PANACEA Project
Platform for **A**utomatic, **N**ormalized **A**nnotation and
Cost-**E**ffective **A**cquisition
Grant Agreement no. 248064

Documentation of Bayesian parameter estimation Web Service[1]

Author: Muntsa Padró. Barcelona, 2012

Contact: muntsa.padro@upf.edu

1 Overview

Given a training set encoded as vectors of cue (or feature) occurrences, this web service estimates the parameters *P(cue$_i$|class)*: the probability of seeing each cue as a member or non-member of the class. This estimation is performed using Bayesian inference, which combines prior knowledge with observed data. The parameters estimated with this web service can be used, for example, to classify new instances using a Naive Bayes classifier.

In this document, first of all we present the inputs needed by the web service and their format and the ouput it produces. The theoretical and methodological details can be found in section 2.

2 Inputs, outputs and formats

Inputs:

- ***Training_data*** (training data): classified instances encoded as cue vectors. Each slot of the vector contains the number of times each feature has been observed for that instance. Also, we add special slots: total number of occurrences in the first slot and correct class (1 or 0) and lemma (or any identifier of the instance) in the two last slots.

  Format: The vectors should be encoded in a weka file. For example, for the class of eventive nouns in English we would have some examples of eventive nouns and some of non-eventive nouns:

  ```
  @relation eventive.arff'
  @attribute "[total_occurences]" numeric
  @attribute "[cue_1]" numeric
  @attribute "[cue_2]" numeric
  …
  @attribute "[cue_n]" numeric
  @attribute "[eventive]" {0,1}
  @attribute "[lemma]" string

  @data
  5,0,0,…,3,0,visa
  386,0,1,…,162,0,characteristic
  23,1,0,… ,0,1,ceremony
  270,0,2,…,0,1,assembly
  ```

  Important: the cue counts must be encoded as integers, this is, no relative frequency needs to be given but the number of times each cue has been seen.

[1] This documented is licensed under a Creative Commons Attribution 3.0 Spain License. To view a copy of this license, visit http://creativecommons.org/licenses/by/3.0/es/.

- ***Virtual_frequency***: This file encodes our prior expectations about which cues are more likely for each class (see sections 2.2 and 2.3). We need to encode the *virtual relative frequency* for each cue of belonging not belonging to the class. This is, with what frequency we expect a priori to see each cue for the instances belonging or not belonging to the class.

  Format: Data separated by ";", each line contains the *virtual relative frequency* of each cue (identified by its label) of seeing and not seeing the cue if we are or are not in the class. First line explains the fields: cue name, probability of seeing the cue with members of the class, probability of not seeing the cue with members of the class (note that this two must sum one), probability of seeing the cue with non-members of the class, probability of not seeing the cue with non-members of the class. For example:

  ```
  #cue;p(cue|class=yes);p(!cue|class=yes);p(cue|class=no);p(!cue|class=no)
  cue_1;0.03;0.97;0.0;1.0
  ...
  cue_n;0.05;0.95;0.09;0.91
  ```

- ***virtual_size_class*** and ***virtual_size_no_class***. Integers representing the weights given to the priors. By default, they are set equal to the data size. If these integers are bigger than data size, the priors will have more influence on the estimated parameters than the data. If they are smaller, the data will be more important than the priors. If these two parameters are equal to 0 the estimation of the parameters will be equivalent to Maximum Likelihood Estimation, this is, only data will be used to estimate the probabilities. See section 2.3 for more details.

Outputs:

Using input information the web service will estimate the parameters using a Bayesian approach and return a comma separated file with them. The output file can be directly used to classify new instances with the ***Naive Bayes classifier Web Service***.

Format: data separated by ";", each line containing the information of each cue. First field labels the cue. The first line explains the contents of each column. There are some fields that report the parameters given as input in order to ease posterior study or debugging of the system. The fields of the output file are:

- *cue*: label of the cue
- *data size class*: number of tokens in the training data that belong to the class
- *virtual size class*: virtual size given to the class (input parameter)
- *data p class*: probability of seeing this cue for members of the class estimated only from data (MLE)
- *prior p class*: prior *virtual relative frequency* given in the input file for members of the class
- *theta class*: probability of seeing this cue for members of the class estimated using a Bayesian approach, i.e. combining observed data and virtual frequencies.
- *data size no class, virtual size no class, data p no class, prior p no class, theta no class*: same fields presented above but for non-class members.
- *log(theta[class]/theta[noclass]), log((1-theta[class])/(1-theta[noclass]))*: pre-computed log-quotients to be used in the Naïve Bayes classifier (see equation 2).

Output Example:

```
#cue;data size class;virtual size class;data p class;prior p class;theta class;data size no class;virtual size no class;data p no class;prior p no class;theta no class;log(theta[class]/theta[noclass]);log((1-theta[class])/(1-theta[noclass]))

cue_1;55262;522008;1.80956172415e-05;1e-05;1.07749926378e-05;36745;15474000;2.72145870186e-05;0.0;6.44714357692e-08;5.11876096228;-1.07105792506e-05
…

cue_n;55262;522008;0.00152003184829;0.01;0.00918821348762;36745;15474000;0.00144237311199;0.0001;0.000103180085805;4.4892009316;-0.00912750007329
```

3 Theoretical background

3.1 Cue-based classification of nouns

As an example of use of this web service, we will present the cue-based classification of nouns, but this can be extended to any task that performs classification based on studying the number of times each feature has been observed.

Traditional Naive Bayes approach to cue-based noun classification computes the probability of a given noun belonging to a particular class k given a vector w that represents the number of times the word to classify has been seen with each cue (n_i). This probability is calculated by Bayesian inversion as:

Where $P(k)$ is the prior probability of the class k, and $P(w|k)$ is the likelihood of vector w given a class k. This likelihood is estimated (assuming independence of cues) as the product over all components n_i of the vector.

In our approach, we built the classifier as a comparison of two hypotheses: the word belongs to the class ($k=1$), or it does not ($k=0$). The hypothesis with higher probability (assessed using the log quotient in 2) is considered the correct one.

Due to the Bernoulli distribution of cue occurrences, we can calculate the probability of a particular vector component given the class we need for 2 using equation 3:

Where N is the total number of occurrences of the word and $P(cue_i|k)$ refers to the probability of the cue given the class for a single occurrence.

Thus, in order to classify a noun using a Naive Bayes classifier (equation 2), we estimated $P(cue_i|k)$ for each cue and class and the prior probabilities of the classes $P(k=0)$ and $P(k=1)$. Usually, these prior probabilities are considered equal to 0.5.

In the next section we focus on how we calculated $P(cue_i|k)$, for each cue_i and k value using a Bayesian approach.

3.2 Bayesian Estimation of Probabilities

Bayesian methods (Griffiths et al., 2008; Mackay, 2003) are a formal framework to introduce prior knowledge when estimating the parameters (probabilities) of a given system. The parameters estimating with Bayesian models are the same that we could estimate with Maximum Likelihood Estimation (MLE). The latter use only data to estimate parameters, while the former use both data and prior knowledge to estimate them.

A canonical example of Bayesian learning is determining the probability of a coin producing heads in a short throw series. A pure frequentist approach, such as MLE, will determine this probability as . Thus, after observing a sequence of 5 heads in a row, MLE would assess that the probability of the coin producing heads is 1. Nevertheless, because of our knowledge, we would rather say that a tail is more than possible, and that the coin probability can still be close to 0.5. Bayesian models allow us to formally introduce this knowledge when estimating the probabilities.

As we have just seen, it is necessary to estimate $P(cue_i|k)$, i.e. the probability that in one occurrence each cue is observed or not given each class. This assessment is usually done using MLE, which infers the probability from the data, following a frequentist approach. Alternatively, we can approach it as an update of our *a priori* expectations with data evidence by using a Bayesian estimation of parameters as follows.

For the sake of the explanation, from now on we will refer to $P(cue_i|k)$ as θ, but note that all assessments were done for each cue_i and for $k=0$ and $k=1$. In order to infer each parameter $\theta=P(cue_i|k)$, we followed a canonical Bayesian modelling approach: we chose as θ its Maximum a Posteriori (MAP) estimator. This is, we looked for the θ value that maximized its probability given the observed data, $P(\theta|data)$:

To calculate $P(\theta|data)$we applied Bayes theorem:

Where $P(\theta|data)$ is the posterior probability to be maximized, the likelihood $P(data|\theta)$ is the probability of these data given a value of θ, and $P(\theta)$ is the prior probability given to θ.

In order to maximize $P(\theta|data)$ in 5 we assessed $P(data|\theta)$ and $P(\theta)$. The first term is computed in terms of the observed data that we will refer to as N_{yes} and N_{no}. N_{yes} represents the number of times the studied cue has been actually observed with words of the class and N_{no} is the number of times this cue has not been seen for words in the same class.

The prior probability $P(\theta)$ is the key point of Bayesian inference. It is where we can introduce our *a priori* expectations about observing a particular cue given a class. For example, we know that it is likely that *during* is observed for the members of the class EVENT and conversely that it is not observed for the non-members of the class.

In order to formally introduce these expectations (that is to encode into the model our abstract knowledge), we used what can be called virtual data: V_{yes} and V_{no}. V_{yes} will be the number of occurrences we expect (*a priori*) for a particular cue and class, and V_{no} will be the number of times we expect the cue not to be observed for words in the same class. Canonically, $P(\theta)$ is formalized in terms of these virtual data assuming it follows a Beta distribution (conjugate prior of Bernoulli distribution) with parameters $V_{yes}+1$ and $V_{no}+1$.

Therefore, we assess $P(data|\theta)$ in 5 in terms of observed occurrences in real data (N_{yes} and N_{no}) and $P(\theta)$ in terms of virtual data (V_{yes} and V_{no}) and we maximize the posterior probability $P(\theta|data)$, to get the following MAP estimator (Griffiths et al, 2008; Mackay 2003):

From 6, we can see that we add some virtual data to actual evidence to compensate their sparseness there. Virtual data becomes the representation of linguistically motivated expectations about the significance of a particular cue of which there will not be enough evidence in the data set to be considered useful by the learner. In the cases where the evidence, N_{yes} and N_{no}, is small or zero (this is our case), adding virtual data results in a smoothed value which preserves its significance.

Recall that we wanted to estimate $P(cue_i|k)$, for each cue and for $k=0$ and $k=1$. Therefore, $P(cue_i|k)$ is assessed in terms of the number of times cue_i had been seen in the training corpus with nouns of class k, the number of times that this cue had not been seen with the same nouns, and the virtual times we added to the positive evidence and to the negative evidence.

An important point of our approach was how to set the size of virtual data. If the amount of virtual data was much larger than that of the actual data, the actual data would not have much influence on the parameters. But, if it was much smaller, the prior would have little influence on the parameter estimation. Note that in the extreme case of $V_{yes} = V_{no}=0$ our MAP estimator becomes equivalent to the Maximum Likelihood estimator, that is, a standard frequentist approach.

3.3 Methodology

As we have seen in the previous section, the key point of the Bayesian parameter estimation is to define our prior knowledge in terms of an amount of virtual data. Note that we needed to define two values: V_{yes} and V_{no}, for each cue and for the two possible values of k. In order to set a specific figure for these values, we started formulating it in terms of an expected occurrence ratio, which we called *virtual relative frequency*: , i.e. a ratio of times we expected *a priori* to see a cue given a class.

Thus, the first step of the learning process is to hypothesize virtual relative frequencies for each cue and for both k values according to our linguistic knowledge. Then, these relative frequencies will be converted into integers in order to obtain the amount of virtual data V_{yes} and V_{no}, to be used in equation 7 to compute $P(cue_i|k)$. Since , we defined an integer that represents the total amount of virtual data that will be used to convert the relative frequencies into V_{yes} and V_{no} for each cue and class:

In order to ease the In our approach, we set two values for V: one for the priors positive for the class ($V(k=1)$) and one for the negative ones ($V(k=0)$). Note that $V(k=1)$ and $V(k=0)$ can be seen as the total amount of virtual data for each class. They are parallel to the number of observed tokens inside or outside the class ($N(k)=N_{yes}(k)+N_{no}(k)$), so it makes sense to consider a unique value for each class: $V(k=1)$ represents the total amount of virtual data we add to the class, and is parallel to the number of observed tokens in the class and $V(k=0)$ represents the total amount of virtual data we add to the non-members of the class.

Thus, the inputs of the web service are:

- The training data, to extract and
- The *virtual relative frequencies* (called priors in the webservice) for each cue and for $k=0$ and $k=1$
- $V(k=1)$ (*virtual_size_class*) and $V(k=0)$ (*virtual_size_no_class*). Note that if $V(k=0) = V(k=1) = 0$ the estimation of the parameters will be equivalent to Maximum Likelihood Estimation.

Using these information the web service will estimate the parameters and return a comma separated file with them. This file can be directly used to classify new instances with the web service ***naive_bayes_classifier***.

References

T. L. Griffiths, C. Kemp, and J.B. Tenenbaum. 2008. Bayesian models of cognition. In Ron Sun (ed.), *Cambridge Handbook of Computational Cognitive Modeling*.Cambridge University Press.

D. J. C. MacKay. 2003. Information Theory, Inference, and Learning Algorithms. *Cambridge University Press,* 2003. ISBN 0-521-64298-1

Related Web Service:

Bayesian parameter estimation Web Service: http://lod.iula.upf.edu/resources/228