Bayes Classifier

Probabilistic Classification

- Generative classification with Maximum A Posterior Rule

$$p(C_j|x_1, x_2, \dots, x_N) = \frac{p(x_1, x_2, \dots, x_N|C_j)p(C_j)}{p(x_1,x_2,\dots,x_N)},$$

  where $p(C_j|x_1, x_2, \dots, x_N)$ is the posterior probability of the class membership, i.e., the probability that X belong to C_j.

- Learning the joint probability $p(x_1, x_2, \dots, x_N|C_j)$ is a difficult task.

- We used Naïve Bayes classification, with the assumption that all the input features are conditionally independent!

$$p(x_1, x_2, \dots, x_N|C_j) \propto \prod_{k=1}^{N} p(x_k|C_j)$$

- Using Bayes' rule above, we label a new case X with a class level C_j that achieves the highest posterior probability.

Naïve Bayes - Pros

- Naïve Bayes is based on the independence assumption and can therefore train 1000+ features easily.
  - Training is very easy and fast; just requiring considering each attribute in each class separately.
  - Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions.
- A popular generative model.
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption.
  - Many successful applications, e.g., spam mail filtering.

Relevant Issues

- **Violation of Independence Assumption**
  - For many real world tasks, $p(x_1, x_2, \dots, x_N | C_j) \neq \prod_{k=1}^{N} p(x_k | C_j)$
  - Nevertheless, Naïve Bayes works surprisingly well anyway!

- **Zero conditional probability Problem**
  - If there is a class C_i, and feature x_k contains the attribute value, such that none of the samples in C_i has that attribute value, $p(x_k = a_{ki} | C_j) = 0$
  - In this circumstance, $p(x_1 | C_j) \times \cdots \times p(x_k = a_{ki} | C_j) \times \cdots \times p(x_N | C_j) = 0$ during test, wiping out all information in the other probabilities when they are multiplied
  - For a remedy, conditional probabilities estimated with: **Laplace Smoothing**

Laplace Smoothing – Correction to Zero probability

- To eliminate zeros joint probability, we use add-one or Laplace smoothing, whose basic idea is to pretend that you saw every feature-class outcome pair k extra times.

- Adds arbitrary low probabilities in such cases so that the probability computation does not become zero.

$$p(x_k = x_{ki}|C_j) = \frac{count(x_k = x_{ki}|C_j) + k}{count(C_j) + k|x_k|}$$

- k: Laplace Smoothing Factor
- $|x_k|$: Number of different values feature x_k can take

Laplace Smoothing – Correction to Zero probability

- Consider class = 1, and a feature x_k containing values = [low, medium, high] in a dataset containing 1000 samples, such that:
  - 0 samples with income = low
  - 990 sample with income = medium
  - 10 samples with income = high
- The likelihood (conditional probability) without the Laplacian correction is:
  - $p(\text{income} = \text{low}|C_1) = 0$
  - $p(\text{income} = \text{medium}|C_1) = \frac{990}{1000} = 0.99$
  - $p(\text{income} = \text{high}|C_1) = \frac{10}{1000} = 0.01$
- We will use $k = 1$, as the **Laplace Smoothing Factor** for each of the three feature values.

Laplace Smoothing – Correction to Zero probability

$$p(x_k = x_{ki}|C_j) = \frac{\text{count}(x_k = x_{ki}|C_j) + k}{\text{count}(C_j) + k|x_k|}$$

- Using Laplace Correction, the dataset now contains 1003 samples, such that:
  - 1 samples with income = low ⇒ $p(\text{income} = \text{low}|C_1) = \frac{1}{1003} = 0.001$
  - 991 sample with income = medium ⇒ $p(\text{income} = \text{medium}|C_1) = \frac{991}{1003} = 0.988$
  - 11 samples with income = high ⇒ $p(\text{income} = \text{high}|C_1) = \frac{11}{1003} = 0.011$

- The “corrected” probability estimates are close to their “uncorrected” counterparts, with the zero probability value being avoided

Relevant Issues

- **<u>Continuous Valued Features:</u>**
  - When a feature is continuous, computing the probabilities by the traditional method of frequency counts is not possible.
  - Two possible solutions:
    - Discretization: Convert the feature values to discrete values - Binning
    - Probability Density Functions: Compute probability densities instead of actual probabilities

$$p(x_k|C_j) = \frac{1}{\sqrt{2\,\pi}\sigma_{kj}} e^{\left(-\frac{(x_k-\mu_{kj})^2}{2\sigma_{kj}^2}\right)}$$

- μ_{kj}: mean of the feature values x_k of examples for which $C = C_j$
- σ_{kj}: standard deviation of the feature values x_k of examples for which $C = C_j$

Naïve Bayes - Types

- The different Naïve Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $p(x_k|C_j)$

- Scikit-learn implements three naïve Bayes variants based on the different probabilistic distributions:

1. Bernoulli – Deals with a binary distribution, useful when a feature can be present or not

2. Multinomial – This is a discrete distribution and is used whenever a feature must be represented by a whole number (in NLP, it can be the frequency of a term)

3. Gaussian - This is a continuous distribution characterized by its mean and variance

Bernoulli Naïve Bayes (BNB)

$$p(A) = \frac{4}{10} = 0.4, \qquad p(B) = \frac{6}{10} = 0.6$$

- The features in BNB is assumed to be a binary-valued variable (0 and 1), with each class requiring samples to be represented as binary-valued feature vectors. Mainly used in text classification.

$$p(x_1^0|A) = \frac{3}{4} = 0.75, \qquad p(x_1^0|B) = \frac{2}{6} = 0.33$$

$$p(x_1^1|A) = \frac{1}{4} = 0.25, \qquad p(x_1^1|B) = \frac{4}{6} = 0.66$$

$$p(x_2^0|A) = \frac{2}{4} = 0.5, \qquad p(x_2^0|B) = \frac{3}{6} = 0.5$$

$$p(x_2^1|A) = \frac{2}{4} = 0.5, \qquad p(x_2^1|B) = \frac{3}{6} = 0.5$$

$$p(A|x_1, x_2) \propto p(A)p(x_1|A)p(x_2|A)$$

$$p(B|x_1, x_2) \propto p(B)p(x_1|B)p(x_2|B)$$

x_1	x_2	Class
0	1	A
1	1	B
0	0	A
1	1	A
1	0	B
0	0	A
1	1	B
0	0	B
0	1	B
1	0	B

Bernoulli Naïve Bayes (BNB)

- Using Naïve Bayes: $\mathrm{p}(\mathrm{A}) = 0.4, \quad \mathrm{p}(\mathrm{B}) = 0.6$

$\mathrm{p}(x_1^0|\mathrm{A}) = 0.75, \quad \mathrm{p}(x_1^0|\mathrm{B}) = 0.33 \quad \mathrm{p}(x_2^0|\mathrm{A}) = 0.5, \quad \mathrm{p}(x_2^0|\mathrm{B}) = 0.5$

$\mathrm{p}(x_1^1|\mathrm{A}) = 0.25, \quad \mathrm{p}(x_1^1|\mathrm{B}) = 0.66 \quad \mathrm{p}(x_2^1|\mathrm{A}) = 0.5, \quad \mathrm{p}(x_2^1|\mathrm{B}) = 0.5$

$$\mathbf{p}(\mathbf{B}|\mathbf{x}_1^1, \mathbf{x}_2^0) \propto \mathrm{p}(\mathrm{B})\mathrm{p}(\mathrm{x}_1^1|\mathrm{B})\mathrm{p}(\mathrm{x}_2^0|\mathrm{B}) = 0.6 \times 0.66 \times 0.5 = 0.2 = \frac{0.2}{0.05 + 0.2} = 0.8$$

$$\mathbf{p}(\mathbf{A}|\boldsymbol{x}_1^1, \boldsymbol{x}_2^0) \propto \mathrm{p}(\mathrm{A})\mathrm{p}(x_1^1|\mathrm{A})\mathrm{p}(x_2^0|\mathrm{A}) = 0.4 \times 0.25 \times 0.5 = 0.05 = \frac{0.05}{0.05 + 0.2} = 0.2$$

$$\mathbf{p}(\mathbf{B}|\mathbf{x}_1^0, \mathbf{x}_2^0) \propto \mathrm{p}(\mathrm{B})\mathrm{p}(\mathrm{x}_1^0|\mathrm{B})\mathrm{p}(\mathrm{x}_2^0|\mathrm{B}) = 0.6 \times 0.33 \times 0.5 = 0.4 \text{ (Normalized)}$$

$$\mathbf{p}(\mathbf{A}|\boldsymbol{x}_1^0, \boldsymbol{x}_2^0) \propto \mathrm{p}(\mathrm{A})\mathrm{p}(x_1^0|\mathrm{A})\mathrm{p}(x_2^0|\mathrm{A}) = 0.4 \times 0.75 \times 0.5 = 0.6 \text{(Normalized)}$$

$\mathbf{p}(\mathbf{B}|\mathbf{x}_1^1, \mathbf{x}_2^1) = 0.8$ (Normalized) $\quad \mathbf{p}(\mathbf{A}|\boldsymbol{x}_1^1, \boldsymbol{x}_2^1) = 0.2$(Normalized)

$\mathbf{p}(\mathbf{B}|\mathbf{x}_1^0, \mathbf{x}_2^1) = 0.4$ (Normalized) $\quad \mathbf{p}(\mathbf{A}|\boldsymbol{x}_1^0, \boldsymbol{x}_2^1) = 0.6$(Normalized)

Multinomial Naïve Bayes

- Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. Mainly used for document classification.

- Each value of the feature vector represents for example the number of occurrences of a term or its relative frequency.

- The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y,
  - n is the number of features (in text classification, the size of the vocabulary).
  - θ_{yi} is the probability $\mathrm{p}(\mathrm{x_i}|\mathrm{y})$ of feature i appearing in a sample belonging to class y.

Multinomial Naïve Bayes

- The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y,
  - n is the number of features (in text classification, the size of the vocabulary).
  - θ_{yi} is the probability $p(x_i|y)$ of feature i appearing in a sample belonging to class y.

- The parameters $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ is estimated by relative frequency counting:

$$\widehat{\theta_{yi}} = \frac{N_{yi} + k}{N_y + k\,n}, \qquad \text{with}$$

  - $N_{yi} = \sum_{x \in T} x_i$, i.e. the number of times feature i appears in a sample of class y in the training set T.
  - $N_y = \sum_{i=1}^{n} N_{yi}$, is the total count of all features for class y.
  - k is the Laplace smoothing factor.

Gaussian Naïve Bayes

- In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution.

$$p(x_k|C_j) = \frac{1}{\sqrt{2\,\pi\sigma_{kj}}} e^{\left(-\frac{(x_k-\mu_{kj})^2}{2\sigma_{kj}^2}\right)}$$

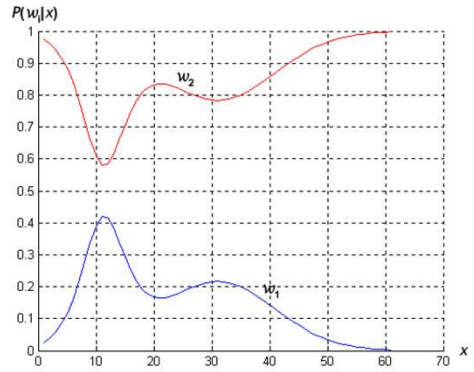
- The conditional probabilities $p(x_k|C_j)$ are also Gaussian distributed, therefore it is necessary to estimate the mean and variance of each of them using the maximum likelihood approach.

$$L(\mu; \sigma^2; x_i|C_j) = \log \prod_k p\left(x_i^{(k)}\middle|C_j\right) = \sum_k \log p\left(x_i^{(k)}\middle|C_j\right)$$

Probability of Error

- For a given observation x, we would be inclined to let the posterior govern our decision.
- What is the probability of error?
- For the 2 class situation, we have:

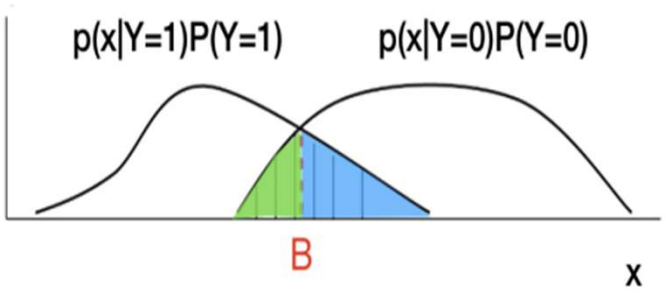
$$\mathrm{p(error|x)} = \begin{cases} \mathrm{p(C_0|x)} & \text{if we decide } \mathrm{C_1} \\ \mathrm{p(C_1|x)} & \text{if we decide } \mathrm{C_2} \end{cases}$$



- This is the probability of making a mistake because we always pick the label i with the highest $\mathrm{p(C_i|x)}$
- However, there is still a probability that we are wrong: this probability is the probability of the less likely label.

Bayes Error Rate

- Suppose that we knew the true probabilities: $p(x|C_0), p(x|C_1)$
- Observe any x: $\Rightarrow p(y = C_0|x), p(y = C_1|x)$ (at any x)
- Optimal decision at any particular x is:



$$\hat{y} = f(x) = \arg\max_{c_i} p(y = C_i|x)$$

- Error rate is: $\mathbb{E}_{xy}[y \neq \hat{y}] = \mathbb{E}_x[1 - \max_c p(y = C_i|x)] =$ Bayes Error Rate
  - This is the best that any classifier can do!
  - Measures fundamental hardness of separating y –values given only features x

Conditional Loss Function

- Let $\{C_0, C_1, \ldots, C_c\}$ be the finite set of c classes and $\{\alpha_1, \alpha_2, \ldots, \alpha_a\}$ be the finite set of a possible actions.
- The Loss function (λ) can be defined as the loss suffered on taking an action α of assigning an input to class C_i when it should have been in class C_j, represented as $\lambda(\alpha_i|C_j)$.
- Provided $\mathbf{i} = \mathbf{j}$, then we get a smaller value of the loss as compared to the alternative cases because it corresponds to a correct decision.
- As $\mathrm{p}(\mathrm{C_j}|\mathrm{x})$ is the probability that the true state of nature is C_j, the **expected loss** associated with taking action α_i is:

$$\mathrm{R}(\alpha_\mathrm{i}|\mathrm{x}) = \sum_{\mathrm{j}=1}^{\mathrm{c}} \lambda(\alpha_\mathrm{i}|\mathrm{C_j}) \times \mathrm{p}(\mathrm{C_j}|\mathrm{x})$$

Conditional Risk

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|C_j) \times p(C_j|x)$$

- $R(\alpha_i|x)$ is called the **conditional risk**.

- We can minimize our expected loss by selecting the action that minimizes the conditional risk.
- Bayes decision procedure provides the optimal performance. We need to find a decision rule that minimizes the overall risk.
  - A general decision rule is a function $\alpha(x)$ that tells us which action to take for every possible observation x.
  - Because conditional risk is associated with action α_i and because the decision rule specifies the action, the overall risk is given by:

$$R = \int R(\alpha(x)|x) \times p(x)dx$$

Bayesian Decision Theory

- Let λ be symmetrical or zero-one loss function.

$$\lambda(\alpha_i|C_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}$$

- Then the conditional risk becomes:

$$R(\alpha_i|\bar{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|C_j) \times p(C_j|\bar{x})$$

$$\Longrightarrow R(\alpha_i|\bar{x}) = \sum_{j \neq i}^{c} p(C_j|\bar{x}) = 1 - p(C_i|\bar{x})$$

- The very decision rule of Bayes' Classifier automatically minimizes the conditional risk associated with an action.

Which algorithm should I use?

Choice of the ML Model

- **Problem Type**: Nature of the problem you are trying to solve is an essential factor: classification problem, regression, clustering, or something else.
- **Dataset Size**: The size of your dataset can influence the choice of models.
- **Dataset Complexity**: Complexity of dataset refers to the number of features, the presence of nonlinear relationships, and the level of noise or outliers.
- **Interpretability**: Prefer models that are more transparent and interpretable, such as linear models or decision trees.
- **Performance Requirements**: Do you need high predictive accuracy, or is it more important to have a model that runs quickly or is memory-efficient?

Choice of the ML Model

- **Available Data**: If your dataset has missing values, outliers, or class imbalance, certain models may be more robust or offer specific techniques to handle these challenges.

- **Scalability**: Provided your model will need to scale to handle large amounts of data or real-time streaming data, consider models that are specifically designed for scalability, such as distributed computing frameworks or online learning algorithms.

- **Resource Constraints**: Consider the computational resources available for training and deploying the model.

- **Trade-offs**: There is often a trade-off between model complexity, interpretability, training time, and performance. Evaluate the pros and cons of each model.

Choice of ML Algorithm

- **The size of data**

  - Small Data: Linear Regression, Naïve Bayes, Decision Trees
  - Medium Data: Random Forests, Support Vector Machines, Gradient Boosting, KNN
  - Large Data: Deep Learning Models

- **Dataset Complexity**

  - Low Complexity: Linear Regression, Logistic Regression, Naïve Bayes, Decision Trees
  - Medium Complexity: Random Forests, Support Vector Machines, Gradient Boosting, KNN
  - Large Complexity: Deep Learning, CNN, RNN, LSTM

Bias vs. Variance

1. **Low Bias, High Variance**: KNN, Decision Trees, SVM
2. **High Bias, Low Variance**: Linear Regression, Logistic Regression, Naïve Bayes

a) A model with a high bias error underfits data and makes very simplistic assumptions on it

b) A model with a high variance error overfits the data and learns too much from it

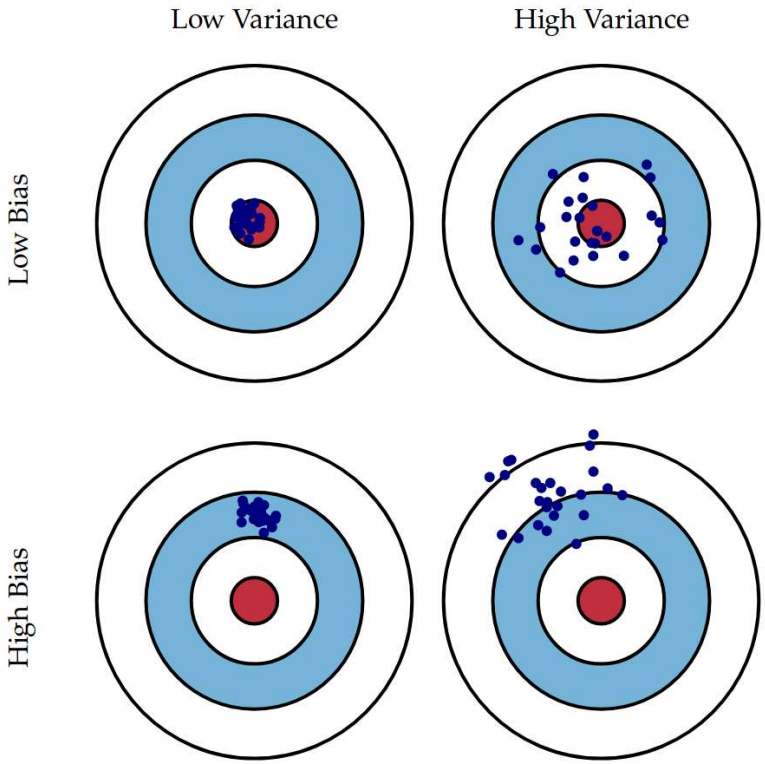


Image Source: From the internet

Choice of ML Algorithm

- **Available Data: Robust Algorithms**

Robust algorithms are the ones, which can handle missing values, outliers as well as class imbalance more effectively

- Decision Trees, Random Forests, Gradient Boosting
- Support Vector Machines
- Nearest Neighbours Algorithms
- Neural Networks

Accuracy vs. Speed

- Which is of more value to your project?

Accuracy?

Go for complex algorithms like SVM, Neural Networks, and Random Forests

Time?

Go for a simpler algorithm like Naïve Bayes, Linear/Logistic Regression