TP Naive Bayes

Thursday 13th April, 2023
deadline: Wednesday 26th April, 2023, 23:59

In this TP, you will implement the Naive Bayes classifier algorithm. The Naive Bayes classifier is a simple but effective algorithm for classification that is widely used in machine learning and data science. It is based on the Bayes' theorem and the assumption of feature independence, which makes it particularly suitable for high-dimensional and sparse datasets.

You are going to fill a few missing functions in the python scripts to implement the exercises that we ask. So, first read and understand the given python scripts. To run your code, you have to run the main_NB.ipynb notebook.

Theory

Naive Bayes is a classification algorithm that uses Bayes' theorem to predict the probability of a particular class label given some input features. Bayes' theorem states that:

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(y)P(x_1, x_2, \ldots, x_n|y)}{P(x_1, x_2, \ldots, x_n)} \tag{1}$$

where $P(y)$ is the prior probability of class y, $P(x_1, x_2, \ldots, x_n|y)$ is the likelihood of the input features given the class label y, and $P(x_1, x_2, \ldots, x_n)$ is the marginal likelihood of the input features.

The "naive" assumption in Naive Bayes is that the input features are conditionally independent given the class label. That is:

$$P(x_1, x_2, \ldots, x_n|y) = \prod_{i=1}^{n} P(x_i|y) \tag{2}$$

This assumption simplifies the calculation of the likelihood, since we can estimate the probability of each feature independently given the class label, rather than having to estimate the joint probability of all the features.
Since the denominator $P(x_1, x_2, \ldots, x_n)$ is constant across all possible classes, we can ignore it when comparing the relative probabilities of different classes. Therefore, we can calculate the posterior probability as:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3}$$

To calculate the likelihood $P(x_i|y)$ for each feature i we can use the appropriate probability distribution (e.g., a normal distribution for continuous features or the frequency of occurrence for categorical features) or to learn it non parametrically (following TP).

Likelihood
For continuous features, we can assume that the data follows a normal distribution, and estimate the likelihood $P(x_i|y)$ using the mean and variance of the feature values in class y:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_{i|y}^2}} \exp\left(-\frac{(x_i - \mu_{i|y})^2}{2\sigma_{i|y}^2}\right) \tag{4}$$

where $P(x_i \mid y)$ is the conditional probability of feature x_i given class y, $\mu_{y|i}$ is the mean of feature i for class y, $\sigma_{y|i}$ is the standard deviation of feature i for class y.

For categorical features, we can estimate the probability $P(x_i|y)$ as the frequency of each value v in the training data for a given feature i and class y as:

$$P(x_i = v|y = y_k) = \frac{number\,of\,training\,samples\,with\,feature\,i\,equal\,to\,v\,and\,label\,y_k}{number\,of\,training\,samples\,with\,label\,y_k} \tag{5}$$

Prior
For the prior probability $P(y)$ of class y_k, we simply calculate the proportion of the training samples that belong to class y_k:

$$P(y_k) = \frac{number\,of\,training\,samples\,with\,label\,y_k}{total\,number\,of\,training\,samples} \tag{6}$$

Prediction

To use Naive Bayes for classification, we compute the posterior probability $P(y|x_1, x_2, \ldots, x_n)$ for each possible class label, and choose the class label with the highest probability as the predicted label for the input instance.

Finally, we can predict the class with the highest posterior probability:

$$\hat{y} = \arg\max_y \log P(y|x_1, x_2, \ldots, x_n) \tag{7}$$

You are going to build the Naive Bayes classifier using both continious data (assuming that they folllow normal distribution) and categorical data.

You are going to fill a few missing functions in the python scripts to implement the exercises that we ask. So, first read and understand the given python scripts. To run your code, you have to run the main_NB.ipynb notebook.

Exercises

1. Fill the missing functions in NaiveBayes() class in naive_bayes.py to implement the NB algorithm for continuous data and discrete data. It should work not only for continuous or categorical datasets but also for datasets with both continuous and categorical features.

   Use main_NB.ipynb in order to run the functions that you implement in NaiveBayes() class.

2. Write a function named compute_accuracy(y_true, y_pred) in the utils.py script. The function takes as arguments the true and the predicted class labels and returns the accuracy. **Use only numpy.**

3. Once your implementation is ready, perform Naive Bayes classification and compute the classification accuracy for the following datasets:

   (a) Continuous dataset
   - Iris dataset
   - Breast cancer dataset
   - For the iris dataset, **draw the decision surfaces** (code is given). For the visualization purposes, we chose two attributes as predictive attributes and color the plane defined by these two attributes based on class labels that Naive Bayes predicts. Draw the decision surfaces for **3 different combinations of attributes** (1st-2nd), (1st-3rd) and (1st-3rd) Comment/ discuss the result.

   (b) Categorical dataset:
   - Handwritten digits dataset

   (c) Dataset with both continuous and categorical features:
   - Forest covertypes dataset

General instruction

The code should be well written, with detailed comments to explain what you do at each step. Avoid the for loops (they are not allowed), instead use the NumPy library. Your code should be generic and use the given functions. For example, the code for the *iris data* should be applicable, to the *Digits data* without any modification. Your final submission should include your code and a report (.pdf) which includes all the results along with the visualization on iris and your comments on the decision surfaces, what form do they have? linear? quadratic? other?