# Less naive Bayes spam detection

Hongming Yang
Eindhoven University of Technology
Dept. EE, Rm PT 3.27,
P.O.Box 513, 5600MB Eindhoven
The Netherlands.
E-mail:h.m.yang@tue.nl
also
CoSiNe Connectivity Systems and Networks
Philips Research, Eindhoven
Email: hongming.yang@philips.com

Maurice Stassen
NXP Semiconductors
HTC 31, WDC 3.39
5656 AE Eindhoven
The Netherlands
Email: maurice.stassen@nxp.com

Tjalling Tjalkens
Eindhoven University of Technology
Dept. EE, Rm PT 3.16
P.O.Box 513, 5600MB Eindhoven
The Netherlands
Email: t.j.tjalkens@tue.nl

***Abstract*— We consider a binary classification problem with a feature vector of high dimensionality. Spam mail filters are a popular example hereof. A naive Bayes filter assumes conditional independence of the feature vector components. We use the context tree weighting method as an application of the minimum description length principle to allow for dependencies between the feature vector components. It turns out that, due to the limited amount of training data, we must assume conditional independence between groups of vector components. We consider several ad-hoc algorithms to find good groupings and good conditional models.**

## I. Introduction and problem statement

We consider a supervised binary classification problem in which the class is conditioned on a binary feature vector of high dimensionality. Also the amount of training data is relatively small. Thus we are limited in the complexity of the models we may consider. Traditionally one can tackle this problem with the use of a naive Bayes filter. The simple model underlying this method is the assumption that the features are independent conditioned on the class. This results in a model with a small number of parameters that can easily be estimated from a small amount of training data.

However, this assumption is too restrictive to be realistic. We shall consider models that allow some dependencies between features. We attempt to find a good trade-off between model complexity and the amount of training data.

A good example of this problem and the motivation for this work is the classification of e-mail messages into spam or non-spam (ham). Naive Bayes filters have been very successful in separating spam and ham. However, the simple model allows spammers to fool the filter easily. We expect that more complex models are less easy to fool.

An e-mail is characterized by its class, denoted by $c$, and the vector of features, $\boldsymbol{f} = f_1, f_2, \ldots, f_k$. The class is binary, where $c = 0$ denotes ham and $c = 1$ is spam. A feature $f = 1$ if it is present and $f = 0$ if it is absent.

We consider the *likelihood ratio* of a feature vector. It is given as

$$\xi(\boldsymbol{f}) = \frac{\Pr\{c = 0|\boldsymbol{f}\}}{\Pr\{c = 1|\boldsymbol{f}\}}. \tag{1}$$

An e-mail with a feature vector $\boldsymbol{f}$ is classified as spam if $\xi(\boldsymbol{f}) < 1$.

In the following section we briefly describe the naive Bayes filter and thus introduce the problem setting. In section III we discuss models with dependent features and describe a mechanism to select a good model. This approach suffers from a too high complexity and thus cannot use all relevant features. We consider the selection of the most relevant features in section IV. But we still cannot make use of all necessary features. In section V we discuss models that use several trees. Section VI gives some ad-hoc approaches to find good trees and gives some experimental results.

## II. The naive Bayes filter

We have a set of $n$ e-mail feature vectors and classes. We denote the feature vector of the $i^{\text{th}}$ e-mail by $\boldsymbol{f}^i$ and the corresponding class by $c^i$. So $f^i_j$ is the $j^{\text{th}}$ feature from the $i^{\text{th}}$ training message.

We wish to determine the likelihood of a feature vector $\boldsymbol{f}$ using the assumption that the features are conditionally independent. We use Bayes formula to write:

$$\Pr\{c|\boldsymbol{f}\} = \frac{\Pr\{c\}\Pr\{\boldsymbol{f}|c\}}{\Pr\{\boldsymbol{f}\}} \tag{2}$$

$$\Pr\{\boldsymbol{f}|c\} = \prod_{i=1}^{k} \Pr\{f_i|c\} \tag{3}$$

$$\xi(\boldsymbol{f}) = \frac{\Pr\{c = 0\}}{\Pr\{c = 1\}} \prod_{i=1}^{k} \frac{\Pr\{f_i|c = 0\}}{\Pr\{f_i|c = 1\}} \tag{4}$$

It is in (3) where we use the independence assumption.

We must make estimates of $\Pr\{c\}$ and $\Pr\{f_i|c\}$ for each $i$. For each of these parameters we have $n$ instances in the training set.

## III. A tree model and tree model selection

First consider the full dependence of class and features. We must estimate $\Pr\{c|\boldsymbol{f}\}$. This implies however that we have to estimate $2^k$ parameters. Moreover, every e-mail contributes to precisely one parameter. So, with large $k$ we quickly run out

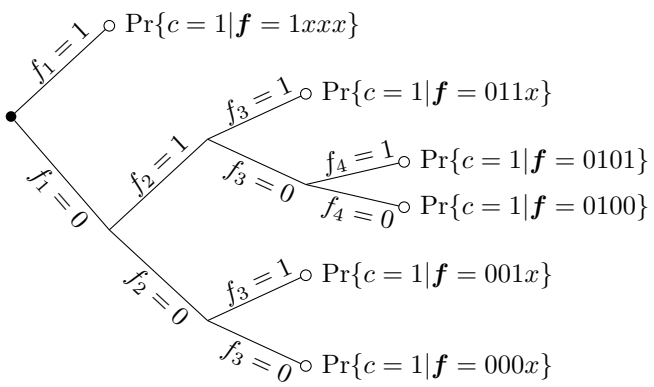


Fig. 1. A tree of relevant features

of training data. This well-known problem is called "the curse of dimensionality", see e.g. [1].

A feature is often selected to be an indication in favor of spam or ham. So, a properly chosen subsequence of features might suffice for the purpose of classification. Formally, we can define a prefix-free and complete set of varying length sequences of features that describe the relevant subsequences of features. Consider for example a problem with four features, $f_1$, $f_2$, $f_3$, and $f_4$. In principle we have sixteen parameters, but let the features and the e-mail generating process be such that if $f_1 = 1$ the other features do not contribute to the classification, so $\Pr\{c|\boldsymbol{f} = 1xxx\} = \Pr\{c|f_1 = 1\}$. See Fig. 1 for a graphical representation of this dependency where we have a total of six parameters for a tree set $S = \{1, 011, 0101, 0100, 001, 000\}$.

Given a tree $S$ we define $\pi_S(\boldsymbol{f})$ as the prefix of $\boldsymbol{f}$ that is in $S$. Note that because $S$ is prefix-free and complete such a prefix always exist and is unique.

$$\pi_S(\boldsymbol{f}) \triangleq x \text{ where } x \in S \text{ and } x \text{ is a prefix of } \boldsymbol{f}. \quad (5)$$

So we can write the a-posteriori class probability as

$$\Pr\{c|\boldsymbol{f}\} = \Pr\{c|\pi_S(\boldsymbol{f})\}. \quad (6)$$

The number of possible tree models grows doubly exponential with the length of the feature vector. We must find the best possible tree model in this set. Following the MDL principle, we can use the CTW method to find a good tree structure.

With the CTW method we compute a probability of the sequence of classes in the training set conditioned on the corresponding feature vectors, $\Pr\{c^1, c^2, \ldots, c^n|\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^n\}$. The CTW algorithm defines a prior, $P(S)$, over all possible tree sources of height $k$ or less. The height of a tree is the length of its longest path from root to leaf. The prior is given as

$$P(S) \triangleq 2^{-\Gamma_k(S)} \quad (7)$$

where $\Gamma_k(S)$, see [2], equals

$$\Gamma_k(S) = 2|S| - 1 - |\{s \in S : |s| = k\}|. \quad (8)$$

This results in the class sequence probability

$$\Pr\{c^1, c^2, \ldots, c^n|\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^n\} = \sum_{S \in \mathcal{S}_k} P(S) \prod_{i=1}^{n} \Pr\{c^i|\pi_S(\boldsymbol{f}^i)\}. \quad (9)$$

Here $\mathcal{S}_k$ is the set of all permissible tree models. For details on the CTW algorithm we refer to [3].

From (9) we see that CTW realizes an MDL solution, see e.g. [4],

$$\Pr\{c^1, c^2, \ldots, c^n|\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^n\} \geq \max_{S \in \mathcal{S}_k} P(S) \prod_{i=1}^{n} \Pr\{c^i|\pi_S(\boldsymbol{f}^i)\}. \quad (10)$$

We can use this probability and method for classification in two different ways.

C1 We find the MAP model $S^*$. This is described in [2] and also in [5]. This model defines the likelihood ratio of the e-mail with feature vector $\boldsymbol{f}$

$$\xi(\boldsymbol{f}) = \frac{\Pr\{c = 0|\pi_{S^*}(\boldsymbol{f})\}}{\Pr\{c = 1|\pi_{S^*}(\boldsymbol{f})\}}.$$

The conditional class probability is calculated with the aid of (9) as

$$\Pr\{c|\pi_{S^*}(\boldsymbol{f})\} = \frac{\Pr\{c^1, \ldots, c^n, c|\pi_{S^*}(\boldsymbol{f}^1), \ldots, \pi_{S^*}(\boldsymbol{f}^n), \pi_{S^*}(\boldsymbol{f})\}}{\Pr\{c^1, \ldots, c^n|\pi_{S^*}(\boldsymbol{f}^1), \ldots, \pi_{S^*}(\boldsymbol{f}^n)\}}. \quad (11)$$

Again, see [2], this can be computed more efficiently than (11) implies.

C2 We can also continue the weighting procedure one step more and calculate the likelihood ratio as

$$\xi(\boldsymbol{f}) = \frac{\sum_{S \in \mathcal{S}_k} P(S) \prod_{i=1}^{n} \Pr\{c^i|\pi_S(\boldsymbol{f}^i)\} \cdot \Pr\{c = 0|\pi_S(\boldsymbol{f})\}}{\sum_{S \in \mathcal{S}_k} P(S) \prod_{i=1}^{n} \Pr\{c^i|\pi_S(\boldsymbol{f}^i)\} \cdot \Pr\{c = 1|\pi_S(\boldsymbol{f})\}}. \quad (12)$$

Still this approach suffers from another form of the "curse of dimensionality" and too many relevant features are ignored by this method. This can be explained with the MDL principle. We briefly repeat the results as presented in [3]. As in that paper we shall consider $-\log_2 P$, or the code word-length, in stead of the probability $P$.

- $\boldsymbol{-\log_2 P(S)}$ This is the model cost, see (7). A more complex model, with more parameters, results in a longer description in terms of bits, or equivalently in a smaller probability.
- $\boldsymbol{\sum_{i=1}^{n} -\log_2 \Pr\{c^i|\pi_S(f^i)\}}$ This splits into two parts namely a parameter cost $-\log_2 P(\theta|S)$ which equals $-\log_2 \sqrt{n}$ per parameter and the cost of describing the sequence given the model and the parameters (this is an entropy term).

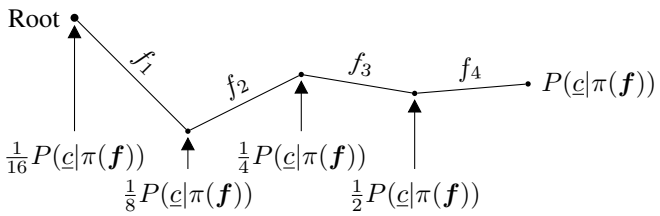


Fig. 2. Averaging in the CTW tree

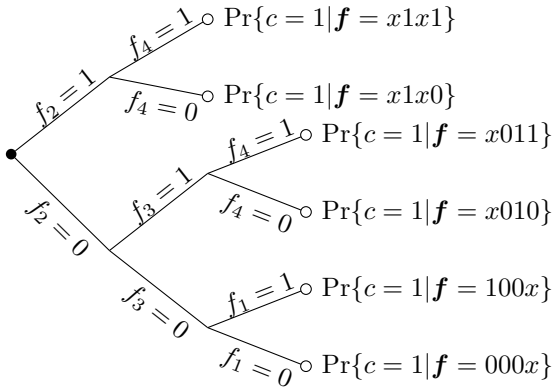


Fig. 3. A tree with arbitrary ordered features

A more complex model usually decreases the entropy term, which is linear in $n$, but increases the parameter and model cost. For small $n$ the increase in parameter and model cost is not overcome by the decrease of the entropy term so, initially, models will be small. Thus for small $n$ only a few features can play a role in the classification.

Another way of understanding this lies in the actual procedure of the CTW algorithm. In this algorithm the probability of a sequence, $\underline{c} = c^i, c^j, \ldots, c^k$, where each e-mail $i, j, \ldots, k$ has the same feature vector part $\pi(\boldsymbol{f})$, is calculated along the path from the node given by $\pi(\boldsymbol{f})$ to the root by averaging. The contribution of $P(\underline{c}|\pi(\boldsymbol{f}))$ is thus reduced by several factors $\frac{1}{2}$, see Fig. 2. Therefor long feature vectors in the tree model will only contribute significantly when the difference in probabilities is large enough and this only happens after many symbols (or e-mails).

## IV. Tree model with flexible feature ordering

We can rearrange the feature vector and move the relevant features to the front. A feature is most relevant if it gives the largest root probability in the CTW algorithm. In Section VI we give more detail on how we determine the relevant ordering of features. Another approach is to use the "class III" CTW method as described in [6]. In this method we consider tree sources where each "split" can be labeled by another feature, see Fig. 3.

Even those approaches do not result in good e-mail classifications. We must conclude that attempting to model the full feature dependencies is not feasible.

## V. Less naive Bayes filter

The success of the naive Bayes approach lies in the fact that the few parameters can be estimated quite well with the little training data that is available. Let us for now assume that the features can be split into sets where the sets are conditionally independent but dependencies exist within a set. So we assume the existence of an integer $m$ that describes the number of sets and a sequence of integers $a_0 = 1 < a_1 < a_2 < \ldots < a_m = k$. The conditional dependence is now given as

$$\Pr\{\boldsymbol{f}|c\} = \prod_{i=1}^{m} \Pr\{\boldsymbol{f}_{[a_{i-1},a_i-1]}|c\}. \tag{13}$$

Here we introduced the notation for a sub-vector $\boldsymbol{f}_{[i,j]} = f_i, f_{i+1}, \ldots, f_j$.

Consider the likelihood ratio taking (13) into account.

$$\begin{aligned}
\xi(\boldsymbol{f}) &= \frac{\Pr\{c=0|\boldsymbol{f}\}}{\Pr\{c=1|\boldsymbol{f}\}} \\
&= \frac{\Pr\{c=0\}}{\Pr\{c=1\}} \prod_{i=1}^{m} \frac{\Pr\{\boldsymbol{f}_{[a_{i-1},a_i-1]}|c=0\}}{\Pr\{\boldsymbol{f}_{[a_{i-1},a_i-1]}|c=1\}} \\
&= \left\{\frac{\Pr\{c=0\}}{\Pr\{c=1\}}\right\}^{1-m} \prod_{i=1}^{m} \frac{\Pr\{c=0|\boldsymbol{f}_{[a_{i-1},a_i-1]}\}}{\Pr\{c=1|\boldsymbol{f}_{[a_{i-1},a_i-1]}\}} \\
&= \left\{\frac{\Pr\{c=0\}}{\Pr\{c=1\}}\right\}^{1-m} \prod_{i=1}^{m} \xi(\boldsymbol{f}_{[a_{i-1},a_i-1]})
\end{aligned} \tag{14}$$

Every tree that corresponds to a sub-vector $\boldsymbol{f}_{[a_{i-1},a_i-1]}$ can train on the complete training set of $n$ e-mails. By keeping the sub-vectors small, in terms of dimension, the corresponding model parameters can be estimated quite well. We can use any approach that we discussed before for one tree. We prefer the use of CTW or maybe the "class III" CTW.

The remaining problem is to find a good subdivision of the feature vector.

## VI. Experimental results on good tree models

We obtained a set of email classes and features from Sam Roweis course on Machine Learning [7]. This set contains 5000 e-mails, each one classified as spam or ham and each one with a feature vector of $k = 185$ features. 1000 are chosen as training data and 4000 are used to test the classifier.

In the naive Bayes filter we need an estimate of $\Pr\{c\}$ and $\Pr\{f|c\}$. As in [8] we use the Bernoulli distribution with the beta prior and hyper-parameters $\alpha = \beta = 1/2$. Let

$$\begin{aligned}
n &= \text{ \#(e-mails) in the training set;} \\
N(c=1) &\triangleq \text{ \#(spam e-mails) in the training set;} \\
N(f_i=1, c=0) &\triangleq \text{ \#(ham e-mails) with } f_i\text{;} \\
N(f_i=1, c=1) &\triangleq \text{ \#(spam e-mails) with } f_i\text{.}
\end{aligned}$$

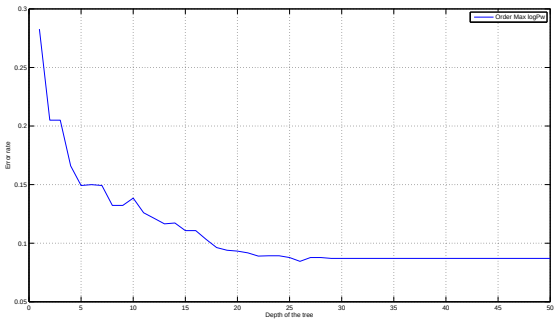


Fig. 4. One tree with feature ordering

Then we find the following parameter estimates

$$\Pr\{c=1\} \text{ estimate} = \frac{N(c=1)+\frac{1}{2}}{n+1},$$

$$\Pr\{f_i=1|c=0\} \text{ estimate} = \frac{N(f_i=1,c=0)+\frac{1}{2}}{n-N(c=1)+1},$$

$$\Pr\{f_i=1|c=1\} \text{ estimate} = \frac{N(f_i=1,c=1)+\frac{1}{2}}{N(c=1)+1}.$$

We applied this in the naive Bayes filter, trained on 1000 e-mails and tested on 4000 other e-mails, and determined the number of misclassifications. The result is listed in Table I. This serves as the benchmark for the other methods.

For the CTW method with one tree we had to determine a good order of the features. First we ran the algorithm on the training data with a feature vector of dimension 1 and tried all features. We selected the feature that gave the largest probability as given in (9). Then we appended successively every other feature and again selected the feature that maximized (9). This is repeated until all features are used. After training this algorithm gave a result, see Table I, that is far worse than the naive Bayes filter. The reason is that several relevant features were left out because of the small size of the tree models used, as we explained before in Section III. In Fig. 4 we show the classification error rate as a function of the number of features used. We observe an error floor starting at about 25 features. We list the result in Table I under "CTW (one tree)".

Then we build a model with several trees. Our approach was to start with all features and build a tree with the best features as described above, except that we stop adding another feature when the resulting increase in the root probability drops below a fixed ratio threshold. We selected a threshold ratio of 2, in line with the MDL. A factor of 2 implies a one bit shortening of the resulting description length. With the remaining features we build a second tree and so on until all features are used. This is a greedy approach. In Fig. 5 we show the error rate as a function of the number of trees used and also the number of features per tree. We list the final result in Table I under "Greedy multi-tree CTW". The performance is significantly better than with the naive Bayes approach.

Finally we attempted building trees with flexible feature assignments, such as shown in Fig. 3. The "Class III" CTW

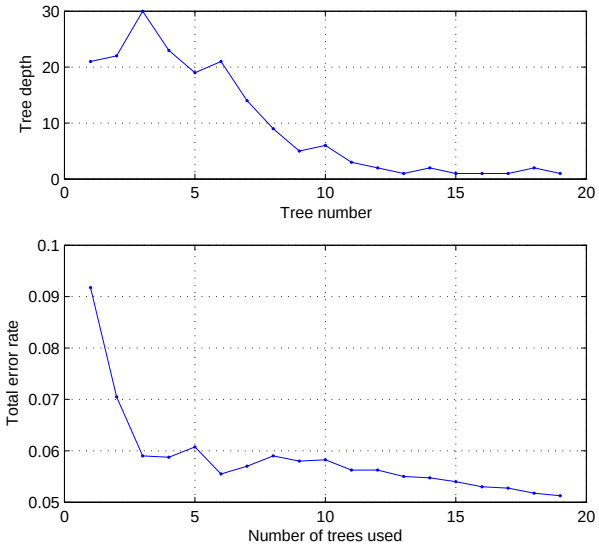


Fig. 5. Greedy multi-tree CTW

TABLE I
NUMBER OF ERRORS ON THE ROWEIS DATA

| Algorithm | nr. misclassifications |
|---|---|
| Naive Bayes filter | 235 |
| CTW (one tree) | 348 |
| Greedy multi-tree CTW | 205 |
| Multi-tree with flexible feature order | 212 |

from [6] is too complex to implement for a feature vector of length 185. So we used another greedy approach.

We build the tree, first finding the single best feature but then extending each leaf of the tree independently with another feature that maximizes that node's probability. We stop with a leaf when the probability does not increase anymore. The result of this approach is listed under "Multi-tree with flexible feature order" in Table I. Although the model is more flexible and the performance is better than that of the naive approach, it is not the best.

## VII. CONCLUSION

The small experiment we conducted showed that e-mail feature vectors are not conditionally independent. Modeling this dependency with a set of context trees is possible and can give a good trade-off between classification power and the amount of training data needed. We employed a compression based approach to find sets of (almost) independent feature vectors and used as decision criterion the (non-)decrease of the conditional entropy. Using a "class III" method could reduce the number of parameters needed in a tree. However, our experiment does not show the expected gain. The experiment is too small to draw a clear conclusion. Either rearranging the features in a flexible way is not useful for e-mail feature vectors or the greedy approach does not result in a good

arrangement. Of course it is important to put the features in the correct sub-set. How to do this in an optimal way is still an open question.

## REFERENCES


[1] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[2] F. M. J. Willems, T. J. Tjalkens, and T. Ignatenko, "Context-tree weighting and maximizing: processing betas," in *Proceedings of the Inaugural workshop of the ITA*. UCSD, La Jolla, USA, feb 2006, pp. 1–5.

[3] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The contex-tree weighting method: Basic properties," *IEEE Trans. Inform. Theory*, vol. 41, pp. 653–664, May 1995.

[4] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2743–2760, Oct 1998.

[5] T. J. Tjalkens and F. M. J. Willems, "Map model selection for context trees," in *Proceedings of the 2006 IEEE International Workshop on Machine learning for Signal processing*. MLSP (CDROM), september 6-8 2006, pp. 1–6.

[6] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "Context weighting for general finite context sources," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1514–1520, Sep 1996.

[7] S. Roweis. [Online]. Available: http://www.cs.toronto.edu/~roweis/csc2515/

[8] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 199–207, Mar 1981.