

Inter-element dependency models for sequence classification

Adrian Silvescu, Carson Andorf, Drena Dobbs, Vasant Honavar

Artificial Intelligence Laboratory
Department of Computer Science and
Graduate Program in Bioinformatics and Computational Biology
Iowa State University
Ames, IA 50011, USA
{silvescu|andorf|ddobs|honavar}@iastate.edu

Abstract

Naive Bayes is a fast to train model for sequence classification. We develop and experiment with two methods that are equally fast (they require only one pass through the training data), but they are also able to model interactions among close neighbours in the sequence (unlike the Naive Bayes independence assumption). The first method basically runs Naive Bayes on overlapping k-grams obtained from the sequence. The second method, called NB(k), constructs a classifier associated with an undirected graphical model over the sequence that takes into account k-wise dependencies. We test our algorithms on protein function classification tasks based on functional families from the Gene Ontology (GO) database. Our results show significant improvements of the proposed methods over Naive Bayes with NB(k) leading over NB k-grams in all test cases. The two proposed methods despite having improved modelling accuracy and demonstrated improved test accuracy maintain the “one pass through the data only” training property of Naive Bayes thus yielding fairly accurate and efficient classifiers.

1 Introduction

Sequence classification is a task that appears in many natural scenarios such as protein sequence function prediction, intrusion detection, text classification, etc.

A simple solution that offers a quick and practical approach to this problem is Naive Bayes. Naive Bayes assumes that the elements in the sequence are independent of each other given the class. This independence assumption however seldom holds in practice. Furthermore most of the time the dependencies are very strong among close neighbours in the sequence. In order to address this concern we would like to be able to relax the strong independence assumption of Naive Bayes and allow for a limited amount of interaction among close neighbours, and thus get better modelling

accuracy. Despite the additional sophistication we would like to maintain the “one pass through the data only” training property of Naive Bayes. Note that since we are preoccupied with sequence classification whenever we talk about Naive Bayes we actually refer to the so called Naive Bayes Multinomial which is more appropriate for our case (sequence classification) as opposed to the Multivariate model (see [7] for a discussion about it).

We are going to provide two methods to deal with this problem. The first one basically splits the sequence into a sequence of overlapping k -grams and runs a Naive Bayes algorithm in the transformed space. We call the resulting algorithm NB k -grams. This method however violates the assumption of independence that Naive Bayes holds in an obvious way: Since the k -grams are overlapping, every two subsequent k -grams share $k-1$ elements, which makes them evidently non-independent.

In the second method, in order to deal with the above mentioned problem properly we construct an undirected graphical model for the k -grams [4], which will adequately discount the contributions of the overlaps. We show the equivalence of this undirected model with the Markov Model of order $k-1$. This contributes to dispelling a misconception that undirected models (i.e., random fields) are more suitable than directed models such as Markov Models for modelling protein sequences. This view may be held based on the fact that the order in the protein sequence has a spatial nature, which is better modelled by undirected models (random fields), rather than a temporal nature as it is the case with Markov Models. While we do not dispute the spatial nature of the order in protein sequences, we show that both the undirected (spatial) and the directed models (temporal = Markov Models) associated with k -grams yield the same probability distribution.

In order to use our undirected models for classification, we apply the basic trick of turning a generative model into a predictor: We train one generative model per class and then when asked to predict a class for a newly seen sequence we output the one whose corresponding generative model outputs the highest probability for the given sequence, weighted with the probability of the class. We call this model NB(k) from Naive Bayes order k (which is when we model dependencies among k attributes). Note that Naive Bayes will turn out to be NB(1) (we model dependencies among 1 attribute, that is no dependencies, i.e., the independence)

We put to test our methods on three protein function prediction problems with functional classes extracted from GO (the Gene Ontology [5]) and corresponding sequences extracted from SWISSPROT [2]. In our experiments NB k -grams outperformed Naive Bayes on all the tests and NB(k) outperformed NB k -grams on all tests too.

To recapitulate, we will present two methods that relax the independence assumption of Naive Bayes by modelling interaction among close neighbours in the sequence. The two methods preserve the “one pass through the data” training property of the Naive Bayes and they are also easily update-able. Additionally our second method - NB(k) which is not incidentally, also our best performer, represents and end in itself since it is the most complete undirected graphical model for k -grams modelling. We further explore the equivalence of this undirected model to the (directed) Markov Model of order $k-1$ thus clearing a misconception about the presumed superiority of undirected graphical models vs. the directed ones in terms of modelling sequences where the order is spatial as opposed to temporal.

The rest of the paper will proceed as follows: First, in the next section we will have an in depth exploration of the intuition and theory behind the two methods to be explored. Then, we put them to test in section three and produce some experimental results. Finally, in section four we conclude with a summary and discussion.

2 Method

In this section we will preoccupy ourselves with unraveling the details of the new models. As a basic strategy we will first start from basic principles and try to develop some intuition behind the new methods followed by more precise and general formulations, which will be furthermore supplemented by theoretical results.

We start in the first subsection with an outline of the Naive Bayes Classifier and the general method for producing classifiers from generative models. Then we examine a way to produce undirected probabilistic models that take into account correlations among proximal elements within a sequence. Subsequently, we outline the associated directed graphical models and comment on the relation with the directed ones. We then briefly present the parameter fitting method.

2.1 Naive Bayes and the General Method for producing classifiers from generative models.

Before plunging into the details, a little bit of **Notation**: We will denote a whole sequence with \bar{S} , the i -th element of the sequence by S_i and the value of the i -th element in the sequence by s_i , furthermore let V be the set of all possible values that the elements in the sequence can take. Let C be the class variable and let $\{c_j\}_{j=1,m}$ be the possible values that C can take. And let n be the length of the current sequence i.e., $\bar{S} = [S_1 = s_1, \dots, S_n = s_n]$.

The Naive Bayes algorithm for sequence classification assumes that the successive elements in a given sequence are independent of each other given the class.

One way to interpret Naive Bayes is as a generative model. A generative model produces the observed data by the means of a probabilistic generation process. The generation process for Naive Bayes is very simple:

Algorithm 1 Naive Bayes Sequence Generative Model

1. First generate a class c_j according to the probability $P(C)$
2. Then for $i = 1$ to n
 generate s_i according to $P(S_i | C = c_j)$.

Note: Since we also assume stationarity $P(S_i | C = c_j) = P(S_l | C = c_j) := P(S | C = c_j)$ for every $i, l \in \{1, \dots, n\}$ and $c_j \in C$

Note that in step 2 we are generating each element in the sequence independent of the previous ones or the ones that are going to come. The generative model implemented by the Algorithm 1 can be depicted as a Bayesian Network as shown in Figure

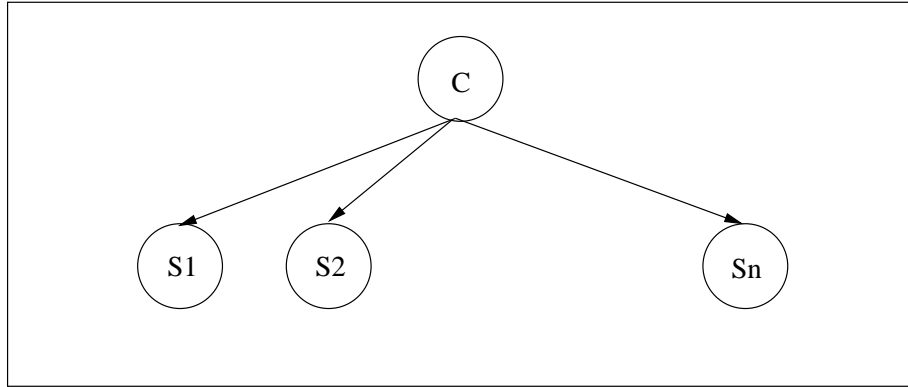


Figure 1: Naive Bayes Bayesian Network

1. In this figure the arrows from the class variable to the elements of the sequence show the dependencies in the generative process.

The Naive Bayes classifier is a particular example of a general scheme of producing a classifier for a given generative model for the input data (sequence in our case). We will try to outline it in what follows with the purpose of using it with other more complicated generative models.

Let us assume we have a probabilistic generative model for sequences GM . That is a probabilistic model which given a sequence $\bar{S} = s_1, \dots, s_n$ can give us the probability $P_{GM}(\bar{S} = s_1, \dots, s_n)$ according to the generative model GM . Then if we want to obtain a classifier using the generative model GM we can use the following (standard) procedure:

1. For each class c_j train a generative model $GM(c_j)$ using the data from class c_j .
2. Then when asked to classify a new sequence $\bar{S} = s_1, \dots, s_n$ predict it as belonging to class:

$$classification = \arg \max_{c_j \in C} P_{GM(c_j)}(\bar{S} = s_1, \dots, s_n) P(c_j)$$

or equivalently:

$$classification = \arg \max_{c_j \in C} P(\bar{S} = s_1, \dots, s_n | GM(c_j)) P(c_j)$$

or equivalently if, by abuse of notation, $P(\bar{S} = s_1, \dots, s_n | c_j)$ stands for $P(\bar{S} = s_1, \dots, s_n | GM(c_j))$ we will have:

$$classification = \arg \max_{c_j \in C} P(\bar{S} = s_1, \dots, s_n | c_j) P(c_j)$$

which in the case of the Naive Bayes generative model is the well known:

$$classification = \arg \max_{c_j \in C} P(S_i = s_i | c_j) P(c_j)$$

because under the independence assumption given the class we have:

$$P(\bar{S} = s_1, \dots, s_n | c_j) = \prod_{i=1}^n P(S_i = s_i | c_j)$$

Note that in the case of Naive Bayes the generative model for the input data is assumed to be a set of independent samples drawn from the same multinomial distribution $P(S_i = s_i | c_j)$ (so the joint $P(S = s_1, \dots, s_n | c_j)$ can be obtained by taking the product). Furthermore under the stationarity assumption the probability distribution does not depend on the position i (i.e., $P(S_i | C = c_j) = P(S_l | C = c_j) = P(S | C = c_j)$ for every $i, l \in \{1, \dots, n\}$ and $c_j \in C$).

We are interested however in capturing more interactions among the elements in the sequence, as they are seldom independent and as such, we will need a more complicated model for the data. Given that this model is established, we can then produce a classifier using the scheme outlined above. So from now on we will forget about classes and we will only be concerned with finding probabilistic models that take into account dependencies among elements in the sequence. And then whenever we shall be asked produce classifications with our derived a probabilistic model PM we will apply the schema outlined in this section by replacing in the formulas $P_{GM(c_j)}(\dots)$ with $P_{PM(c_j)}(\dots)$.

Now that we have outlined the general scheme for producing classifiers from Probabilistic Models we reduced our problem to the one of finding some appropriate probabilistic models that capture correlations among close elements within a sequence. Upon finding such models we can turn them into a classifier using the scheme outlined above. Hence, classification being set aside, we are ready to launch into the exploration of dependency models among sequence elements.

2.2 Models for direct interaction among k consecutive elements

In graphical depiction we show the interaction model for Naive Bayes in Figure 2 a) for a sequence of five elements. What we would like now to model is dependencies among two or more attributes. One way to represent these dependencies in a graphical form is by drawing edges between the nodes that are supposed to be directly dependent on each other. The graph for pairwise dependencies is illustrated in Figure 2 b) and the one for 3-wise dependency is depicted in Figure 2 c).

The models show in Figure 2 are called Probabilistic Undirected Graphical Models / Markov Networks / Random Fields. The arcs in the graphs denote the direct dependencies. By complementarity the absence of an arc between two nodes denotes the absence of a direct dependency. This does not necessarily mean independence, and most often than not, it isn't independence. Two nodes are going to be indirectly dependent if they are going to be connected in the graph by a path containing more than one edge. The absence of an arc between two nodes means that there may be situations where if we know some of the values of nodes on paths connecting our two nodes, this might yield the two nodes independent. More exactly in order to get independence between two indirectly connected nodes (connected by a path of length greater than one) we have to know the value of at least one node on each of the paths between these two

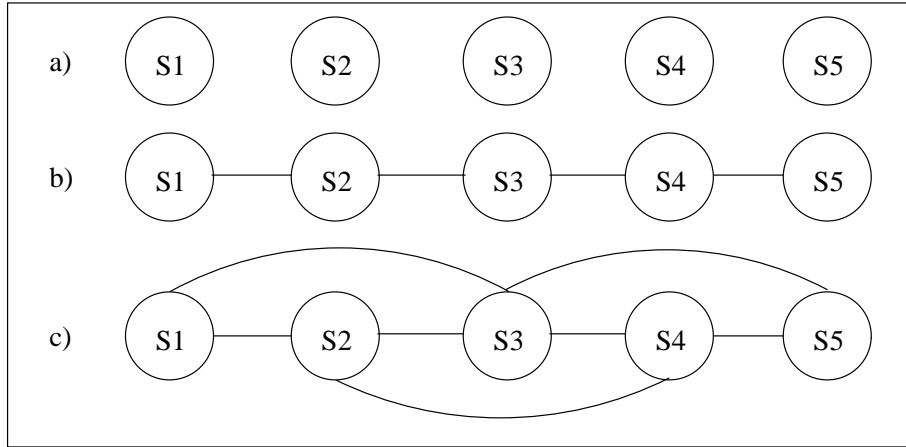


Figure 2: Graphical depiction of the dependence between the elements in a sequence of five elements using Undirected Graphical Models: a) shows the Naive Bayes [= independence model]; b) shows pairwise dependence and c) shows 3-wise dependence.

nodes. The knowledge of the values of these nodes is going to “block” the indirect dependence between our two unconnected nodes, thus yielding them independent.

Now that we have explained the intuition behind the graphical representation we are going to concern ourselves with a way of capitalising on this intuition and try to derive a formula for the probability of a data-point (sequence in our case). That will allow us to provide $P_{PM}(\bar{S})$ which is all that is needed in order for our classification scheme to work. In the independence assuming case (Figure2 a) the formula for our 5 elements of the sequence is as follows:

$$P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5]) = \prod_{i=1}^5 P(S_i = s_i)$$

and in general for a sequence of n elements we have:

$$P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n]) = \prod_{i=1}^n P(S_i = s_i)$$

2.2.1 UNPkgP & NB k-grams

One intuitive way to try to define a probability distribution for the case of the graphical model representing pairwise dependencies (Figure2 b) is to take the product of the marginals of the directly dependent node as proposed by the following formula (for our 5 elements sequence from Figure2 b) [Since the nodes are pairwise dependent the information in the pairwise marginals is presumably important and we cannot derive it from products of single variable marginals]:

$$P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5]) ? = \prod_{i=1}^4 P(S_i = s_i, S_{i+1} = s_{i+1})$$

This will not produce a probability distribution however, and the intuition behind it is that we are somehow “double counting” or more exactly “double multiplying” the influence of some of the nodes (such as $S_2 = s_2, S_3 = s_3, S_4 = s_4; S_2 = s_2$ for example appears both in $P(S_1 = s_1, S_2 = s_2)$ and $P(S_2 = s_2, S_3 = s_3)$).

There are two ways to deal with this problem. The first one is to ignore the double counting and in order to get a probability distribution just normalise it over all the possible values that the elements in the sequence can take. So in general we will have

$$P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n]) = \frac{1}{Z} \prod_{i=1}^{n-1} P(S_i = s_i, S_{i+1} = s_{i+1})$$

Where Z is given by the following formula:

$$Z = \sum_{S=(s_1, \dots, s_n) \in V^n} \prod_{i=1}^{n-1} P(S_i = s_i, S_{i+1} = s_{i+1})$$

We will call this model Probabilistic 2-gram Propositionalisation or shortly P2gP. More generally, for the case of k -wise direct dependencies we will call this model Probabilistic k -gram Propositionalisation or shortly PkgP.

$$P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n]) = \frac{1}{Z} \prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})$$

and Z being given by:

$$Z = \sum_{S=s_1, \dots, s_n \in V^n} \prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})$$

Note that the normalisation constant Z (a.k.a. partition function) is a very complicated thing to calculate and with the exception of the cases when we have very small sequences we cannot compute it. One possible way to deal with this problem is to just ignore it, or basically set Z to 1 and use the corresponding results (which are not going to be true probabilities) instead of the true probability $p(\bar{S})$ when doing classification. We will call this model UnNormalised Probabilistic k -gram Propositionalisation or in short UNPkgP. This model turns out, from the point of view of classification to be the same as transforming the sequence into a bag of overlapping k -grams and using the Naive Bayes classifier on it. Here is a more precise statement of the claim and the proof for it.

Claim: *The class conditional (classifier) version of UNPkgP [denoted C-UNPkgP] yields the the same decision as Naive Bayes on k -grams [denoted NB k -grams].*

Proof.

$$\begin{aligned}
C - UNPkgP &= \arg \max_{c_j \in C} P_{C-UNPkgP(c_j)}(\bar{S} = [S_1 = s_1, \dots, S_n = s_n])P(c_j) \\
&= \arg \max_{c_j \in C} \prod_{i=1}^{n-k+1} P_{C-UNPkgP(c_j)}(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})P(c_j) \quad (1) \\
&= \arg \max_{c_j \in C} \prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1} | c_j)P(c_j) \quad (2) \\
&= \arg \max_{c_j \in C} \prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1} | c_j)P(c_j) = NB\ k - grams
\end{aligned}$$

Where the transition from equation (1) to equation (2) is due to the fact that $P_{C-UNPkgP(c_j)}(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})$ is estimated in the same way as $P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1} | c_j)$ from the training data, namely as:

$$P(s_1, \dots, s_k | c_j) = \frac{\text{counts}(s_1, \dots, s_k, c_j)}{\text{counts}(c_j)}$$

(see Section 2.3 Training ... for more details)□

As a consequence of this claim we are going to use Naive Bayes applied to k-grams as our first proposed method which we showed to be identical in classification results to turning an UNPkgP probabilistic model into a classifier.

2.2.2 NB(k)

In a different vein however we can try to fix the problem of “double multiplying” by dividing by the probabilities associated with the double-counted nodes. That is, for our 5 letter sequence from Figure 2 b):

$$P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5]) ? = \frac{\prod_{i=1}^4 P(S_i = s_i, S_{i+1} = s_{i+1})}{\prod_{i=2}^4 P(S_i = s_i)}$$

It turns out that this is the right thing to do and what we get is a full blown probability distribution which is the “correct” one. The generalisation for $k > 2$ is as follows:

$$P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n]) = \frac{\prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})}{\prod_{i=2}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-2} = s_{i+k-2})} \quad (3)$$

The obtained probability model is “correct” in the following sense: It is the same probability model that would be obtained from considering a Markov Network / Random Field over the sequence, whose graph is constructed by drawing directed arcs between all k consecutive elements of the sequence. Additionally the parameters for the

marginal probabilities of each k consecutive elements (the maximal cliques of these graphs) are shared for all positions. We are producing in this case a position independent model (or equivalently the probability distribution associated with it is stationary). More exactly if we model dependence among k attributes we will call the model NB(k) (from Naive Bayes of k). Note that in this case NB(1) is really the independence case which yields Naive Bayes.

The fact that the probability distribution obtained in the previous formula is correct is supported by the Junction Tree Theorem (see [6]):

Theorem (Junction Tree): *Let the Markov Network for a set of random variables be described by a chordal graph G , then the probability distribution associated with the Markov network can be written as the product of the marginals of the maximal cliques divided by the product of the marginals of the separators. i.e.,*

$$P(\bar{X} = [X_1 = v_1, \dots, X_n = v_n]) = \frac{\prod_{c \in \text{MaxCliques}} P(X_c = x_c)}{\prod_{s \in \text{Separators}} P(X_s = x_s)}$$

Before proceeding with a proof sketch a few potentially useful **Definitions**:

Definition: (**elementary cycle**) A cycle is called elementary if it contains no sub-cycles.

Definition: (**chordal graph**) A graph G is chordal iff it has no elementary cycles of length bigger than 3.

Definition: (**clique**) A clique is a set of nodes from a graph G such that there is an edge between every pair of nodes in the clique.

Definition: (**maximal clique**): A clique c from a graph G is called maximal clique if any superset of nodes that contains the clique c is not a clique.

Definition: (**separator**) Let c_1 and c_2 be two cliques from a graph G . The separator of these two cliques c_1 and c_2 is $c_1 \cap c_2$.

Remark: It is obvious that the sequence graphs that we are dealing with are chordal and the application of the theorem will give us our previously derived formula i.e., equation (3).

Proof Idea. We will give a proof of the theorem for the sequence graphs that we are dealing with and refer the reader to [6] for the proof in the case of general chordal graphs.

Intuitive example: For our sequence of 5 elements the probability of a sequence is as follows:

$$\begin{aligned} P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5]) &= \frac{\prod_{i=1}^4 P(S_i = s_i, S_{i+1} = s_{i+1})}{\prod_{i=2}^4 P(S_i = s_i)} \\ &= \frac{P(S_1 = s_1, S_2 = s_2)P(S_2 = s_2, S_3 = s_3)P(S_3 = s_3, S_4 = s_4)P(S_4 = s_4, S_5 = s_5)}{P(S_2 = s_2)P(S_3 = s_3)P(S_4 = s_4)} \end{aligned}$$

where if we write $P(S_1 = s_1, S_2 = s_2) = P(S_1 = s_1)P(S_2 = s_2|S_1 = s_1)$ and if we also group together the $i+1$ -th term from the numerator with the i -th term from the denominator into a conditional. For example for the 2-nd term from numerator and

the 1-st term from the denominator we have: $P(S_2 = s_2, S_3 = s_3)/P(S_2 = s_2) = P(S_3 = s_3|S_2 = s_2)$ we get:

$$P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5]) =$$

$$P(S_1 = s_1)P(S_2 = s_2|S_1 = s_1)P(S_3 = s_3|S_2 = s_2)P(S_4 = s_4|S_3 = s_3)P(S_5 = s_5|S_4 = s_4)$$

which is $P(\bar{S} = [S_1 = s_1, \dots, S_5 = s_5])$ given the independences represented by the graphical model.

Similarly we can show that for k-wise dependencies:

$$P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n]) = \frac{\prod_{i=1}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})}{\prod_{i=2}^{n-k+1} P(S_i = s_i, \dots, S_{i+k-2} = s_{i+k-2})} \quad (4)$$

$$= P(S_1 = s_1, \dots, S_{k-1} = s_{k-1}) \prod_{i=k}^n P(S_i = s_i | S_{i-1} = s_{i-1}, \dots, S_{i-k+1} = s_{i-k+1}) \quad (5)$$

Which proves the theorem for sequence graphs. \square

Note that the previous formula is identical to the probability that a Markov(k-1) model will assign. Which brings us to the next Section.

2.2.3 Directed Graphical Model Interpretation of NB(k) and the Random Fields vs. Markov models debate.

If we are to construct a directed graphical model instead of an undirected one we will get, for our sequence of length 5, the graphs depicted in Figure 3a) b) c) for independence, pairwise and 3-wise dependency respectively. These are graphs representing the Markov(k-1) models in the general case (see [4, 8]). (Note that the Markov(k-1) models dependency among k elements: the previous k-1 and current one, hence the correspondence with NB(k) and not with NB(k-1))

The probability distributions represented by the undirected graphical models are the same as the one of the directed one as mentioned in the Proof Idea of the Junction Tree Theorem. More exactly we have shown that $P(\bar{S} = [S_1 = s_1, \dots, S_n = s_n])$ can be written either as in equation (4) which is the formula for the undirected model or as in equation (5) which is the formula for the directed / Bayesian network / Markov(k-1) model.

We may think that Markov Models, since they are time oriented probabilistic models, might be less suited for modelling sequences that do not have this temporal dimension but where the dimension along which the sequence spans of is rather spatial than temporal nature (as in the case of protein sequences vs. speech utterances for example). As a consequence a need for using undirected graphical models may be felt. We have proved however [see Proof Idea] that the Markov(k-1) Models (temporally oriented) and the Markov Networks / Random Fields (undirected - spatially based) / NB(k) models produce the same probability distribution. Hence, which type of model to choose becomes more of a matter of preference or taste rather than one of choosing between models that yield different probabilities.

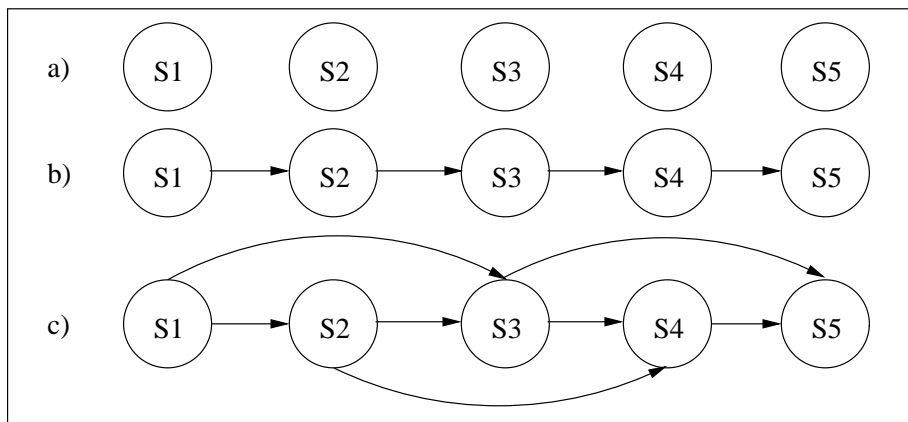


Figure 3: Bayesian Network (Markov Model) representation of the k -wise dependency. a) shows the Naive Bayes = independence model; b) shows pairwise dependence and c) shows 3-wise dependence

2.3 Training, i.e., Probability Estimation from Data

In order to train our models we need to get estimates from the probabilities $P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1})$ for each possible class value c_j . Because of the stationarity assumption these probabilities are going to be the same for each i so $P(S_1 = s_1, \dots, S_k = s_k | c_j) = P(S_i = s_i, \dots, S_{i+k-1} = s_{i+k-1} | c_j)$ for all values of i . We will denote their common value for a set of elements s_1, \dots, s_k and a class c_j by notation abuse with $P(s_1, \dots, s_k | c_j)$. We will estimate this quantity in the standard way as follows:

$$P(s_1, \dots, s_k | c_j) = \frac{\text{counts}(s_1, \dots, s_k, c_j)}{\text{counts}(c_j)}$$

Where $\text{counts}(s_1, \dots, s_k, c_j)$ are the number of times the elements s_1, \dots, s_k appear in a sequence of belonging to class c_j at any position and $\text{counts}(c_j)$ is how many such k -grams exist within sequences from class c_j .

In the case of directed models the estimation can be done in a similar way namely by the following formula:

$$P(s_1 | s_2, \dots, s_k, c_j) = \frac{\text{counts}(s_1, \dots, s_k, c_j)}{\text{counts}(s_2, \dots, s_k, c_j)}$$

Now that we have showed how to fit the parameters of the model we are ready to test it experimentally.

Furthermore in order to avoid zero probabilities and to get a mild regularisation effect we used the Laplace correction for the previous probability estimates (Which basically initialises the counts with 1 instead of 0 and adds to the denominator an appropriate count in order to ensure that the probabilities sum up to 1).

3 Experimental setup and results

We will test the two classification algorithms associated with the probabilistic models derived in Sections 2.2.1 and 2.2.2 against Naive Bayes. More precisely we will test Naive Bayes, NB k-grams and NB(k) on three protein function classification datasets.

3.1 Data Sets

The first dataset is based on families of yeast and human kinases. These families were chosen for this study because many of them are well-characterized, with known structures and functions. The data set used in this study consisted of proteins belonging to the Gene Ontology, functional family GO0004672, Protein Kinase Activity. We classified them according to the highest level below GO0004672. This consists of 5 groups. In GO their labels are GO0004672, Protein Kinase Activity; GO0004674, protein serine/threonine kinase activity; GO0004713, protein-tyrosine kinase activity; GO0004712, protein threonine/tyrosine kinase activity; and GO0004716, receptor signaling protein tyrosine kinase activity. These families represent all the yeast and human proteins that can be extracted from SWISSPROT that have the given GO classification. Since GO is represented as a directed acyclic graph, some proteins may be represented in multiple classifications. We filter the data to remove any multi-labeled proteins to guarantee disjoint classes. This dataset includes a total of 396 proteins: 102 proteins from GO0004672; 209 proteins from GO0004674, 69 proteins from GO0004713, 10 proteins from GO0004712, and 6 proteins from GO0004716.

The second dataset is based on two subfamilies of GO0003824, Catalytic Activity. This division is at a higher level of GO than the previous dataset. The data set used in this study consisted of proteins belonging to the Gene Ontology functional family GO0004672, Protein Kinase Activity and GO001684, Protein Ligase Activity. These families represent all the yeast proteins that can be extracted from SWISSPROT that have the given GO classification.. This data includes a total 376 proteins: 158 proteins from Kinase, and 218 proteins from Ligase.

The third dataset is a super set of dataset two. It contains the Kinase data and Ligase data in addition to two other subfamilies of GO0003824, Catalytic Activity. These families are GO0004386, Protein Helicase Activity, and GO0016853, Protein Isomerase Activity. This dataset tests the classifiers ability on a larger number of classes at a high level of GO. This data includes a total of 572 proteins: 158 proteins from Kinase, 218 proteins from Ligase, 110 proteins from Helicase, and 86 proteins from Isomerase.

[Shorter version] Datasets: The three Datasets are as follows: The first one Kinase, contains 5 classes of Kinases amounting to a total of 396 proteins with the following per class count (102, 209, 69, 10, 6); The second one Kinase/Isomerase aims to discriminate between Kinases and Ligases (i.e., a two class problem) and contains a total of 376 protein sequences with a per class count of (158, 218); The third dataset Kinase/Ligase/Helicase/Isomerase contains four classes of proteins: Kinases, Ligases, Helicases and Isomerases respectively, amounting to 572 proteins with a per class count of (158, 218, 110, 86). All the protein functions have been Extracted from GO (the Gene Ontology,[5]) and the corresponding protein sequences were extracted from SWISSPROT [2]. The datasets are available for download at [1].

k	NB	NB k-grams	NB(k) [\Leftarrow MM(k-1)]	Improvement
1	66.1	66.1	66.1	0.0
2	X	81.3	88.6	7.3
3	X	89.9	92.7	2.8
4	X	90.4	91.6	1.6

Table 1: Kinase (GO0004672) dataset results. There were a total of 396 proteins and five classes. k refers to the number of dependencies used in each algorithm. NB shows the accuracy of running Naive Bayes on the dataset. NB k-grams shows the accuracy of running Naive Bayes using k-grams on the dataset. NB(k) [\Leftarrow MM(k-1)] shows the accuracy of running NB(k) using k dependencies on the dataset.

3.2 Experiments and Results

The computational experiments were motivated by the following questions:

How do the two presented methods NB k-grams and NB(k) compare among each other and against the baseline represented by Naive Bayes (i.e., NB(1), a.k.a. NB 1-grams)?

Furthermore we inquire upon the effect that the number of dependencies, k, has on the classification accuracy of the dependency ?

Tables 1,2 and 3 show the results of our experiments using the three datasets. For each dataset we ran the three separate algorithms Naive Bayes, (i.e., NB(1) a.k.a. NB 1-grams), Naive Bayes with k-grams, and NB(k).

We ran each algorithm with the size of dependency k sizes ranging from 1 to 4 (except for Naive Bayes, of course). For sizes of k larger than 4, e.g., for 5 we need to calculate over three million probabilities ($=20^5$) per class; and we do not have enough data to reliably estimate these probabilities. Therefore all of our data gets washed out by prior probabilities given by the Laplace correction and everything is classified as belonging to the largest class, the class with the highest probability of occurring. For our alphabet we chose the standard 20 letter amino acid alphabet (although using a smaller alphabet will decrease the complexity and therefore the runtime of these algorithms, it offers no benefit in terms of accuracy, neither of which is our ultimate goal, so we exclude these results[3]. Note that the reduced alphabets are obtained by grouping some amino acids under the same symbol and treating the as one type of entity). As already hinted all of our probability estimates for all methods used Laplace correction. The accuracy estimates in all the tables are based on stratified 10-fold cross validation.

The tables 1,2 and 3 show from left to right: the dependency number k (k-wise dependency); the accuracy for Naive Bayes, the accuracy for Naive Bayes k-grams, the accuracy for NB(k), and the improvement in accuracy of NB(k) over NB k-grams

[Can Be omitted in a short paper] Table 1 shows the results from the Kinase dataset. Using Naive Bayes alone we were able to get a 66% classification rate. By increasing our value of k to 2 we were able to increase accuracy to 81.3% for NB 2-grams and 88.6% with NB(2). For NB(2) this was over a 22% improvement over Naive

k	NB	NB k-grams	NB(k) [\Leftarrow MM(k-1)]	Improvement
1	77.9	77.9	77.9	0.0
2	X	83.5	84.6	1.1
3	X	84.0	85.6	1.6
4	X	85.9	90.7	4.8

Table 2: Kinase(GO0004672) / Ligase(GO001684) dataset results. There were a total of 396 proteins and two classes. K refers to the number of dependencies used in each algorithm. NB shows the accuracy of running Naive Bayes on the dataset. NB k-grams shows the accuracy of running Naive Bayes using k-grams on the dataset. NB(k) [\Leftarrow MM(k-1)] shows the accuracy of running NB(k) using k dependencies on the dataset.

k	NB	NB k-grams	NB(k) [\Leftarrow MM(k-1)]	Improvement
1	56.1	56.1	56.1	0.0
2	X	70.3	72.2	1.9
3	X	79.5	80.8	1.3
4	X	79.4	82.0	2.6

Table 3: Kinase(GO0004672) / Ligase(GO001684) / Helicase Activity(GO0004386) / Isomerase(GO0016853) dataset results. There was a total of 572 proteins and four classes. K refers to the number of dependencies used in each algorithm. NB shows the accuracy of running Naive Bayes on the dataset. NB k-grams shows the accuracy of running Naive Bayes using k-grams on the dataset. NB(k) [\Leftarrow MM(k-1)] shows the accuracy of running NB(k) using k dependencies on the dataset.

Bayes and over 7% improvement over NB 2-grams. By increasing k to 3, we again see an improvement. NB 3-grams and NB(3) improved to 89.9% and 92.7% respectively. When we increase k again to 4 we get little improvement on this dataset. NB 4-grams improved by only 0.5% and NB(4) while still doing better than NB 4-grams actually decreased in accuracy over in NB(3). The main reason here is the poor estimates for the probability distribution. We are estimating 20^4 160,000 probabilities per class. And we have 5 classes.

Table 2, Kinase/Ligase dataset and Table 3, Kinase/Ligase/Isomerase/Helicase dataset show similar results. When increasing k from 2 to 3 we get a dramatic improvement with NB(3) over NB(2), over 5% in Table 2 and nearly 15% in Table 3. As we increase k from 2 to 4 we again see improvement in both datasets. We can get up to a 90.7% accuracy in Table 2 and 82% accuracy in Table 3. Another important observation is that NB(k) always outperformed NB k-grams. It is worth noting that both of these experiments used the same splits when doing cross validation and used the same way (Laplace correction) to estimate the probabilities when building classifiers. So there is no bias in terms of how these two algorithms were trained.

[To recapitulate] The experimental results show the superiority of both NB k-grams and NB(k) over Naive Bayes on all test cases. Furthermore NB(k) ended up being superior in accuracy to NB k-grams on all test cases. In the case of the Kinase dataset

DataSet	NB k-grams			NB(k)			Improvement
	k	Model Size	Acc.	k	Model Size	Acc.	
Kinase	4	160,000	90.4	3	8000	92.7	2.3
K/L	4	160,000	85.9	4	160,000	90.7	4.8
K/L/H/I	3	8000	79.5	4	160,000	82.0	2.5

Table 4: A comparison of the best accuracy for each dataset using both the Naive Bayes with k-grams and NB(k) methods. k refers to the number of dependencies modelled by each algorithm. Model size refers to the total number of probabilities per class computed by each algorithm for the given value of k. Acc. refers to the accuracy. For each algorithm we show the value k for which it obtained the best accuracy on each dataset. Improvement shows the percent improvement that NB(k) had over NB with k-grams.

for NB(4) while still doing better than NB 4-grams did worse than NB(3). This result is probably due to an insufficient amount of data available for the estimation of the probabilities.

Table 4 reports the best k size for each of the two methods. In this table model size refers to the overall alphabet size used by the algorithm or equivalently how many probabilities per class we estimate. It is computed by taking the alphabet size and raising it to the k-th power. For example an alphabet size of 20, our amino acid alphabet, with a k size of 4 would have a $20^4 = 160,000$ letter alphabet. This is also, again the number of probabilities per class needed to be estimated for each algorithm. Note that in the case of Naive Bayes we estimate 20 probabilities per class. Also note that the estimations of these big probabilities are very fast as long as the table fits into memory (it involves one addition and one multiplication per entry into the table). Table 4 shows that k equal to 3 or 4 is usually optimal in terms of classification.

4 Summary and Discussion

To recapitulate, we presented two methods that relax the independence assumption of Naive Bayes by modelling interaction among close neighbours in the sequence. The two methods model k-wise dependency among successive elements in sequence data. Despite the modeling accuracy gain, the two proposed algorithms maintain the “one pass through the data only” training property of Naive Bayes.

We put to test our methods on three protein function prediction problems with functional classes extracted from GO (the Gene Ontology, [5] and corresponding sequences extracted from SWISSPROT [2]. The proposed methods compared favorably with Naive Bayes. Furthermore NB(k) showed (as expected) the best performance in terms of accuracy on all test cases (over NB k-grams). NB(k) is a natural generalization of Naive Bayes (for k=1) and it constitutes an end in itself. That is, under the stationarity assumption it is the most complete and correct probabilistic model of k-wise interaction among k successive elements of the sequence.

The improved modelling accuracy and also the demonstrated improvement in test

accuracy for increased values of k need to be taken with a grain of salt. That is they will hold, under the proviso that we have enough data to fit the model probabilities.

We have also shown the equivalence of the NB(k) undirected model with the Markov Model of order $k-1$. This contributes to dispelling a misconception that undirected models (i.e., random fields) are more suitable than directed models such as Markov Models for modelling protein sequences since the order in the protein sequence has a spatial nature rather than a temporal one as it is the case with Markov Models. While we do not dispute the spatial nature of protein sequences, we show that both the undirected (spatial = NB(k)) and the directed models (temporal = Markov($k-1$) Models) associated with k -wise dependency yield the same probability distribution. Additionally NB(k), which is not incidentally also our best performer, represents an end in itself since it is the most complete undirected graphical model for k -grams modelling.

Some directions for future work include: Studying the effect of varying the alphabet size on the resulting accuracy. More in-depth examination of the results, addressing questions such as: Are NB(k) picking out probabilities similar to those of multiple sequence alignment motifs? A more in-depth study using a wider range of datasets such as Intrusion Detection, ...

References

- [1] www.cs.iastate.edu/~andorfc/pcdatasets.html.
- [2] Bairoch A. and Apweiler R. The swiss-prot protein sequence database and its supplement trembl. *Nucleic Acids Res.*, (28):45–48, 2000.
- [3] Dobbs D. Andorf, C. and V. Honavar. Discovering protein function classification rules from reduced alphabet representations of protein sequences. In *Proceedings of the Conference on Computational Biology and Genome Informatics*. Durham, North Carolina, 2002.
- [4] Eugene Charniak. *Statistical Language Learning*, Cambridge: 1993. MIT Press, 1993.
- [5] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genet.*, (25):25–29, 2000.
- [6] Dawid A.P. Lauritzen S.L. Spiegelhalter D.J. Cowell, R.G. *Probabilistic Networks and Expert Systems*. Springer, 1999.
- [7] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on "Learning for Text Categorization"*, 1998.
- [8] Gregory R. Grant Warren J. Ewens. *Statistical Methods in Bioinformatics*. Springer, 2001.