IIT-GOA CS-561
Probability & Statistics

November 2020

NAIVE BAYES CLASSIFIER

— Rohit S —
< rohit2013107@iitgoa.ac.in >

Abstract

The project report discusses the concept and implementation of *Naïve Bayes Classifier* — a fast and simple classifier that uses Bayes' theorem under the assumption that all features are mutually independent for a given class. Then, the results of testing one such (Guassian) Naïve Bayes Classifier against an EMNIST dataset of A–Z handwritten alphabets are presented.
Python codes for the above classifier are provided at `https://github.com/s-rohit/naive-bayes`

Keywords: machine learning, data mining, probabilistic classifier, Bayesian network models

Contents

1 Theory **1**
- 1.1 Backgroud 1
- 1.2 Bayes Classifier 1
  - 1.2.1 Naïve Bayes Classifier 2
  - 1.2.2 Gaussian Naïve Bayes 2

2 Implementation **2**

3 Experiment **4**
- 3.1 Mean and Variance 4
- 3.2 Confusion matrix and Accuracy 5

4 Conclusion **6**

References **6**

1 Theory

1.1 Backgroud

Our goal is to construct a **classifier** – a system that accepts an 'object' as input and based on certain 'features' of the object, attempts to assign a 'label' to it. To do so, the classifier must first have:

i) A *training dataset* to learn about the different objects, their labels and possible features.

ii) A *prediction algorithm* to predict the label of an unknown object given as input.

Depending on the prediction algorithm used, a wide variety of classifiers have been created.

Probabilistic Classifier a classifier that gives the probability distribution over all labels for a given input. The predicted label for that object then is the label with the highest probability.

Confusion Matrix a tabular representation of how often the system got 'confused' between any two labels (mislabelled one as another). Rows represent the actual label of the object tested and columns represent the predicted labels (or vice-versa). Observe that:

- Diagonal elements = number of successful predictions for a particular label
- Trace of the matrix = total number of successful predictions
- Row-wise (or column-wise) sum = total number of times an object of that label was tested
- Total sum = total number of tests performed

We use these to calulcating the overall and class-wise accuracies in section 3.

1.2 Bayes Classifier

Bayes Classifier is a probabilistic classifier that uses Bayes' theorem to find the probability of the given object belonging to a particular label. That is, given an un-labelled object $\mathbf{x} = (x_1, \ldots, x_n)$ with n features, we use Bayes' theorem to find the *posterior* probabaility $\Pr(C_k \mid \mathbf{x})$ of object $\mathbf{x}$ having a label C_k

$$\Pr(C_k \mid \mathbf{x}) = \frac{\Pr(C_k)\ \Pr(\mathbf{x} \mid C_k)}{\Pr(\mathbf{x})}$$

- Label likelihood, $\Pr(C_k) = \dfrac{\text{number of } C_k\text{-labelled objects}}{\text{total number of objects}}$ in training dataset (Discrete Uniform Law)
- Evidence (feature) probability, $\Pr(\mathbf{x}) = \sum_k \Pr(C_k)\ \Pr(\mathbf{x} \mid C_k)$ (Total Probability Theorem)
- Prior probability, $\Pr(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} \Pr(x_i \mid x_1, \ldots, x_{i-1}, C_k)$ (Chain Rule in Probability)

Using this, the predicted label $C_{\hat{k}}$ for $\mathbf{x}$ is found by calculating $\hat{k} = \underset{k \in \{1 \ldots K\}}{\operatorname{argmax}} \left\{ \Pr(C_k \mid \mathbf{x}) \right\}$

Note $\Pr(\mathbf{x})$ remains the same for a given $\mathbf{x}$. Hence we simpify as:

$$\hat{k} = \underset{k\in\{1...K\}}{\operatorname{argmax}} \left\{ \Pr(C_k)\ \Pr(\mathbf{x} \mid C_k) \right\}$$
$$= \underset{k\in\{1...K\}}{\operatorname{argmax}} \left\{ \Pr(C_k) \prod_{i=1}^{n} \Pr(x_i \mid x_1, \dots, x_{i-1}, C_k) \right\}$$

1.2.1 Naïve Bayes Classifier

Naïve Bayes classifier is a Bayes classifier that computes $\hat{k}$ with the assumption that all features of $\mathbf{x}$ are mutually independent, conditional on the class C_k.
Under this assumption: $\Pr(x_i \mid x_1, \dots, x_{i-1}, C_k) = \Pr(x_i \mid C_k)$
Hence, for input $\mathbf{x} = (x_1, \dots, x_n)$ the classifier predicts label $C_{\hat{k}}$ where $\boxed{\hat{k} = \underset{k\in\{1...K\}}{\operatorname{argmax}} \left\{ \Pr(C_k) \prod_{i=1}^{n} \Pr(x_i \mid C_k) \right\}}$

1.2.2 Gaussian Naïve Bayes

In case of continuous data, we assume that the values of features for the same class follow a Gaussian distribution. Let μ_{ki} and σ^2_{ki} be the mean and variance of values of i^{th} feature for class C_k.
Then, we get the probability density $\boxed{p(x_i = v_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma^2_{ki}}} \exp\left\{ -\frac{(v_i - \mu_{ki})^2}{2\sigma^2_{ki}} \right\}}$

For a small fixed δ, $\Pr(x_i = v_i \pm \delta \mid C_k) \ \propto \ p(x_i = v \mid C_k)$

2 Implementation

The two boxed equations need to be modified before implementing on them on a computer, because:

1. The product of the probabilites would be a very small number (especially for large n); and such numbers cannot be accurately stored on a computer as floating points. If the calculated probability is very small, it may even get approximated to zero!
2. A smoothing factor will have to be introduced to improve the classifier's tolerance and also support cases where the observed variance of feature in a class is close to 0.
3. Operations like `exp`, `pow`, `sqrt`, etc are usually very resource-intensive. If we can minimize the number of operations by simplifying those equations, removing unecessary constant terms, or calculating some values beforehand, then we can speed up the classifier.

Hence, in this section we will modify the boxed equations to get a working equation as follows:

$$
\begin{aligned}
\hat{k} &= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \Pr(C_k) \prod_{i=1}^{n} \Pr(x_i \mid C_k) \right\} && \text{Boxed-eqn-1} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left[\Pr(C_k) \prod_{i=1}^{n} \Pr(x_i \mid C_k) \right] \right\} && \log(\cdot) \text{ is a monotonic function} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left[\Pr(C_k) \right] + \sum_{i=1}^{n} \log \left[\Pr(x_i \mid C_k) \right] \right\} && \text{Simplifying} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(\frac{N_k}{|D|} \right) + \sum_{i=1}^{n} \log \left[\Pr(x_i \mid C_k) \right] \right\} && \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(N_k \right) - \log \left(|D| \right) + \sum_{i=1}^{n} \log \left[\Pr(x_i \mid C_k) \right] \right\} && \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(N_k \right) + \sum_{i=1}^{n} \log \left[\Pr(x_i \mid C_k) \right] \right\} && \text{removing } |D| \text{ constant} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(N_k \right) + \sum_{i=1}^{n} \log \left[\frac{1}{\sqrt{2\pi\sigma_{ki}^2}} \exp \left\{ -\frac{(v_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right\} \right] \right\} && \text{Using boxed-eqn-2} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(N_k \right) - \sum_{i=1}^{n} \left[\frac{1}{2} \log \left(2\sigma_{ki}^2 \right) + \frac{\log(\pi)}{2} + \frac{(v_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right] \right\} && \text{Simplifying} \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmax}} \left\{ \log \left(N_k \right) - \sum_{i=1}^{n} \left[\frac{1}{2} \log \left(z_{ki} \right) + \frac{(v_i - \mu_{ki})^2}{z_{ki}} \right] \right\} && \text{Simplify \& take } z_{ki} = 2\sigma_{ki}^2 \\
&= \underset{k\in\{1\ldots K\}}{\operatorname{argmin}} \left\{ \underbrace{-\log \left(N_k \right) + \sum_{i=1}^{n} \left[\frac{1}{2} \log \left(z_{ki} \right) \right]}_{= m_k,\ \text{constant for class } C_k} + \sum_{i=1}^{n} \left[\frac{(v_i - \mu_{ki})^2}{z_{ki}} \right] \right\} && \text{Reorganizing}
\end{aligned}
$$

Hence, we get the working formula $\boxed{\hat{k} = \underset{k\in\{1\ldots K\}}{\operatorname{argmin}} \left\{ m_k + \sum_{i=1}^{n} \frac{(v_i - \mu_{ki})^2}{z_{ki}} \right\}}$

Redefine $z_k = 2\sigma_{ki}^2 + \xi$ to include smoothing factor ξ

3 Experiment

Based on the working formula for $\hat{k}$, a Gaussian Naïve Bayes classifier in Python was created and tested against an EMNIST dataset of over 370,000 grayscale 28x28 images of A–Z handwritten alphabets.

- Hence, the dataset has 26 classes (A–Z) with 784 features (pixels) each taking one of 256 values.
- The entire dataset was randomly divided into testing sets and training sets in 9:1 ratio.
- The classifier uses Gaussian distribution model despite the pixel taking discrete values – because the training set may not have enough samples to plot the discrete distribution of each pixel accurately.

The results of one such experiment are presented in this section:

3.1 Mean and Variance

The mean and variance of the pixels each alphabet are presented as 28x28 grayscale images.
As expected, the mean represents "what that alphabet on average looks like", and variance increases towards edges and the 'protruding' parts of each alphabet.

Figure 1: Mean of pixel values of each alphabet

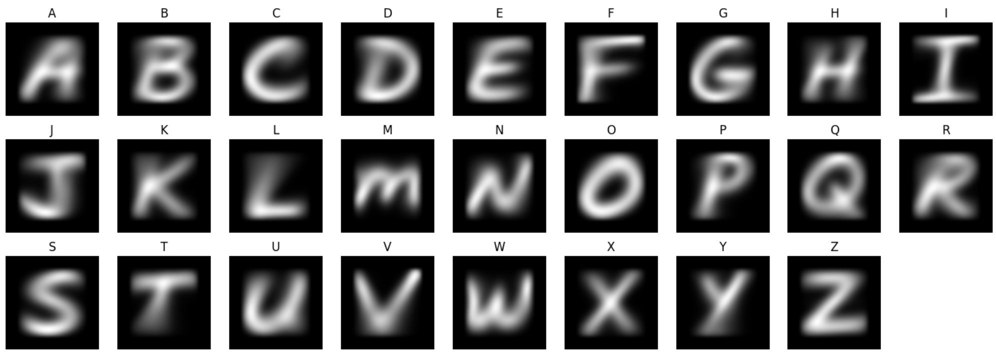
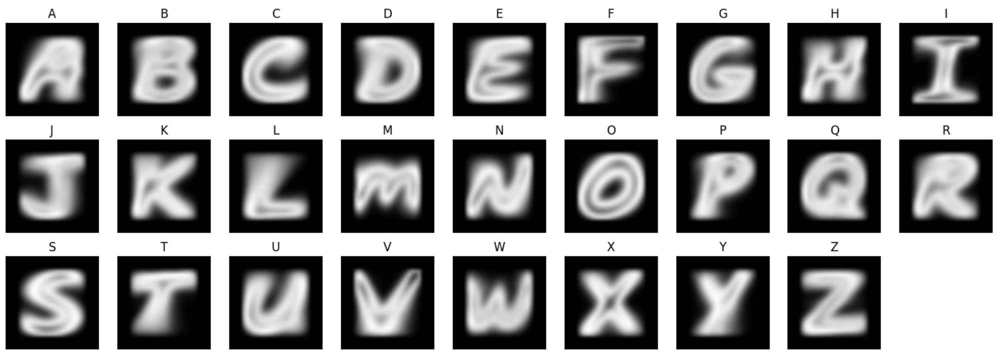


Figure 2: Variance of pixel values of each alphabet



3.2 Confusion matrix and Accuracy

As described in section 1, the overall and class-wise accuracy are found using the confusion matrix.

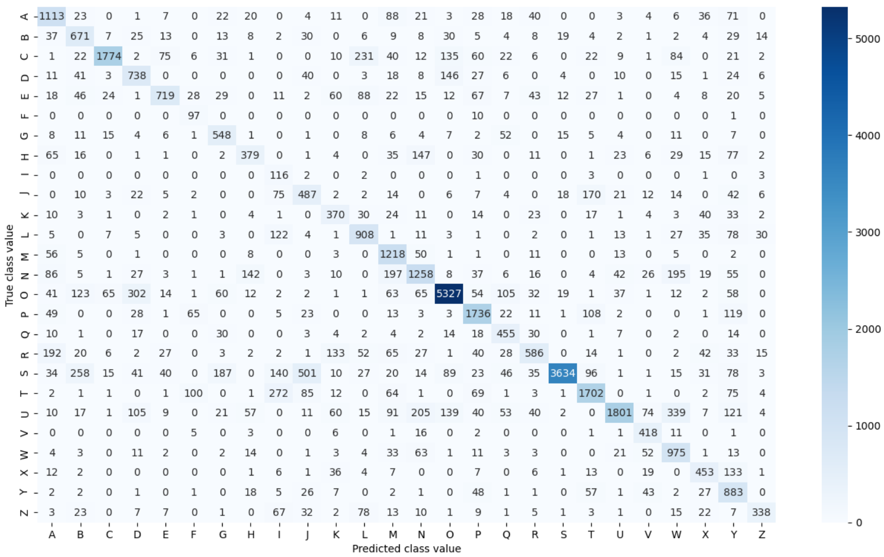
- Overall accuracy = trace of matrix ÷ total sum
- Class-wise accuracy = value$_{ii}$ ÷ row-wise sum

Figure 3: Accuracy of prediction

Overall accuracy ≈ 69.82%

A	B	C	D	E	F	G	H	I
73.27%	70.56%	69.11%	67.03%	56.66%	89.81%	76.54%	44.85%	90.62%
J	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**
52.82%	62.29%	72.18%	88.65%	58.73%	83.23%	79.27%	74.10%	45.29%
S	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	
68.07%	70.98%	55.90%	89.89%	79.92%	64.53%	78.28%	52.32%	

Figure 4: Confusion matrix



Observations

- Letter ‘H’ has the least accuracy, and is confused for ‘N’ about a third of the time.
- Letter ‘R’ has a low accuracy, and is often confused for ‘A’ and ‘K’.
- Letters ‘F’ and ‘I’ seem to have accuracy, but were not tested as much as others.
- Letters ‘O’ and ‘S’ were tested the most; letters ‘I’ and ‘V’ have the highest accuracy.

4 Conclusion

The project provides a basic introduction into the field of machine learning and shows a small example of how probability theorems are used in practical applications.

The Naïve Bayes Classifier is fast and easy but the biggest disadvantage is that features are assumed to be independent, which is not true for most real life cases. An improved classifier can be built by removing these (naïve) independence assumptions and using multivariate Gaussian probability distribution.

References

[1] Arnabp. (May 27, 2020). Naive bayes tutorial using mnist dataset, [Online]. Available: `https://medium.com/data-sensitive/na%C3%AFve-bayes-tutorial-using-mnist-dataset-2b4a82b124d2` (visited on 11/14/2020).

[2] R. Nagaraj. (May 17, 2020). Mnist handwritten image classification with naive bayes and logistic regression., [Online]. Available: `https://medium.com/@rnagara1/mnist-handwritten-image-classification-with-naive-bayes-and-logistic-regression-9d0dd2b6edc0` (visited on 11/14/2020).

[3] (Mar. 19, 2015). Bayes classifier and naive bayes tutorial (using the mnist dataset), [Online]. Available: `https://lazyprogrammer.me/bayes-classifier-and-naive-bayes-tutorial-using/` (visited on 11/14/2020).

[4] R. Dwivedi. (Apr. 29, 2020). What is naive bayes algorithm in machine learning? [Online]. Available: `https://www.analyticssteps.com/blogs/what-naive-bayes-algorithm-machine-learning` (visited on 11/14/2020).

[5] J. Brownlee. (Aug. 15, 2020). Naive bayes for machine learning, [Online]. Available: `https://machinelearningmastery.com/naive-bayes-for-machine-learning/` (visited on 11/14/2020).