Naive Bayes Classifiers

Shrey Gupta

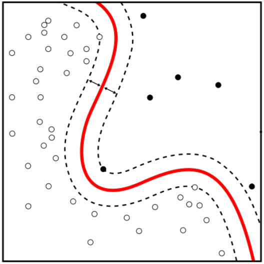
Applied Machine Learning (HOUSECS 59-01), Duke University

October 3, 2018

Supervised Learning

- Data (a subset from a larger distribution) is labeled, and we attempt to generalize to (predict) the larger distribution.
- Classification: predicts a discrete class output.

Classification: Examples



- Given data about temperature, humidity, and wind speed, predict whether it will be sunny, cloudy, or raining.
- Predict whether the price of an equity will increase or decrease.

Image source: Wikipedia

Recall

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & \dots & x_2^{(m)} \\ \vdots & & & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & x_n^{(3)} & \dots & x_n^{(m)} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Data is stored in matrices and vectors.
- Given n (training) data points and m features (per data point).
- Given labeled data vector y.

Recall

$$X_{test} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & \dots & x_2^{(m)} \\ \vdots & & & \ddots & \vdots \\ x_k^{(1)} & x_k^{(2)} & x_k^{(3)} & \dots & x_k^{(m)} \end{bmatrix}, \hat{y}_{test} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_k \end{bmatrix}$$

- Given k testing data points and m features (per data point).
- $\hat{y}_{test} = f(X_{test})$ contains *predictions* of the classification algorithm, where $f(\cdot)$ is learned by the algorithm.
- How do we define $f(\cdot)$, and how does the algorithm "learn" it?

Introduction

- The Naive Bayes algorithm is a simple probabilistic classification algorithm.
- It is “naive” in the sense that it assumes features to be *independent* and *equal*. Note this doesn’t always hold true, but the algorithm often works well in practice.
- The argument is largely based on *Bayes’ theorem*, a crucial probability theorem.

Notation

- Define $P(A)$ to be the *probability* of an *event* A.
- Let $P(A|B)$ be the *conditional probability* of event A on event B. That is, assuming event B happens, what is the probability of event A?

Goal

$$P(y|X) = P(y|x^{(1)}, x^{(2)}, ..., x^{(m)})$$

$$\hat{y} = f(X) = \arg\max_{y} P(y|X)$$

- What is the probability of class y given features $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$?
- Our prediction $\hat{y}$ will be the class y that maximizes such probability.

Optional: Bayes' Theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} = \frac{\textit{likelihood} \times \textit{prior}}{\textit{evidence}}$$

- Prior: $P(A)$ is the probability of event A prior to observing any "evidence".
- Likelihood: $P(B|A)$ is the likelihood of observing evidence B given event A.
- Evidence: B is the evidence observed.
- Posterior: $P(A|B)$ is the probability of event A after observing evidence B.

Optional: Theorem Visualization

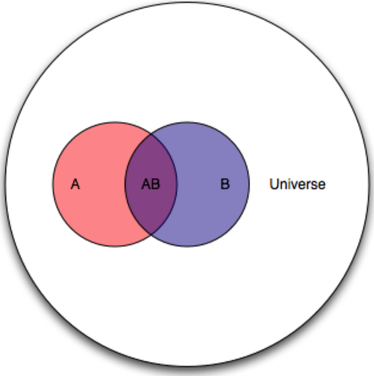


Image source: Oscar Bonilla

Optional: Theorem Derivation

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

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

Optional: Algorithm Derivation

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$
$$\propto P(X|y) \cdot P(y)$$

- $P(X)$ can be treated as a constant across all different y under examination, and hence can be "discarded".
- Therefore, our goal is to compute $P(X|y) \cdot P(y) \; \forall \; y$.

Optional: Algorithm Derivation

$$P(X|y) \cdot P(y) = P(X, y) = P(x^{(1)}, x^{(2)}, ..., x^{(m)}, y)$$
$$= P(x^{(1)}|x^{(2)}, ..., x^{(m)}, y) \cdot P(x^{(2)}, ..., x^{(m)}, y)$$
$$= P(x^{(1)}|x^{(2)}, ..., x^{(m)}, y) \cdot P(x^{(2)}|x^{(3)}, ..., x^{(m)}, y) \cdot P(x^{(3)}, ..., x^{(m)}, y)$$
$$= P(x^{(1)}|x^{(2)}, ..., x^{(m)}, y) \cdots P(x^{(m)}|y) \cdot P(y)$$

- We can "naively" assume conditional independence between features so that $P(x^{(i)}|x^{(i+1)}, ..., x^{(m)}, y) = P(x^{(i)}|y)$.

Optional: Algorithm Derivation

$$
\begin{aligned}
& P(x^{(1)}|x^{(2)}, ..., x^{(m)}, y) \cdots P(x^{(m)}|y) \cdot P(y) \\
&= P(x^{(1)}|y) \cdot P(x^{(2)}|y) \cdots P(x^{(m)}|y) \cdot P(y) \\
&= P(y) \cdot \prod_i P(x^{(i)}|y)
\end{aligned}
$$

- We can normalize over all classes y_j to find the probability $P(y|X)$. More formally,

$$
P(y|X) = \frac{P(y) \cdot \prod_i P(x^{(i)}|y)}{\sum_j P(y_j) \cdot \prod_i P(x^{(i)}|y_j)}.
$$

Practicalities

- In practice, it is not necessary to normalize and compute $P(y|X)$.
- Remember: $P(X)$ is effectively a constant across all y, so we only need to maximize the numerator.

Algorithm

$$g(X, y) = P(y) \cdot \prod_i P(x^{(i)}|y)$$

$$\hat{y} = f(X) = \arg\max_y g(X, y)$$

- Compute *score* $g(X, y)$ over classes y.
- Assign prediction $\hat{y}$ to the class y that maximizes such score.
- $P(y)$ and $P(x^{(i)}|y)$ can be easily calculated using the data. However, this assumes the features are *discrete*.

Continuous Features

$$P(x^{(i)}|y) = \frac{1}{\sqrt{2\pi(\sigma_y^{(i)})^2}} \cdot e^{-\frac{(x^{(i)}-\mu_y^{(i)})^2}{2(\sigma_y^{(i)})^2}}$$

- If a feature is instead *continuous* and we assume it to be *normally distributed*, we can calculate the feature's mean $\mu_y^{(i)}$ and variance $(\sigma_y^{(i)})^2$ over class y.
- $P(x^{(i)}|y)$ can then be calculated using the above formula.

Notebook

- We've introduced several theoretical concepts today, so some (or a lot) of it might be confusing.
- Today's notebook will work through an example of the Naive Bayes classifier with discrete and continuous features.