

An approach to overcome imbalance datasets of eukaryotic genomes during the analysis by machine learning technique (SVM)

Mohd. Faheem Khan^{1,2}, Gaurav Chauhan² and A. K. Jaitly¹

¹Department of Plant Sciences, MJP Rohilkhand University Bareilly (U.P) India.

²Bioinformatics Centre, IVRI, Izatnagar, Bareilly-243122 (U.P) India.

kingsfaheem@gmail.com

Abstract

In biology, Support Vector Machines (SVM) is most frequently used tool for the analysis of gene expression, microarray experiments and other biological applications. In human genome dataset, only a small proportion of the DNA sequences represent genes, and the rest do not. In our work, we highlighted the reasons why, particular SVM, fails and what can be done to overcome this.

Keywords: Imbalanced dataset, BioSVM

Introduction

Support Vector Machines (SVM) introduced by Vapnik and its colleagues (Vapnik, 1995) have been very successful in the scientific application. Complex biological data can be more accurately analysed by using SVM in comparison to other machine learning techniques. SVM is the choice to analyse the problem of imbalanced data because of its strong theoretical foundation (Vapnik, 1995) and it performs well with moderately imbalanced data even without any modifications (Akbari *et al.*, 2004). SVM has its strengths in that the final model is only dependent on the support vectors, whereas the rest of the instances are discarded. Its unique learning mechanism makes it an interesting candidate for dealing with imbalanced datasets, since SVM only takes into account those instances that are close to the boundary, i.e. the support vectors. This means that SVM is unaffected by non-noisy negative instances far away from the boundary even if they are huge in number. Another advantage is that even though there may be more majority class support vectors than minority class, because of Karush-Kuhn-Tucker conditions, the weights of the minority class support vectors would be higher, resulting in some offsetting.

But, when faced with imbalanced datasets where the number of negative instances far outnumbers the positive instances, the performance of SVM drops significantly (Wu & Chang, 2003). There are many applications in which instances belonging to one class are heavily outnumbered by instances belonging to another class. Such datasets are called imbalanced datasets, since the class distributions are not evenly balanced. Examples of these imbalanced datasets in biology include the eukaryotic genome datasets.

We used the human genome dataset. It is a good example of eukaryotic genome because it satisfies all eukaryotic genomic properties therefore it is a good model genome for analyzing this type of problem. The major difference between the properties of these datasets is that the imbalance ratio for the human genome dataset

is very large, but we know what that ratio is at the time of training. Both the training and test sets are derived from the same distribution (the human genome), so the imbalance ratio in the train and test sets would be the same. The imbalance ratio is expected to change dynamically and any algorithm needs to adapt to that change. In this work, we present techniques to deal with these changes.

Method

The genome was collected from FTP NCBI site from internet. The PERL program was written to extract the start and stop codons from the genome. The PERL is most frequently used programming language in bioinformatics therefore we used the BIOPERL program for the analysis. The scripts takes input one by one chromosome from whole genome and return the number of ATG (start codon), TAG, TGA, and TAA (stop codon). This data was used for further analysis to analyze the imbalance of datasets as described below.

Effects of Imbalance on SVM

In order to combat the effects of imbalance, we need to understand exactly why SVM's performance deteriorates with high imbalance ratios. To do that, we need to look at how soft margin SVMs work. Given a set of labeled instances $X_{train} = \{x_i, y_i\}$ $N_i=1$ and a kernel function K ; SVM finds the optimal a_i for each x_i to maximize the margin γ between the hyper plane and the closest instances to it. The class prediction for a new test instance x is made through:

$$\text{sign} \left(f(x) = \sum_{i=1}^n y_i \alpha_i K(x, x_i) + b \right) \quad (1)$$

Where b is the threshold. 1-norm soft-margin SVMs minimize the primal Lagrangian:

$$L_p = \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i (w \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^n r_i \xi_i \quad (2)$$

Where $a_i \geq 0$ and $r_i \geq 0$ (Cristianini & Shawe-Taylor, 2000). The penalty constant C represents the trade-off between the empirical error ξ and the margin. In order to meet the Karush-Kuhn-Tucker (KKT) conditions, the value of a_i must satisfy:

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^n \alpha_i y_i = 0 \quad \text{-- (3)}$$

Reasons for performance loss with imbalanced data

Weakness of soft-margins: The most significant factor for the loss in performance of SVMs is the weakness of soft margin SVMs. Mathematically, we can see from eq. 2 that minimizing the first term on the right hand side $\|w\|^2/2$, is equivalent to maximizing the margin γ , while minimizing the second term $C \sum \xi$ minimizes the associated error. The constant C specifies what trade-off we are willing to tolerate between maximizing the margin and minimizing the error. If C is not very large, SVM simply learns to classify everything as negative because that makes the margin the largest, with zero cumulative error on the abundant negative examples. The only trade-off is the small amount of cumulative error on the few positive examples, which does not count for much. This explains why SVM fails completely in situations with a high degree of imbalance.

Positive points lie further from the ideal boundary. Wu and Chang (Wu & Chang, 2003) point out this factor as one source of boundary skew. They mention that the imbalance in the training data ratio means that the positive instances may lie further away from the "ideal" boundary than the negative instances. This is illustrated with the example that if we were to draw n randomly chosen numbers between 1 to 100 from a uniform distribution, our chances of drawing a number close to 100 would improve with increasing values of n , even though the expected mean of the draws is invariant of n . As a result of this phenomenon, SVM learns a boundary that is too close to and skewed towards the positive instances.

Imbalanced Support Vector Ratio: Another source of boundary skew according to Wu and Chang (Wu & Chang, 2003) is the imbalanced support vector ratio. They found that as the training data gets more imbalanced, the ratio between the positive and negative support vectors also becomes more imbalanced. They hypothesize that as a result of this imbalance, the neighborhood of a test instance close to the boundary is more likely to be dominated by negative support vectors and hence the decision function is more likely to classify a boundary point negative. We would like to point out however, that because of the KKT conditions in eq. 3, the sum of the a 's associated with the positive support vectors must be equal to the sum of the a 's associated with the negative support vectors. Because there are fewer positive support vectors with correspondingly fewer a 's, each positive support vector's a must be larger than

the negative support vector's a on average. These a 's act as weights in the final decision function (eq. 1) and as a result of larger a 's the positive support vectors receive a higher weight than the negative support vectors which offsets the effect of support vector imbalance to some extent. This shows why SVM does not perform too badly compared to other machine learning algorithms for moderately skewed datasets.

SVM and biological system

To illustrate the degree of imbalance encountered in bioinformatics, we use the example of trying to identify parts of genes in eukaryotic DNA. Human DNA consists of 23 pairs of chromosomes. The Human Genome Project sequenced these chromosomes and we now have almost the entire DNA sequence of 3 billion base pairs. However, not all of the 3 billion base pairs in DNA code for proteins. In fact, the vast majority of DNA does not code for proteins. Portions of DNA that code for proteins are called genes. Genes have several components:

1st and 2nd keys (promoters): These regions aid in initiating gene expression (i.e. protein production). They may be absent.

5' UnTranslated Region (UTR): This is the point where mRNA transcription begins.

Start (Translation Initiation Site - TIS): This is the position where translation begins i.e. proteins start to be coded from this site.

Exon: This is region of DNA that codes for the protein.
(IInd key) (Ist key)

5'---TF---Promoter---5'UTR start Exon--Intron--Exon--
Intron--Exons stop 3'UTR--3'

Intron: This region of DNA is interspersed between two exons and it does not code for proteins. It is spliced out from the mRNA before translation begins.

Donor Site (not shown): The junction where an exon meets an intron.

Acceptor Site (not shown): The junction where an intron meets an exon. Together the donor and acceptor sites are called *splice sites*.

Stop: This is the termination site where protein synthesis stops. It always consists of one of three possible codons, TGA, TAA or TAG.

3' UTR: This is the region after the stop codon that does not code for a protein but it forms the tail of the mRNA that is produced by the gene.

Work done

The problem of gene finding is to identify the locations of each of these components on the DNA sequence. In our example, we only identified the start and stop codons. Almost all start codons have the sequence ATG, while all stop codons consist of either TAA, TAG or TGA. But not all ATG sequences are start codons and not all TAA, TAG and TGA sequences are stop codons or else the problem would become trivial. We counted the occurrence of ATG in the entire human genome and

found it to be approximately 104 million. Note that if the DNA sequence was random than ATG would occur $(1/43) \times 3 \times 10^9 = 47$ million times. By contrast, there are an estimated 23,000 confirmed and unconfirmed genes that have ATG as the start codon. So assuming 23,000 genes, the estimated degree of imbalance in predicting when an ATG sequence is start codons (henceforth called positive instances), and when it is not (negative instances) is:

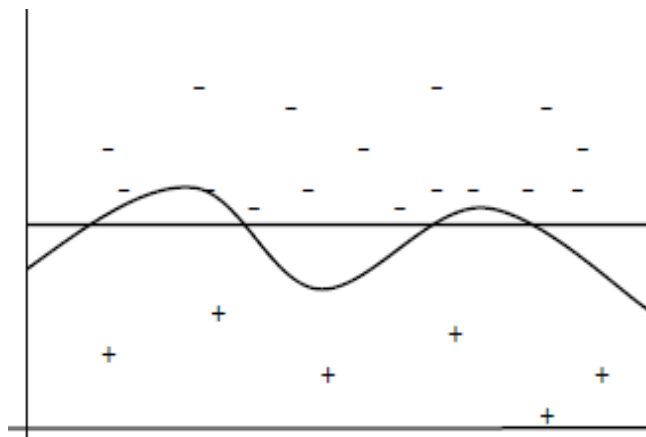
ATG is start codons: ATG is not start codons
23, 0000 : 104,000,000 (1: 4,522)

Looking at the problem of identifying stop codons independently of other predictions poses an even greater degree of imbalance. We have found that there are about 300 million TAG, TAA or TGA sequences in the human genome. Only 23,000 of them are suspected to be actual stop codons. The imbalance ratio for stop codons is therefore:

23,000: 300,000,000 (1: 13,043)

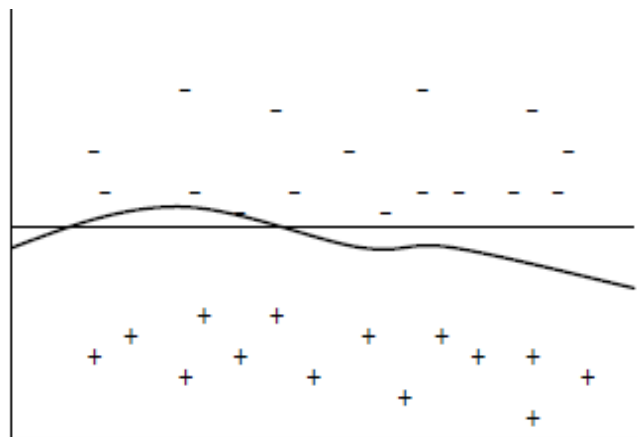
Unfortunately, ordinary machine learning algorithms are incapable of handling this extremely high degree of imbalance. Since an imbalance ratio of over 1:1000 is well beyond the performance capabilities of any ML algorithm, we decided to generate the TIS data from the human genome with an imbalance of 1:100 for our current scheme. Even this ratio causes most ML algorithms to perform very poorly. This ratio is still much higher than the P&N dataset which has an imbalance ratio of only 1:3. Our first strategy was to construct a dataset containing sequences from the human genomic data and then use it to generate several candidate features for our algorithm. We then used feature selection algorithms to select the best attributes among them. This technique was originally proposed by Zeng et al. (Zeng et al., 2002). To begin with, we randomly chose known ATG TIS sites from the NCBI database for our positive examples. Then we randomly picked ATG sites from the

Fig. 1. The learned boundary (curved line) in the input space closely follows the distribution of the positive instances. The ideal boundary is denoted by the horizontal line.



genome that are not known to be TIS sites, for our negative examples. We maintained a ratio of 1:100 for positive to negative examples. A window of 200 nucleotides was chosen for every example, running from 100 bps upstream of the ATG to 100 bps downstream of the ATG. This set constituted our raw dataset. From this raw dataset, we generated several features. Every position in the raw data was used as a candidate feature. In addition, we generated the frequency of occurrence of all possible monomers, dimers, trimers, all the way up to hexamers that lie upstream of the ATG and also for those that lie downstream of the ATG. This gave us a total of 11120 features. Then we ran several different feature selection algorithms on this large set of attributes to determine the top attributes. We ran the Correlation Feature Selection (CFS) algorithm, which prefers those set of attributes that have a high correlation with the class label, but low correlation among themselves, and also Information Gain, Gain Ratio, and chi-square test. By observing their results, we were able to choose the top 15 of the 11120 attributes, which were found to be the following (in order of importance): dn-CG, dn-TA, dn-AT, up-AT, up-CG, dn-GC, dn-G, up-TA, dn-CGG, up-CGG, dn-T, dn-ATT, pos -3, pos -1, pos +4, where dn-CG means the frequency of occurrence of CG downstream of the ATG, and up-CG means the frequency of CG upstream of the ATG, pos -3 means the nucleotide at position -3. Although we found pos -3, pos -1 and pos +4 to be the most important positions, the relevance score for these was much lower than the relevance score for the frequency counts, but we included them in our experiments nevertheless. It should also be noted that these positions correspond to the Kozak consensus sequence (Kozak, 1996). Our final dataset consisted of these 15 selected features. We used a similarly generated separate test set for evaluation.

Fig. 2. After over sampling, the positive instances are now more densely distributed and the learned boundary (curved line) is better defined



overcome some of the problems mentioned above i.e imbalanced datasets; the learned boundary is too close to the positive instances. We need to bias SVM in a way that will push the boundary away from the positive instances. Veropoulos et al. (Veropoulos et al., 1999) suggest using different error costs for the positive (C^+) and negative (C^-) classes. Specifically, they suggest changing the primal Lagrangian (eq. 2) to:

$$L_p = \frac{\|w\|^2}{2} + C^+ \sum_{\{i|y_i=+1\}} \xi_i + C^- \sum_{\{j|y_j=-1\}} \xi_j - \sum_{i=1}^n \alpha_i [y_i (w \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^n r_i \xi_i \quad \text{-- (4)}$$

The constraints on α_i then become:

$$0 \leq \alpha_i \leq C^+ \text{ if } y_i = +1 \text{ and } 0 \leq \alpha_i \leq C^- \text{ if } y_i = -1 \quad \text{-- (5)}$$

Furthermore, we note that $\xi_i > 0$ only when $\alpha_i = C$ (Liu et al., 2004). Therefore non-zero errors on positive support vectors will have larger α_i while non-zero errors on negative support vectors will have smaller α_i . The net effect is that the boundary is pushed more towards the negative instances. However, a consequence of this is that SVM becomes more sensitive to the positive instances and obtains stronger cues from the positive instances about the orientation of the plane than from the negative instances. If the positive instances are sparse, as in imbalanced datasets, then the boundary may not have the proper shape in the input space as illustrated in Fig.1.

The solution we adopted to remedy the problem of sparse positive instances was to generate several synthetic minority instances, in line with Chawla et al's technique (Chawla et al., 2002). We repeatedly randomly selecting two neighboring positive instances using the Euclidean distance measure and then generated a new instance that lies somewhere randomly in between these instances. The underlying assumption was that the space between two positive neighboring instances was assumed to be positive. We found this assumption to hold for our dataset. We found that over sampling the minority class in this way was much more effective than the traditional over sampling technique of generating multiple identical copies of existing minority instances. Simply resampling the minority instances merely overlaps the instances on top of each other and does not help in "smoothing out" the shape of the boundary. We synthetically generated new instances between two existing positive instances which helped in making their distribution more well-defined. After this over sampling, the input space may look like Fig.2.

Synthetic over sampling also alleviates the problem of soft margins, since now the cost of misclassifying minority instances is significant. This, together with higher costs of misclassifying minority instances levels the playing field for both classes. To summarize, our method consists of: Generating and selecting the most relevant features for the problem. Using different error costs for different classes to push the boundary away from the

positive instances and overcome soft margin weakness. Generating synthetic minority instances to make the positive instances more densely distributed in order to make the boundary better defined.

Results

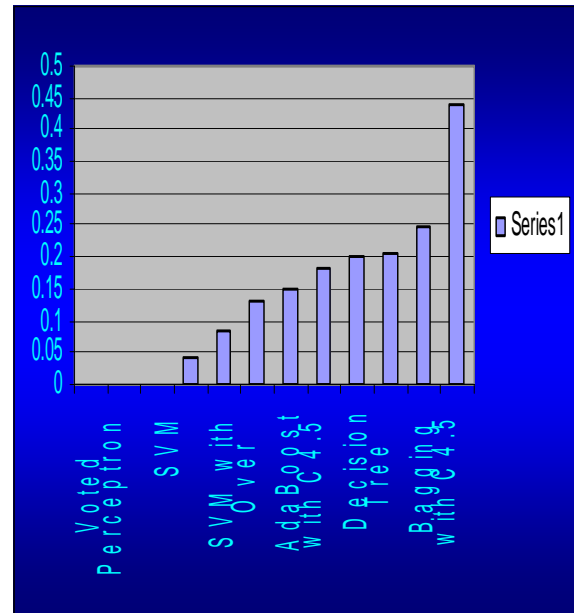
We compared our algorithm with several other standard ML algorithms for predicting TISs in the human genome. We also compared our technique with the common approach of over sampling the minority class or under sampling the majority class in order to reduce imbalance ratio in the dataset prior to training. For evaluation, we used the F-measure as our metric, which is the harmonic mean of the recall and precision. The results are shown below in Table along with plotted in Fig.3. They illustrate how poorly standard ML approaches perform for predicting TISs at the genomic level due to the high imbalance ratio. Our approach improves the performance significantly. By varying the parameters of our algorithm we are able to obtain different recall and precision values. Some examples of recall/precision obtained respectively are: 15%/100%, 29%/95%, 85%/4%. Thus, depending on the application the algorithm parameters can be varied to obtain the desired level of recall vs. precision. While the results are encouraging, the TIS dataset we generated was a watered down version with 1:100 imbalance ratio, compared to the 1:4522 imbalance ratio found in the human genome. The search for algorithms that can deal with such large imbalances is far from over. However, we suggest using heuristics based on domain knowledge to discard those ATG codons which are unlikely to be TISs, instead of directly feeding them into the ML classifier. These heuristics may include ATG codons that are too far from, or too close to stop codons or splice sites. This will reduce the imbalance ratio to some extent and may improve performance.

Discussion

This work discusses the issues faced when trying to train SVM on imbalanced datasets. The main reason why SVM performs poorly for such datasets is because of the weakness of soft margins. Soft margins were introduced in order to make SVM resilient against non separable datasets. The idea was to tolerate some classification error as a trade off for maximizing the margin between the support vectors. But this has an adverse effect when it comes to imbalanced datasets. SVM ends up with a hyper plane that is far from the entire cluster of instances. Any instance that is on the same side of the plane as the cluster is classified as the majority class. Having the hyper plane further from the instances maximizes the margin, at the cost of misclassifying the minority class. But since there are only a few instances of the minority class, the error is rather small and the benefit of larger margins overcomes this. Therefore, everything is classified as the majority class. Some solutions to this

Fig. 3. Plotted F-Measure of various ML algorithms vs. our algorithm on the TIS dataset

Algorithm	F-Measure
Voted Perceptron	0
ZeroR	0
SVM	0
SVM with Under Sampling	0.041
SVM with Over Sampling	0.082
Neural Network	0.133
AdaBoost with C4.5	0.148
3 Nearest Neighbours	0.182
Decision Tree	0.2
Naive Bayes	0.205
Bagging with C4.5	0.25
Our Algorithm	0.44



problem are presented in the work and evaluated against eukaryotic genome. The imbalance ratio in the genome dataset can be as high as 1:4500. This is well beyond the capability of traditional ML algorithms. However, we can use under sampling of the majority class and heuristics to reduce the imbalance ratio. Then we can use the techniques presented in this work to improve the performance of SVM on this dataset. These techniques include generating and selecting good features, using different error costs for majority and minority instances, and generating synthetic minority instances to even out the imbalance.

References

1. Akbani R, Kwek S and Japkowicz N (2004) Applying support vector machines to imbalanced datasets. Proc. 15th Eur. Conf. on Machine Learning (ECML). Pisa, Italy, Sept., Springer-Verlag, Germany. pp: 39-50.
2. Chawla N, Bowyer K, Hall L and Kegelmeyer W (2002) SMOTE: Synthetic Minority Over-sampling Technique. *J. Artificial Intelligence Res.* 16, 321-357.
3. Cristianini N and Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK. ISBN 0521780195.
4. Joachims T (1998) Text categorization with SVM: Learning with many relevant features. Proc. 10th Eur. Conf. on Machine Learning (ECML).
5. Kozak M (1996) Interpreting cDNA sequences: Some insights from studies on translation. *Mammalian Genome.* 7, 563-574.
6. Vapnik V (1995) The nature of statistical learning theory. Springer, NY. ISBN 0387987800.
7. Veropoulos K, Campbell C and Cristianini N (1999) Controlling the sensitivity of support vector machines. Proc. Intl. Joint Conf. on AI. pp: 55-60.
8. Wu G and Chang E (2003) Class-Boundary alignment for imbalanced dataset learning. Proc. ICML 2003 Workshop on Learning from Imbalanced Data Sets II, Washington DC, USA.
9. Zeng F, Yap HC and Wong L (2002) Using feature generation and feature selection for accurate prediction of translation initiation sites. Proc. of 13th Workshop on Genome Informatics, Universal Academy Press. pp: 192-200.