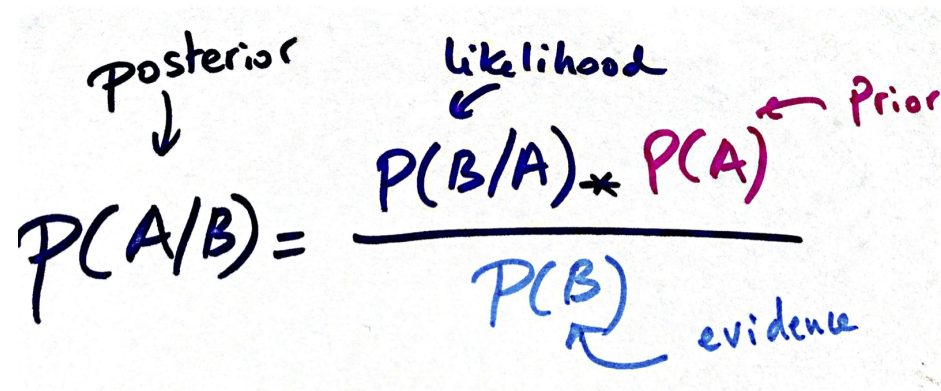


# Artificial Intelligence

## Machine Learning

### Naive Bayes



Handwritten formula for Naive Bayes theorem with annotations:

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

Annotations:

- posterior (points to  $P(A/B)$ )
- likelihood (points to  $P(B/A)$ )
- Prior (points to  $P(A)$ )
- evidence (points to  $P(B)$ )

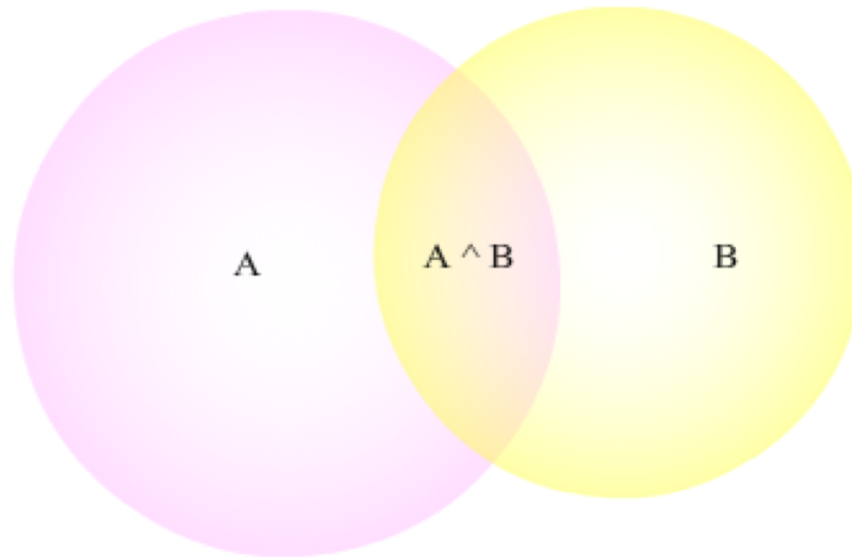
# Outline

---

1. Generative models
2. Naive Bayes Classifier.
3. Setting
4. Example
5. Estimating probabilities

# Conditional Probability

---



$$p(A|B) = \frac{p(A \wedge B)}{p(B)}$$

$$p(A \wedge B) = p(A|B) * p(B)$$

# Bayes Rule

---

Writing  $p(A \wedge B)$  in two different ways:

$$p(A \wedge B) = p(B|A) * p(A)$$

$$p(A \wedge B) = p(A|B) * p(B)$$

# Bayes Rule

---

Writing  $p(A \wedge B)$  in two different ways:

$$p(A \wedge B) = p(B|A) * p(A)$$

$$p(A \wedge B) = p(A|B) * p(B)$$

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B)}$$

# Bayes Rule

---

Writing  $p(A \wedge B)$  in two different ways:

$$p(A \wedge B) = p(B|A) * p(A)$$

$$p(A \wedge B) = p(A|B) * p(B)$$

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B)}$$

$p(A|B)$  is called posterior (posterior distribution on  $A$  given  $B$ .)

$p(A)$  is called prior.

$p(B)$  is called evidence.

$p(B|A)$  is called likelihood.

# Bayes Rule

---

	A	not A	Sum
B	$P(A \text{ and } B)$	$P(\text{not } A \text{ and } B)$	$P(B)$
Not B	$P(A \text{ and not } B)$	$P(\text{not } A \text{ and not } B)$	$P(\text{not } B)$
	$P(A)$	$P(\text{not } A)$	1

- This table divides the sample space into 4 mutually exclusive events.
- The probability in the margins are called marginals and are calculated by summing across the rows and and the columns.

# Bayes Rule

---

	A	not A	Sum
B	P(A and B)	P(not A and B)	P(B)
Not B	P(A and not B)	P(not A and not B)	P(not B)
	P(A)	P(not A)	1

- This table divides the sample space into 4 mutually exclusive events.
- The probability in the margins are called marginals and are calculated by summing across the rows and and the columns.

Another form:

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$



# Example of Using Bayes Rule

---

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

**A:** patient has cancer.

**B:** patient has a positive lab test.

# Example of Using Bayes Rule

---

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

**A:** patient has cancer.

**B:** patient has a positive lab test.

$$p(A) = 0.008$$

$$p(\neg A) = 0.992$$

# Example of Using Bayes Rule

---

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

**A:** patient has cancer.

**B:** patient has a positive lab test.

$$p(A) = 0.008$$

$$p(B|A) = 0.98$$

$$p(\neg A) = 0.992$$

$$p(\neg B|A) = 0.02$$

# Example of Using Bayes Rule

---

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

**A:** patient has cancer.

**B:** patient has a positive lab test.

$$p(A) = 0.008$$

$$p(B|A) = 0.98$$

$$p(B|\neg A) = 0.03$$

$$p(\neg A) = 0.992$$

$$p(\neg B|A) = 0.02$$

$$p(\neg B|\neg A) = 0.97$$

# Example of Using Bayes Rule

---

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

**A:** patient has cancer.

**B:** patient has a positive lab test.

$$p(A) = 0.008$$

$$p(B|A) = 0.98$$

$$p(B|\neg A) = 0.03$$

$$p(\neg A) = 0.992$$

$$p(\neg B|A) = 0.02$$

$$p(\neg B|\neg A) = 0.97$$

$$p(A|B) = \frac{0.98 * 0.008}{0.98 * 0.008 + 0.03 * 0.992} = 0.21$$

# Why probabilities?

---

Why are we bringing here Bayes rule?

# Why probabilities?

---

Why are we bringing here Bayes rule?

Recall Classification framework:

**Given:** Training data:  $(x_1, y_1), \dots, (x_n, y_n) / x_i \in \mathbb{R}^d$  and  $y_i \in \mathbb{Y}$ .

**Task:** Learn a classification function:  $f : \mathbb{R}^d \longrightarrow \mathbb{Y}$

# Why probabilities?

---

**Why are we bringing here a Bayesian framework?**

Recall Classification framework:

**Given:** Training data:  $(x_1, y_1), \dots, (x_n, y_n) / x_i \in \mathbb{R}^d$  and  $y_i \in \mathbb{Y}$ .

**Task:** Learn a classification function:  $f : \mathbb{R}^d \longrightarrow \mathbb{Y}$

Learn a mapping from  $x$  to  $y$ .

We would like to find this mapping  $f(x) = y$  through  $p(y|x)$ !



# Discriminative Algorithms

---

- **Discriminative Algorithms:**
  - Idea: model  $p(y|x)$ , conditional distribution of  $y$  given  $x$ .
  - In Discriminative Algorithms: find a decision boundary that separates positive from negative example.
  - To predict a new example, check on which side of the decision boundary it falls.
  - Model  $p(y|x)$  directly.

# Generative Algorithms

---

- **Generative Algorithms** adopt a different approach:
  - Idea: Build a model for what positive examples look like.  
Build a different model for what negative example look like.
  - To predict a new example, match it with each of the models and see which match is best.
  - Model  $p(x|y)$  and  $p(y)$ !
  - Use Bayes rule to obtain  $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$ .

# Generative Algorithms

---

- **Generative Algorithms** adopt a different approach:
  - Idea: Build a model for what positive examples look like.  
Build a different model for what negative example look like.
  - To predict a new example, match it with each of the models and see which match is best.
  - Model  $p(x|y)$  and  $p(y)$ !
  - Use Bayes rule to obtain  $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$ .
  - To make a prediction:

$$\operatorname{argmax}_y p(y|x) = \operatorname{argmax}_y \frac{p(x|y)p(y)}{p(x)}$$

$$\operatorname{argmax}_y p(y|x) \approx \operatorname{argmax}_y p(x|y)p(y)$$

# Naive Bayes Classifier

---

- Probabilistic model.
- Highly practical method.
- Application domains to natural language text documents.
- Naive because of the strong independence assumption it makes (not realistic).
- Simple model.
- Strong method can be comparable to decision trees and neural networks in some cases.

# Setting

---

- A training data  $(x_i, y_i)$ ,  $x_i$  is a feature vector and  $y_i$  is a discrete label.
- $d$  features, and  $n$  examples.
- Example: consider document classification, each example is a documents, each feature represents the presence or absence of a particular word in the document.
- We have a training set.
- A new example with feature values  $x_{new} = (a_1, a_2, \dots, a_d)$ .
- We want to predict the label  $y_{new}$  of the new example.

# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p(a_1, a_2, \dots, a_d|y) * p(y)}{p(a_1, a_2, \dots, a_d)}$$

# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p(a_1, a_2, \dots, a_d|y) * p(y)}{p(a_1, a_2, \dots, a_d)}$$

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(a_1, a_2, \dots, a_d|y) * p(y)$$



# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p(a_1, a_2, \dots, a_d|y) * p(y)}{p(a_1, a_2, \dots, a_d)}$$

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(a_1, a_2, \dots, a_d|y) * p(y)$$

**Can we estimate these two terms from the training data?**

# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p(a_1, a_2, \dots, a_d|y) * p(y)}{p(a_1, a_2, \dots, a_d)}$$

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(a_1, a_2, \dots, a_d|y) * p(y)$$

**Can we estimate these two terms from the training data?**

1.  $p(y)$  can be easy to estimate: count the frequency with which each label  $y$  occurs in the training data.

# Setting

---

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y|a_1, a_2, \dots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} \frac{p(a_1, a_2, \dots, a_d|y) * p(y)}{p(a_1, a_2, \dots, a_d)}$$

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(a_1, a_2, \dots, a_d|y) * p(y)$$

**Can we estimate these two terms from the training data?**

1.  $p(y)$  can be easy to estimate: count the frequency with which each label  $y$ .
2.  $p(a_1, a_2, \dots, a_d|y)$  is not easy to estimate unless we have a very very large sample. (We need to see every example many times to get reliable estimates)

# Naive Bayes Classifier

---

Makes a simplifying assumption that the feature values are conditionally independent given the label.

Given the label of the example, the probability of observing the conjunction  $a_1, a_2, \dots, a_d$  is the product of the probabilities for the individual features:

$$p(a_1, a_2, \dots, a_d | y) = \prod_j p(a_j | y)$$

# Naive Bayes Classifier

---

Makes a simplifying assumption that the feature values are conditionally independent given the label.

Given the label of the example, the probability of observing the conjunction  $a_1, a_2, \dots, a_d$  is the product of the probabilities for the individual features:

$$p(a_1, a_2, \dots, a_d | y) = \prod_j p(a_j | y)$$

**Naive Bayes Classifier:**

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y) \prod_j p(a_j | y)$$

# Naive Bayes Classifier

---

Makes a simplifying assumption that the feature values are conditionally independent given the label.

Given the label of the example, the probability of observing the conjunction  $a_1, a_2, \dots, a_d$  is the product of the probabilities for the individual features:

$$p(a_1, a_2, \dots, a_d | y) = \prod_j p(a_j | y)$$

**Naive Bayes Classifier:**

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y) \prod_j p(a_j | y)$$

**Can we estimate these two terms from the training data?**

# Naive Bayes Classifier

---

Makes a simplifying assumption that the feature values are conditionally independent given the label.

Given the label of the example, the probability of observing the conjunction  $a_1, a_2, \dots, a_d$  is the product of the probabilities for the individual features:

$$p(a_1, a_2, \dots, a_d | y) = \prod_j p(a_j | y)$$

**Naive Bayes Classifier:**

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y) \prod_j p(a_j | y)$$

**Can we estimate these two terms from the training data?**

Yes!

# Algorithm

---

**Learning:** Based on the frequency counts in the dataset:

1. Estimate all  $p(y)$ ,  $\forall y \in \mathbb{Y}$ .
2. Estimate all  $p(a_j|y)$   $\forall y \in \mathbb{Y}$ ,  $\forall a_i$ .

**Classification:** For a new example, use:

$$y_{new} = \operatorname{argmax}_{y \in \mathbb{Y}} p(y) \prod_j p(a_j|y)$$

Note: No model per se or hyperplane, just count the frequencies of various data combinations within the training examples.



# Example

---

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

Can we predict the class of the new example?

# Example

---

$$y_{new} = \operatorname{argmax}_{y \in \{yes, no\}} p(y) * p(Masters|y) * p(UX \ Design|y) * p(Java|y) * p(TRUE|y)$$

$$p(yes) = 8/14 = 0.572$$

$$p(no) = 6/14 = 0.428$$

Conditional probabilities:

$$p(masters|yes) = 4/8 \quad p(masters|no) = 1/6$$

$$p(UX \ Design|yes) = 2/8 \quad p(UX \ Design|no) = 2/6$$

$$p(Java|yes) = 6/8 \quad p(Java|no) = 1/6$$

$$p(TRUE|yes) = 4/8 \quad p(TRUE|no) = 3/6$$

$$p(yes) * p(Masters|yes) * p(UX \ Design|yes) * p(Java|yes) * p(TRUE|yes) = 0.026$$

$$p(no) * p(Masters|no) * p(UX \ Design|no) * p(Java|no) * p(TRUE|no) = 0.002$$

$$y_{new} = yes$$

# Estimating probabilities

---

**m-estimate of the probability:**

$$p(a_j|y) = \frac{n_c + m * p}{n_y + m}$$

where:

$n_y$ : total number of examples for which the class is  $y$ .

$n_c$ : total number of examples for which the class is  $y$  and feature  $x_j = a_j$ .

$m$ : called *equivalent sample size*

# Estimating probabilities

---

**m-estimate of the probability:**

$$p(a_j|y) = \frac{n_c + m * p}{n_y + m}$$

where:

$n_y$ : total number of examples for which the class is  $y$ .

$n_c$ : total number of examples for which the class is  $y$  and feature  $x_j = a_j$ .

$m$ : called *equivalent sample size*

## **Intuition:**

Augment the sample size by  $m$  virtual examples, distributed according to prior  $p$  (prior estimate of each value).

If prior is unknown, assume uniform prior: if a feature has  $k$  values, we can set  $p = \frac{1}{k}$ .

# Text Classification

---

Learning to classify text. Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic
- Classify Spam from non Spam emails
- Naive Bayes is among most effective algorithms
- What attributes shall we use to represent text documents?

# Text Classification

---

- Given a document (corpus), define an attribute for each word position in the document.
- The value of the attribute is the English word in that position.
- To reduce the number of probabilities that needs to be estimated, besides NB independence assumption, we assume that: The probability of a given word  $w_k$  occurrence is independent of the word position within the text. That is:

$$p(x_1 = w_k | c_j), p(x_2 = w_k | c_j), \dots$$

estimated by:

$$p(w_k | c_j)$$

# Text Classification

---

- m-estimate of the probabilities:

$$p(w_k|c_j) = \frac{n_k + 1}{n_j + |\text{Vocabulary}|}$$

where:

$n_j$ : total #word positions in all training examples of class  $c_j$ .

$n_k$ : number of times the word  $w_k$  is found in among these  $n_j$  word positions.

- The following function learns the probabilities  $P(w_k/c_j)$  describing the probability that a randomly drawn word from a document with class  $c_j$  is the English word  $w_k$ . It also learn the class priors  $P(c_j)$ .

# Naive Bayes for text

---

1. Naive Bayes is a linear classifier.
2. Incredibly simple and easy to implement.
3. Works wonderful for text.



# Credit

---

- Machine Learning. Tom Mitchell 1997.