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 the *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 & C++ codes for the above classifier are provided at Github.

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

Contents

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.

Objects having the same label are said to belong to the same “class” of objects.
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 possible labels for a given input. The predicted label for that object is then the label with the highest probability.

Confusion Matrix a tabular representation of how often the system got ‘confused’ between objects of any two classes (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 class
- Trace of the matrix = total number of successful predictions
- Row-wise (or column-wise) sum = number of times an object of that class was tested
- Total sum = total number of tests performed

1.2 Bayes Classifier

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

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

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

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

Note Pr(**x**) remains the same for a given **x**. Hence we simpify as:

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

1.2.1 Naïve Bayes Classifier

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

1.2.2 Gaussian Naïve Bayes

In case of continuous data, we assume that the values of features for a class follow Gaussian distribution.
Let μ_{ki} and σ^2_{ki} be the mean and variance of the values of the i^{th} feature of 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\} && \text{Frequency } N_k \text{ in training set } D \\
&= \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 augmented variance $z_{ki} = 2\sigma_{ki}^2 + \xi$ to include smoothing factor ξ

3 Experiment

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

- Hence, our dataset had 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 1:9 ratio.
- The classifier uses Gaussian distribution model despite the pixels taking discrete values – because the training set may not have enough samples to plot the distribution of each pixel accurately.
- Smoothing factor $\xi \approx 2048$ was found to give the best results. This corresponds to the assumption of a minimum variance of 32^2 in the values of the features of every class.

The results of one such experiment are presented in this section:
Python & C++ codes for the classifier are available at Github.

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 the variance increases towards the edges and the protruding parts of each alphabet.

Figure 1: Mean of pixel values of each alphabet (scaled).

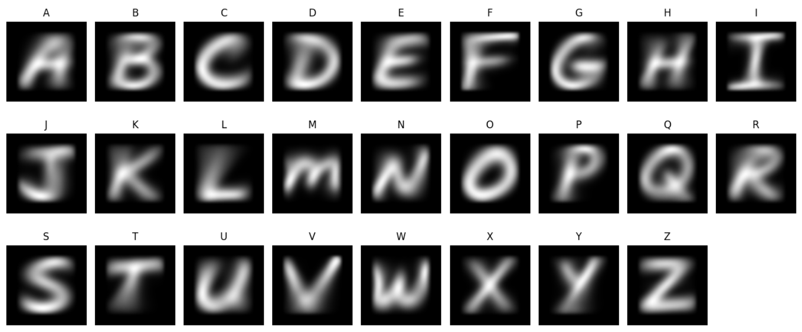
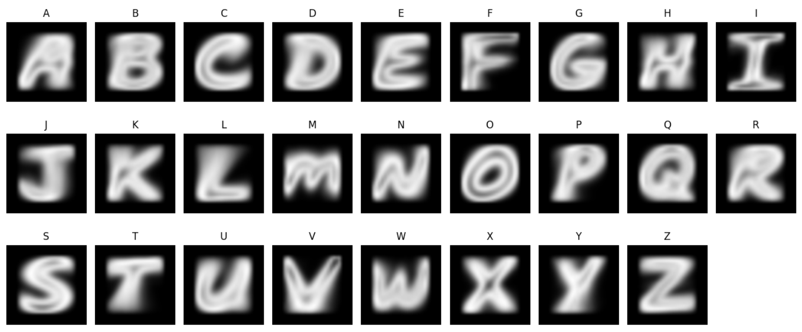


Figure 2: Variance of pixel values of each alphabet (scaled).



3.2 Confusion matrix and Accuracy

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

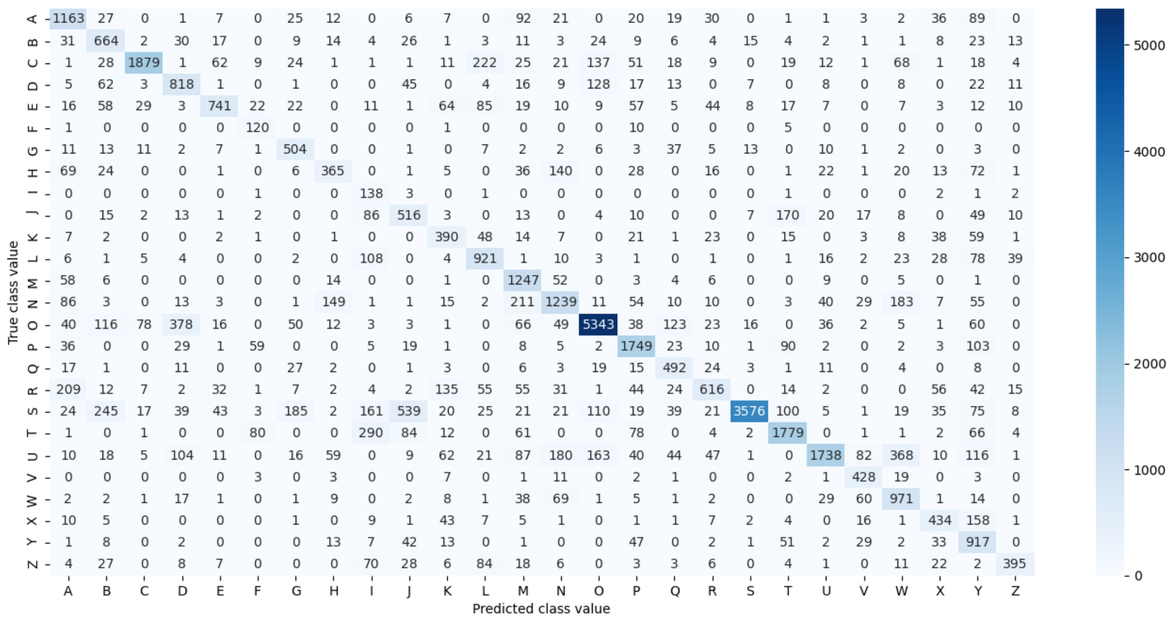
- Overall accuracy = trace of matrix ÷ total sum
- Class-wise accuracy = value_{ii} ÷ row-wise sum

Figure 3: Accuracy of prediction

Overall accuracy ≈ 70.05%

A	B	C	D	E	F	G	H	I
74.46%	71.78%	71.61%	69.44%	58.81%	87.59%	78.63%	44.46%	92.62%
J	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**
54.55%	60.84%	73.44%	88.69%	58.28%	82.72%	81.42%	75.93%	45.03%
S	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	
66.80%	72.14%	54.45%	88.98%	78.62%	61.39%	78.31%	56.03%	

Figure 4: Confusion matrix


True class value
Predicted class value
5000
4000
3000
2000
1000
0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	1163	27	0	1	7	0	25	12	0	6	7	0	92	21	0	20	19	30	0	1	1	3	2	36	89	0
B	31	664	2	30	17	0	9	14	4	26	1	3	11	3	24	9	6	4	15	4	2	1	1	8	23	13
C	1	28	1879	1	62	9	24	1	1	1	11	222	25	21	137	51	18	9	0	19	12	1	68	1	18	4
D	5	62	3	818	1	0	1	0	0	45	0	4	16	9	128	17	13	0	7	0	8	0	8	0	22	11
E	16	58	29	3	741	22	22	0	11	1	64	85	19	10	9	57	5	44	8	17	7	0	7	3	12	10
F	1	0	0	0	0	120	0	0	0	0	1	0	0	0	0	10	0	0	0	5	0	0	0	0	0	0
G	11	13	11	2	7	1	504	0	0	1	0	7	2	2	6	3	37	5	13	0	10	1	2	0	3	0
H	69	24	0	0	1	0	6	365	0	1	5	0	36	140	0	28	0	16	0	1	22	1	20	13	72	1
I	0	0	0	0	0	1	0	0	138	3	0	1	0	0	0	0	0	0	0	1	0	0	0	2	1	2
J	0	15	2	13	1	2	0	0	86	516	3	0	13	0	4	10	0	0	7	170	20	17	8	0	49	10
K	7	2	0	0	2	1	0	1	0	0	390	48	14	7	0	21	1	23	0	15	0	3	8	38	59	1
L	6	1	5	4	0	0	2	0	108	0	4	921	1	10	3	1	0	1	0	1	16	2	23	28	78	39
M	58	6	0	0	0	0	0	14	0	0	1	0	1247	52	0	3	4	6	0	0	9	0	5	0	1	0
N	86	3	0	13	3	0	1	149	1	1	15	2	211	1239	11	54	10	10	0	3	40	29	183	7	55	0
O	40	116	78	378	16	0	50	12	3	3	1	0	66	49	5343	38	123	23	16	0	36	2	5	1	60	0
P	36	0	0	29	1	59	0	0	5	19	1	0	8	5	2	1749	23	10	1	90	2	0	2	3	103	0
Q	17	1	0	11	0	0	27	2	0	1	3	0	6	3	19	15	492	24	3	1	11	0	4	0	8	0
R	209	12	7	2	32	1	7	2	4	2	135	55	55	31	1	44	24	616	0	14	2	0	0	56	42	15
S	24	245	17	39	43	3	185	2	161	539	20	25	21	21	110	19	39	21	3576	100	5	1	19	35	75	8
T	1	0	1	0	0	80	0	0	290	84	12	0	61	0	0	78	0	4	2	1779	0	1	1	2	66	4
U	10	18	5	104	11	0	16	59	0	9	62	21	87	180	163	40	44	47	1	0	1738	82	368	10	116	1
V	0	0	0	0	0	3	0	3	0	0	7	0	1	11	0	2	1	0	0	2	1	428	19	0	3	0
W	2	2	1	17	1	0	1	9	0	2	8	1	38	69	1	5	1	2	0	0	29	60	971	1	14	0
X	10	5	0	0	0	0	1	0	9	1	43	7	5	1	0	1	1	7	2	4	0	16	1	434	158	1
Y	1	8	0	2	0	0	0	13	7	42	13	0	1	0	0	47	0	2	1	51	2	29	2	33	917	0
Z	4	27	0	8	7	0	0	0	70	28	6	84	18	6	0	3	3	6	0	4	1	0	11	22	2	395

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 high accuracy, but were not tested as much as others.
- Letters 'O' and 'S' were tested the most; letter 'I' has 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).