# Naïve Bayes Classifier

Seoul Al Meetup Martin Kersner, 2017/06/10

## Introduction

- Classifier.
- Handles multiple classes.
- Nominal values (in case of Multinomial Naïve Bayes Classifier)
- Does not need much of training data.
- Decent classifier, bad estimator.
- Types
  - Gaussian Naïve Bayes Classifier
  - Multinomial Naïve Bayes Classifier
  - Bernoulli Naïve Bayes Classifier
- Example usage
  - Document classification
  - Spam Filtering

```
How does it work?
```

$$P(y_{1}|X) \ge P(y_{2}|X)$$
  

$$P(y_{1}|X) < P(y_{2}|X)$$
  

$$argmax_{n} P(y_{n}|X)$$
 multi-class classifier

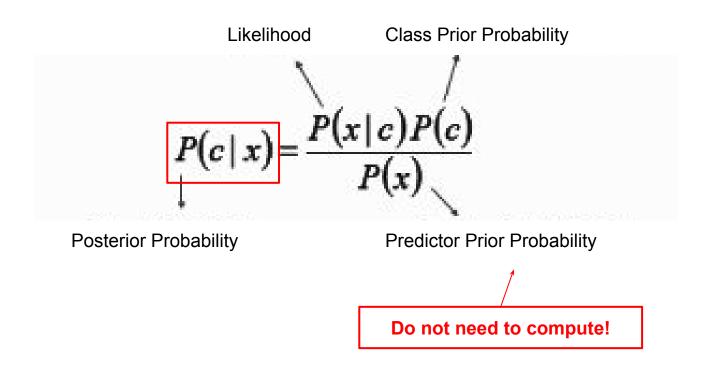
**X** represents feature vector.

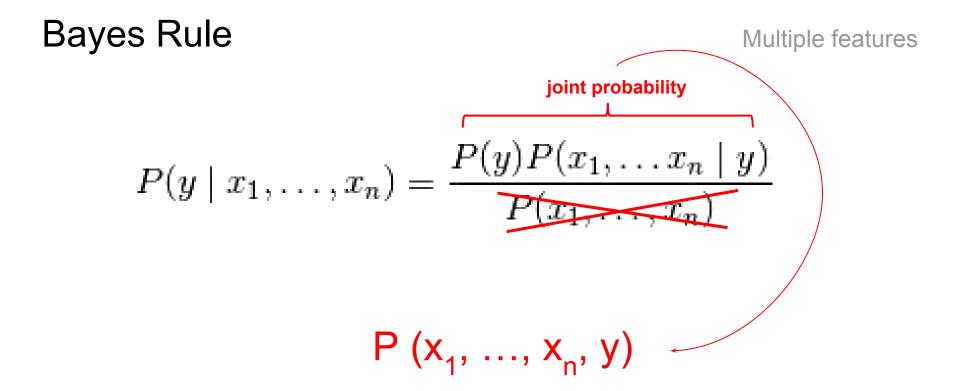
y and c symbols in following slides represent the same thing; class.

From (Con)joint probability to Bayes Rule P(A and B) = P(A)P(B|A)P(B and A) = P(B)P(A|B)P(A and B) = P(B and A)P(A)P(B|A) = P(B)P(A|B)P(B|A) = (P(B)P(A|B)) / P(A)

#### **Bayes Rule**

Single feature





#### Chain Rule of Conditional Probability

$$egin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \ &= p(x_1 \mid x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \ &= p(x_1 \mid x_2, \dots, x_n, C_k) p(x_2 \mid x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \ &= \dots \ &= p(x_1 \mid x_2, \dots, x_n, C_k) p(x_2 \mid x_3, \dots, x_n, C_k) \dots p(x_{n-1} \mid x_n, C_k) p(x_n \mid C_k) p(C_k) \end{aligned}$$

#### Naïvity of Naïve Bayes Classifier

$$p(C_k \mid x_1, \dots, x_n) \propto p(C_k, x_1, \dots, x_n) \ \propto p(C_k) \ p(x_1 \mid C_k) \ p(x_2 \mid C_k) \ p(x_3 \mid C_k) \ \cdots \ \propto p(C_k) \prod_{i=1}^n p(x_i \mid C_k) \,.$$

 $\propto$  represents relation called "is proportional to".

# Multinomial Naïve Bayes Classifier

- 1. Create vocabulary and convert features accordingly.
- 2. Compute  $p(C_k)$ .
  - a. Sum number of  $C_k$  instances and divide by total number of training examples (scikit-learn *fit\_priors=True* parameter).
  - b. OR set manually based on prior knowledge about class distribution (scikit-learn class\_prior=[...] parameter).
- 3. Compute for every  $\mathbf{x}_i$ ,  $\mathbf{p}(\mathbf{x}_i | \mathbf{C}_k)$ , where  $\mathbf{x}_i$  is i-th feature.
  - a. What is the probability of feature occurrence  $\mathbf{X}_{i}$  in class  $\mathbf{C}_{k}$ ?
  - b. Element-wise sum of all feature vectors in from each class.
  - c. Normalize. Divide each element by a sum of all feature occurrences from one class.
- 4. For each class we will end up with <u>vector of probabilities for each feature from</u> <u>vocabulary</u>.

## Multinomial Naïve Bayes Classifier

Predict

- 1. Convert features according to vocabulary  $\Rightarrow x_{+}$
- 2. Multiply all relevant  $p(x_i | C_k)$  and  $p(C_k)$  probabilities.
  - a. Perform for each class.
  - b. Compare.
  - c. Select class with the highest **output**.

- Potential issues
  - If any of  $p(x|C_k)$  probabilities is zero, then **probability of whole document is zero as well**!
  - Multiplication of small values  $\Rightarrow$  underflow  $\Rightarrow$  ln(a\*b) = ln(a) + ln(b)

# scikit-learn MultinomialNB

- For multinomial data distribution.
- Default parameters
  - alpha [= 1.0]
  - fit\_prior [= True]
  - class\_prior [= None]

#### >>> import numpy as np

- >>> X = np.random.randint(5, size=(6, 100))
- >>> y = np.array([1, 2, 3, 4, 5, 6])
- >>> from sklearn.naive\_bayes import MultinomialNB

```
>>> clf = MultinomialNB()
```

>>> clf.fit(X, y)

MultinomialNB(alpha=1.0, class\_prior=None, fit\_prior=True)

```
>>> print(clf.predict(X[2:3]))
```

# cl-ml [1] naive-bayes-classifier

- Inspired by 4th chapter of Machine Learning in Action [2].
- Multinomial Naive Bayes **Binary** Classifier.

<u>https://github.com/martinkersner/cl-ml</u>
 <u>https://www.manning.com/books/machine-learning-in-action</u>

#### References

- <u>http://scikit-learn.org/stable/modules/naive\_bayes.html</u>
- <u>https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/</u>
- <u>http://greenteapress.com/wp/think-bayes/</u>
- https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier
- https://en.wikipedia.org/wiki/Joint\_probability\_distribution