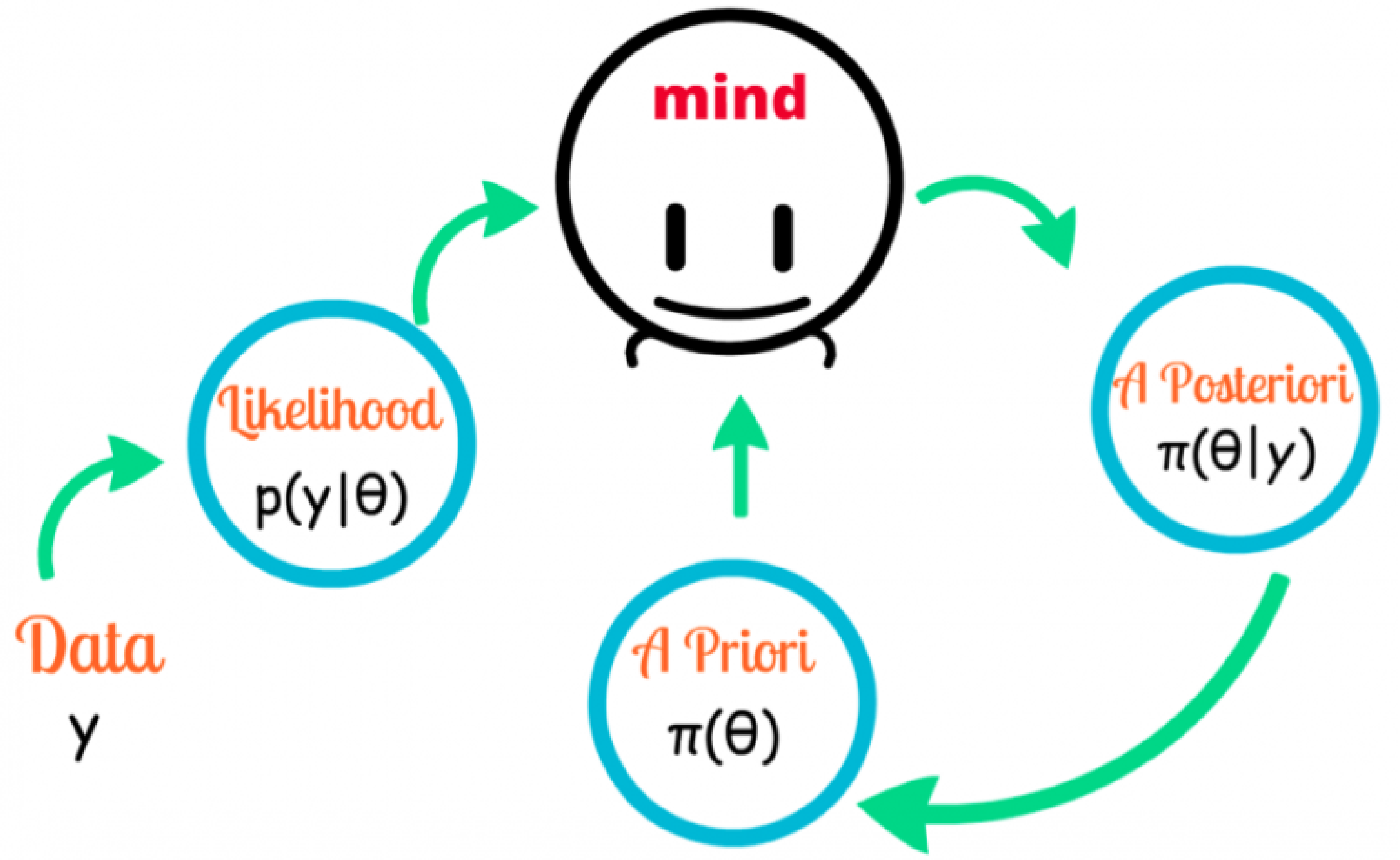
Machine Learning Blog & Software Development News

Machine Learning Tutorial: The Naive Bayes Text Classifier (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/)

October 13, 2013 Vasilis Vryniotis (http://blog.datumbox.com/author/bbriniotis/)
. 5 Comments (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/#comments)
Machine Learning & Statistics (/topics/machine-learning-statistics)



In this tutorial we will discuss about Naive Bayes text classifier. Naive Bayes is one of the simplest classifiers that one can use because of the simple mathematics that are involved and due to the fact that it is easy to code with every standard programming language including PHP, C#, JAVA etc.

Update: The Datumbox Machine Learning Framework is now open-source and free to download (https://github.com/datumbox/datumbox-framework). Check out the package com.datumbox.framework.machinelearning.classification to see the implementation of Naive Bayes Classifier in Java.

Note that some of the techniques described below are used on Datumbox's Text Analysis service (http://www.datumbox.com/features/) and they power up our API (http://www.datumbox.com/machine-learning-api/).

What is the Naive Bayes Classifier?

The Naive Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem with strong and naïve independence assumptions. It is one of the most basic text classification techniques with various applications in email spam detection, personal email sorting, document categorization, sexually explicit content detection, language detection and sentiment detection. Despite the naïve design and oversimplified assumptions that this technique uses, Naive Bayes performs well in many complex real-world problems.

Even though it is often outperformed by other techniques such as boosted trees, random forests, Max Entropy, Support Vector Machines etc, Naive Bayes classifier is very efficient since it is less computationally intensive (in both CPU and memory) and it requires a small amount of training data. Moreover, the training time with Naive Bayes is significantly smaller as opposed to alternative methods.

Naive Bayes classifier is superior in terms of CPU and memory consumption as shown by Huang, J. (2003) (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.751&rep=rep1&type=pdf), and in several cases its performance is very close to more complicated and slower techniques.

When to use the Naive Bayes Text Classifier?

You can use Naive Bayes when you have limited resources in terms of CPU and Memory. Moreover when the training time is a crucial factor, Naive Bayes comes handy since it can be trained very quickly. Indeed Naive Bayes is usually outperformed by other classifiers, but not always! Make sure you test it before you exclude it from your research. Keep in mind that the Naive Bayes classifier is used as a baseline in many researches.

Which Naive Bayes Variation to use?

There are several Naive Bayes Variations. Here we will discuss about 3 of them: the Multinomial Naive Bayes, the Binarized Multinomial Naive Bayes and the Bernoulli Naive Bayes. Note that each can deliver completely different results since they use completely different models.

Usually Multinomial Naive Bayes is used when the multiple occurrences of the words matter a lot in the classification problem. Such an example is when we try to perform Topic Classification. The Binarized Multinomial Naive Bayes is used when the frequencies of the words don't play a key role in our classification. Such an example is Sentiment Analysis, where it does not really matter how many times someone mentions the word "bad" but rather only the fact that he does. Finally the Bernoulli Naive Bayes can be used when in our problem the absence of a particular word matters. For example Bernoulli is commonly used in Spam or Adult Content Detection with very good results.

The Theoretical Background of Naive Bayes

As stated earlier, the Naive Bayes classifier assumes that the features used in the classification are independent. Despite the fact that this assumption is usually false, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of Naive Bayes classifiers as Zhang (2004) (http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf) shown. It can be proven (Manning et al, 2008 (http://nlp.stanford.edu/IR-book/html/htmledition/properties-of-naive-bayes-1.html)) that even though the probability estimates of Naive Bayes are of low quality, its classification decisions are quite good. Thus, despite the fact that Naive Bayes usually over estimates the probability of the selected class, given that we use it only to make the decision and not to accurately predict the actual probabilities, the decision making is correct and thus the model is accurate.

In a text classification problem, we will use the words (or terms/tokens) of the document in order to classify it on the appropriate class. By using the "maximum a posteriori (MAP)" decision rule, we come up with the following classifier:

$$c_{map} = \arg\max_{c \in C}\left(P(c \mid d)\right) = \arg\max_{c \in C}\left(P(c)\prod_{1 \le k \le n_d} P(t_k \mid c)\right)$$

Where t_k are the tokens (terms/words) of the document, C is the set of classes that is used in the classification, $P(c \mid d)$ the conditional probability of class c given document d, $P(c)$ the prior probability of class c and $P(t_k \mid c)$ the conditional probability of token t_k given class c.

This means that in order to find in which class we should classify a new document, we must estimate the product of the probability of each word of the document given a particular class (likelihood), multiplied by the probability of the particular class (prior). After calculating the above for all the classes of set C, we select the one with the highest probability.

Due to the fact that computers can handle numbers with specific decimal point accuracy, calculating the product of the above probabilities will lead to float point underflow. This means that we will end up with a number so small, that will not be able to fit in memory and thus it will be rounded to zero, rendering our analysis useless. To avoid this instead of maximizing the product of the probabilities we will maximize the sum of their logarithms:

$$c_{map} = \arg\max_{c \in C}\left(\log P(c) + \sum_{1 \le k \le n_d} \log P(t_k \mid c)\right)$$

Thus instead of choosing the class with the highest probability we choose the one with the highest log score. Given that the logarithm function is monotonic, the decision of MAP remains the same.

The last problem that we should address is that if a particular feature/word does not appear in a particular class, then its conditional probability is equal to 0. If we use the first decision method (product of probabilities) the product becomes 0, while if we use the second (sum of their logarithms) the log(0) is undefined. To avoid this, we will use add-one or Laplace smoothing by adding 1 to each count:

$$P(t \mid c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'}) + B'}$$

Where B' is equal to the number of the terms contained in the vocabulary V.

Naive Bayes Variations

Let's examine three common Naive Bayes variations which differ on the way that they calculate the conditional probabilities of each feature and on the scoring of each category.

Multinomial Naive Bayes model

This variation, as described by Manning et al (2008) (http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html), estimates the conditional probability of a particular word/term/token given a class as the relative frequency of term t in documents belonging to class c:

$$P(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Thus this variation takes into account the number of occurrences of term t in training documents from class c, including multiple occurrences.

Both the training and the testing algorithms are presented below in the form of pseudo code:

```
TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)
  $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
  $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
  for each $c \in \mathbb{C}$
  do $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
     $prior[c] \leftarrow N_c / N$
     $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
     for each $t \in V$
     do $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
     for each $t \in V$
     do $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
  return $V, prior, condprob$
```

```
APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)
  $W \leftarrow$ EXTRACTTOKENSFROMDOC($V, d$)
  for each $c \in \mathbb{C}$
  do $score[c] \leftarrow \log prior[c]$
     for each $t \in W$
     do $score[c] += \log condprob[t][c]$
  return $\arg\max_{c \in \mathbb{C}} score[c]$
```

Binarized (Boolean) Multinomial Naive Bayes model

This variation, as described by Dan Jurafsky (http://www.stanford.edu/class/cs124/lec/sentiment.pptx), is identical to the Multinomial Naive Bayes model with only difference that instead of measuring all the occurrences of the term t in the document, it measures it only once. The reasoning behind this is that the occurrence of the word matters more than the word frequency and thus weighting it multiple times does not improve the accuracy of the model.

Both the training and the testing algorithms remain the same with only exception that instead of using multiple occurrences we clip all word counts in each document at 1.

Bernoulli Naive Bayes model

The Bernoulli variation, as described by Manning et al (2008) (http://nlp.stanford.edu/IR-book/html/htmledition/the-bernoulli-model-1.html), generates a Boolean indicator about each term of the vocabulary equal to 1 if the term belongs to the examining document and 0 if it does not. The model of this variation is significantly different from Multinomial not only because it does not take into consideration the number of occurrences of each word, but also because it takes into account the non-occurring terms within the document. While in Multinomial model the non-occurring terms are completely ignored, in Bernoulli model they are factored when computing the conditional probabilities and thus the absence of terms is taken into account.

Both the training and the testing algorithms are presented below:

```
TRAINBERNOULLINB($\mathbb{C}, \mathbb{D}$)
  $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
  $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
  for each $c \in \mathbb{C}$
  do $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
     $prior[c] \leftarrow N_c / N$
     for each $t \in V$
     do $N_{ct} \leftarrow$ COUNTDOCSINCLASSCONTAININGTERM($\mathbb{D}, c, t$)
        $condprob[t][c] \leftarrow (N_{ct} + 1)/(N_c + 2)$
  return $V, prior, condprob$
```

```
APPLYBERNOULLINB($\mathbb{C}, V, prior, condprob, d$)
  $V_d \leftarrow$ EXTRACTTERMSFROMDOC($V, d$)
  for each $c \in \mathbb{C}$
  do $score[c] \leftarrow \log prior[c]$
     for each $t \in V$
     do if $t \in V_d$
          then $score[c] += \log condprob[t][c]$
          else $score[c] += \log(1 - condprob[t][c])$
  return $\arg\max_{c \in \mathbb{C}} score[c]$
```

Bernoulli model is known to make many mistakes while classifying long documents, primarily because it does not take into account the multiple occurrences of the words. Moreover we should note that it is particularly sensitive to the presence of noisy features.

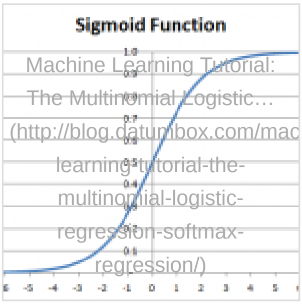
If you are confused by all the statistics and maths, don't worry! Datumbox made it simple to use Machine Learning from your software. Just sign-up (http://www.datumbox.com/users/register/) for a free account and start using our easy-to-use REST Machine Learning API (http://www.datumbox.com/machine-learning-api/) in no time.

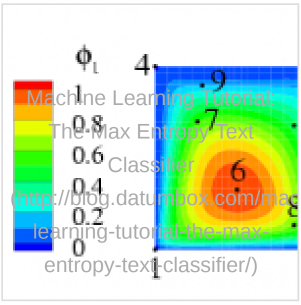
Did you like the article? Please take a minute to share it on Twitter. 🙂

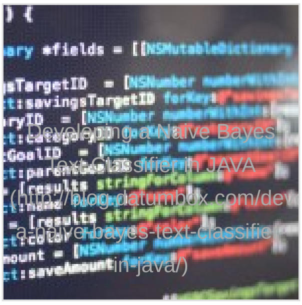
About Vasilis Vryniotis (http://blog.datumbox.com/author/bbriniotis/)

My name is Vasilis Vryniotis. I'm a Data Scientist, a Software Engineer, author of Datumbox Machine Learning Framework and a proud geek. Learn more (http://blog.datumbox.com/author/bbriniotis/)

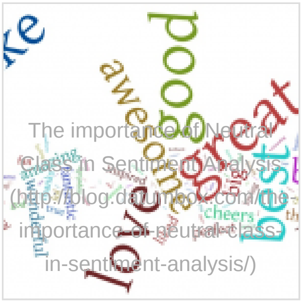
Related Posts:

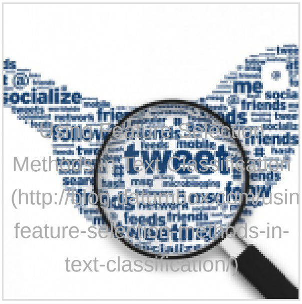

Machine Learning Tutorial: The Multinomial Logistic… (http://blog.datumbox.com/mac learning-tutorial-the-multinomial-logistic-regression-softmax-regression/)


Machine Learning Tutorial: The Max Entropy Text Classifier (http://blog.datumbox.com/mac learning-tutorial-the-max-entropy-text-classifier/)


Developing a Naive Bayes Text Classifier in JAVA (http://blog.datumbox.com/dev a-naive-bayes-text-classifie in-java/)

10 Tips for Sentiment Analysis projects (http://blog.datumbox.com/10-tips-for-sentiment-analysis-projects/)


The importance of Neutral Class in Sentiment Analysis (http://blog.datumbox.com/the-importance-of-neutral-class-in-sentiment-analysis/)


Meth (http://b feature-se ds-in-text-classification/)

Clustering documents and gaussian data with Dirichlet… (http://blog.datumbox.com/clus documents-and-gaussian-data-with-dirichlet-process-mixture-models/)

How to build your own Facebook Sentiment Analysis Tool (http://blog.datumbox.com/how to-build-your-own-facebook-sentiment-analysis-tool/)

Latest Comments

Behnaz

December 8, 2015

Hi Vasilis,
I was reading your blog today for the first time. Your insights to machine learning problems are very impressive and you are able to explain concepts very clearly.
Thank you for doing this!

Reply (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/?

dilipbobby

September 16, 2016

Hi vasilis,

you have given a good explanation but you used the Multinomial navies for the sentiment analysis in classifier code example of Datumboxframe work but in this blog post you said Binarized Multinomial Naive Bayes is used for sentiment analysis can I know why you used

Multinomial navies instead of Binarized Multinomial?

Reply (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/?

krunal

May 11, 2016

hello,

there's sentiment package in R, where it is using algorithm = naive bayes. so i want to understand how it works over there.

Reply (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/?

Shantanu

October 5, 2016

Hi
Your article gives all details in one go

Reply (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/?

Reddy

May 11, 2017

What if the test document contains words not in the vocabulary which was created during training phase?

Reply (http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/?

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Captcha *

−

4

=

1

Post Comment

Recent Posts

- The Batch Normalization layer of Keras is broken (http://blog.datumbox.com/the-batch-normalization-layer-of-keras-is-broken/)
- 5 tips for multi-GPU training with Keras (http://blog.datumbox.com/5-tips-for-multi-gpu-training-with-keras/)
- Ubuntu 17.10: a last minute review (http://blog.datumbox.com/ubuntu-17-10-a-last-minute-review/)

- Datumbox Machine Learning Framework v0.8.1 released (http://blog.datumbox.com/datumbox-machine-learning-framework-v0-8-1-released/)
- Drilling into Spark's ALS Recommendation algorithm (http://blog.datumbox.com/drilling-into-sparks-als-recommendation-algorithm/)

Archives

- April 2018 (http://blog.datumbox.com/2018/04/)
- January 2018 (http://blog.datumbox.com/2018/01/)
- October 2017 (http://blog.datumbox.com/2017/10/)
- August 2017 (http://blog.datumbox.com/2017/08/)
- February 2017 (http://blog.datumbox.com/2017/02/)
- January 2017 (http://blog.datumbox.com/2017/01/)
- March 2016 (http://blog.datumbox.com/2016/03/)
- January 2016 (http://blog.datumbox.com/2016/01/)
- May 2015 (http://blog.datumbox.com/2015/05/)
- November 2014 (http://blog.datumbox.com/2014/11/)
- October 2014 (http://blog.datumbox.com/2014/10/)
- July 2014 (http://blog.datumbox.com/2014/07/)
- June 2014 (http://blog.datumbox.com/2014/06/)
- May 2014 (http://blog.datumbox.com/2014/05/)
- April 2014 (http://blog.datumbox.com/2014/04/)
- March 2014 (http://blog.datumbox.com/2014/03/)
- February 2014 (http://blog.datumbox.com/2014/02/)
- January 2014 (http://blog.datumbox.com/2014/01/)
- December 2013 (http://blog.datumbox.com/2013/12/)
- November 2013 (http://blog.datumbox.com/2013/11/)
- October 2013 (http://blog.datumbox.com/2013/10/)
- September 2013 (http://blog.datumbox.com/2013/09/)
- July 2013 (http://blog.datumbox.com/2013/07/)

Categories

Machine Learning & Statistics (/topics/machine-learning-statistics) Programming (/topics/programming)

Framework (/topics/framework) Online Marketing (/topics/online-marketing) Misc (/topics/misc)

About

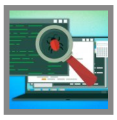
Datumbox offers an open-source Machine Learning Framework and an easy to use and powerful API.

Subscribe To Newsletter

Subscribe to our newsletter and get our latest news!

Email Address Subscribe

Recent Blog Posts


The Batch Normalization layer of Keras is broken (http://blog.datumbox.com/the-batch-normalization-layer-of-keras-is-broken/)

5 tips for multi-GPU training with Keras (http://blog.datumbox.com/5-tips-for-multi-gpu-training-with-keras/)

Ubuntu 17.10: a last minute review (http://blog.datumbox.com/ubuntu-17-10-a-last-minute-review/)

Contact Us

Email: info@datumbox.com (mailto:info@datumbox.com)

Social Media