the performance of the prediction on the test data. Varying the number of training data, what do you observe ?

1.2.2 Naive Bayes

Now, we propose to implement a statistical method called Naive Bayes. This method assumes the following model for posterior distribution of the training data $\mathbf{x}_i = \left[x_i^1, \dots, x_i^P\right]^\top$

$$\mathbb{P}(y_i = k|\mathbf{x}_i) \propto \left(\prod_{p=1}^{P} p(x_i^p|y_i = k)\right) \underbrace{\mathbb{P}(y_i = k)}_{\pi_k} \quad \forall k \in \{0, 1\} \tag{1}$$

7) Explain what properties allows us to obtain the equation (1).

In the following, we assume that $p(x_i^p|y_i = k)$ *is a* ***Gaussian distribution*** *with mean* μ_k^p *and variance* $(\sigma_k^p)^2$*.*

8) By assuming the independence of each $\mathbf{x}_i$, provide the expression for the posterior distribution $\mathbb{P}(y_1, \dots, y_N|\mathbf{x}_1, \dots, \mathbf{x}_N)$. Explain how to estimate μ_k^p, $(\sigma_k^p)^2$, and π_k.

We can show that the estimators are given by

$$\widehat{\mu}_k^p = \frac{1}{N_k} \sum_{i, y_i = k}^{N_k} x_i^p, \quad (\widehat{\sigma}_k^p)^2 = \frac{1}{N_k} \sum_{i: y_i = k} (x_i^p - \widehat{\mu}_k^p)^2, \quad \widehat{\pi}_k = \frac{N_k}{N} \tag{2}$$

$N_k = \mathrm{card}(\{\mathbf{x}_i|y_i = k\})$.

Once that the estimators are computed, they are used to predict the class $y^\star$ *of* $\mathbf{x}^\star = \left[x^{*1}, \dots, x^{*P}\right]$ *by maximizing the posterior distribution*

$$p(y^\star = k|\mathbf{x}^\star) \propto \left(\prod_{p=1}^{P} p(x^{\star p}|y = k)\right) \widehat{\pi}_k \;\; \forall k \in \{0, 1\}.$$

9) Demonstrate that:

$$y^\star = \underset{k \in \{0,1\}}{\mathrm{argmax}} - 0.5 \sum_{p=1}^{P} \left(\frac{(x^{\star p} - \widehat{\mu}_k^p)^2}{(\widehat{\sigma}_k^p)^2} + \log(\sigma_p^2)\right) + \log(\widehat{\pi}_k) \tag{3}$$

Now, locate the “NaiveBayes” class in the code. This class includes three methods: “fit”, “decision”, and “predict”. It also has four attributes: the three sets of parameters to learn (self.prior, self.variances, self.means) and the number of classes (self.classes).

10) Complete the method “fit” implementing the estimators $\widehat{\mu}_{p,k}$, $(\widehat{\sigma}_k^p)^2$ and $\widehat{\pi}_k$.

11) Complete the method “predict” implementing the rule decision given by the equation (3).

12) In the same way as for the KNN, compute the prediction performance on the **test data**. Interpret and explain the results. How we could improve the performance ?

1.2.3 LDA

Now, we propose using a more sophisticated method than Naive Bayes, called Linear Discriminant Analysis (LDA). Recall that LDA is based on the following modeling of the likelihood each data point:

$$p(\mathbf{x}_i|y_i = k) = \frac{1}{\sqrt{(2\,\pi)^p\,|\boldsymbol{\Sigma}_k|}} \exp\left(-0.5\,(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}\,(\mathbf{x}_i - \boldsymbol{\mu}_k)\right) \tag{4}$$

To simplify, we assume that $\boldsymbol{\Sigma}_1 = .. = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}$.

13) Give the new expression of the posterior $\mathbb{P}(y_1, \ldots, y_N|\mathbf{x}_1, \ldots, \mathbf{x}_N)$

14) Demonstrate that the estimator of $\boldsymbol{\mu}_k$ maximizing the posterior distribution is given by

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i:y_i=k} \mathbf{x}_i, \tag{5}$$

In a intuitive way, give the expression of $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\pi}_k$.

15) Now, deduce that the new rule decision to predict the class of $\mathbf{x}^\star$ is given by

$$y^\star = \underset{k\in\{0,1\}}{\operatorname{argmax}}\ -0.5\,(\mathbf{x}^\star - \widehat{\boldsymbol{\mu}}_k)^\top \widehat{\boldsymbol{\Sigma}}^{-1}\,(\mathbf{x}^\star - \widehat{\boldsymbol{\mu}}_k) + \log\left(\widehat{\pi}_k\right) \tag{6}$$

Now, identify, in the code, the class "LDA". It consists in three methods: the constructor, the method "predict". and the method "fit".

16) In the function "fit", implement the estimators $\widehat{\boldsymbol{\mu}}_k$, $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\pi}_k$.

17) In the function "predict", implement the rule decision given by (3).

18) Now, as before, compare the performance obtained with KNN and Naive Bayes. Interpret and comment on the results.

19) Study the accuracy of the three methods for different set of training data.

2 Second part: Breast cancer detection

In this section, we propose to work with a new dataset. It contains medical information about several patients who are potentially affected by breast cancer. This informations are features (30) computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and describe the characteristics of the cell nuclei present in the image.

Based on these features, a diagnosis has been made to determine whether the tumor is malignant or benign. If the tumor is malignant, the features are associated with class 1; if benign, they are associated with class 0. Then, we aim to use this data to automatically predict whether a patient is affected by this pathology.

2.1 Pre-processing

1) Open the file "PartII.py": **this is the Python file that you have to complete now.**

2) Load the data thanks to the command "load_breast_cancer": retrieve the features $\{\mathbf{x}_i\}_{i=1}^N$ and the associated class $\{y_i\}_{i=1}^N \in \{0, 1\}$.

3) Normalize the features by using the function "StandardScaler".

2.2 Logistic regression: a bit of theory..

We propose in this section, to implement a new classification method called logistic regression. The idea of this approach is always to assume a statistical model but directly parametrized on the posterior distribution

$$\mathbb{P}(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}) = \frac{1}{1 + \exp\left(-\left(\theta^0 + \sum_{p=1}^{P} \theta^p\, x_i^p\right)\right)} \tag{7}$$

where $\boldsymbol{\theta} = \{\theta^0, \theta^1, \ldots, \theta^P\} \in \mathbb{R}^{P+1}$ is the set of the unknown parameters to learn.

4) What is the value of $\mathbb{P}(y_i = 0|\mathbf{x}_i, \boldsymbol{\theta})$? Demonstrate that the posterior follows a binomial distribution i.e

$$\mathbb{P}(y_1, \ldots, y_N|\mathbf{x}_1, \ldots, \mathbf{x}_N, \boldsymbol{\theta}) = \prod_{i=1}^{N} \mathbb{P}(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta})^{y_i}\ (1 - \mathbb{P}(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}))^{(1-y_i)} \tag{8}$$

5) To your opinion, what is the interest to use this modelling ?

To perform prediction, we first need to learn/estimate the set of parameters $\boldsymbol{\theta}$ *on the set of training data* $(\mathbf{x}_i, y_i)_{i=1}^N$.

6) In a straightforward manner, how could we estimate $\boldsymbol{\theta}$? Why is it not feasible ?

A common solution to determine $\boldsymbol{\theta}$ *is using a numerical approach and to find a sequence of values* $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^L$ *(L: number of iterations) converging to* $\boldsymbol{\theta}$.

$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \alpha \boldsymbol{\nabla} J(\boldsymbol{\theta}^{(l)}) \tag{9}$$

where $J(\boldsymbol{\theta}^{(l)}) = -\log \mathbb{P}(y_1, \ldots, y_N|\mathbf{x}_1, \ldots, \mathbf{x}_N, \boldsymbol{\theta}^{(l)})$ and $\alpha > 0$ the learning rate.

7) Compute the gradient of $J(\boldsymbol{\theta})$, by using (7) and demonstrate that:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta^p} = \sum_{i=1}^{N} (\mathbb{P}(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}) - y_i)\, x_i^p \quad \forall p \in \{1, \ldots, P\} \tag{10}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta^0} = \sum_{i=1}^{N} (\mathbb{P}(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}) - y_i) \tag{11}$$

Once the parameters have been estimated, class prediction can be performed. This naturally involves computing:

$$\mathbb{P}(y_i = 1|\mathbf{x}^\star, \widehat{\boldsymbol{\theta}}) \tag{12}$$

and affect $\mathbf{x}^\star$ *to the class* 1 *if* $\mathbb{P}(y_i = 1|\mathbf{x}^\star, \widehat{\boldsymbol{\theta}}) > 0.5$*.*

2.3 RL implementation

In this section, we propose to implement the method on our breast cancer dataset. As before, a RegressionLogistic class has been pre-created. The constructor of this class includes two attributes: the number of iterations and the learning rate (self.learning_rate), which will be used in the following questions.

8) Complete the method "update_weights". It should implement the gradient of $J(\boldsymbol{\theta})$ and perform one step of the recursion (8).

9) Integrate this function into the method "fit", which iterates the recursion for a fixed number of iterations.

10) Now, complete the method "predict" to allow for class prediction of test data as established in (12).

11) Now, test the logistic regression model. To begin, consider using a learning rate of 0.01 and a number of iterations equal to 1000.

    In the previous section, we observed that the most accurate algorithm was LDA. Therefore, we aim to compare the performance of logistic regression with that of LDA.

12) Test the LDA previously implemented on the first dataset with the breast cancer dataset and assess the prediction performance by varying the number of iterations (until 20000 for instance).