Data2060 Final Project

Gaussian Naive Bayes for classification

Never Converge

Zi Tao
Jiaqing Dai
Yingzhi Ma
2025.12.10

Introduction

- classification tasks involving continuous-valued features.
- assume conditional independence between features.
- Unlike the Bernoulli Naive Bayes models that assume discrete features, Gaussian Naive Bayes models each feature using a Gaussian (normal) distribution for every class.

- mathematical simplicity, fast training time, and competitive performance

Representation - Math

Given a feature vector

$\mathbf{x} = (x_1, x_2, \ldots, x_d)$,

GNB models the conditional likelihood for each class $y \in \{1, \ldots, K\}$ as:

$$P(x_i \mid y) = \mathcal{N}(x_i; \mu_{y,i}, \sigma^2_{y,i}) = \frac{1}{\sqrt{2\pi\sigma^2_{y,i}}} \exp\left(-\frac{(x_i - \mu_{y,i})^2}{2\sigma^2_{y,i}}\right).$$

Under the independence assumption, the joint likelihood becomes:

$$P(\mathbf{x} \mid y) = \prod_{i=1}^{d} P(x_i \mid y).$$

Using Bayes' rule, the posterior is:

$$P(y \mid \mathbf{x}) \propto P(y) \prod_{i=1}^{d} P(x_i \mid y).$$

To avoid floating-point underflow, we compute the \textbf{log-posterior}:

$$\log P(y \mid \mathbf{x}) = \log P(y) + \sum_{i=1}^{d} \log \mathcal{N}(x_i; \mu_{y,i}, \sigma^2_{y,i}).$$

The predicted class is:

$$\hat{y} = \arg\max_y \log P(y \mid \mathbf{x}).$$

Thus, the feature representation converts continuous inputs into probabilities governed by class-specific Gaussian distributions.

Bayes Rule:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

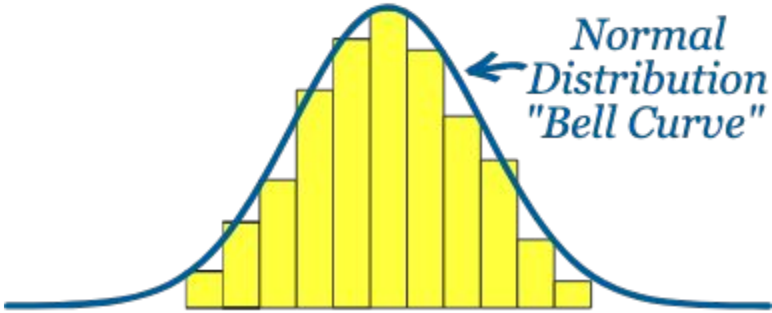


Image source: Pierce, Rod, 2025, 'Normal Distribution, Math Is Fun, Available at: <https://www.mathsisfun.com/data/standard-normal-distribution.html>.

Loss - Maximum Likelihood Estimates

Class priors:

$$P(y) = \frac{N_y}{N}$$

Feature mean:

$$\mu_{y,i} = \frac{1}{N_y} \sum_{x \in y} x_i$$

Feature variance:

$$\sigma^2_{y,i} = \frac{1}{N_y} \sum_{x \in y} (x_i - \mu_{y,i})^2$$

At evaluation time, models are typically assessed using classification accuracy or cross-entropy loss:

$$\text{Loss} = -\sum_{n=1}^{N} \log P(y_n \mid x_n)$$

Optimizer

GNB does not require iterative optimization. All parameters have closed-form MLE solutions, making the algorithm extremely efficient.

Parameters learned:

Class prior probabilities $P(y)$ Feature means $\mu_{y,i}$ Feature variances $\sigma_{y,i}^2$

Numerical Techniques

1. Using log-space to prevent numerical underflow

$$P(x \mid y) = \prod_{i=1}^{d} \mathcal{N}(x_i \mid \mu_{y,i}, \sigma_{y,i}^2) \quad \longrightarrow \quad \log P(x \mid y) = \sum_{i=1}^{d} \log \mathcal{N}(x_i \mid \mu_{y,i}, \sigma_{y,i}^2)$$

2. Variance smoothing: adding ε to the variance：

$\sigma_{y,i}^2 = 0$

$$\boldsymbol{\sigma_{y,i}^2 = 0} \quad \longrightarrow \quad \sigma_{y,i}^2 \leftarrow \sigma_{y,i}^2 + \epsilon$$

3. Closed-form MLE for comp

$$\mu_{y,i} = \frac{1}{N_y} \sum_{x \in y} x_i$$

$$\sigma_{y,i}^2 = \frac{1}{N_y} \sum_{x \in y} (x_i - \mu_{y,i})^2$$

Numerical Techniques

Train:

initialize priors $P(y)$, means $\mu[y][i]$, variances $\sigma^2[y][i]$

for each class c :

$X_c \leftarrow$ all rows of X where label = c

$N_c \leftarrow$ number of samples in class c

$P(c) \leftarrow N_c/N$

for each feature $i \in \{1, \dots, d\}$:

$\mu[c][i] \leftarrow$ mean of $X_c[:, i]$

$\sigma^2[c][i] \leftarrow$ variance of $X_c[:, i] + \epsilon$ (smoothing)

return priors, means, variances

Predict:

To predict a class for a given input $\mathbf{x}$:

for each class c :

log_post$[c] \leftarrow \log P(c)$

for each feature $i \in \{1, \dots, d\}$:

log_post$[c] += \log \text{Gaussian}(x[i] \mid \mu[c][i], \sigma^2[c][i])$

return class with maximum log_post

Compare With sklearn GaussianNB

Motivation

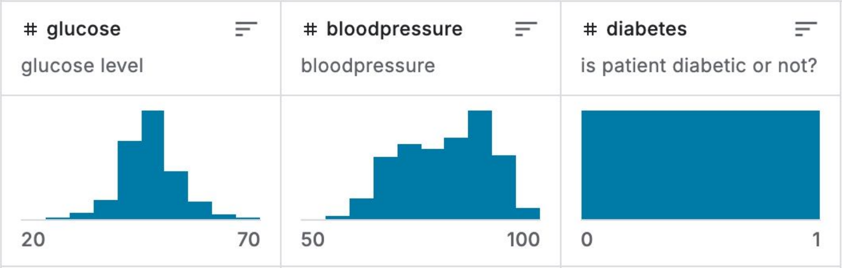
- We implemented Gaussian Naive Bayes from scratch
- We need to verify that our math and implementation are correct
- sklearn’s GaussianNB is a trusted reference model
- Matching sklearn confirms correctness and numerical stability

Dataset Introduction

Public Dataset from Kaggle: Naive-Bayes-Classification-Data.csv

The dataset contains 995 entries and 3 columns:

- `glucose`: numeric feature (blood glucose level)
- `bloodpressure`: numeric feature (blood pressure)
- `diabetes`: binary label (0 = no diabetes, 1 = diabetes)



NaKrani, H. (2023). Naive Bayes Classification Data [Dataset]. Kaggle. https://www.kaggle.com/datasets/himanshunakrani/naive-bayes-classification-data

Data Preparation & Experiment Setup

```python
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('Naive-Bayes-Classification-Data.csv')
df.head()

X = df[['glucose', 'bloodpressure']].values
y = df['diabetes'].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

- Selected continuous features: glucose, blood pressure
- Target label: diabetes (0 or 1)
- Train-test split: 70% / 30%
- `random_state = 42` for reproducibility
- Used `stratify=y` to keep class balance

Results: Our Model vs sklearn GaussianNB

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Train our model
model_my = GaussianNaiveBayes()
model_my.train(X_train, y_train)

# Predict with our model
y_pred_my = model_my.predict(X_test)
acc_my = accuracy_score(y_test, y_pred_my)

# Train sklearn model
model_sk = GaussianNB()
model_sk.fit(X_train, y_train)

# Predict with sklearn model
y_pred_sk = model_sk.predict(X_test)
acc_sk = accuracy_score(y_test, y_pred_sk)

# Compare results
print("My model accuracy:       ", acc_my)
print("Sklearn model accuracy:  ", acc_sk)
print("Predictions identical:   ", np.array_equal(y_pred_my, y_pred_sk))
```

```
My model accuracy:        0.9264214046822743
Sklearn model accuracy:   0.9264214046822743
Predictions identical:    True
```

- Trained both models on the same training split
- Tested both on the same test split
- Our model accuracy: **0.9264**
- sklearn accuracy: **0.9264**
- Predictions were identical sample-by-sample

Why Do the Results Match?

- Gaussian NB uses closed-form estimates for parameters
- Both implementations compute same:
  - class priors
  - means
  - variances
- Both use log-likelihood for stability
- So identical predictions are expected when math is correct

Summary

- Interesting
  - Gaussian NB has very clean math
    - Parameters have closed-form formulas (no optimization)
    - Only needs simple statistics (means & variances)
    - Extremely fast to train
- Challenges
  - Equations

Thank You