
Augmenting Classifiers with Domain Knowledge: A Comparative Study

America Holloway

Department of Computer Science
University of California, Irvine
Irvine, CA 92617
ahollowa@ics.uci.edu

Sidharth Shekhar

Department of Computer Science
University of California, Irvine
Irvine, CA 92617
shekhars@ics.uci.edu

Abstract

Standard machine learning classification algorithms do not utilize domain knowledge for building or training classifiers. However, intuition suggests that augmenting classifiers with domain knowledge can increase classification performance. We explore three different methods for adding domain knowledge to neural networks and support vector machines.

1 Introduction

Supervised learning techniques, such as neural networks, support vector machines, and generalized linear models, rely only upon labeled training data for classification. Intuition suggests, however, that classification performance can be improved by taking advantage of domain knowledge. Domain knowledge is a formal representation of expert information in a particular field. In this paper we only consider domain knowledge stored in the form of logical statements, in particular, propositional logic. The problem we choose to focus on is that of recognizing splice junctions in DNA sequences.

2 Problem Description

Observations in molecular biology show that large portions of m-RNA are removed before being translated to proteins. The method by which the non coding segments of the precursor m-RNA, called introns, are removed, is known as splicing. The utilized sequences are called exons and the points where splicing occur are known as splice junctions. The recognition of these splice junctions is non-trivial and therefore, it is essential to develop techniques that can identify these junctions without having to look at the entire sequence. The initial domain theory for recognizing splice junctions is shown in Table 1. We present three methods that attempt to use this knowledge base to build classifiers that can accurately predict if the junction in question is either an Exon/Intron boundary, called a donor, or an Intron/Exon boundary, called an acceptor, or neither.

Table 1: Knowledge Base for Splice-Junctions

donor :- @-3=M, @-2=A, @-1=G, @1=G, @2=T, @3=R, @4=A, @5=G, @6=T, not(don-stop).	
don-stop :- @-3=T, @-2=A, @-1=A.	don-stop :- @-4=T, @-3=A, @-2=G.
don-stop :- @-3=T, @-2=A, @-1=G.	don-stop :- @-4=T, @-3=G, @-2=A.
don-stop :- @-3=T, @-2=G, @-1=A.	don-stop :- @-5=T, @-4=A, @-3=A.
don-stop :- @-4=T, @-3=A, @-2=A.	don-stop :- @-5=T, @-4=A, @-3=G.
don-stop :- @-5=T, @-4=G, @-3=A.	
acceptor :- pyr-rich, @-3=Y, @-2=A, @-1=G, @1=G, @2=K, not(acc-stop).	
pyr-rich :- 6 of (@-15=Y, @-14=Y, @-13=Y, @-12=Y, @-11=Y, @-10=Y, @-9=Y, @-8=Y, @-7=Y, @-6=Y.)	
acc-stop :- @1=T, @2=A, @3=A.	acc-stop :- @2=T, @3=A, @4=A.
acc-stop :- @1=T, @2=A, @3=G.	acc-stop :- @2=T, @3=A, @4=G.
acc-stop :- @1=T, @2=G, @3=A.	acc-stop :- @2=T, @3=G, @4=A.
acc-stop :- @3=T, @4=A, @5=A.	acc-stop :- @3=T, @4=A, @5=G.
acc-stop :- @3=T, @4=G, @5=A.	
R :- A. R :- G. Y :- C. Y :- T. M :- C. M :- A. K :- G. K :- T	

The notation @-2 = A references the location, 2 nucleotides before the candidate junction and states that it must be A.

3 The KBANN Algorithm

Knowledge Based Artificial Neural Networks [NTS90, NTS91] use propositional formulas to specify the structure and weights of an artificial neural network (ANN). KBANN requires the formulas to be conjunctive and non-recursive. Disjuncts are treated as multiple formulas.

Let \mathcal{F} be a formula with n mandatory antecedents and m prohibitory antecedents. The consequent and each of the antecedents are represented as units in the ANN with an edge between every antecedent and the consequent. Edges of mandatory antecedents have weight ω , and edges of prohibitory antecedents have weight $-\omega$. The bias of the consequent is set to be $n * \omega - \phi$ where the parameter ϕ is set so that the consequent has activation ~ 0.9 when its antecedents are satisfied and ~ 0.1 otherwise. If \mathcal{F} contains disjuncts, an additional unit is added to the ANN that mediates between the consequent and the disjuncts. The bias of this additional node is set to $1 * \omega - \phi$ since only one of the disjuncts must be true for the consequent to be satisfied. The edge weights and biases are fixed throughout the training of the ANN. As we discovered, it is important to add a small, random number to each weight in order to break the symmetry of the ANN.

As an example, suppose the knowledge base contains two formulas,

$$\begin{aligned} A &\leftarrow B \vee C \wedge D \\ B &\leftarrow \neg E \wedge F \end{aligned}$$

The associated ANN is shown in Figure 1 for $\omega = 3$ and $\phi = 2.3$. Once this initial structure is set, additional edges are added to fully connect adjacent layers. The weights for these edges are randomly initialized, and the ANN is then trained using backpropagation.

4 The EASVM Algorithm

Explanation Augmented SVMs [SD05] use the domain knowledge to generate explanation vectors for each training instance. These explanations contain non-zero values for features

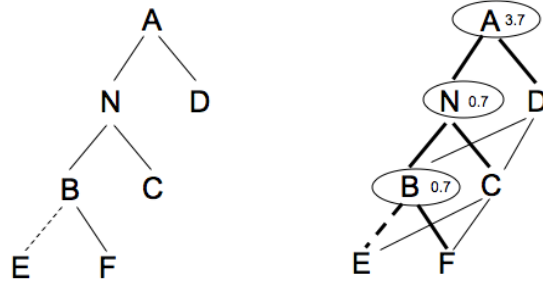


Figure 1: Example of a KBANN

of the data that may be relevant for predicting the class label. Given a training vector x that satisfies all of the antecedents for some rule in the knowledge base, an explanation is created as follows:

$$v_i = \begin{cases} x_i & \text{if the } i\text{th position is an antecedent} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

For a perfect explanation, we require that $w^\top v^i = w^\top x^i$.¹ This forces w to treat the explanation and the training example equivalently since they have the same label for the same reason. However, assuming the domain knowledge to be incomplete and noisy we formulate the SVM with slack as follows

$$\begin{aligned} \min_{w, b, \delta^i} \quad & \frac{1}{2} \|w\|^2 + Q \sum_i \delta^i \\ \text{s.t.} \quad & y^i (w^\top x^i + b) - 1 \geq 0 \quad \forall i \\ & -\delta^i \leq w^\top x^i - w^\top v^i \leq \delta^i \quad \forall i \\ & \delta^i \geq 0 \quad \forall i \end{aligned}$$

5 KSVM

Knowledge-Based SVMs [FMS02] attempt to generalize training examples to training *regions*. These regions, which are entirely contained within one class, arise from the domain knowledge. The SVM then maximizes the margin between the regions as well as the training examples. These regions are encoded as constraints of the form:

$$\forall x \text{ if } Cx \leq c \Rightarrow x \in \text{Class } A+ \quad (2)$$

This gives us,

$$Cx \leq c \Rightarrow x^\top w \geq b + 1 \quad (3)$$

which is equivalent to

$$\text{for a given } (w, b), Cx \leq c, x^\top w < b + 1, \text{ has no solution } x \quad (4)$$

The above statement is implied by the following

$$C^\top u + w = 0, C^\top u + b + 1 \leq 0, u \geq 0, \text{ has a solution } (u, w) \quad (5)$$

¹We use the convention that a superscript i refers to the i th vector in a set, whereas subscript i refers to the i th element of the vector

Thus, for each rule, we add the generated constraints to the SVM. However, since it is likely that we do not have linearly separable data, we add in slack for each of the constraints. Assuming we have k rules for class $A+$ and l rules for class $A-$, the final optimization problem is written as follows

$$\begin{aligned} \min_{w, b, \xi^i, u^t, r^t, \rho^t, v^j, s^j, \sigma^j} & \|w\|_1 + Q \sum_{i=1}^m \xi^i + \\ & \mu \left(\sum_{t=1}^k (r^t + \rho^t) + \sum_{j=1}^l (s^j + \sigma^j) \right) \\ \text{s.t.} & y^i (w^\top x^i + b) + \xi^i \geq 1 \\ & -r^t \leq C^{t\top} u^t + w \leq r^t \\ & c^{t\top} u^t + b + 1 \leq \rho^t \\ & -s^j \leq D^{j\top} v^j - w \leq s^j \\ & d^{j\top} v^j - b + 1 \leq \sigma^j \\ & \xi^i \geq 0, \quad i = 1, \dots, m \\ & u^t, r^t, \rho^t \geq 0, \quad t = 1, \dots, k \\ & v^j, s^j, \sigma^j \geq 0, \quad j = 1, \dots, l \end{aligned}$$

6 Experimental Results

The dataset² contains 3175 DNA sequences. Each sequence is 60 nucleotides long, consisting of 30 nucleotides before the candidate junction and 30 nucleotides after it. 24% of the data are Exon/Intron boundaries, 24% are Intron/Exon boundaries and the remaining 52% are neither. Each nucleotide is encoded as a 4 bit binary string. A is represented as 1000, C as 0100, G as 0010 and T as 0001.

6.1 KBANN

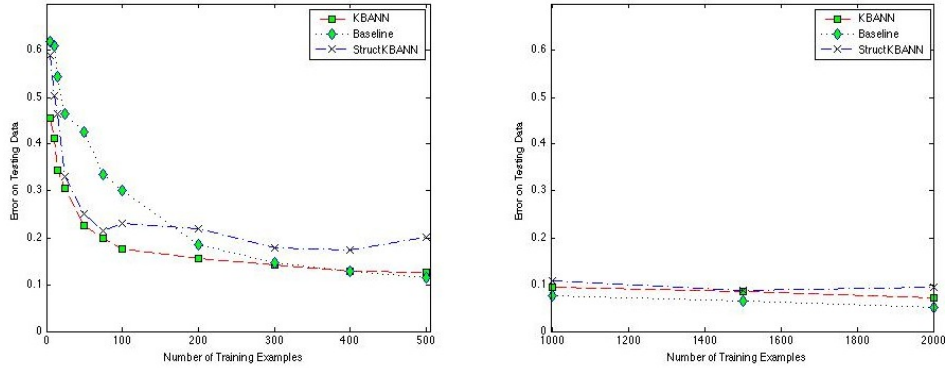


Figure 2: Performance Results for KBANN

The knowledge-based neural network has three hidden layers with seventeen, six and six nodes respectively. There are three outputs corresponding to the three possible class labels.

²<http://archive.ics.uci.edu/ml/datasets/Molecular+Biolog+Gene+Sequences>

The sigmoid function is used as the activation function for the hidden layers and the outputs are calculated using soft max. All of the nodes have a fixed bias, except for 4 nodes in the third hidden layer. These nodes are added because the knowledge base contains no rules for identifying sequences that do not contain splice junctions. Using cross-validation, ω is set to 3 and ϕ is set to 2.

We compare the performance of KBANN with an ANN with one hidden layer containing twenty-five units. The number of hidden units is chosen by cross-validation. We also consider an ANN (StructKBANN) with the same structure suggested by KBANN but having no fixed weights or biases. Figure 2 shows the performance of all three ANNs. For sparse training data, both KBANN and StructKBANN outperform the baseline ANN. As the number of training examples increases, the baseline ANN surpasses KBANN. It is important to note, however, that KBANN remains competitive even for 2000 training examples.

6.2 EASVM

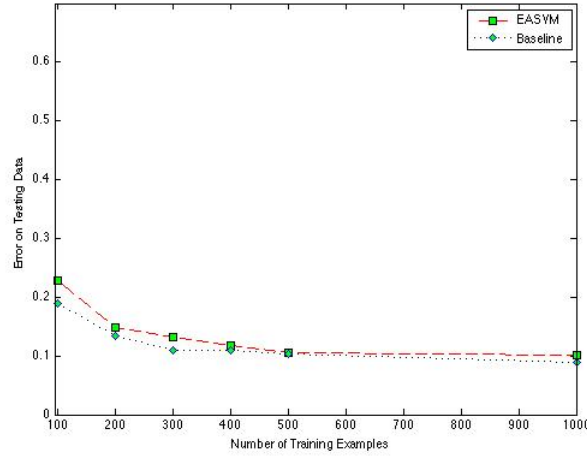


Figure 3: Performance Results for EASVM

To perform multi-class classification, we train three SVMs and use a majority vote for the final classification. Initially, a standard two-norm SVM was optimized using MATLAB's quadprog function. We tried a variety of kernel functions including the Gaussian kernel, and polynomial kernel with various dimensions. However, there were many instabilities such as inconsistent intercept values (b) for the support vectors. In the end, a one-norm SVM is optimized using MATLAB's linprog function. The one-norm SVM is formulated as follows:

$$\begin{aligned}
 \min_{w, \xi_i, \delta_i} \quad & \lambda \|w\|_1 + Q_1 \sum_i \xi^i + Q_2 \sum_i \delta^i \\
 \text{s.t.} \quad & y^i (w^\top x^i + b) \geq 1 - \xi^i \quad i = 1, 2, \dots, m \\
 & -\delta_i \leq w^\top x^i - w^\top v^i \leq \delta^i \quad i = 1, 2, \dots, m \\
 & \xi_i, \delta^i \geq 0 \quad i = 1, 2, \dots, m
 \end{aligned}$$

where v^i is the explanation for x^i . If x^i does not have an explanation (i.e. it follows no rule in the knowledge base) then $v^i = x^i$. For implementation, the one-norm, $\|w\|_1 = \sum_j |w_j|$, must be re-written as a series of constraints on the individual components of w . Each

component of w , denoted w_j , is written as the difference $(w_j^+ - w_j^-)$, and the constraints $w_j^+, w_j^- \geq 0$ are added to the SVM.

Some rules in the knowledge base are not easily encoded as explanations. For example, the rule for pyr-rich states, “At least 6 out of 10 nucleotides must be either a C or T”. Since encoding this rule as a series of explanations is infeasible we remove it from the knowledge base. Also, some training examples have more than one explanation. In this case, we randomly choose one.

Using cross-validation, λ is set to .001, and Q_1 and Q_2 are both set to 1. We compare the performance of the EASVM with a standard one-norm SVM. The performance results are shown in Figure 3. In our experiments, EASVM fails to provide any improvement over the baseline SVM.

6.3 KSVM

The KSVM is formulated as a one-norm SVM. The components w_j are again replaced by $(w_j^+ - w_j^-)$, and the constraints $w_j^+, w_j^- \geq 0$ are added. Rules are encoded as regions in the input space, $Cx \leq c$, where C is a binary string of length 240 with zeros in every position except those specified by the rule. Unfortunately, the optimizer could not reach a solution, even when the number of iterations was increased to 100,000 and the optimization was restarted at the position returned by the previous 10,000 iterations.

7 Conclusion and Future Work

We have explored three methods for adding domain knowledge to artificial neural networks and support vector machines.

We observed that encoding propositional statements is far more natural for KBANN than for EASVM or KSVM. Indeed, neither algorithm could accommodate the complex rules found in the knowledge base. Specifically for EASVM, it is unclear how to reconcile a training example that has multiple explanations.

For KSVM, the constraint matrix has $O(n(k + l) + m)$ rows and $O(n + m + k + l)$ columns. Even for a relatively small training set of size 100, the constraint matrix already has dimensionality 12606 X 660. This makes optimization difficult. In the case of EASVM, the optimization failed to return a solution for training sets of size 1500 and 2000. Thus both SVMs necessitate the use, or development, of better optimizers.

KBANN explicitly requires the knowledge base to be expressed in the form of propositional logic. One extension would be to allow for first-order statements. Another interesting area to explore would be different values of ω and ϕ for different units, and to see if it is possible to learn the optimal values.

Acknowledgments

We would like to acknowledge Professor Deva Ramanan and Professor Xiaohui Xie for their unwavering support and guidance.

References

- [FMS02] G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based support vector machine classifiers. Technical report, Data Mining Institute, Computer Science Dept., University of Wisconsin, 2002.

- [NTS90] M. O. Noordewier, G. G. Towell, and J. W. Shavlik. Refinement of approximate domain theories by knowledge-based neural networks. In *Proc. 8th Nat'l. Conf. A.I. (AAAI)*, pages 861–866, 1990.
- [NTS91] M. O. Noordewier, G. G. Towell, and J. W. Shavlik. Training knowledge-based neural networks to recognize genes in dna sequences. *Advances in Neural Information Processing Systems*, 3, 1991.
- [SD05] Q. Sun and G. DeJong. Explanation-augmented svm: an approach to incorporating domain knowledge into svm learning. In *ICML '05: Proceedings of the 22nd Int'l. Conf. Machine Learning*, pages 864–871. ACM, 2005.