

RBNBC: Repeat Based Naive Bayes Classifier for Biological Sequences

Pratibha Rani
Center for Data Engineering
IIIT Hyderabad, India
pratibha_rani@research.iiit.ac.in

Vikram Pudi
Center for Data Engineering
IIIT Hyderabad, India
vikram@iiit.ac.in

Abstract

In this paper, we present RBNBC, a Repeat Based Naive Bayes Classifier of bio-sequences that uses maximal frequent subsequences as features. RBNBC's design is based on generic ideas that can apply to other domains where the data is organized as collections of sequences. Specifically, RBNBC uses a novel formulation of Naive Bayes that incorporates repeated occurrences of subsequences within each sequence. Our extensive experiments on two collections of protein families show that it performs as well as existing state-of-the-art probabilistic classifiers for bio-sequences. This is surprising as it is a pure data mining based generic classifier that does not require domain-specific background knowledge. We note that domain-specific ideas could further increase its performance.

1 Introduction

A critical problem in biological data analysis is to classify bio-sequences based on their features and functions [19]. Predicting the family of an unclassified sequence reduces the time and cost required for performing lab experiments to determine its functions and structure as sequences belonging to the same family have similar characteristics.

In this paper, we propose a data mining based, simple but effective solution, which does not require domain knowledge, called *Repeat Based Naive Bayes Classifier* (RBNBC)—a new Bayesian classifier which is able to incorporate *repeats* of subsequences within a sequence. A direct implementation of *Naive Bayes*, which is a surprisingly successful classifier for many application domains, will not work well for bio-sequences. We have adapted it to work for bio-sequences. Our algorithm drastically improves the accuracy from 32% (for the direct *Naive Bayes*) to 98%.

RBNBC has the following desirable features, which can be incorporated in any *Bayesian* classifier: (1) It uses a novel formulation of *Naive Bayes* to incorporate repeated occurrences of subsequences within each sequence of a

family. (2) Unlike direct *Naive Bayes*, it works for a *nonuniform feature set* (where all features are not present in each class). (3) It uses a *bit-vector based optimization* that drastically reduces the time required to extract frequent subsequences. (4) It uses an *entropy based feature selection* method to find the discriminating features for a class and to reduce the number of irrelevant features. (5) Being a *Bayesian* classifier, it is scalable with the database size and with the number of classes. (6) It does not require domain knowledge based ideas such as alignment based similarity (like in FASTA [17] and BLAST [15]), complex feature extraction or data transformation (like in SVM).

2 Problem Definition

Given a training dataset $D = \{F_1, F_2, \dots, F_n\}$ as a set of n families, where each *family*¹ is a collection of sequences, the goal of the classifier is to label a query sequence S with family F_i for which the posterior probability $P(F_i|S)$ is maximum. *Bayes formula* allows us to compute this probability from the prior probability $P(F_i)$ and the class-conditional probability $P(S|F_i)$ as follows:

$$P(F_i|S) = \frac{P(S|F_i)P(F_i)}{P(S)} \quad (1)$$

Since $P(S)$ is common for all families, it is left out. $P(F_i)$ is trivial to compute as relative frequency of family F_i in D . Hence the classification problem reduces to the correct estimation of $P(S|F_i)$ from D .

3 Estimating Feature Probabilities

Starting with a few definitions, in this section, we describe our method of estimating feature probabilities.

Definition 1. *Sequence Count* of a feature X_j in family F_i is the number of sequences of family F_i in which feature X_j is present at least once.

¹In this paper, the terms “family” and “class” are used interchangeably.

Definition 2. *Repeat Count* of a feature X_j in family F_i is the sum of the number of occurrences of that feature in each sequence of the family.

Definition 3. A feature X_j is *frequent* in family F_i iff

$$\text{Sequence Count of } X_j \text{ in } F_i \geq \sigma$$

where σ is the *MinsupCount* for family F_i calculated using the user given support threshold *minsup* and total number of sequences N_i in family F_i as: $\sigma = N_i \times \text{minsup}$.

Definition 4. Let $F = \{X_1, X_2, \dots, X_{|F|}\}$ be the set of all frequent features extracted from family F_i . Feature $X_j \in F$ is *maximal frequent* in family F_i iff

$$\nexists X_k \in F \text{ such that } X_k \supset X_j.$$

Since *Sequence Count* does not account for multiple occurrences of X_j in a sequence, we present the following method using *Repeat Counts* to estimate the probability $P(X_j|F_i)$ of a feature X_j in a family F_i . In our study we found that *Repeat Count* results in better accuracy as it uses all the occurrences of a feature.

1. Find the number of *slots* available for X_j in family F_i (containing N_i sequences).

- If we consider that the features may overlap:

$$\text{slots}_{ij} = \sum_{k=1}^{N_i} [\text{length of } S_k - \text{length of } X_j + 1] \quad (2)$$

- If we consider non overlapping features:

$$\text{slots}_{ij} = \sum_{k=1}^{N_i} \left\lfloor \frac{\text{length of } S_k}{\text{length of } X_j} \right\rfloor \quad (3)$$

2. Find the probability of feature X_j in family F_i as:

$$P(X_j|F_i) = \frac{\text{Repeat Count of } X_j \text{ in } F_i}{\text{slots}_{ij}} \quad (4)$$

Equations 2 and 3 find the total slots for feature X_j in family F_i by summing the available slots in each sequence S_k of F_i . Next, the feature probability is estimated as the fraction of times X_j actually occurs over the slots.

4 Problems with Naive Bayes

For a query sequence S represented as a feature vector $\vec{X} = \{X_1, X_2, \dots, X_m\}$, *Naive Bayes* (NB) assumes strong independence among the features and estimates

$P(S|F_i)$ by multiplying the feature probabilities. It calculates the posterior probability of each family as:

$$P(F_i|S) = P(F_i) \prod_{j=1}^m P(X_j|F_i) \quad (5)$$

Along with the advantages of simplicity and speed, the NB classifier has the merit that even in cases where the independence assumption is not strictly satisfied it performs surprisingly well on a variety of datasets [5, 11].

One of the problems with the above formulation is that when very small feature probability values are multiplied in Equation 5, the product can go below the available minimum number range of the processor. An appropriate scaling factor or log scaled formulation is used to avoid this problem. The second problem is discussed below.

4.1 Problem of Features not Represented in the Training Data

Since calculation of $P(X_j|F_i)$ is based on the presence of X_j in the *training data* of class F_i , a problem can arise if X_j is completely absent in the training data of class F_i . The absence of X_j is quite common because training data is typically too small to be comprehensive, and not because $P(X_j|F_i)$ is really zero. Evidence based on other subsequences of query sequence S may point to a significant presence of S in F_i . Due to this problem, the existing NB formulation (Equation 5) cannot be applied directly on bio-sequences when frequent subsequences are used as features. Known solutions are:

1. Use a *nonuniform* feature vector, i.e., use different feature vectors of query sequence S for each class which include only those features of S which are present in that class. Then set $P(S|F_i) = 0$ only if *none* of the features of S is present in class F_i .

This solution has a drawback: Classes with *more* matching features of S could be computed as having *less* posterior probability due to the multiplication of more feature probabilities whose values are always less than one. This results in wrong classification.

2. Incorporate sample-correction such as the *Laplace correction factor* [11] in all feature probabilities. It is infeasible for datasets with a large feature set.
3. If a feature value does not occur in a given class, then set its probability to $\frac{1}{N}$, where N is the number of examples in the training set [11].

We experimented with two models of the NB classifier for bio-sequences—model A using solution (1) and model B using solution (3)—and found that model B performed better than model A. In RBNBC we use another solution described in Section 5.3, which outperformed both A and B models.

5 The RBNBC Classifier

The RBNBC classifier runs in three phases:

1. **Feature Extraction:** This is the training phase in which first *maximal frequent subsequences* are extracted as features from each family and stored with their *Repeat* and *Sequence Counts*. Then for each family, the *Repeat* and *Sequence Counts* for maximal features from other families, which are not maximal in this family, are also stored. This is to ensure that all families share the same feature set.
2. **Feature Selection:** The extracted feature set is pruned in this phase using an *entropy based selection criterion*. This results in a smaller set of features and their *Repeat* and *Sequence Counts* within each family. The feature extraction and selection phases are executed only once to train the classifier. After this the original dataset is no longer required and the classifier works with the feature set left after pruning.
3. **Classification:** This phase is executed for labeling a query sequence with the family having the maximum posterior probability. The classifier first separates all the features belonging to the query sequence from the available feature set from the second phase. It then uses this feature set to find the posterior probability of each family and outputs the one with the maximum posterior probability.

5.1 Feature Extraction

Many sophisticated feature mining algorithms [8, 13] exist for bio-sequences, but we have used simple features, avoiding the need for complex data transformations and domain knowledge. We believe that frequent subsequences capture everything that is significant in a collection of sequences. This assumption has borne out well in the results.

Since the number of extracted frequent features increases exponentially as *minsup* decreases, to reduce the feature set we have opted to use *maximal frequent subsequences* as features. There may be some loss in information by using maximal frequent subsequences as features. However, they satisfy the following criteria set by [13], which are necessary for features of any classifier: (1) Significant features: we ensure this by considering only *frequent* features (i.e., $Sequence\ Count \geq MinsupCount$). (2) Non-redundant Features: we ensure this by using *maximal* frequent subsequences as features. (3) Discriminative Features: for ensuring this, we use the *entropy based selection* criteria described in Section 5.2 after extraction of features.

We extracted maximal frequent subsequences using an *Apriori*-like method using the same minimum support

threshold for all families. To extract all possible features, we set *maxlen*—maximum length of the feature—as the length of the largest sequence of the training set.

5.1.1 Bit-Vector based Optimization of Frequent Subsequence Extraction

We have optimized the time consuming and memory intensive frequent subsequence extraction process by avoiding extraction of infrequent subsequences by storing information of their location in a bit-vector. This optimization proved to be very effective and reduced the feature extraction time from days to hours.

```
for each sequence  $S$  of the family:
  Initialize a bitvector  $B_S$  of '1's of length equal to that of  $S$ 
for  $l = 1$  to  $maxlen$ : # length of subsequences to consider
  for each sequence  $S$  of the family:
    for each position  $p$  in  $B_S$ :
      if  $p = 1$ :
        extract feature  $X$  of length  $l$  starting from position  $p$ 
        increment counters measuring occurrences of  $X$ 
        set  $p = 0$ 
    for each extracted feature  $X$  of length  $l$ :
      if  $Sequence\ Count\ of\ X \geq MinsupCount$ :
        for each sequence  $S$  containing  $X$ :
          for each occurrence  $q$  of  $X$  in  $S$ :
            set  $p = 1$  in  $B_S$  corresponding to the position of  $q$ 
```

Figure 1: Bit-Vector based optimization.

The procedure first initializes a bit-vector of '1's for each sequence in a family which is of the same length as the sequence. Then it starts extracting frequent subsequences of length one and iteratively proceeds to longer subsequences. The presence of a '1' in a bit-vector indicates that a frequent subsequence of length l can be extracted from the corresponding position in the sequence. The presence of a '0' indicates that the subsequence of length l at the corresponding position in the sequence is *infrequent*. It follows that subsequences longer than l from this position will also be infrequent. Hence the bit will remain '0'.

In the first phase of each iteration, candidate subsequences of length l are counted. In the second phase, the bit positions corresponding to frequent subsequences of length l are set to '1', to be considered in the next iteration.

5.2 Feature Selection

As is typical of frequent pattern mining, the feature extraction phase outputs *too many* features especially for low *minsup*. Entropy based criteria [11] like *information gain* and *gain ratio* have been used to tackle this problem by selecting only the important features.

We select discriminating features [13] for each family based on low values of $H(D|X_j = present)$, i.e., *entropy*

of the dataset in the presence of a feature, defined as:

$$H(D|X_j = \text{present}) = - \sum_{i=1}^N [P(F_i|X_j = \text{present}) \times \log[P(F_i|X_j = \text{present})]]$$

where

$$P(F_i|X_j = \text{present}) = \frac{\text{Sequence Count of } X_j \text{ in } F_i}{\sum_k \text{Sequence Count of } X_j \text{ in } F_k}$$

Analysis of this criterion gives the following observations:

(1) $H(D|X_j = \text{present}) = 0$ when a feature X_j is present in one and only one family. (2) $H(D|X_j = \text{present})$ is higher when a feature X_j is present in all families.

For selecting features we compare a user-given threshold H_{th} with the calculated value of $H(D|X_j = \text{present})$, and select all the features satisfying the criteria $H(D|X_j = \text{present}) \leq H_{th}$ while pruning the others.

5.3 Classification

RBNBC uses a very simple assumption to handle the problem of *zero probabilities* and the problem arising from the use of a *nonuniform feature set* (discussed in Section 4.1). It assumes that the probability of any feature to be present in any family is *never zero*. So for the features of other families which are not present in a given family, it uses a correction probability ϵ , which is the *minimum possible feature probability computed using repeat counts*.

$$\epsilon = \frac{1}{\text{Sum of the lengths of sequences of the largest family}} \quad (6)$$

For classifying a query sequence S , RBNBC finds the uniform feature set F , which is the set of features present in S , collected from all families. It then uses Equation 4 for finding probabilities of features present in a family and uses ϵ as the probability for features not present in that family. It uses these probabilities to compute the posterior probability of all families using Equation 5. Finally, it classifies the query sequence into the family with the largest posterior probability. The pseudo-code is shown in Figure 2.

6 Experiments and Results

We have used two collections of protein families to evaluate the performance of RBNBC. The first collection is a large collection of 8435 *G protein-coupled receptors* (GPCRs) arranged in 13 families called superfamilies, taken from the March-2005 Release 9.0 of GPCRDB [9] (<http://www.gpcr.org/7tm>). This collection is a skewed dataset with the peculiar property that the largest family class A contains 71% of the total sequences of the dataset.

1. Find features from all families which are present in the query sequence S and make *uniform feature set* $F = \{X_1, X_2, \dots, X_m\}$.
2. **for each** family F_i do
 - (a) **for each** feature $X_j \in F$ do
 - if** $X_j \in F_i$: use Equation 4 to compute $P(X_j|F_i)$
 - else** : set $P(X_j|F_i) = \epsilon$
 - (b) Use Equation 5 to compute $P(F_i|S)$.
3. Find family F_k having the largest value of $P(F_i|S)$.
4. Classify S into F_k .

Figure 2: Classification phase of RBNBC.

The second collection of 2097 proteins arranged in 26 families was obtained from the Feb-2008 Release 55.0 of SWISSPROT [4] using the list of SWISSPROT protein IDs obtained from Pfam [2] version 22.0. This set of families has already been used in [3, 6, 7], so using it allows a direct comparison with their methods. We should note that, due to the constant refinement in the topology of the Pfam database, there are significant differences in the families common to the two collections.

All the classifiers were assessed based on the standard *Accuracy* measure, which gives the percentage of correctly classified test sequences. For comparison with the *Naive Bayes* classifier, we implemented models A and B described in Section 4.1. In all experiments we used the same set of features for all the classifiers.

Table 1 gives the number of features before and after pruning (Section 5.2), for different *minsup* values, used in the experiments with the GPCRDB dataset. For 5% *minsup* we used threshold $H_{th} = 0.0$ and for other *minsup* values we used $\frac{1}{2}H(D)$. The table shows that the *entropy based selection* reduced the number of features by 40% for low *minsup* values.

Table 2 presents the results of experiments using the set of features after pruning. This table shows the family-wise and average accuracies of the classifiers with corresponding minimum support values. The *Size* column gives the number of sequences of each family before dividing it into training and test sets. Table 3 summarizes the results of experiments done on the classifiers with different *minsup* values using the feature set before and after pruning.

7 Discussion

From the results of Table 2, it is evident that the NB model A, which uses a *nonuniform feature set*, is biased towards the largest family; but RBNBC is able to break the biasing effect of large families and so performs very well on the skewed dataset also.

Table 2. Classification results of NB models (A and B) and RBNBC for the collection of GPCRs from the GPCRDB dataset divided into training set of 7587 and test set of 848 GPCRs

Family Name	Size	Model A (%)	Model B (%)	RBNBC (%)
Class A	6031	83.2	57.5	98.35
Class B	309	0.0	96.7	93.55
Class C	206	0.0	100.0	100.0
Class D	65	0.0	66.7	66.67
Class E	10	0.0	100.0	100.0
Ocular albinism proteins	8	0.0	100.0	100.0
Frizzled family	130	0.0	92.3	92.31
Insect odorant receptors	236	0.0	4.2	37.5
Plant Mlo receptors	52	0.0	100.0	100.0
Nematode chemoreceptors	755	0.0	42.7	90.67
Vomeranosal receptors	286	10.3	82.7	89.65
Taste receptors T2R	237	0.0	87.5	95.83
Class Z Archaeal opsins	110	9.0	100.0	100.0
Average Accuracy (%)		60.14	60.38	95.17
<i>minsup</i> (%)		50	5	5

Table 1. Number of features for GPCRDB

<i>minsup</i> (%)	Without Pruning	With Pruning
5	12087	7081
10	6663	5069
30	1491	1305
50	524	489

The results of Table 3 show that performance of NB models A and B improves when feature set is pruned, while RBNBC outperforms them for all *minsup* values for both pruned as well as non pruned feature set. This indicates that the performance of RBNBC is not dependent on the pruning phase, so we can conclude that RBNBC can perform well on datasets with large feature sets, whereas the other NB classifiers can not. These results also indicate that *Repeat Count* gives a better model of the sequence families than *Sequence Count*.

Comparison of the results obtained from experiments on the Pfam dataset with the published results of the C classifier of [7] and PST [3] on the same dataset indicates that performance of RBNBC is comparable to the performances of C and PST classifiers. Due to lack of space, we have not included the results in this paper. The complete set of results is available in [18].

The families of the Pfam dataset were constructed on the basis of sequence similarity using HMM [2], while the families of GPCRDB were constructed manually on the basis of the function of proteins [9], so similarity among sequences

Table 3. Classification results for different minsup on the test set from GPCRDB

Classifier	<i>minsup</i> (%)	Without Feature Pruning	With Feature Pruning
		Accuracy (%)	Accuracy (%)
NB Model A	5	0.83	17.57
	10	0.12	0.94
	30	0.24	14.27
	50	32.08	60.14
NB Model B	5	27.21	60.38
	10	16.51	29.60
	30	8.84	12.15
	50	9.08	9.20
RBNBC	5	98.0	95.17
	10	97.88	94.46
	30	92.81	94.69
	50	81.49	87.03

of a family of the Pfam dataset is higher than the GPCRDB dataset. Due to this, classifiers like PSTs which use *Variable Length Markov Model* perform very well on the Pfam dataset. Comparison of the results obtained on GPCRDB and Pfam dataset shows that although GPCRDB is larger in size and skewed than Pfam, RBNBC performs better on it than Pfam. So RBNBC can be better than other sequence classifiers for large and skewed datasets with less similarity

among sequences.

Classifiers like PSTs [3], SMTs [6], *Bayesian* classifier of [1] and the C classifier [7] require parameters other than *minsup*, such as feature length, sequence length, etc. to be supplied by the user for the feature extraction process. Performance of a classifier is very sensitive towards these parameters. RBNBC does not require any parameter other than *minsup* for the feature extraction process and the parameter required for pruning can be set easily.

8 Related Work

The classifiers of bio-sequences can be broadly divided into the following categories:

Bayesian Classifiers [1, 7, 10]: The authors of [1] propose that the NB classifier can be used for protein classification by representing protein sequences as class conditional probability distribution of *k-grams* (short subsequences of amino acids of length *k*). [7] uses a query-sequence-driven gapped subsequence extraction method [8] to find features which are used in the NB classifier. This method defers the feature extraction phase till classification time. [10] presents a recursive NB classifier RNBL–MN, which constructs a tree of Naive Bayes classifiers, where each individual NB classifier is based on a multinomial event model.

Probabilistic Suffix Tree based Classifiers [3, 6]: A PST [3] is a variable length Markov Model, where the probability of a symbol in a sequence depends on the previous symbols. SMT [6] generalizes PST by incorporating wildcard support (a symbol that denotes a gap of size one and matches any symbol on the alphabet).

HMM based Classifiers [12]: These classifiers use HMM to build a model for each family based on multiple alignment of sequences. They are very complex to implement and the generated models tend to be space inefficient and require large memory.

Similarity based Classifiers [15, 17]: These classifiers compare an unlabeled sequence with all the sequences of the database and assess sequence similarity using sequence alignment methods like FASTA [17] or BLAST [15] and classify a sequence using the *nearest neighbor* approach.

SVM based Classifiers [14, 16]: These classifiers either use a set of features to train SVM or use *kernel* based SVMs alone or with some standard similarity measure like BLAST or with some structural information. They require a lot of data transformation but report the best accuracies.

9 Conclusions

In this paper, we proposed a new *Bayesian* classifier which uses the simplest possible features of sequence families—maximal frequent subsequences— and incorporates

the *Repeat Count* of features present in the query sequence. We proposed simple but effective solutions to handle the problems associated with NB classifiers. Experiments on two datasets demonstrated that RBNBC's performance is comparable to the PSTs and the C classifier; it outperforms the other NB classifiers which use *Sequence Count* and it performs very well even on large and skewed datasets.

References

- [1] C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar. Learning classifiers for assigning protein sequences to gene ontology functional families. In *Fifth Int. Conf. on KBCS*, 2004.
- [2] A. Bateman et al. The Pfam protein families database. *Nucleic Acids Research*, 36(Database-Issue), 2008.
- [3] G. Bejerano and G. Yona. Modeling protein families using probabilistic suffix trees. In *RECOMB*, 1999.
- [4] B. Boeckmann et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1), 2003.
- [5] P. Domingos and M. J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *ICML*, 1996.
- [6] E. Eskin, W. S. Noble, and Y. Singer. Protein family classification using sparse markov transducers. *Journal of Computational Biology*, 10(2), 2003.
- [7] P. G. Ferreira and P. J. Azevedo. Protein sequence classification through relevant sequence mining and bayes classifiers. In *Progress in AI, 12th Portuguese Conf. on AI, EPIA*, 2005.
- [8] P. G. Ferreira and P. J. Azevedo. Query driven sequence pattern mining. In *SBBDB*, 2006.
- [9] F. Horn et al. GPCRDB information system for G protein-coupled receptors. *Nucleic Acids Research*, 31(1), 2003.
- [10] D. Kang, J. Zhang, A. Silvescu, and V. Honavar. Multinomial event model based abstraction for sequence and text classification. In *SARA*, 2005.
- [11] S. B. Kotsiantis and P. E. Pintelas. Increasing the classification accuracy of simple bayesian classifier. In *AIMSA*, 2004.
- [12] A. Krogh et al. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235, 1994.
- [13] N. Lesh, M. J. Zaki, and M. Ogihara. Mining features for sequence classification. In *ACM SIGKDD*, 1999.
- [14] C. Leslie, E. Eskin, and W. Noble. Mismatch string kernels for SVM protein classification. In *Neural Information Processing Systems 15*, 2003.
- [15] D. J. Lipman et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3), 1990.
- [16] K. Marsolo and S. Parthasarathy. On the use of structure and sequence-based features for protein classification and retrieval. In *ICDM*, 2006.
- [17] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. National Academy Sciences USA*, 85(8), 1988.
- [18] P. Rani and V. Pudi. RBNBC: Repeat Based Naive Bayes Classifier for Biological Sequences. Technical report, IIIT/TR/2008/126, IIIT Hyderabad, India, 2008.
- [19] J. T. L. Wang, M. J. Zaki, H. Toivonen, and D. Shasha, editors. *Data Mining in Bioinformatics*. Springer, 2005.