Prediction and modeling with partial dependencies

Citation for published version (APA):

Tjalkens, T. J. (2008). Prediction and modeling with partial dependencies. In *Information Theory and Applications Workshop, 2008, 3rd, 27 January -1 February 2008, San Diego, U.S.A.* (pp. 1-11). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ITA.2008.4601082>

DOI:

[10.1109/ITA.2008.4601082](https://doi.org/10.1109/ITA.2008.4601082)

Document status and date:

Published: 01/01/2008

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Prediction and modeling with partial dependencies

Tjalling Tjalkens

Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract

I consider a binary classification problem with a feature vector of high dimensionality. Spam mail filters are a popular example hereof. A Bayesian approach requires us to estimate the probability of a feature vector given the class of the object. Due to the size of the feature vector this is an unfeasible task. A useful approach is to split the feature space into several (conditionally) independent subspaces. This results in a new problem, namely how to find the "best" subdivision. In this paper I consider a weighing approach that will perform (asymptotically) as good as the best subdivision and still has a manageable complexity.

1 Problem statement

An object $\mathcal{O}$ belongs to a particular class c and is described by a vector of features, f^k . Given a sequence of objects $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n$, we wish to estimate the conditional features probabilities given the classes.

An extreme, and extremely simple, model for these probabilities is known as the *Naive Bayes filter*, see e.g. [1], where all features are assumed to be conditionally independent or

$$P(f^k|c) = \prod_{i=1}^k P(f_i|c). \quad (1)$$

This simple model is applied, with much success, see [2], to spam filtering although the model is obviously too simple to be correct.

Apart from the unexpected success of the naive Bayes filter, the main reason to use this model is its computational simplicity. The model classes that I will consider, partition the feature vector into conditionally independent parts, each one containing a variable number of features. In [3] ad-hoc methods are discussed that create models with partial dependencies and experimental results indicate that the success rate of spam detection improved significantly over the Naive Bayes filter. But now a new question arises, namely what is the most appropriate partitioning model, given a set of training data, and can this model be determined or approximated in a computationally efficient manner?

I assume binary classes and binary features and also assume that the objects are drawn independently from the same distribution $P(c)P(f^k|c)$. Of course it is possible to relax the restriction on the alphabet and the method applies to non-binary discrete alphabets too.

2 Model class description

Let the feature vector index set $\{1, 2, \dots, k\}$ be written as $\mathfrak{F}$. A model $\mathcal{M}$ is described by a number of subsets $\mathfrak{s}_1, \mathfrak{s}_2, \dots, \mathfrak{s}_g$ for some number g . These subsets have a (sub-) partitioning property defined by $\mathfrak{s}_i \cap \mathfrak{s}_j = \emptyset$; if $i \neq j$, and $\bigcup_{i=1}^g \mathfrak{s}_i \subset \mathfrak{F}$. Apart from the subsets a model also contains parameters $\underline{\theta}$ that describe the probabilities of the feature vector given the class.

A subset $\mathfrak{s}$ selects some features from the feature vector f^k . If $\mathfrak{s} = \{i_1, i_2, \dots, i_s\}$ then this selection is written as $f^{\mathfrak{s}} = f_{i_1}, f_{i_2}, \dots, f_{i_s}$. The model $\mathcal{M} = (\mathfrak{s}_1, \mathfrak{s}_2, \dots, \mathfrak{s}_g)$, together with its parameters $\underline{\theta}$, define the following conditional feature vector probability.

$$P(f^k|c, \mathcal{M}, \underline{\theta}) = \prod_{i=1}^g P(f^{\mathfrak{s}_i}|c, \underline{\theta}). \quad (2)$$

A possible model class results if the subsets $\mathfrak{s}_i$ for each model form a complete partitioning. e.g. let $k = 3$, then the following models belong to this class.

Model	feature probability
$\mathcal{M}_1 = (\{1\}, \{2\}, \{3\})$	$P(f_1 c)P(f_2 c)P(f_3 c),$
$\mathcal{M}_2 = (\{1, 2\}, \{3\})$	$P(f_1, f_2 c)P(f_3 c),$
$\mathcal{M}_3 = (\{1, 3\}, \{2\})$	$P(f_1, f_3 c)P(f_2 c),$
$\mathcal{M}_4 = (\{2, 3\}, \{1\})$	$P(f_2, f_3 c)P(f_1 c),$
$\mathcal{M}_5 = (\{1, 2, 3\})$	$P(f_1, f_2, f_3 c).$

Given is a sequence of objects $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n$, where the i^{th} object is described as $\mathcal{O}_i = (c_{(i)}, f_{(i)}^k)$. Assume a given model $\mathcal{M} = (\mathfrak{s}_1, \mathfrak{s}_2, \dots, \mathfrak{s}_g)$. Due to the independence of the objects, the conditional features probability is written as

$$P(f_{(1\dots n)}^k|c_{(1\dots n)}, \mathcal{M}, \underline{\theta}) \triangleq P(f_{(1)}^k, \dots, f_{(n)}^k|c_{(1)}, \dots, c_{(n)}, \mathcal{M}, \underline{\theta}) = \prod_{i=1}^n P(f_{(i)}^k|c_{(i)}, \mathcal{M}, \underline{\theta}), \quad (3)$$

where $P(f_{(1\dots n)}^k|c_{(1\dots n)})$ is just a shorthand notation for this sequence probability.

2.1 Unknown parameters, Bayesian mixture, and the log-regret

First of all, the parameters $\underline{\theta}$ for a given model are unknown and must be estimated. Just as in [4] formula (8), I will use the Krichevsky-Trofimov probability assignment, written as $P_{KT}(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}) = \prod_{i=1, g} P_{KT}(f_{(1\dots n)}^{s_i} | c_{(1\dots n)})$.

The second problem is that I do not know the “best” (or (“real” or “ML” model) model, $\mathcal{M}_*$, in a given model class $\mathfrak{M}$. To solve this problem I will compute the following *Bayesian mixture*

$$P_e(f_{(1\dots n)}^k | c_{(1\dots n)}) = \sum_{i=1}^{|\mathfrak{M}|} P(\mathcal{M}_i) P_{KT}(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}_i), \quad (4)$$

where the model class is given as $\mathfrak{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_{|\mathfrak{M}|}\}$ and $P(\mathcal{M})$ is a prior over $\mathfrak{M}$. The log-regret measures the decrease in probability of a given probability assignment as compared to the “target” $\mathcal{M}_*$. The log-regret r can now be written as

$$r = -\log_2 P_e(f_{(1\dots n)}^k | c_{(1\dots n)}) + \log_2 P(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}_*, \underline{\theta}). \quad (5)$$

From the fact that $P_e(f_{(1\dots n)}^k | c_{(1\dots n)}) \geq P(\mathcal{M}_*) P_{KT}(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}_*)$ we obtain

$$r \leq -\log_2 P(\mathcal{M}_*) + \log_2 \frac{P(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}_*, \underline{\theta})}{P_{KT}(f_{(1\dots n)}^k | c_{(1\dots n)}, \mathcal{M}_*)}. \quad (6)$$

In this paper I am mainly concerned with the complexity of the computation of (4), but I will also consider the contribution of the mixture to the log-regret, $-\log_2 P(\mathcal{M}_*)$.

3 Introducing four model classes

I will consider *ordered* and *unordered* partitions. A partition is *ordered* if the subsets contain only consecutive feature indices, so $\mathfrak{s}_i = \{a, a+1, a+2, \dots, a+b\}$. In an *unordered* partition the subsets can contain any combination of feature indices. For both cases the partitioning can be complete (*full partitioning*) or incomplete (*sub-partitioning*).

$$\bigcup_{i=1}^g \mathfrak{s}_i \begin{cases} = \mathfrak{F}; & \text{full partitioning} \\ \subset \mathfrak{F}; & \text{sub-partitioning} \end{cases} \quad (7)$$

3.1 Class I: ordered features and full partitioning

Here I wish to compute (4) in the case where the model subsets contain consecutive indices and form a complete partition of $\mathfrak{F}$. If $\mathfrak{s}_i$ is a subset in the

model, then we call $P_{KT}(f_{(1\dots n)}^{5i}|c_{(1\dots n)})$ the corresponding *basic* probability. Obviously there are $\frac{1}{2}k(k+1)$ basic probabilities.

I consider two methods of computing (4). The *brute force* or direct computation of (4), and the *Network* method that makes use of the distributive law of algebra. I will use the *short-hand notation* P_{123} for $P_{KT}(f_{(1\dots n)}^{\{1,2,3\}}|c_{(1\dots n)})$ and so on. Also I will compute *partial mixture results*

$$N_{1,2,3} = \alpha_1 P_{KT}(f_{(1\dots n)}^{\{1,2,3\}}|c_{(1\dots n)}) + \alpha_2 P_{KT}(f_{(1\dots n)}^{\{1,2\}}|c_{(1\dots n)}) P_{KT}(f_{(1\dots n)}^{\{3\}}|c_{(1\dots n)}) \quad (8)$$

$$+ \alpha_3 P_{KT}(f_{(1\dots n)}^{\{1\}}|c_{(1\dots n)}) P_{KT}(f_{(1\dots n)}^{\{2,3\}}|c_{(1\dots n)}) \quad (9)$$

$$+ \alpha_4 P_{KT}(f_{(1\dots n)}^{\{1\}}|c_{(1\dots n)}) P_{KT}(f_{(1\dots n)}^{\{2\}}|c_{(1\dots n)}) P_{KT}(f_{(1\dots n)}^{\{3\}}|c_{(1\dots n)}). \quad (10)$$

The α 's are to be selected in an appropriate or convenient way. The final mixture result will be written as $N_{\mathfrak{F}}$. The network computations are explained by the following graph.

This graph describes e.g. the following computations.

$$N_1 = P_1; \quad N_2 = P_2; \quad N_3 = P_3; \quad N_4 = P_4. \quad (11)$$

$$N_{12} = P_{12} + N_1 \cdot N_2 = P_{12} + P_1 P_2. \quad (12)$$

$$N_{\mathfrak{F}} = N_{1234} = P_{1234} + P_1 P_{234} + P_{12} P_{34} + P_{123} P_4 + 2P_1 P_2 P_{34} + 2P_1 P_{23} P_4 + 2P_{12} P_3 P_4 + 5P_1 P_2 P_3 P_4. \quad (13)$$

I am interested in

- $T_1(k)$: The total number of terms in $N(\mathfrak{F})$. This is the normalization factor needed to turn $N(\mathfrak{F})$ into the probability $P_e(f_{(1\dots n)}^k|c_{(1\dots n)})$.
- $M_1(g)$ the multiplicity of a model with g subsets in $N(\mathfrak{F})$. Together $\frac{M_1(g)}{T_1(k)}$ define the model prior.
- $W_1(k)$ the number of additions and multiplications needed to compute $N(\mathfrak{F})$.

$T_1(k)$ is described by the recursion $T_1(k) = 1 + \sum_{i=1}^{k-1} T_1(i)T_1(k-i)$; and $T_1(1) = 1$. This results in

$$T_1(k) = \sum_{i=0}^{k-1} C_i \binom{k-1}{i}. \quad (14)$$

Here C_i is the i^{th} Catalan number, $C_i = \frac{1}{i+1} \binom{2i}{i}$.

$M_1(g)$ is described by the recursion $M_1(g) = \sum_{i=1}^{g-1} M_1(i)M_1(g-i) = C_{g-1}$. This results in a contribution to the log-regret of $r_{1,\mathcal{M}}(k, g) = -\log_2 \frac{M_1(g)}{T_1(k)}$.

$$\begin{aligned} r_{1,\mathcal{M}}(1, 1) &= 0, \\ r_{1,\mathcal{M}}(50, 50) &= 16.263, \\ r_{1,\mathcal{M}}(50, 1) &= 104.982. \end{aligned}$$

The following table lists the amount of work, $W_1(k)$, that is needed for the brute-force method and for the network method as a function of the feature length k . The graph plots the number of operations as a function of k .

Brute force

$(k-1)2^{k-2}$ multiplications.
 $2^{k-1} - 1$ additions.

Network model

$1/6(k-1)k(k+1)$ multiplications.
 $1/6(k-1)k(k+1)$ additions.

3.2 Class II: unordered features and full partitioning

In this model class the features can be distributed arbitrarily over the feature groups. The network computations are now performed with the following graph.

So we finally obtain the (unnormalized) mixture probability

$$N(\mathfrak{F}) = N_{123} = P_{123} + P_1 P_{23} + P_2 P_{13} + P_3 P_{12} + 2P_1 P_2 P_3. \quad (15)$$

- $T_2(k)$: The total number of terms in $N(\mathfrak{F})$.

$$T_2(k) = 1 + \sum_{i=1}^{k-1} \binom{k-1}{i-1} T_2(i) T_2(k-i); \quad T_2(1) = 1. \quad (16)$$

$$= \sum_{i=1}^k (2i-3)!! \left\{ \begin{matrix} k \\ i \end{matrix} \right\}. \quad (17)$$

Here $a!!$ is the double factorial and $\left\{ \begin{matrix} a \\ b \end{matrix} \right\}$ denotes a Stirling number of the second kind.

- $M_2(g)$ the multiplicity of a model with g subsets in $N(\mathfrak{F})$.

$$M_2(g) = (2g-3)!! \quad (18)$$

This results in a contribution to the log-regret of $r_{2,\mathcal{M}}(k, g) = -\log_2 \frac{M_2(g)}{T_2(k)}$.

$$\begin{aligned} r_{2,\mathcal{M}}(1, 1) &= 0, \\ r_{2,\mathcal{M}}(50, 50) &= 15.263, \\ r_{2,\mathcal{M}}(50, 1) &= 269.191. \end{aligned}$$

$W_2(k)$

Brute force

$\sum_{g=1}^k (g-1) \left\{ \begin{matrix} k \\ g \end{matrix} \right\}$ multiplications.

$\sum_{g=1}^k \left\{ \begin{matrix} k \\ g \end{matrix} \right\} - 1$ additions.

Network model

$1/2(3^k - 2^{k+1} + 1)$ multiplications.

$1/2(3^k - 2^{k+1} + 1)$ additions.

3.3 Class III: ordered features and sub-partitioning

It seems reasonable to request a model class that allows some features not to be used at all. I might have added all features I could think of, not knowing how relevant they are. So I wish to compute

$$P_e(f_{(1...n)}^k | c_{(1...n)}) = 2^{-n\alpha} \sum_{i=1}^{|\mathfrak{M}|} P(\mathcal{M}_i) P_e(f_{(1...n)}^{\mathfrak{s}_i} | c_{(1...n)}, \mathcal{M}_i), \quad (19)$$

where the model subsets contain consecutive indices and form a (partial) partition of $\mathfrak{F}$ and α is the number of unused features. As we shall see, this can be accommodated in a simple using the method for Class I. I use the additional short-hand $Z = 2^{-n}$.

The only difference from (13) is $N_1 = P_1 + Z$; $N_2 = P_2 + Z$; $N_3 = P_3 + Z$; and $N_4 = P_4 + Z$. This results in

$$\begin{aligned} N(\mathfrak{F}) = N_{1234} = & P_{1234} + P_1 P_{234} + P_{12} P_{34} + P_{123} P_4 + 2P_1 P_2 P_{34} + 2P_1 P_{23} P_4 \\ & + 2P_{12} P_3 P_4 + 5P_1 P_2 P_3 P_4 + P_{123} Z + P_{234} Z + 2P_1 P_{23} Z + \dots + 5Z^4. \end{aligned} \quad (20)$$

Remember: k is the length of the feature vector; g is the number of subsets $\mathfrak{s}$ in a model $\mathcal{M}$ plus the number of unused features. So every unused feature counts as a subset.

- $T_3(k)$: The total number of terms in $N(\mathfrak{F})$. $T_3(k) = 1 + \sum_{i=1}^{k-1} T_3(i)T_3(k-i)$; and $T_3(1) = 2$.

$$T_3(k) = \frac{1}{k} \binom{2(k-1)}{k-1} + \sum_{g=1}^k \frac{1}{g} \binom{2(g-1)}{g-1} \sum_{a=0}^{g-1} \binom{g}{a} \binom{k-a-1}{g-a-1}. \quad (21)$$

- $M_3(g)$ the multiplicity of a model with g subsets in $N(\mathfrak{F})$.

$$M_3(g) = \frac{1}{g} \binom{2(g-1)}{g-1}. \quad (22)$$

This results in a contribution to the log-regret of $r_{3,\mathcal{M}}(k, g) = -\log_2 \frac{M_3(g)}{T_3(k)}$.

$$\begin{aligned} r_{3,\mathcal{M}}(1, 1) &= 1, \\ r_{3,\mathcal{M}}(50, 50) &= 54.6877, \\ r_{3,\mathcal{M}}(50, 1) &= 143.407. \end{aligned}$$

$W_3(k)$

First define $n_3(k, g) = \sum_{a=0}^{g-1} \binom{g}{a} \binom{k-a-1}{g-a-1}$ if $g < k$; and $n_3(k, g) = 2^k$ if $g = k$.

Brute force

$\sum_{g=1}^k n_3(k, g)(g-1)$ multiplications.

$\sum_{g=1}^k n_3(k, g) - 1$ additions.

Network model

$\frac{1}{6}(k-1)k(k+1)$ multiplications.

$\frac{1}{6}(k-1)k(k+1) + k$ additions.

3.4 Class IV: unordered features and sub-partitioning

Just as in Class III we only add Z to each of the nodes $N_1, \dots, N_k$ and obtain the following.

$$N(\mathfrak{F}) = N_{123} = P_{123} + P_1P_{23} + P_2P_{13} + P_3P_{12} + 3P_1P_2P_3 + P_{12}Z + P_{13}Z + P_{23}Z + 3P_1P_2Z + 3P_1P_3Z + 3P_2P_3Z + 3P_1Z^2 + 3P_2Z^2 + 3P_3Z^2 + 3Z^3. \quad (23)$$

- $T_4(k)$: The total number of terms in $N(\mathfrak{F})$.

$$T_4(k) = 1 + \sum_{i=1}^{k-1} \binom{k-1}{i-1} T_4(i) T_4(k-i); \quad T_4(1) = 2. \quad (24)$$

$$= 2^k (2k-3)!! + \sum_{g=1}^{k-1} (2i-3)!! \sum_{a=0}^{g-1} \binom{k}{a} \left\{ \begin{matrix} k-a \\ g-a \end{matrix} \right\}. \quad (25)$$

- $M_4(g)$ the multiplicity of a model with g subsets in $N(\mathfrak{F})$.

$$M_4(g) = (2g-3)!! \quad (26)$$

This results in a contribution to the log-regret of $r_{4,\mathcal{M}}(k, g) = -\log_2 \frac{M_4(g)}{T_4(k)}$.

$$\begin{aligned} r_{4,\mathcal{M}}(1, 1) &= 1, \\ r_{4,\mathcal{M}}(50, 50) &= 54.4977, \\ r_{4,\mathcal{M}}(50, 1) &= 308.425. \end{aligned}$$

$W_4(k)$

$$n_4(k, g) = \sum_{a=0}^{g-1} \binom{g}{a} \begin{cases} k-a \\ g-a \end{cases} \text{ if } g < k; \quad n_4(k, g) = 2^k \text{ if } g = k. \quad (27)$$

Brute force

$\sum_{g=1}^k n_4(k, g)(g-1)$ multiplications.
 $\sum_{g=1}^k n_4(k, g) - 1$ additions.

Network model

$\frac{1}{2}(3^k - 1) - (2^k - 1)$ multiplications.
 $\frac{1}{2}(3^k - 1) - (2^k - 1) + k$ additions.

4 Remarks and conclusions

A basic probability $P(f^{\mathfrak{s}}|c)$ is described by an exponential (in $|\mathfrak{s}|$) number of parameters. It can be useful to restrict the sizes of the subsets $\mathfrak{s}$.

4.1 Limiting the group size

We might wish to consider only those models that contain subsets of limited size. A reason for this can be that we have a-priori knowledge that the subsets are limited in size as will often be the case in spam filters. Another reason could be to limit the complexity of the problem even further.

The following graph shows as an example a feature vector of length $k = 5$ where the subset size can be at most 2.

$$N_{12345} = P_{12}P_{34}P_5 + P_{12}P_3P_{45} + P_1P_{23}P_{45} + \quad (28)$$

$$3P_{12}P_3P_4P_5 + P_1P_{23}P_4P_5 + P_1P_2P_{34}P_5 + 3P_1P_2P_3P_{45} + \quad (29)$$

$$4P_1P_2P_3P_4P_5. \quad (30)$$

This approach can be applied to all four model classes.

4.2 Conclusions

The network method gives a substantial reduction in number of operations.

The unordered features classes are still very complex.

The method is flexible and can be used for several model classes.

The class and feature variables need not be binary. The influence of the alphabet size is only in the ‘basic’ probabilities.

The graph can easily be adapted to allow for limited group sizes, and thus allows a trade-off between computational complexity and the precision to fit the proper model.

The method can be implemented sequentially for on-line prediction or classification.

References

- [1] T.M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- [2] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A Bayesian approach to filtering junk e-mail,” In *Learning for Text Categorization Papers from the AAAI Workshop*, pages 5562, Madison, Wisconsin, 1998.
- [3] H. Yang, M. Stassen, and T. Tjalkens, “Classification with the CTW algorithm,” *Proc. 28th Symposium on Information Theory in Benelux*, Enschede, The Netherlands, pp. 321-328, May 24 – 25, 2007.
- [4] F.M.J. Willems, Yu.M. Shtarkov, and Tj.J. Tjalkens, “The context-tree weighting method: basic properties,” *IEEE Trans. Inform. Th.*, vol IT-41, no 3, pp. 653–664, May 1995.