Assignment Report

# Assignment 1: Naive Bayes

Artem Bisliouk (abisliou), Elizaveta Nosova (enosova)

October 13, 2024

## 1. Training - implemented in the Jupiter notebook

## 2. Prediction - implemented in the Jupiter notebook

## 3. Experiments on MNIST digits data

### a) Accuracy of the Naive Bayes model

A Naive Bayes classifier for categorical data was trained on the 60 000 images of the MNIST digit training datasets and used to make label prediction on the 10 000 MNIST digit test images.

The first performance metric reported is accuracy. It represents the ratio of correctly predicted labels to all predictions. The accuracy achieved by the presented model is 0.8363, i.e. correct label predictions were produced for 83,63% of the test data.

To draw conclusions about the prediction quality of the model, we need to compare the obtained accuracy with a baseline. One simple baseline is a classifier that always predicts the most frequent class (observed during training), also known as a zero-rule classifier (1, p.180). Our training set is not perfectly balanced, there is a slight skew in the class distribution, with class 1 making up 11,23% of the training set and thus being the most frequent class.

Predicting the labels of the test set with such a zero-rule classifier gives us an accuracy of 00,1135, which corresponds to the proportion of class 1 in the test set. If we now compare the accuracy of the Naive Bayes classifier with that of the baseline classifier, we can see that the performance is significantly improved, making correct predictions about 7 times more often. Nevertheless, it is worth mentioning the baseline is quite weak.

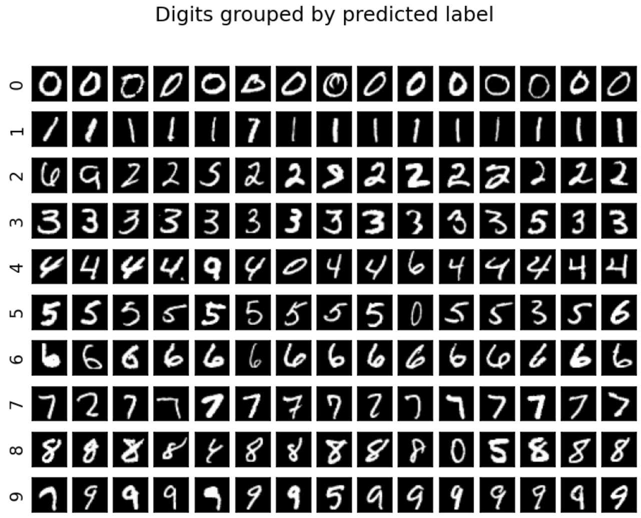

Figure 1: A sample of predictions per class

## b) Error analysis

Randomly sampled 15 images of each class, according to the model's predictions, are presented in Figure 1. About 12% of given samples are misclassified, with most errors easily detectable by human observation.

A set of misclassified examples is shown in Figure 2. Some narrow-shaped instances of various digits get misclassified as 1. Label 3 is frequently assigned to images of 5 and 8, likely due to similar patterns in the lower half of these digits and a high presence of light pixels in the upper half. The model seems to capture this concentration but not its exact distribution, i.e. shape, leading to this type of error.

To better understand the prediction quality of each class and how it differs across classes, we will analyze metrics such as precision, recall and f1-score (2, p.174) as presented in the classification report in Table 1 in appendix A.

The precision measures what fraction of all predictions of certain label are correct. For classes, it varies between 0,75 and 0,91. For instance, the model correctly predicts classes 0 and 7 in 91% of the cases, indicating better identification for these classes compared to others, such as classes 8 and 9, which have lower precision.

Recall shows what fraction of all instances of a certain class the model detected correctly. It varies between 0,78 and 0.97, with class 1 being the most accurately identified. This may be attributed to class 1 prevalence in the training set. Conversely, class 5 has the lowest recall, likely due to being the most underrepresented class in the training set, accounting for 9,04%.

F1-score is computed as the harmonic mean of precision and recall, thus a high value of F1-score implies that both precision and recall are high. This metric ranges between

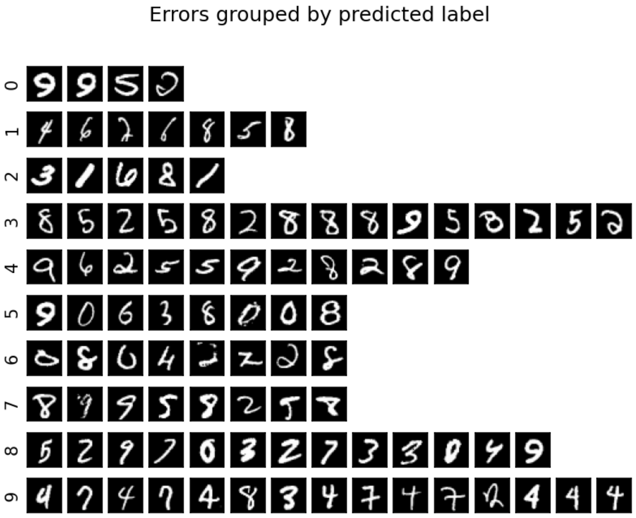

Figure 2: Incorrect predictions per class

0,72 and 0,91, again demonstrating higher scores for more frequent classes.

The confusion matrix in Figure 3 provides detailed information on misclassification frequencies between classes. The largest misclassification occurs when images of 5 are labeled as 3: approximately 14% of the images labeled 5 in the test dataset were misclassified as 3. This misclassification was already illustrated in Figure 1. At the same time, less than 3% of images labeled as 3 were assigned label 5, indicating that misclassification is not necessarily symmetrical, as can be seen in other label-prediction pairs.

Another example of mutually mistaken classes is 4 and 9. In this case, misclassification symmetry is observed; they are the most commonly mistaken labels for each other. However, a 4 is twice as likely to be misclassified as a 9 than vice versa (119 instances versus 64).

There are only a few combinations of true labels and predicted labels that the classifier never confuses in given test set. For example, 0 is never labeled as 1 and vice versa. Also, 1 is never labeled as 4, 7 or 9. This may not be solely because class 1 is the most frequent in the training data – the class imbalance is far from extreme – but also because the relatively simple handwriting pattern of 1, resembling a straight vertical or slightly inclined line, is distinct from more complex digit patterns, making it easier for the model to learn the difference from the pixel probability distribution.

## 4. Model selection (optional)

See appendix B

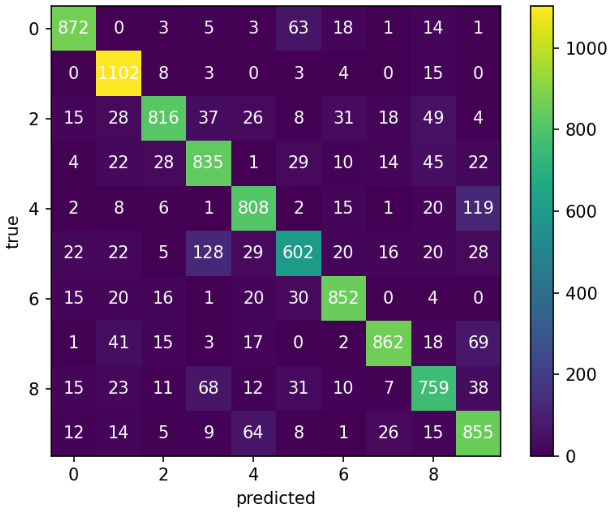

Figure 3: Confusion matrix

## 5. Generating data

The hyperparameter $\alpha$ in Naive Bayes acts as a smoothing parameter for the class-conditional probabilities, especially in situations where certain feature values might not be observed frequently. By varying $\alpha$, we can observe how this parameter influences the generation of digits and the model's behavior.

### Generating data

As we vary $\alpha$ and generate digits for each class as shown in Figure 4, we observe the following:

- Small $\alpha$ values (e.g., $\alpha = 1$): With smaller values of $\alpha$, the generated digits tend to resemble more specific patterns from the training data. This is because the model is not overly smoothed, so it heavily relies on the empirical frequencies of feature values in the dataset. As a result, the generated digits have sharper and more distinct features, closely mimicking the digits seen during training. However, this also means that the model might overfit to rare occurrences or zero-count issues in the dataset.

- Moderate $\alpha$ values (e.g., $\alpha = 2$): With slightly larger values of $\alpha$, the generated digits remain fairly representative of the data while reducing the risk of overfitting. The smoothing introduced by $\alpha$ helps mitigate the effects of zero-counts, making the model more general and slightly more robust to noise in the training data. The generated digits are still recognizable, but the model avoids some of the sharper details, resulting in more general representations.

- Large $\alpha$ values (e.g., $\alpha = 10$ and $\alpha = 50$): As $\alpha$ increases, the smoothing effect

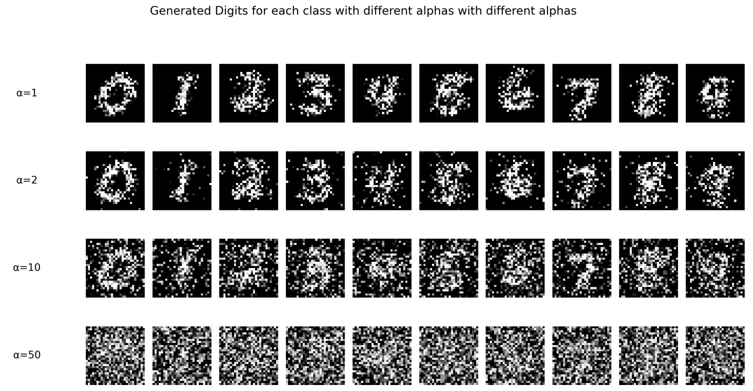
Generated Digits for each class with different alphas with different alphas
α=1
α=2
α=10
α=50

Figure 4: Generated Digits for each class with different alphas

becomes more pronounced. The generated digits begin to lose sharpness and distinct features, appearing more ”blurred” or ”average” across the class. This happens because the model assigns more uniform probabilities across feature values, reducing the influence of rare patterns in the training data. Consequently, the digits generated with large $\alpha$ values may not closely resemble any particular example from the training set, as the model treats all feature values more equally. This indicates underfitting, where the model fails to capture the detailed structure of the data.

## Interpretation of Results

The hyperparameter $\alpha$ controls the balance between reliance on the observed data and smoothing towards a uniform distribution.

- Small $\alpha$ values lead to overfitting, where the model might produce highly detailed digits, closely following specific examples in the training data. - Large $\alpha$ values induce underfitting, where the model generates more generic or average-looking digits that may lose critical distinguishing features.

For practical applications, selecting a moderate $\alpha$ provides a balance between these two extremes. The generated digits retain recognizable features while avoiding overfitting to rare occurrences in the data.

## Most-Likely and Expected Values

In addition to generating digits, we visualize the most likely value of each feature for each class as well as the expected value of each feature in Figures 5 and 6 respectively, which remain the same across different values of $\alpha$:

- Most Likely Values: These reflect the feature values that maximize the class-conditional probability $p(x_j|y)$. Since $\alpha$ mainly smooths the distribution, the most likely feature value remains consistent, as smoothing does not significantly alter the location of the probability mass (the maximum).

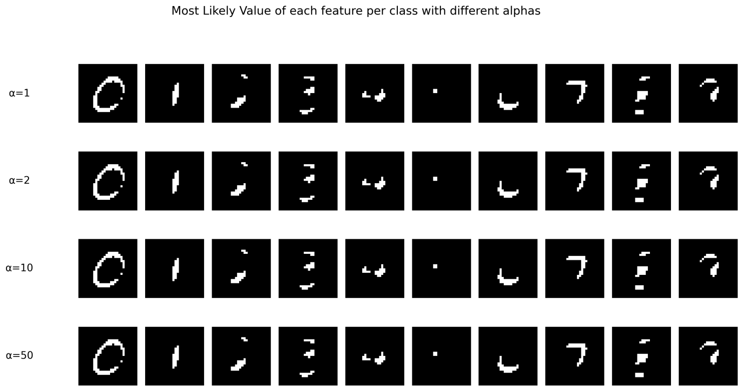

Figure 5: Most Likely Value of each feature per class with different alphas

- Expected Values: These values are calculated as the weighted average of all possible feature values for each class. The averaging effect of the expected value smooths out variations caused by $\alpha$, and hence, the expected values appear uniform across different values of $\alpha$.

## 6. Missing Data

### (a) $p(y|x_{1:D})$

We can compute the posterior distribution using Bayes' theorem:

$$p(y|x_{1:D}) = \frac{p(y)p(x_{1:D}|y)}{p(x_{1:D})}$$

where, under the Naive Bayes assumption, the likelihood factorizes as:

$$p(x_{1:D}|y) = \prod_{j=1}^{D} p(x_j|y)$$

The denominator $p(x_{1:D})$ is the marginal likelihood, which sums over all possible class labels $y'$ as follows:

$$p(x_{1:D}) = \sum_{y'} p(y') \prod_{j=1}^{D} p(x_j|y')$$

Thus, the posterior becomes:

$$p(y|x_{1:D}) = \frac{p(y) \prod_{j=1}^{D} p(x_j|y)}{\sum_{y'} p(y') \prod_{j=1}^{D} p(x_j|y')}$$

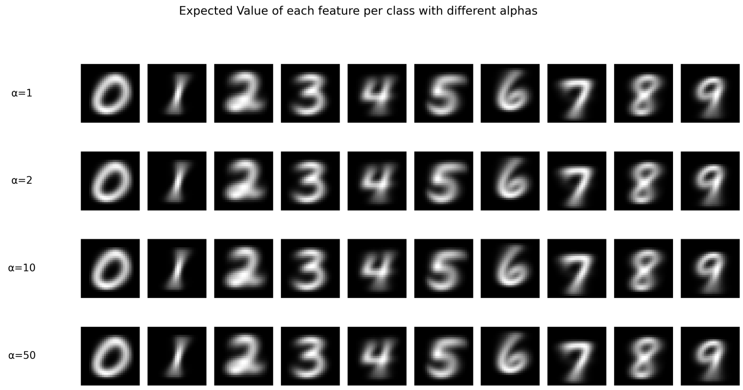

Figure 6: Expected Value of each feature per class with different alphas

This formula is central to the Naive Bayes classifier. The term $p(y)$ represents the prior probability of the class $y$, which is essential when we don't have any feature data yet—it reflects the overall frequency of the class in the dataset. The product $\prod_{j=1}^{D} p(x_j|y)$ is the likelihood of observing the specific features $x_1, \ldots, x_D$ given the class $y$. This assumes independence between the features, allowing us to factor the joint likelihood into a product of individual probabilities.

The denominator $p(x_{1:D})$ is the marginal likelihood, which ensures that the posterior is properly normalized over all possible classes. This marginal probability sums over all possible class labels and their corresponding likelihoods, thus accounting for all ways the feature vector could be generated.

This formula is useful in fully observed datasets where all features $x_1, \ldots, x_D$ are available. The posterior $p(y|x_{1:D})$ gives us the probability of a class given a complete feature vector, which directly informs our classification decision. The independence assumption enables computational efficiency, even with high-dimensional data like images (e.g., MNIST with 784 pixels as features).

### (b) $p(y|x_{1:D'})$ for $1 \leq D' < D$

When only a subset of features $x_{1:D'}$ is available, the posterior is computed as:

$$p(y|x_{1:D'}) = \frac{p(y) \prod_{j=1}^{D'} p(x_j|y)}{p(x_{1:D'})}$$

with the marginal likelihood given by:

$$p(x_{1:D'}) = \sum_{y'} p(y') \prod_{j=1}^{D'} p(x_j|y')$$

In this scenario, we are dealing with missing data or a situation where only a subset of features is available. The numerator $p(y) \prod_{j=1}^{D'} p(x_j|y)$ includes the prior $p(y)$ and the likelihood based on the available features $x_1, \ldots, x_{D'}$. By using only a subset of the features, we implicitly assume that the omitted features $x_{D'+1}, \ldots, x_D$ do not strongly influence the prediction, or that they are not highly correlated with the outcome. This assumption simplifies the problem and allows us to make predictions even with incomplete data.

In some cases, this approach can also be viewed as a form of dimensionality reduction, where we rely on the most informative features while ignoring less important ones. This is useful in practical applications where high-dimensional data may be noisy or redundant, and where focusing on a smaller set of features leads to more efficient computation without significantly sacrificing accuracy.

This formula is particularly valuable in real-world scenarios where some features may be missing or unavailable (e.g., incomplete surveys or sensor failures). By making predictions based on the available features, the model remains robust to incomplete data and can still provide meaningful outputs.

### (c) $p(x_{D'+1:D}|x_{1:D'})$ for $1 \leq D' < D$

To compute the distribution of the missing features $x_{D'+1:D}$ given the observed features $x_{1:D'}$, we use the law of total probability:

$$p(x_{D'+1:D}|x_{1:D'}) = \sum_y p(x_{D'+1:D}|y)p(y|x_{1:D'})$$

where $p(x_{D'+1:D}|y) = \prod_{j=D'+1}^{D} p(x_j|y)$.

Here, the posterior distribution $p(x_{D'+1:D}|x_{1:D'})$ describes the likelihood of the missing features $x_{D'+1}, \ldots, x_D$ conditioned on the observed features $x_1, \ldots, x_{D'}$. The term $p(y|x_{1:D'})$ serves as a weighting factor that determines how strongly each class label $y$ influences the prediction of the missing features.

The term $p(x_{D'+1:D}|y) = \prod_{j=D'+1}^{D} p(x_j|y)$ reflects how the missing features are distributed within a given class $y$. By summing over all possible class labels, we account for the uncertainty about which class label is most likely, based on the available features.

This formula is particularly useful for imputing missing data. In scenarios where some features are unobserved, we can predict their likely values based on the observed data. For example, in image reconstruction tasks, we might predict missing pixels based on the observed ones, leveraging the class-specific patterns learned by the model.

## References

[1] C. Huyen, *Designing Machine Learning Systems*, 2022.

[2] K. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

## A. Classification Report

Classification report is presented in Table 1

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0.91 | 0.89 | 0.90 | 980 |
| 1 | 0.86 | 0.97 | 0.91 | 1135 |
| 2 | 0.89 | 0.79 | 0.84 | 1032 |
| 3 | 0.77 | 0.83 | 0.80 | 1010 |
| 4 | 0.82 | 0.82 | 0.82 | 982 |
| 5 | 0.78 | 0.67 | 0.72 | 892 |
| 6 | 0.88 | 0.89 | 0.89 | 958 |
| 7 | 0.91 | 0.84 | 0.87 | 1028 |
| 8 | 0.79 | 0.78 | 0.79 | 974 |
| 9 | 0.75 | 0.85 | 0.80 | 1009 |

Table 1: Classification report of the Naive Bayes model on the MNIST test set

## B. Model Selection

To define a suitable value for the hyperpatameter $\alpha$, 5-fold cross-validation was implemented on a range of 17 values from 0,5 to 50. Cross-validation proved to be relatively time-consuming, with each iteration (5 sets of folds for each value of $\alpha$) taking approximately 7 minutes. As shown seen in Figure 7, after reaching an optimal accuracy above 0,8 at $\alpha$=2, accuracy decreases with decreasing values of $\alpha$. This may indicate underfitting caused by an increasing influence of the prior over the likelihood of the observed data.

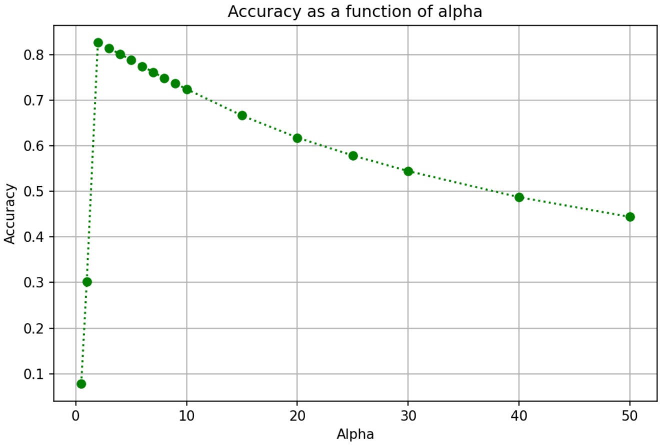

Figure 7: Accuracy at hyperparameter tuning

## Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

**Declaration of Used AI Tools**

| Tool | Purpose | Where? | Useful? |
|---|---|---|---|
| ChatGPT | Rephrasing | Throughout | + |
| DeepL | Style Edits | Throughout | ++ |
| GPT-4 | Code debugging | Throughout | +- |

Unterschrift
Mannheim, den 13. Oktober 2024