91258 - Natural Language Processing

Lesson 4. Naïve Bayes

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna
a.barron@unibo.it @_albarron_

10/10/2023

Previously

- Pre-processing (e.g., tokenisation, stemming, stopwording)
- BoW representation
- One rule-based sentiment analyser

Table of Contents

Into ML

Machine Learning

"[...] an umbrella term for **solving problems** for which development of algorithms by human programmers would be cost-prohibitive"

"[...] the problems are solved by helping machines **"discover" their "own" algorithms**, without needing to be explicitly told what to do by any human-developed algorithms."

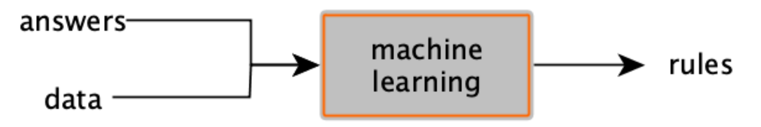
https://en.wikipedia.org/wiki/Machine_learning

Machine Learning

A change of paradigm

From hand-crafted rules

To training



Diagrams borrowed from L. Moroney's Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning

Supervised vs Unsupervised

Supervised The algorithms build a mathematical model of a set of data including...

- **the inputs**
- **desired outputs**

Unsupervised The algorithms take a set of data that contains...

- **only inputs**

...and find structure in the data

https://en.wikipedia.org/wiki/Machine_learning

Naïve Bayes

Naïve Bayes

1. Introduced in the IR community by Maron (1961)
2. First machine learning approach
3. It is a **supervised** model
4. It applies Bayes' theorem with strong (naïve) independence assumptions between the features
   - they are independent
   - they contribute "the same"

Naïve Bayes

A conditional probability model

Given an instance represented by a vector

$$\mathbf{x} = (x_1, \ldots, x_n) \tag{1}$$

representing n **independent** features $x_1, x_2, x_3, \ldots, x_{n-2}, x_{n-1}, x_n$

n could be $|V|$ (the size of the vocabulary)

The model assigns to instance $\mathbf{x}$ the probability

$$p(C_k \mid \mathbf{x}) = p(C_k \mid x_1, \ldots, x_n) \tag{2}$$

for each of the k possible outcomes C_k

where $C_k = \{c_1, \ldots, c_k\}$

From
`https://en.wikipedia.org/wiki/Naive_Bayes_classifier`

Naïve Bayes'

Using Bayes' Theorem

The conditional probability $p(C_k \mid x_1, \ldots, x_n)$ can be decomposed as

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \tag{3}$$

Which can be read as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

But $p(x)$ does not depend on the class (since it is constant):

$$p(C_k \mid \mathbf{x}) \sim p(C_k)\ p(\mathbf{x} \mid C_k) \tag{4}$$

From
`https://en.wikipedia.org/wiki/Naive_Bayes_classifier`

Naïve Bayes

Going deeper (assuming a binary classifier)

$$p(C \mid \mathbf{x}) = \frac{p(C)\ p(\mathbf{x} \mid C)}{p(\mathbf{x})} \tag{5}$$

$$\text{posterior probability} = \frac{\text{class prior probability} \times \text{likelihood}}{\text{predictor prior probability}}$$

$p(C \mid \mathbf{x})$ Posterior probability of the class given the input[1]

```
if p > 0.5:
    class = positive
else:
    class = negative
```

[1] Symbol $\mid$ means "given": the probability of the class given the representation vector

Naïve Bayes

Going deeper (assuming a binary classifier)

$$p(C \mid \mathbf{x}) = \frac{p(C)\ p(\mathbf{x} \mid C)}{p(\mathbf{x})} \tag{6}$$

$$\text{posterior probability} = \frac{\text{class prior probability} \times \text{likelihood}}{\text{predictor prior probability}}$$

$p(C)$ Class **prior** probability
How many **positive** instances I have seen (during training)?

Naïve Bayes

Going deeper (assuming a binary classifier)

$$p(C \mid \mathbf{x}) = \frac{p(C)\ p(\mathbf{x} \mid C)}{p(\mathbf{x})} \tag{7}$$

$$\text{posterior probability} = \frac{\text{class prior probability} \times \text{likelihood}}{\text{predictor prior probability}}$$

$p(\mathbf{x} \mid C)$ Likelihood
The probability of the document given the class

Rough Idea

- The value of a particular feature is **independent** of the value of any other feature, given the class variable
- All features contribute the same to the classification

- Naïve Bayes' tries to find keywords in a set of documents that are predictive of the target (output) variable
- The internal coefficients will try to map tokens to scores
- Same as VADER, but without manually-created rules
  the machine will estimate them!

From (Lane et al., 2019, p. 65–68)

Naïve Bayes

A toy example: Should I ride my bike today?

One single factor: *zone (flag)*

sunny

overcast

rainy

(here come some dense slides)

Naïve Bayes

A toy example: Should I ride my bike today?

Dataset

Flag	🚲
🟨	yes
🟧	yes
🟨	no
🟥	yes
🟨	yes
🟧	yes
🟨	yes
🟧	yes
🟥	yes
🟨	no
🟥	no
🟧	yes
🟥	no
🟥	no

Computing **all** the probabilities by "counting"

Frequency table

	🚲	
Flag	**yes**	**no**
🟨	3	2
🟧	4	0
🟥	2	3

Likelihood table

	🚲	
Flag	**yes**	**no**
🟨	3/9	2/5
🟧	4/9	0/5
🟥	2/9	3/5

Adapted from `http://www.saedsayad.com/naive_bayesian.htm`

Naïve Bayes

A toy example: Should I ride my bike today?

Likelihood table

	🚲	
Flag	**yes**	**no**
🟨	3/9 [1]	2/5
🟧	4/9	0/5
🟥	2/9	3/5
	9/14 [2]	5/14

[1] $p(x \mid c) = p(🟨 \mid \text{yes}) = 3/9 = 0.33$

[2] $p(c) = p(\text{yes}) = 9/14 = 0.64$

$p(x) = p(🟨) = 5/14 = 0.36$

What is the Naïve Bayes' probability of **yes** if 🟨?

$$\begin{aligned} p(c \mid x) &= p(c)p(x \mid c)/p(x) \\ p(\text{yes} \mid 🟨) &= p(\text{yes})p(🟨 \mid \text{yes})/p(🟨) \\ p(\text{yes} \mid 🟨) &= 0.64 * 0.33/0.36 \\ p(\text{yes} \mid 🟨) &= 0.59 \end{aligned}$$

Adapted from `http://www.saedsayad.com/naive_bayesian.htm`

Naïve Bayes

A toy example: Should I ride my bike today?

If. . . 🟨 let's ride 🚲!

Naïve Bayes

A toy example: Should I ride my bike today?

Considering more data

Flag	**Temp**	**Humidity**	**Windy**	🚲
🟥	hot	high	false	no
🟥	hot	high	true	no
🟧	hot	high	false	yes
🟧	mild	high	false	yes
🟨	cool	normal	false	yes
🟨	cool	normal	true	no
🟧	cool	normal	true	yes
🟥	mild	high	false	no
🟥	cool	normal	false	yes
🟨	mild	normal	false	yes
🟥	mild	normal	true	yes
🟧	mild	high	true	yes
🟧	hot	normal	false	yes
🟨	mild	high	true	no

Adapted from `http://www.saedsayad.com/naive_bayesian.htm`

Naïve Bayes

A toy example: Should I ride my bike today?

Frequency tables

Flag	**yes**	**no**
	3	2
	4	0
	2	3

Humidity	**yes**	**no**
high	3	4
normal	6	1

Temp	**yes**	**no**
hot	2	2
mild	4	2
cool	3	1

Windy	**yes**	**no**
false	6	2
true	3	3

Likelihood tables

Flag	**yes**	**no**
	3/9	2/5
	4/9	0/5
	2/9	3/5

Humidity	**yes**	**no**
high	3/9	4/5
normal	6/9	1/5

Temp	**yes**	**no**
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Windy	**yes**	**no**
false	6/9	2/5
true	3/9	3/5

Adapted from `http://www.saedsayad.com/naive_bayesian.htm`

Naïve Bayes

Likelihood tables

Flag	**yes**	**no**
	3/9	2/5
	4/9	0/5
	2/9	3/5

Humidity	**yes**	**no**
high	3/9	4/5
normal	6/9	1/5

Temp	**yes**	**no**
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Windy	**yes**	**no**
false	6/9	2/5
true	3/9	3/5

flag	**temp**	**humidity**	**windy**	ride
	cool	high	true	?

$$p(\text{yes} \mid x) = \frac{p(\text{yes})p(\ \mid \text{yes})p(\text{cool} \mid \text{yes})p(\text{high} \mid \text{yes})p(\text{true} \mid \text{yes})}{p(\ )p(\text{cool})p(\text{high})p(\text{true})}$$

$$= \frac{9/14 \times 2/9 \times 3/9 \times 3/9 \times 3/9}{5/14 \times 4/14 \times 7/14 \times 6/14}$$

$$= 0.00529/0.02811 = 0.188 \sim 0.2 \ \textbf{no}$$

Adapted from `http://www.saedsayad.com/naive_bayesian.htm`

Naïve Bayes

Back to the definition...

$$p(C \mid \mathbf{x}) = \frac{p(C)\ p(\mathbf{x} \mid C)}{p(\mathbf{x})} \tag{8}$$

The probability $p(\mathbf{x})$ is constant for any given input

$$p(C \mid \mathbf{x}) = \frac{p(C)\ p(\mathbf{x} \mid C)}{\cancel{p(\mathbf{x})}} \tag{9}$$

$$p(c \mid \mathbf{x}) \propto p(c)p(\mathbf{x} \mid c) \tag{10}$$

Naïve Bayes

Back to the definition...

$$p(c \mid \mathbf{x}) \propto p(c)p(\mathbf{x} \mid c) \tag{11}$$

Remember that $\mathbf{x}$ is a vector

$$p(c \mid x_1 \ldots x_n) \propto p(c)p(x_1 \mid c) \times p(x_2 \mid c) \times \cdots \times p(x_n \mid c) \tag{12}$$

Eq. (12) can be rewritten as

$$p(c \mid x_1 \ldots x_n) \propto p(c) \prod_{i=1}^{n} p(x_i \mid c) \tag{13}$$

Naïve Bayes

The classification process

Back to the toy example

$$
\begin{aligned}
p(\text{yes} \mid x) &\propto p(\text{yes})p(\text{🚩} \mid \text{yes})p(\text{cool} \mid \text{yes})p(\text{high} \mid \text{yes})p(\text{true} \mid \text{yes}) \\
&\propto 9/14 \times 2/9 \times 3/9 \times 3/9 \times 3/9 \\
&\propto 0.00529\textbf{, which is not a probability}
\end{aligned}
$$

Classification: the maximum for all the classes

$$c \propto \arg\max_{c} p(c) \prod_{i=1}^{n} p(x_i \mid c) \tag{14}$$

```
compute p(yes|x)
compute p(no|x)
if p(yes|x) > p(no|x):
    yes
else:
    no
```

Training a Machine Learning Model

The dataset

We need a bunch of items (documents) with their associated **class**

kind	examples
binary	{positive, negative}
	{0, 1}
	{-1, 1}
multiclass	{positive, neutral, negative}
	{0,1,2}

In our case, we need the sentiment:

d_1	pos	d_5	neg	d_9	neu
d_2	neu	d_6	neg	d_{10}	pos
d_3	pos	d_7	neg	d_{11}	neu
d_4	pos	d_8	pos	d_{12}	neg

The dataset

Option 1 **Use a corpus created by somebody else**

Option 2 Build your own corpus[2]

(a) You have/hire experts to do it

(b) You engage non-experts through gamification

(c) You hire non-experts through explicit crowdsourcing

(d) There are many other ways to get annotated data

[2]Stay tuned: a course on this topic will start in November

Let us go and build a classifier with a corpus built by Hutto and Gilbert (2014)[3]

For this, you have to download and install the software companion of NLP in Action:

`https://github.com/totalgood/nlpia`

[3] http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

What I did on OsX and GNU Linux

I use pipenv[4]

```
$ pipenv install --skip-lock nlpia
```

On Github they explain how to install it with conda or pip if you plan to contribute to the project

</> Let us see it working

[4] `https://pipenv.readthedocs.io/en/latest/`

References

Hutto, C. and E. Gilbert
2014. VADER:A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI.

Lane, H., C. Howard, and H. Hapkem
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning Publication Co.

Maron, M.
1961. Automatic indexing: An experimental inquiry. *Journal of the ACM*, 8:404–417.