

國立交通大學

電機與控制工程研究所

碩士論文

利用 Nearest Neighbor 演算法處理符號性質
資料的分類及其於生物資訊的應用

**Nearest Neighbor Algorithm for Symbolic Data Set
Classification and Its Application in Bioinformatics**

研究生：陳勇成

指導教授：張志永

中華民國九十三年六月

利用 Nearest Neighbor 演算法處理符號性質
資料的分類及其於生物資訊的應用

**Nearest Neighbor Algorithm for Symbolic Data Set
Classification and Its Application in Bioinformatics**

學 生：陳勇成 Student : Yeong-Cheng Chen

指導教授：張志永 Advisor : Jyh-Yeong Chang



A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master in

Electrical and Control Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

利用 Nearest Neighbor 演算法處理符號性質資料的分類及其於生物資訊的應用

學生：陳勇成

指導教授：張志永博士

國立交通大學電機與控制工程研究所

摘要

在過去，Nearest Neighbor 演算法通常都是用來處理資料屬性全部都是數值的例子。在這樣的屬性當中，這些事例都是被視為點，而且彼此之間的距離都適用標準的定義(如歐幾里得距離為基準)。而在符號的領域當中，我們通常需要對特徵向量空間做更複雜的處理；處理符號屬性空間的 Nearest Neighbor 演算法則，是利用特定的距離表，產生事例之間彼此的實值距離，而且指派一些權重在某些有效或可靠事例，以進一步修正特徵空間中的架構。

此篇論文，我們在符號領域中，有效的提出一種典型符號的學習方式，這種典型可以藉由最小距離分類器，學習處理關於符號屬性的問題、屬性的權重、以及在每一種類別當中找到一個典型符號，如此我們都可以由符號性質最近均值分類器(symbolic nearest mean classifier)進行分類。

除了上述之每一種類中當學到典型符號的方法，另外，我們可以把在同一類別當中所有典型的分量均予考慮，這樣我們就可以在同一個類別當中，設計出一個模糊式的典型符號，我們再由模糊典型符號之最近均值分類器(symbolic

nearest mean classifier with fuzzy prototype)進行分類。

我們使用上述演算法，處理機器學習領域中的三個(其中兩個為生物資訊)問題：鏡片辨識、辨識 Promoter 的基因序列及計算 Splice 的接面，皆呈現極佳的分類準確率。藉由不同的測試評估方法，和其他的學習演算法做比較，我們的演算法在那三個所要測試的資料領域中，都是勝過其他演算法或是可與其匹敵的；除此之外，我們的演算法具有訓練簡單及速度快的優點。最後，模擬實驗結果可以證明 Nearest-Neighbor 演算法及相關的延續發展在處理符號屬性資料的辨識是具優勢的。



Nearest Neighbor Algorithm for Symbolic Data Set Classification and Its Application in Bioinformatics

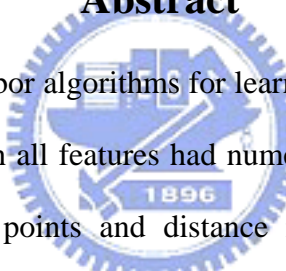
STUDENT: UN-CHENG CHEN

ADVISOR: DR.JYH-YEONG CHANG

Institute of Electrical and Control Engineering

National Chiao-Tung University

Abstract

The logo of National Chiao-Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized representation of a building or industrial structure, and the year '1896' is inscribed at the bottom.

In the past, nearest neighbor algorithms for learning from examples have worked very well in domains in which all features had numeric values. In such domains, the examples can be treated as points and distance metrics can be exploited using standard definitions, such as Euclidean distance. In symbolic domains, a more sophisticated treatment of the feature space is required. The nearest neighbor algorithm used for the symbolic feature space calculates distance tables that allow it to produce real-valued distances between instances, and attaches weight to the instances to further modify the structure of feature space.

In this thesis, we present an empirical analysis of symbolic prototype learners for discrete domains. Our symbolic prototype learner is derived from modifying the minimum distance classifier to solve problems with symbolic attributes and attribute weighting, and learns a prototype to each class. And then the classification is implemented in symbolic nearest mean classifier.

In addition to a prototype to each class, we can consider the contributions of the component prototypes for all samples in each class. Then we can design a fuzzy prototype approach and implement the symbolic nearest mean by fuzzy prototype setting.

We validate our proposed algorithms and on three data sets, majority of them are bioinformatics problems; that have been studied by machine learning researchers, such as Lenses recognition, identifying DNA promoter sequences, and Splice-junction determination. From experimental comparisons with the other learning algorithms, our simulation result has shown that our proposed algorithms are superior or comparable in the classification accuracy. In addition, our algorithms have advantages in training speed, simplicity, and perspicuity. Experimental evidence has demonstrated the promising sign to continue development of nearest neighbor algorithms for symbolic data domains.

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to my advisor, Dr. Jyh-Yeong Chang. Without his patient guidance and inspiration during the two years, it is impossible for me to complete the thesis. In addition, I am thankful to all my lab members for their discussion and suggestion.

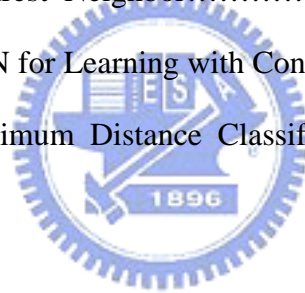
Finally, I would like to express my deepest gratitude to my family. Without their strong support, I could not go through the two years.



Content

ABSTRACT (CHINESE)	i
ABSTRACT (ENGLISH)	iii
ACKNOWLEDGEMENT	v

Chapter 1. Introduction	1
1.1 Introduction to Nearest Neighbor.....	1
1.2 Introduction to K -NN for Learning with Continuous Features.....	2
1.3 Introduction to Minimum Distance Classifier.....	5



Chapter 2. Nearest Neighbor Algorithm for Learning with Symbolic Features	10
2.1 Overlap Metric.....	10
2.2 Information Gain Feature Weighting.....	11
2.3 Modified Value Difference Metric.....	13
2.4 Tie Breaking.....	17
2.5 K - Nearest Neighbor Algorithm.....	18

Chapter 3. Learning Symbolic Prototypes.....20

3.1 Distance Between Values Combing Information Gain.....20

3.2 Mean of Symbolic Prototype.....20

3.3 Nearest Mean Classifier Algorithm.....21

3.4 Mean of Symbolic Fuzzy Prototype.....23

3.5 Nearest Mean Classifier with Fuzzy Prototype Algorithm.....24

Chapter 4. Simulation and Experiment.....26

4.1 Introduction to Data Sets.....26

4.2 Simulation and Results.....33

4.2.1 Information Gain of Data Sets.....33

4.2.2 Performance Comparison38

4.2.3 Comparison the Variance Difference Percentage.....39

4.2.4 Comparison by Leave-One-Out Methodology.....44

4.2.5 Comparison by Ten-Fold Methodology.....45

4.2.6 Comparison by Another Methodology.....47

4.3 Summary.....48

Chapter 5. Conclusion.....49

References.....51

List of Figures

Fig. 1.1.	Simple 2-D case, each instance is described only by two values.....	3
Fig. 1.2.	Minimum-distance classification - two single prototype case.....	8
Fig. 2.1.	The distance between instance X and Y.....	17
Fig. 2.2.	The steps of k -NN algorithm (IBL).....	19
Fig. 3.1.	The steps of SNM algorithm.....	23
Fig. 3.2.	The steps of FSNM algorithm.....	25
Fig. 4.1.	The organization of genes in higher organisms.....	29
Fig. 4.2.	“Canonical” splice-junction.....	31



List of Tables

Table	I. Number of occurrence of each value to each class.....	15
Table	II. Value difference table.....	16
Table	III. Rules for promoter-recognition.....	28
Table	IV. Initial rules for splice-junction determination.....	31
Table	V. Ambiguity code for DNA nucleotides.....	32
Table	VI. Ambiguity code for other characters.....	32
Table	VII. Properties of the data sets.....	32
Table	VIII. Information gain and gain ratio of the Lenses data set.....	33
Table	IX. Information gain and gain ratio of the Promoter data set.....	34
Table	X. Information gain and gain ratio of the Splice data set.....	36
Table	XI. Accuracies for different algorithms.....	38
Table	XII. Normalized variance difference of the Lenses data set.....	39
Table	XIII. Normalized variance difference of the Promoter data set.....	40
Table	XIV. Normalized variance difference of the Splice data set.....	42
Table	XV. Accuracies of Promoter data set on different algorithms by leave -one-out.....	44
Table	XVI. Accuracies of Splice data set on different algorithms by ten-fold cross-validation.....	46
Table	XVII. Accuracies of Lenses data set on different algorithms by ten-fold cross-validation.....	46
Table	XVIII. Accuracies of promoter data set on different algorithms by another methodology.....	47

Chapter 1. Introduction

1.1 Introduction to Nearest Neighbor

Different approaches from pattern recognition, machine learning, and expert systems have been used in intelligent diagnostic systems. One of the most significant developments in this domain is the nearest neighbor algorithm. But the above-mentioned approaches used for classification and approximation are not able to handle symbolic attributes directly. In the past, the nearest neighbor is used to deal with the continuous domain. But recent work suggests that the conventional Euclidean measure does not adequately model the symbolic data set. So the nearest neighbor requires us to define a distance function differ from the Euclidean distance to measure differences among items in a data set, and then to compute the closest items to a query point with respect to this measure. That is to say how to define an appropriate distance function between instances is very important.

Instance-based learning (IBL) programs also called exemplar-based introduced by Salzberg [1] or nearest neighbor introduced by Cover and Hart [2], which learn by storing examples as points in a feature space, require some means of measuring distance between examples and the more advanced concepts are introduced by Aha [3], Aha and Kibler [4], Salzberg [5], Cost and Salzberg [6]. An example is usually a vector of feature values plus a category label. When the features are numeric, normalized Euclidean distance can be used to compare examples. However, when the

feature values have symbolic, unordered values (e.g., the letters of the alphabet, which have no natural inter-letter “distance”), nearest neighbor methods typically resort to much simpler metrics, such as counting the features that match. Towell *et al.* [7] recently used this metric for the nearest neighbor algorithm in their comparative study. Simpler metrics may fail to capture the complexity of the problem domains, and as a result may not perform well. Cost and Salzberg [8] proposed an algorithm PEBLS. It constructs “modified value difference metric” introduced by Stanfill and Waltz [9] to produce a non-Euclidean distance metric and the distance is modified by a weighting scheme that weights instances in memory according to their performance history [5], [10]. On the other hand, Datta and Kibler [11], [12] proposed a symbolic prototype learning algorithm and the prototypes are learned by modifying the minimum-distance classifier to solve problems with symbolic attributes. And the above-mentioned algorithms can be applied in the pronouncing English text, and bioinformatics, such as identifying promoter sequences and predicting protein secondary structure.

1.2 Introduction to K -Nearest Neighbor for Learning with Continuous Features

Nearest neighbor classifiers are based on learning by analogy. The training samples are described by n -dimensional numeric attributes. Each sample represents a point in an n -dimensional space. In this way, all of the training samples are stored in an n -dimensional pattern space. When given an unknown sample, a k -nearest neighbor classifier searches the pattern space for k training samples that are closest to the unknown sample. The k training samples are the k “nearest neighbors” of the

unknown sample. “Closeness” is defined in terms of Euclidean distance, where the Euclidean distance between two points, $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ is

$$d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1.1)$$

Consider the case of m classes $\{C_i\}_{i=1}^m$ and a set of N sample patterns $\{y_i\}_{i=1}^N$ whose classification is a priori known. Let x denote an arbitrary incoming pattern. The nearest neighbor classification approach classifies x in the pattern class of its nearest neighbor in the set $\{y_i\}_{i=1}^N$, i.e., if $\|x - y_j\| = \min_{1 \leq i \leq N} \|x - y_i\|$ then $x \in C_j$. This scheme which is basically another type of minimum-distance classification, can be modified by considering the k nearest neighbors to x and using a majority-rule type classifier. Its advantage is overcoming class noise in the training set. And the example is shown in Fig. 1.1 :

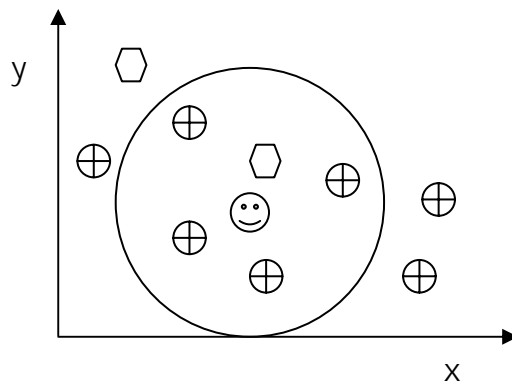


Fig. 1.1. Simple 2-D case, each instance is described only by two values (x, y co-ordinates). The class is either \oplus or \hexagon .

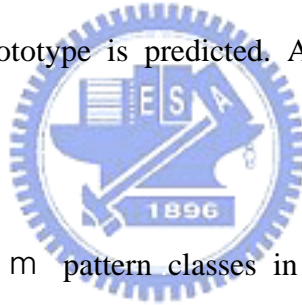
Inside the circle of Fig. 1.1, we can easily see that the class of simple-NN (1-NN) is \square , and the class of 5-NN is \oplus . (\odot is the testing data)

Nearest neighbor classifiers are instance-based or lazy-learners in that they store all of the training samples and do not build a classifier until a new (unlabeled) sample needs to be classified. This contrasts with eager learning methods, such as decision tree induction and back propagation, which construct a generalization model before receiving new samples to classify. Lazy learners can incur expensive computational costs when the number of potential neighbors (i.e., stored training samples) with which to compare a given unlabeled sample is great. Therefore, they require efficient indexing techniques. As expected, lazy learning methods are faster at training than eager methods, but slower at classification since all computation is delayed to that time. Unlike decision tree induction and back propagation, nearest neighbor classifiers assign equal weight to each attribute. This may cause confusion when there are many irrelevant attributes in the data.

Nearest neighbor classifiers can also be used for prediction, that is, to return a real-valued prediction for a given unknown sample. In this case, the classifier returns the average value of the real-valued labels associated with the k nearest neighbors of the unknown sample.

1.3 Introduction to Minimum Distance Classifier

Nearest neighbor learning method is a traditional statistical pattern recognition method for classifying unseen examples. These methods store the training set of examples. To classify an example from the test set, the closet example in the training set is found and the class of that example is predicted. In order to reduce the number of training examples stored, some researchers have stored only a subset of the examples or a generalization of the examples. They finally discuss a variation of the nearest neighbor classifier termed the minimum-distance classifier. This classifier works similarly to the nearest neighbor classifier except that instead of storing each training example, the mean of each class is stored as a prototype. During classification time, the class of closest prototype is predicted. And the detail is shown in the following:



Let C_1, \dots, C_m denote m pattern classes in \mathbb{R}^n , represented by the single prototype vectors y_1, \dots, y_m respectively. The distance between an incoming pattern x and the prototype vectors are

$$D_i = \|x - y_i\| = ((x - y_i)^T(x - y_i))^{1/2}, \quad 1 \leq i \leq m \quad (1.2)$$

and a minimum-distance classifier will classify x at C_j (or to y_j) for which D_j is minimum, i.e.,

$$D_j = \min \|x - y_i\|, \quad 1 \leq i \leq m \quad (1.3)$$

If the minimum is achieved by several j 's, x is classified at the first C_j (for example) for which a minimum is found, or not classified at all. Minimizing D_i^2 is equivalent to minimizing D_i but is more convenient. Indeed

$$D_i^2 = (x - y_i)^T(x - y_i) = x^T x - 2x^T y_i + y_i^T y_i \quad (1.4)$$

and since the constant $x^T x$ can be removed, we should only minimize

$-2x^T y_i + y_i^T y_i$ or instead maximize

$$2x^T y_i - y_i^T y_i, \quad 1 \leq i \leq m \quad (1.5)$$

Thus, we can define

$$d_i(x) = x^T y_i - \frac{1}{2} y_i^T y_i, \quad 1 \leq i \leq m \quad (1.6)$$

as decision functions and apply the classifier

$$x \in C_i \quad \text{iff} \quad d_i(x) > d_j(x), \quad j \neq i \quad (1.7)$$

The decision functions are linear, i.e.,

$$d_i(x) = w_i^T x, \quad 1 \leq i \leq m \quad (1.8)$$

where x is the augmented vector $(x_1, x_2, \dots, x_n, 1)^T$ and

$w_i = (w_{i1}, w_{i2}, \dots, w_{in}, w_{i,n+1})^T$, $1 \leq i \leq m$ are determined by

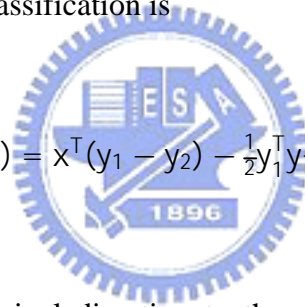
$y_i = (y_{i1}, y_{i2}, \dots, y_{in})^T$, $1 \leq i \leq m$ as

$$w_{ij} = y_{ij}; \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \quad (1.9)$$

$$w_{i,n+1} = -\frac{1}{2}y_i^T y_i, \quad 1 \leq i \leq m$$

For example, in the case of two pattern classes the decision boundary associated with the minimum-distance classification is

$$d_{12}(x) = d_1(x) - d_2(x) = x^T(y_1 - y_2) - \frac{1}{2}y_1^T y_1 + \frac{1}{2}y_2^T y_2 = 0 \quad (1.10)$$



This is a hyper plane in nominal direction to the vector $y_1 - y_2$. By substituting $x = (y_1 + y_2)/2$ in Eq. (1.10) we get

$$d_{12}(x) = \frac{1}{2}(y_1 + y_2)^T(y_1 - y_2) - \frac{1}{2}y_1^T y_1 + \frac{1}{2}y_2^T y_2 = 0 \quad (1.11)$$

The decision boundary is therefore the hyper plane which is perpendicular to the vector connecting the two prototypes and bisects it (Fig. 1.2).

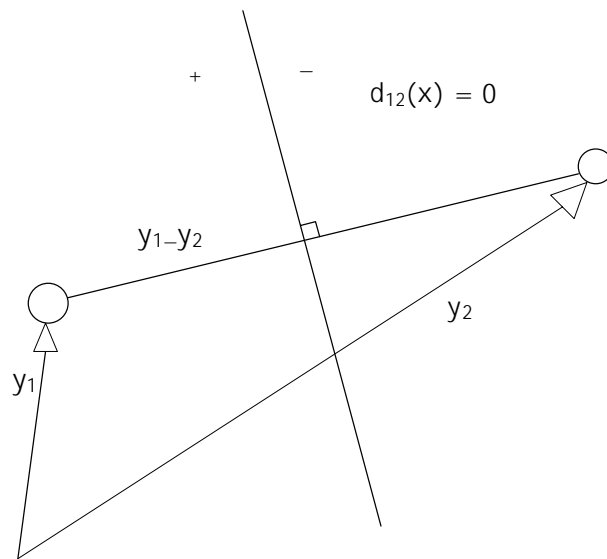


Fig. 1.2. Minimum-distance classification - two single prototype case.

Some researchers have stored only a subset of the examples or a generalization of the examples, such as Duda and Hart [13], Aha *et al* [14], Zhang [15], Skalak [16], and Datta and Kibler [17]. Duda and Hart discuss a variation of the nearest neighbor classifier termed the minimum-distance classifier. This classifier works similarly to the nearest neighbor classifier except that the mean of each class is stored as a prototype. During classification time, the class of closest prototype is predicted.

But the minimum-distance classifier has some drawbacks. It cannot use symbolic (nominal) attributes since the mean for symbolic attributes and the distance between two attribute values is not pre-defined. Another drawback is its inability to weigh attributes. Attributes not relevant for classification may distract the classifier.

In this thesis, we propose three classification algorithms. First, we construct a “value difference” tables in the style of Stanfill and Waltz [9] to produce a non-Euclidean distance metric so that we can deal with the symbolic data and add the information gain to improve classification effectiveness. Second, we use the MVDM metric combing with information gain to select the prototype to each class for different attributes by selecting the minimal variance one. Our method can resolve the problem of the minimum-distance classifier not able to be exploited for symbolic data. Third, we propose the fuzzy prototype to each class, in which each class prototype admits partial belonging to each symbol in each attribute. Finally, we use the MVDM metric and the information gain to predict the class of the data.

The rest of the thesis is structured as follows. We begin with constructing our MVDM metric in Chapter II. We use the information gain to be our weighting for attributes. In Chapter III, we develop two kinds of algorithms. One is the modified Symbolic Nearest Mean and the other is Symbolic Nearest Mean with the fuzzy prototype. In Chapter IV, the result of some experiments on some data sets is presented. Chapter V presents our conclusions.

Chapter 2. Nearest Neighbor Algorithm for learning with symbolic Features

2.1 Overlap Metric

The most basic metric that works for patterns with symbolic features is the overlap metric given in Eq. (2.1) and Eq. (2.2); where $\hat{\delta}(X, Y)$ is the distance between instances X and Y , represented by n features, and $\hat{\delta}$ is the distance per feature. The distance between two patterns is simply the sum of the differences between the features. The k -NN algorithm with this metric is called IB1 introduced by Aha et al. [14] Usually k is set to 1.

$$\hat{\delta}(X, Y) = \sum_{i=1}^n |X_i - Y_i|^k \quad (2.1)$$

where

$$\hat{\delta}(x_i, y_i) = \begin{cases} \text{abs}\left(\frac{x_i - y_i}{\max_i - \min_i}\right), & \text{if } \text{numeric, else} \\ 0, & \text{if } x_i = y_i \\ 1, & \text{if } x_i \neq y_i \end{cases} \quad (2.2)$$

But the overlap metric has a disadvantage that it could not decide the real intrinsic instances in symbolic domain. When the instances are not equal, it assigns a 1 of value to their distance. This assumes that each different symbolic value is equi-distant from another which leads to problems when two different values should be considered equal, and where symbolic values should have varying distances among them. However the rule is ambiguous, since it could not find the degree of difference

between the instances. So the overlap metric is not a proper distance metric for calculating the distance among those instances. We propose a different distance metric for symbolic attributes latter.

2.2 Information Gain Feature Weighting

The distance metric in Eq. (2.2) simply counts the number of (mis)matching feature-values in both patterns. In the absence of information about feature relevance, this is a reasonable choice. Otherwise, we can add domain knowledge bias to weight or select different features (see e.g., [Cardie \[18\]](#) for an application of linguistic bias in a language processing task), or look at the behavior of features in the set of examples used for training. We can compute statistics about the relevance of features by looking at which features are good predictors of the class labels. Information theory [19], [20] gives us a useful tool for measuring feature relevance in this way.

Information Gain (IG) weighting looks at each features in isolation, and measures how much information it contributes to our knowledge of the correct class label. The Information Gain of feature i is measured by computing the difference in uncertainty (i.e., entropy) between the situations without and with knowledge of the value of that feature (Eq. (2.3)).

$$w_i = H(c) - \sum_{v \in V_i} P(v) \times H(C | v) \quad (2.3)$$

where C is the set of class labels, V_i is the set of values for feature i , and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated P from relative frequencies in the training set.

It is important to realize that the IG weight is really a probability-weighted average of the informativity of the different values of the feature. On the one hand, this pre-empts the consideration of values with low frequency but high informativity. Such values “disappear” in the average. On the other hand, this also makes the IG weight very robust to estimation problems. Each parameter (weight) is estimated on the whole data set.

Information Gain, however, tends to overestimate the relevance of features with large numbers of values. Imagine a data set of hospital patients, where one of the available features is a unique “patient ID number”. This feature will have very high Information Gain, but it does not give any generalization to new instances. To normalize Information Gain for features with different numbers of values, Quinlan [20] has introduced a normalized version, called **Gain Ratio**, which is Information Gain divided by $si(i)$ (split info), the entropy of the feature-values (Eq. 2.5) is given by

$$W_i^n = \frac{H(C) - \sum_{v \in V_i} P(v) \times H(C | v)}{si(i)} \quad (2.4)$$

$$si(i) = - \sum_{v \in V_i} P(v) \log_2 P(v) \quad (2.5)$$

The resulting Gain Ratio values can then be used as weights w_i in the weighted distance metric (Eq. 2.6). The k -NN algorithm with this metric is called IB1-IG [21].

$$\Delta(X, Y) = \sum_{i=1}^n w_i^n \hat{I}(x_i, y_i) \quad (2.6)$$

The possibility of automatically determining the relevance of features implies that many different and possibly irrelevant features can be added to the feature set. This is a very convenient methodology if domain knowledge does not constrain the choice enough beforehand, or if we wish to measure the importance of various information sources experimentally. However, because IG values are computed for each feature independently, this is not necessarily the best strategy. Sometimes better results can be obtained by leaving features out than by letting them in with a low weight. Very redundant features can also be challenging for IB1-IG, because IG will overestimate their joint relevance. Imagine an informative feature which is duplicated. This results in an overestimation of IG weight by a factor two, and can lead to accuracy loss, because the doubled feature will dominate the distance metric.

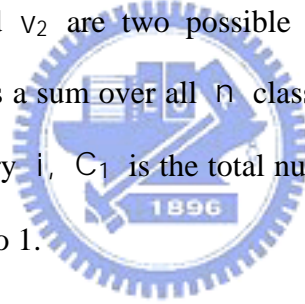
2.3 Modified Value Difference Metric

Overlap and IG Overlap, are limited to exact match between feature-values. This means that all values of a feature are seen as equally dissimilar. However, if we think of an imaginary task in e.g. the phonetic domain, we might want to use the information that ‘b’ and ‘p’ are more similar than ‘b’ and ‘a’. For this purpose a metric was defined by Stanfill and Waltz and further refined by Cost and Salzberg . It

is called the (Modified) Value Difference Metric, and it is a method to determine the similarity of the values of a feature by looking at co-occurrence of values with target classes. For the distance between two values, v_1 and v_2 of a feature, we compute the difference of the conditional distribution of the classes C_i for these values is defined in Eq. (2.7):

$$\begin{aligned}\hat{d}(v_1, v_2) &= \sum_{i=1}^n |P(C_i | v_1) - P(C_i | v_2)| \\ &= \sum_{i=1}^n \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k\end{aligned}\quad (2.7)$$

In the equation, v_1 and v_2 are two possible values for the feature and the distance between the values is a sum over all n classes. C_{1i} is the number of times v_1 was classified into category i , C_1 is the total number of times value 1 occurred, and k is constant, usually set to 1.



Using Eq. (2.7), we compute a matrix of value differences for each feature in the input data. It is interesting to note that the value difference matrices computed in the experiments below are quite similar overall for different features, although they differ significantly for some value pairs.

The idea behind this metric is that we wish to establish that values are similar if they occur with the same relative frequency for all classifications. The term C_{1i}/C_1 represents the likelihood that the central residue will be classified as i given that the feature in question has value v_1 . Thus we say that two values are similar if they give similar likelihood for all possible classifications. Eq. (2.7) computes overall

similarity between two values by finding the sum of the differences of these likelihood over all classifications.

Consider the following example. Say we have a pool of instances for which we examine a single feature that takes one of three values, A, B, and C. Two classifications, \bar{e} and \bar{i} , are possible. From the data we construct Table I, in which the table entries represent the number of times an instance had a given feature value and classification. From this information we construct a table of distances as follows. The frequency of occurrence of A for class \bar{e} is 57.14%, since there were 4 instances classified as \bar{e} out of 7 instances with value A. Similarly, the frequencies of occurrence for B and C are 28.57% and 66.67%, respectively. The frequency of occurrence of A for class \bar{i} is 42.86%, and so on. In order to find the distance between A and B, we use Eq. (2.7), which yields $|4/7 - 2/7| + |3/7 - 5/7| = 0.5714$. The complete table of distances is shown in Table II. Note that we construct a different value difference table for each feature; if there are 20 features, we will construct 20 tables.

TABLE I
NUMBER OF OCCURENCES OF EACH VALUE TO EACH CLASS

Feature Values	Class	
	\bar{e}	\bar{i}
A	4	3
B	2	5
C	4	2

TABLE II
VALUE DIFFERENCE TABLE

	Feature values		
	A	B	C
A	0.0000	0.5714	0.1905
B	0.5714	0.0000	0.7619
C	0.1905	0.7619	0.0000

We can find that the distance between A and C is very small. This is due to their occurrence numbers in the \tilde{e} and \tilde{i} are very similar. Eq. (2.7) defines geometric distance on a fixed, finite set of values. The distances are symmetric, and they obey the triangle inequality. We summarize these properties as follows:

- i. $\hat{d}(a, b) > 0, a \neq b$
- ii. $\hat{d}(a, b) = \hat{d}(b, a)$
- iii. $\hat{d}(a, a) = 0$
- iv. $\hat{d}(a, b) + \hat{d}(b, c) \geq \hat{d}(a, c)$

The total distance Δ between two instances is given by Eq. (2.6). X and Y represent two instances with X being an exemplar in memory (training example) and Y a new example (test example). The variables x_i and y_i are values of the i th feature for X and Y, where each example has n features and w_i is the information gain constructed by the data set. That is to say that we can calculate the weights so that we could find the importance for each feature in advance.

The distance between X and Y is shown in the following figure, Fig. 2.1 :

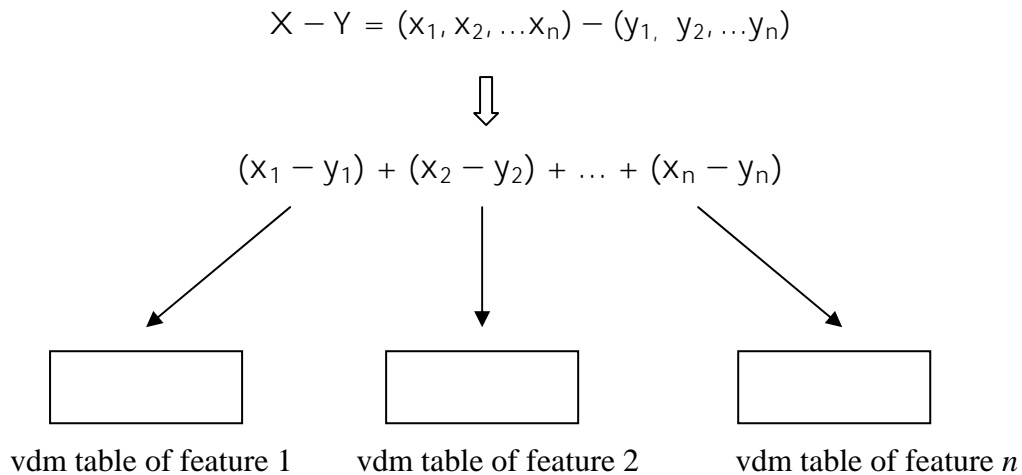


Fig. 2.1. The distance between instances X and Y.



2.4 Tie Breaking

Thus far we have described the last step of k -NN classification as taking the majority category among the set of nearest neighbors. Especially in case of unweighted voting, ties can occur; e.g., of a set of ten nearest neighbors, five votes for class A, and the other five for B. The procedure for breaking this tie in the k -NN classifier in our algorithm is as follows. First, the value of the k parameter is incremented by 1, and the additional nearest neighbors at this new k th distance are added to the current nearest neighbor set (k is subsequently reset to its user-specified value). If the tie in the class distribution persists, then the class label is selected with the highest overall occurrence in the training set. If that is also equal, then the first class is taken that was encountered when reading the training instance file. Optionally,

our algorithm can be set to avoid ties by making a random choice of a classification from a class distribution in a nearest-neighbor set, weighted by the distribution of the classes in the set.

2.5 *K*-Nearest Neighbor Algorithm

The *k*-NN algorithm is summarized in the following flow chart (Fig. 2.2):

Step 1)

Consider, the case of learning a discrete-valued target function of the form :

$$f : \mathbb{R}^n \rightarrow V, \text{ where } V \text{ is the finite set } \{V_1, V_2, \dots, V_S\}$$

Step 2)

For each training example $\langle x, f(x) \rangle$, add the example to the list of training examples

Step 3)

Construct value difference matrices by the training data

Step 4)

Given a query instance x_q to be classified

1. Let x_1, \dots, x_k denote the k instances from training examples that are nearest to x_q
2. Return

$$\hat{c}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k \hat{1}_s(v, f(x_i))$$

$$\text{where } \hat{1}_s(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

To avoid ties by making a random choice of a classification from a class distribution in a nearest neighbor set, weighted by the distribution of the classes in the set.

Fig. 2.2. The steps of k -NN algorithm (IBL).

Chapter 3. Learning Symbolic Prototypes

3.1 Distance Between Values Combing Information Gain

The distance metric for nearest neighbor and minimum-distance classifier is crucial to their predictive capabilities. Euclidean distance, a commonly used metric, is defined as

$$E(x, y) = \sqrt{\sum_{i=1}^a d(x_i, y_i)^2} \quad (3.1)$$

where x and y are two examples, a is the number of attributes and x_i refers to the i th attribute value for example x . For real-value attributes $d(x_i, y_i)$ is defined as their absolute difference (i.e., $|x_i - y_i|$). For symbolic attributes the distance between two values is defined as Eq. (2.4) and Eq. (2.6).

3.2 Mean of Symbolic Prototype

The simplest method for determining the mean of a symbolic attribute is to choose the most common value; however, this does not yield the best classification accuracy. In addition, the mean of a symbolic attribute must be one of its possible values. For numeric data the mean is the value that minimizes the variance. We generalize this notion to deal with symbolic attributes as well. We define the mean of a set of symbolic

values by finding the value of u minimizing the variance, that is

$$u = \arg \min(\sum_{v \in J} D(v, u)) \quad (3.2)$$

where J is the set of values and v is a symbolic value in J . But u is not necessarily a single value. It may be more than one value. So we can find the value in the u which occurs the most frequently in the J , that is

$$u_i = \arg \max_{1 \leq i \leq n} \sum_{v \in J} \hat{I}_m(v, u_i) \quad (3.3)$$

where $\hat{I}_s(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$

u_i will act as the best constant approximation for the symbolic values in J similar to the mean for the real values. Computationally, each symbolic value will be tried as u_i and the symbolic value that minimizes the variance will become the mean for J . We define the mean of a set of examples, S , to be the vector $\langle A1_u, A2_u, \dots, An_u \rangle$ where Ai_u denotes the mean of the i th attribute. We call this vector the prototype for S .

3.3 Nearest Mean Classifier Algorithm

We have developed an algorithm called SNM (Symbolic Nearest Mean). SNM uses the MVDM and the definition of mean described above. SNM learns a prototype to each class, classifies examples by finding the closest prototype using Euclidean

distance, and predicts the prototype's class. And the flow chart is shown in the followings:

Step 1)

Consider, the case of learning a discrete-valued target function of the form :

$$f : \mathbb{R}^n \rightarrow V, \text{ where } V \text{ is the finite set } \{V_1, V_2, \dots, V_s\}$$

Step 2)

For each training example $\langle x, f(x) \rangle$, add the example to the list of training examples

Step 3)

Construct the value difference matrix by the training data

Step 4)

Construct the mean of each class for different attributes in the training data

Step 5)

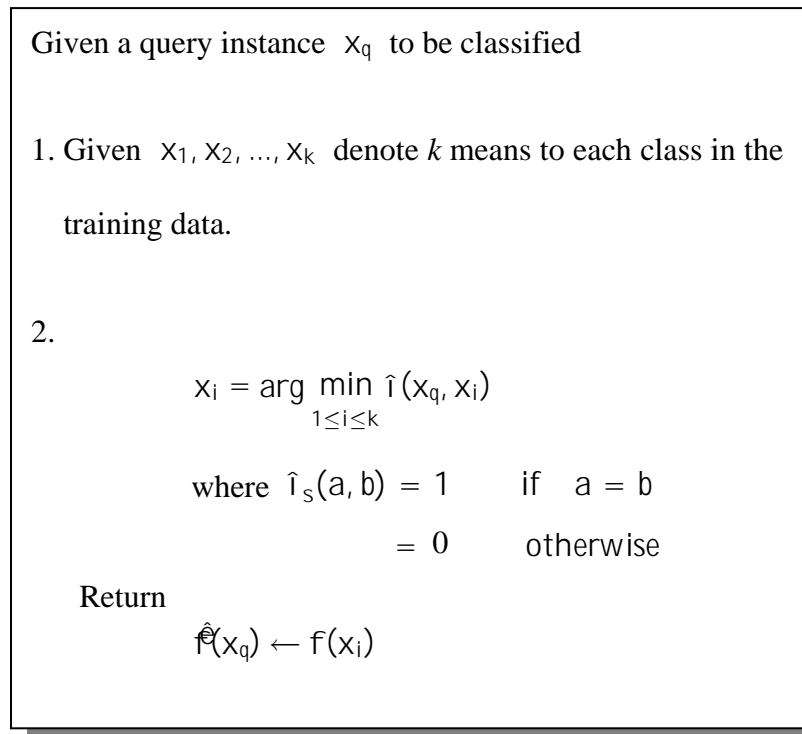


Fig. 3.1. The steps of SNM algorithm.



3.4 Mean of Symbolic Fuzzy Prototype

In the preceding context, we calculate the prototype to each class for different attributes. This is called Symbolic Nearest Mean classifier. In this section, we will develop another algorithm different from SNM. For each attribute, we will consider the contribution of all the possible symbols in the same class. So the mean of each class for different features is composed of each symbolic value multiplied by its factor of occurrence ratio. And the equation is thus defined as the following equation, Eq. (3.4):

$$u_t = \sum_{i=1}^n u_i \frac{V_i}{V_t} \quad (3.4)$$

where u_t is the mean and $\langle u_1, u_2, \dots, u_n \rangle_k$ are the symbol values in the set J_k (k is the k th feature in the data set), V_i is the occurrence number of u_i in the set, and V_t is the total occurrence number in the set. So u_t is called the **Fuzzy Prototype**.

3.5 Nearest Mean Classifier with Fuzzy Prototype Algorithm

We also develop the algorithm Fuzzy Symbolic Nearest Mean classifier (FSNM). It also uses the MVDM and the mean as defined in Eq. (3.4). FSNM learns a prototype composed of each value to each class. Like SNM, it classifies examples by finding the closest prototype using Euclidean distance, and predicts the prototype's class. And the flow chart of FSNM is shown in the following figure, Fig. 3.2 :

Step 1)

Consider, the case of learning a discrete-valued target function of the form :

$$f : \mathbb{R}^n \rightarrow V, \text{ where } V \text{ is the finite set } \{V_1, V_2, \dots, V_s\}$$

Step 2)

For each training example $\langle x, f(x) \rangle$, add the example to the list of training examples

Step 3)

Construct the value difference matrix by the training data

Step 4)

Given a query instance x_q to be classified

1. Given $\langle u_1, u_2, \dots, u_n \rangle_k$ are the symbol values in the training set J_k . (k is the k th feature in the data set), where v_1, v_2, \dots, v_n are their occurrence number and v_t is the total occurrence number.

$$u_t = \sum_{i=1}^n u_i \frac{v_i}{v_t}$$

2. Given $x_{t_1}, x_{t_2}, \dots, x_{t_c}$ denote c means composed of each symbol value each class for different features in the training data,

$$x_{t_i} = \arg \min_{1 \leq i \leq c} \hat{f}(x_q, x_{t_i})$$

where $\hat{f}(x_q, x_{t_i})$ can be found in the vdm matrix

Return


$$\hat{f}(x_q) \leftarrow f(x_{t_i})$$

Fig. 3.2. The steps of FSNM algorithm.

Chapter 4. Simulation and Experiment

4.1 Introduction to Data Sets

We will use three data sets to test the three our proposed algorithm composed of k -NN with our MVDM, SNM, and FSNM. We will use other prediction methods and compare the simulation results of ours with PEBLS [8] and SNM [11], [12] in the next section. Before presenting the result, we provide brief descriptions of the data sets.



Lenses [22], [23]: The Lenses data set with 24 points has four attributes and three classes named the patient should be fitted with hard contact lenses, the patient should be fitted with soft contact lenses, the patient should not be fitted with contact lenses. The features are age of the patient (young, pre-presbyopic, presbyopic), spectacle prescription (myope, hypermetrope), astigmatic (no, yes), tear production rate (reduced, normal) and the number of missing attribute value is zero. This database is complete (all possible combinations of attribute-value pairs are represented). Each instance is complete and correct and there are 9 rules cover the training set.

Promoter [24]–[28]: The data sets are short DNA sequences that precede the beginning of genes. They are can be detected in “wet” biological experiments as they are locations at which the protein *RNA polymerase* binds to the

DNA sequence. The set examples contains 53 sample promoters and 53 nonpromoter sequences. The 53 sample promoters were obtained from a compilation produced by Hawley and McClure [29]. Negative training examples were derived by selecting contiguous substrings from a 1.5 kilobase sequence provided by Prof. T. Record of the University of Wisconsin's Chemistry Department [30]. This sequence is fragment from *E. coli* bacteriophage *T7* isolated with the restriction enzyme *HaeIII*. By virtue of the fact that the fragment does not bind *RNA* polymerase, it is believed to contain no promoters.

The input features for promoter recognition are sequence of 57 DNA nucleotides, starting at position -50 (p -50) and ending at position $+7$ (p7). Each of these fields is filled by one of {A, G, T, C}. Following biological convention, the reference point for promoter recognition is the site at which gene transcription begins (if the example is a promoter). The reference point is located seven nucleotides from the right. (Thus, positive examples contain the first seven nucleotides of the transcribed gene.)

Table III contains the initial rule set used in the promoter recognition task. According to the rules in Table III, there are two sites at which the DNA sequence must bind to *RNA polymerase* – the minus 10 and minus 35 regions. (These regions are named for their distance from the reference point.) The conformation rules attempt to capture the three-dimensional structure of DNA, thereby ensuring that the minus 10 and minus 35 sites are spatially aligned. This set of rules was derived from the biological literature by Noordewier [26].

In the viewpoint of machine learning, the dataset has two classes with 57 dimensional data consisting of 106 points. All of 57 attributes consist of {A, T, C, G}.

Prior to training, the rules in Table III do not classify any of the 106 examples as promoters. Thus, the rules are useless as a classifier. Nevertheless, they do capture a significant amount of information about promoters.

TABLE III
RULES FOR PROMOTER-RECOGNITION

promoter	: - contact, conformation.	
contact	: - minus-35, minus-10 .	
minus-35	: - @-37 “CTTGAC-“ .	minus-35 : - @-37 “-TTG-GA” .
minus-35	: - @-37 “-TTGACA” .	minus-35 : - @-37 “-TTGAC-“ .
minus-10	: - @-14 “TATAAT--” .	minus-10 : - @-14 “-TA-A-T” .
minus-10	: - @-14 “-TATAAT-” .	minus-10 : - @-14 “-- TA- --T” .
conformation	: - @-45 “AA--A” .	
conformation	: - @-45 “A ---A” , @-28 “T---T-AA--T-“ , @-04 “T” .	
conformation	: - @-49 “A----T” , @-27 “T----A--T-TG” , @-01 “A” .	
conformation	: - @-47 “CAA-TT-AC” , @-22 “G---T-C” , @-08 “GCGCC-CC” .	

Splice [31]–[35] : Splice junctions are points on a DNA sequence at which ‘superfluous’ DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as E/I sites), and recognizing intron/exon boundaries (I/E sites). Fig. 4.1 illustrates how splicing occurs during the process of protein creation.

As with the promoter recognition, biologists have attempted to use neural networks from splice-junction determination. The work of Brunak *et al.* [31] is a very complete treatment of the topic.

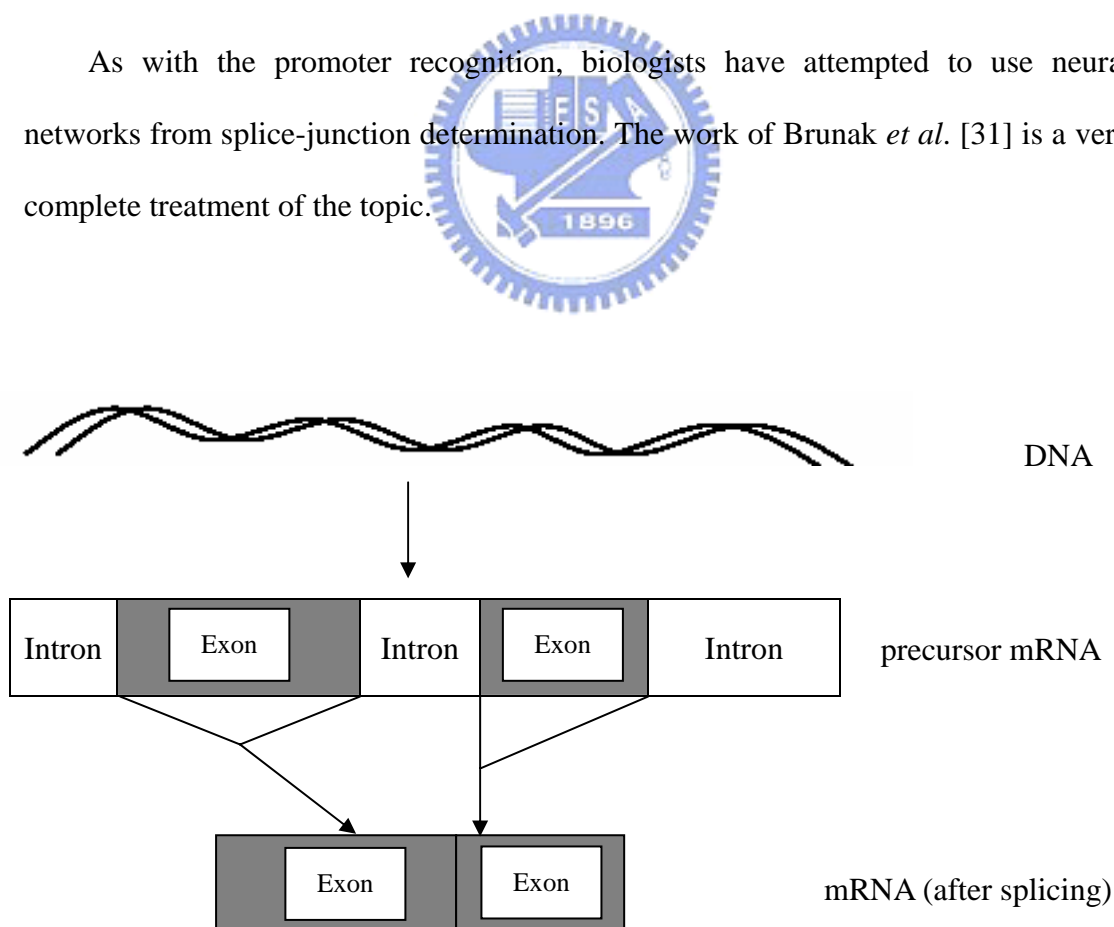


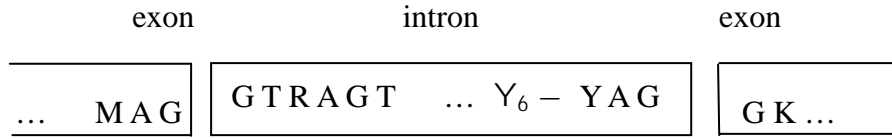
Fig. 4.1. The organization of genes in higher organisms.

Fig. 4.2 illustrates a prototypical DNA sequence showing both the I/E and E/I borders. This prototypical sequence is the basis of the rules for splice-junction determination contained in Table IV.

The dataset used to approach this problem was extracted from the biological literature by Noordewier [32]. The full set of examples contains 3190 examples, and three classes, of which approximately 25% are I/E, 25% are E/I and the remaining 50% are neither. Each example consists of a 60 nucleotide-long DNA sequence categorized according to the type of boundary at the center of the sequence; the center of the sequence is the reference location used for numbering nucleotides. These DNA sequences starts at position -30 and ending at position $+30$. Each of these fields is almost always filled by one of $\{A, G, T, C\}$. Other characters indicate ambiguity among the standard characters according to the following Table VI. The examples were obtained by taking the documented “split” genes from all primate gene entries in Genbank release 64.1 that are described as complete.

In addition to the examples, information about splice-junctions includes a set of 23 rules appearing in Table IV. This count does not include the rules defined by the iterative construct “For i from...” which define the meaning of “Y.” This set of rules was derived from the biological literature [36] by Noordewier.

In order to clearly summarize the three data sets, we list the properties of them in Table VII .



Y₆ means that either a “Y” occurs in six consecutive locations

Fig. 4.2. “Canonical” splice-junction.

TABLE IV

INITIAL RULES FOR SPLICE-JUNCTION DETERMINATION

E/I :- @-3 ‘MAGGTRAGT’ , not (E/I-stop) .

E/I-stop :- @-3 ‘TAA’ . E/I-stop :- @-4 ‘TAA’ . E/I-stop :- @-5 ‘TAA’ .

E/I-stop :- @-3 ‘TAG’ . E/I-stop :- @-4 ‘TAG’ . E/I-stop :- @-5 ‘TAG’ .

E/I-stop :- @-3 ‘TGA’ . E/I-stop :- @-4 ‘TGA’ . E/I-stop :- @-5 ‘TGA’ .

I/E :- pyrimidine-rich, @-3 ‘YAGG’ , not (I/E-stop) .

pyrimidine-rich :- 6 of (@-15 ‘YYYYYYYYYYY’) .

For i from ((-30 to -1) and (+1 to +30))

{@< i > ‘Y’ :- @< i > ‘C’ . @< i > ‘Y’ :- @< i > ‘T’ .}

I/E-stop :- @1 ‘TAA’ . I/E-stop :- @2 ‘TAA’ . I/E-stop :- @3 ‘TAA’ .

I/E-stop :- @1 ‘TAG’ . I/E-stop :- @2 ‘TAG’ . I/E-stop :- @3 ‘TAG’ .

I/E-stop :- @1 ‘TGA’ . I/E-stop :- @2 ‘TGA’ . I/E-stop :- @3 ‘TGA’ .

see Table V for meanings of letters other than A, G, T, C. The notation ‘:- ‘ indicates a rule that is a definition. Hence, it is not to be altered during learning. The construct “For i ...” creates 120 rules that define a disjunction at each location in the input. Consequents with an antecedent of the form ‘n of (...)’ are satisfied if at least n of the parenthesized antecedents are true.

TABLE V
AMBIGUITY CODE FOR DNA NUCLEOTIDES

Code	Meaning
M	A or C
R	A or G
W	A or T
S	C or G
Y	C or T
K	G or T
V	A or C or G
H	A or C or T
D	A or G or T
B	C or G or T
X	A or G or C or T

TABLE VI
AMBIGUITY CODE FOR OTHER CHARACTERS

character	meaning
D	A or G or T
N	A or G or C or T
S	C or G
R	A or G

TABLE VII
PROPERTIES OF THE DATA SETS

Data Set	Lenses	Promoter	Splice
Data Number	24	106	3190
Class Number	3	2	3
Feature Number	4	57	60
Feature Property	Nominal	Nominal	Nominal
Missing Value	None	None	None

4.2 Simulation and Results

4.2.1 Information Gain of Data Set

We adopt the information gain to be our weight in our three algorithms. First we list the information gain of data sets and the information gain and gain ratio of data sets are showed in Tables VIII, IX, and X, respectively.

TABLE VIII
INFORMATION GAIN AND GAIN RATIO
OF THE LENSES DATA SET

Features	Information Gain (w_i)	Gain Ratio (w_i^r)
1	0.039397	0.024856
2	0.039511	0.039511
3	0.375830	0.375830
4	0.547550	0.547550

TABLE IX
INFORMATION GAIN AND GAIN RATIO
OF THE PROMOTER DATA SET

Features	Information Gain (w_i)	Gain Ratio (w_i^r)
1	0.0037745	0.0019560
2	0.0505350	0.0255290
3	0.0024569	0.0012359
4	0.0019557	0.0009824
5	0.0318790	0.0162970
6	0.1479500	0.0770730
7	0.0572380	0.0296240
8	0.0769040	0.0393220
9	0.0667640	0.0340440
10	0.0837850	0.0427210
11	0.0445300	0.0232010
12	0.0216070	0.0108870
13	0.0231640	0.0119700
14	0.0613640	0.0313600
15	0.3472300	0.1976200
16	0.2824700	0.1620100
17	0.3202800	0.1809300
18	0.1788900	0.0931230
19	0.0766450	0.0411360
20	0.1141600	0.0573200
21	0.0284400	0.0143990
22	0.0092367	0.0046613
23	0.0240180	0.0120620
24	0.0210790	0.0106400
25	0.0065717	0.0033171

26	0.0307930	0.0159170
27	0.0325380	0.0163050
28	0.0164910	0.0084008
29	0.0078622	0.0040520
30	0.0586060	0.0297950
31	0.0700260	0.0361900
32	0.0385390	0.0193820
33	0.0354010	0.0184000
34	0.0097125	0.0049977
35	0.0215350	0.0107710
36	0.0042229	0.0021333
37	0.0305820	0.0160730
38	0.0786980	0.0396620
39	0.2351100	0.1217300
40	0.0997090	0.0535690
41	0.1191100	0.0620070
42	0.0681800	0.0345430
43	0.0380690	0.0198790
44	0.0073343	0.0037683
45	0.0245650	0.0126340
46	0.0585780	0.0296540
47	0.0134770	0.0067665
48	0.0336760	0.0169520
49	0.1086300	0.0552150
50	0.0123430	0.0064302
51	0.0367800	0.0184870
52	0.0412570	0.0213320
53	0.0204960	0.0103850
54	0.0293260	0.0148210
55	0.0133500	0.0066934
56	0.0084004	0.0042138
57	0.0206390	0.0105330

TABLE X
INFORMATION GAIN AND GAIN RATIO
OF THE SPLICE DATA SET

Features	Information Gain (w_i)	Gain Ratio (w_i^r)
1	0.0056511	0.0028317
2	0.0060552	0.0030299
3	0.0024874	0.0012469
4	0.0088296	0.0044223
5	0.0163470	0.0081853
6	0.0178410	0.0089460
7	0.0079690	0.0039872
8	0.0069887	0.0035000
9	0.0282700	0.0141690
10	0.0279820	0.0140000
11	0.0112620	0.0056412
12	0.0161990	0.0081392
13	0.0274150	0.0137320
14	0.0274080	0.0137380
15	0.0321370	0.0161700
16	0.0435680	0.0219330
17	0.0478100	0.0240790
18	0.0562910	0.0283270
19	0.0651770	0.0328440
20	0.0718200	0.0361920
21	0.0650180	0.0328290
22	0.0606410	0.0304680
23	0.0746060	0.0376200
24	0.0777150	0.0392370
25	0.1112800	0.0561200
26	0.0779480	0.0394010
27	0.0058059	0.0029136

28	0.2106100	0.1103300
29	0.3409500	0.1912100
30	0.3883200	0.2333100
31	0.3294400	0.1837800
32	0.3304300	0.1770600
33	0.1512500	0.0769090
34	0.1371000	0.0699530
35	0.2304400	0.1186700
36	0.0326220	0.0163850
37	0.0119670	0.0060254
38	0.0075238	0.0037673
39	0.0085445	0.0042825
40	0.0093502	0.0046951
41	0.0150360	0.0075396
42	0.0060594	0.0030408
43	0.0105530	0.0052965
44	0.0038257	0.0019150
45	0.0075266	0.0037714
46	0.0101130	0.0050692
47	0.0111670	0.0055982
48	0.0096923	0.0048606
49	0.0098409	0.0049540
50	0.0068017	0.0034024
51	0.0045416	0.0022824
52	0.0046442	0.0023267
53	0.0038402	0.0019239
54	0.0116450	0.0058496
55	0.0107500	0.0053878
56	0.0062025	0.0031036
57	0.0039486	0.0019792
58	0.0073756	0.0036980
59	0.0035447	0.0017738
60	0.0118150	0.0059359

4.2.2 Performance Comparison

In this section, we compare the performance of the three our proposed methods: k -NN (MVDM), SNM, and FSNM, and adopt the leave-one-out strategy (i.e., each instance is tested after first training on all other instances in the dataset) to test the Lenses and Promoter datasets. In the case of the Splice dataset, whose data number is relatively large, we use ten-fold strategy on 1000 randomly selected from the complete set of 3190. From Table XI, we can easily see that the k -NN (MVDM) is better than the other two algorithms and we just list the best value of k in the table. We can know that using the information gain weighting method boosts the accuracies by 1–5%. And the result is shown in the followings:



TABLE XI
ACCURACIES FOR DIFFERENT ALGORITHMS

	Lenses	Promoter	Splice	Ave Rank
5-NN (unweighted)	83.33 (2)	90.57 (4)	94.60(2)	2.67
5-NN (MVDM)	87.50 (1)	94.34 (1)	95.70(1)	1.00
SNM (unweighted)	83.33 (2)	88.68 (5)	79.70(6)	4.33
SNM (weighted)	87.50 (1)	93.34 (2)	84.90(4)	2.33
FSNM (unweighted)	87.50 (1)	88.68 (6)	81.90(5)	4.00
FSNM (weighted)	87.50 (1)	91.51 (3)	85.10(3)	2.33

4.2.3 Comparison the Variance Difference Percentage

As mentioned in Sec. 4.2.2, we know the performance of SNM approximates to the performance of FSNM. Besides, We compare the average variance between SNM and FSNM and find that the variance of FSNM is larger than the variance of SNM (except for the splice dataset), but the performance of FSNM is a little smaller (or better) than SNM. We calculate the variance difference of SNM and FSNM that is divided by the variance of SNM. From Tables XII, XIII, and XIV, we can know the result easily. Because some symbols of several features in a few classes are the same, it can generate the zero variance in the case of SNM and FSNM and the difference ratio of NAN (the zero is divided by the zero) and INF (the floating point is divided by the zero) will occur in the Lenses and Splice data set. Surprisingly, the variance of SNM is larger than FSNM's in the three class and leads to negative variance difference percentage in the Splice data set.

TABLE XII

NORMALIZED VARIANCE DIFFERENCE OF LENSES DATA SET

Class \ Feature	1	2	3
1	0.19659	0.26832	0.25566
2	NAN	0.20000	0.06667
3	NAN	NAN	0.06667
4	NAN	NAN	0.60000

TABLE XIII**NORMALIZED VARIANCE DIFFERENCE OF PROMOTER DATA SET**

Feature \ Class	Class	
	1	2
1	0.37947	0.39257
2	0.46304	0.42007
3	0.21462	0.15898
4	0.39028	0.39283
5	0.35279	0.31162
6	0.28416	0.41175
7	0.06832	0.48068
8	0.47345	0.17377
9	0.32232	0.32066
10	0.12252	0.33475
11	0.60287	0.26374
12	0.32686	0.23488
13	0.29601	0.43290
14	0.52022	0.25226
15	0.68027	0.34129
16	0.62763	0.54694
17	0.58716	0.50646
18	0.32399	0.33179
19	0.23065	0.31185
20	0.39476	0.33434
21	0.48173	0.34103
22	0.30901	0.27972
23	0.43347	0.47529
24	0.19526	0.20749
25	0.36619	0.39439
26	0.31489	0.27221

27	0.36429	0.31223
28	0.53582	0.51188
29	0.59453	0.52644
30	0.52726	0.20987
31	0.10495	0.45418
32	0.40233	0.45102
33	0.10633	0.33318
34	0.39324	0.45794
35	0.33131	0.29710
36	0.13440	0.26500
37	0.38119	0.36492
38	0.44095	0.25993
39	0.70765	0.24773
40	0.32296	0.24584
41	0.26829	0.36285
42	0.32723	0.34308
43	0.42502	0.20583
44	0.39966	0.26464
45	0.17951	0.42657
46	0.19690	0.45739
47	0.24393	0.20536
48	0.30094	0.22838
49	0.33769	0.30411
50	0.23918	0.26970
51	0.29825	0.23346
52	0.52788	0.44816
53	0.30191	0.24272
54	0.37850	0.49361
55	0.45439	0.46230
56	0.40608	0.33459
57	0.25969	0.44075

TABLE XIV**NORMALIZED VARIANCE DIFFERENCE OF SPLICE DATA SET**

Class Feature	1	2	3
1	0.44743	0.72794	-0.32741
2	0.38454	0.42209	-0.29129
3	0.42007	0.59478	-0.32114
4	0.18044	0.28901	-0.42623
5	0.35891	0.66520	-0.34974
6	0.12127	0.63104	-0.43996
7	0.63804	0.44848	-0.28017
8	-0.20629	-0.06151	-0.66321
9	0.06755	0.29048	-0.47173
10	0.44545	0.60284	-0.39052
11	0.56829	0.78305	-0.32645
12	0.05729	0.43439	-0.46078
13	0.24594	0.81238	-0.41819
14	0.12550	0.45994	-0.46723
15	-0.08267	0.14020	-0.54888
16	0.19282	0.77140	-0.46978
17	0.11679	0.40249	-0.50206
18	0.43634	0.36436	-0.35082
19	0.14978	0.51711	-0.44017
20	0.16972	0.40588	-0.47022
21	0.16078	0.37822	-0.47082
22	0.21989	0.41264	-0.42916
23	0.05456	0.48720	-0.51304
24	-0.05390	0.45879	-0.51009
25	0.01526	0.69872	-0.53889
26	0.08455	1.15870	-0.50932
27	0.05931	0.13232	-0.45634

28	0.06583	0.48776	-0.40005
29	0.23689	NAN	-0.02094
30	0.31988	INF	-0.55758
31	NAN	0.21118	-1.00000
32	INF	0.41000	-0.96855
33	1.42190	0.12705	-0.88755
34	0.35092	0.76008	-0.38646
35	0.54921	1.04740	-0.66353
36	0.12911	0.16759	-0.30196
37	0.46383	0.63186	-0.28625
38	0.45396	0.69831	-0.30689
39	0.55782	0.29449	-0.43510
40	0.49898	0.63533	-0.31138
41	0.34152	0.51178	-0.37654
42	0.41203	0.34480	-0.40801
43	0.71485	0.39814	-0.27207
44	0.38959	0.40317	-0.36746
45	0.34573	0.39305	-0.35445
46	0.56460	0.48819	-0.29116
47	0.04521	0.25798	-0.47594
48	0.59244	0.78030	-0.25480
49	0.25737	0.31963	-0.52225
50	0.67662	0.63602	-0.22120
51	0.50000	0.67744	-0.26658
52	0.28681	0.35101	-0.41207
53	0.33144	0.42098	-0.33320
54	0.36146	0.60999	-0.39381
55	0.57789	0.55490	-0.33133
56	0.15163	0.50049	-0.42813
57	0.81511	0.79865	-0.14426
58	0.13497	0.19400	-0.47673
59	0.97629	1.18930	-0.13455
60	0.19918	0.20985	-0.32580

As mentioned in Chapter 2 and Chapter 3, we use different testing methods in our experiments and compare with other algorithm, such as PEBLS, C4.5, SNM [11], [12] and so on. Hereafter, the accuracy and rank of different prediction methodologies will be displayed in the following tables, in which the accuracies and ranks of our proposed algorithms will be shown in boldface. Moreover, we will try the value of k in k -NN, such as $k=1, 3, 5, 7, 9$, etc and show the best k in our comparison.

4.2.4 Comparison by Leave-One-Out Methodology

We use the leave-one-out strategy to test the promoter data set by our three algorithms (FSNM, SNM, k -NN (MVDM)) and provide comparisons to nearest neighbor (NN) using the overlap metric (which counts the number of feature value mismatches between two examples), PEBLS, and BAYES (Bayesian classifier) [37]. And the comparison is showed in the Table XV. From Table XV, All of our algorithms are better than the other algorithms.

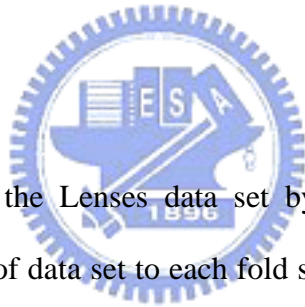
TABLE XV

ACCURACIES OF PROMOTER DATA SET ON DIFFERENT ALGORITHMS BY LEAVE-ONE-OUT

	Promoter	Rank
BAYES [37]	91.50	4
PEBLS [37]	90.60	5
NN [37]	80.50	6
SNM	93.34	2
FSNM	91.51	3
5-NN(MVDM)	94.43	1

4.2.5 Comparison by Ten-Fold Cross-Validation Methodology

Besides, we achieve the results of Splice data set by using ten-fold cross-validation methodology on 1000 randomly selected from the complete set of 3190 (we delete 15 missing values) and randomizing the permutation of data set to each fold so that we could choose the best result and we provide comparisons with other algorithms, such as KBANN [38]–[40], PEBLS, ID3 [41], and so on (all experiments, except BRAIN, carried out at the University of Wisconsin [35], [42]). From Table XVI, the performance of k -NN (MVDM) is better than other algorithms but SNM and FSNM are only better than NN (overlap).



Moreover, we also test the Lenses data set by ten-fold cross-validation and randomizing the permutation of data set to each fold so that we could choose the best result to compare with well-known algorithms, such as C4.5 and C5.0 [43]. From Table XVIII, our experimental results show that our performances are superior to other algorithms.

TABLE XVI
ACCURACIES OF SPLICE DATA SET ON DIFFERENT ALGORITHMS
BY TEN-FOLD CROSS-VALIDATION

	Splice	Rank
KBANN	93.12	3
BACKPROP	92.74	4
PEBLS	92.47	5
5-NN (MVDM)	95.70	1
SNM	84.90	10
FSNM	85.10	9
COBWEB	87.90	7
PERCEPTRON	87.43	8
ID3	88.86	6
Nearest. Neighbor	82.72	11
Brain	95.67	2

TABLE XVII
ACCURACIES OF LENSES DATA SET ON DIFFERENT ALGORITHMS
BY TEN-FOLD CROSS-VALIDATION

	Lenses	Rank
5-NN (MVDM)	90.00	1
SNM	90.00	1
FSNM	90.00	1
C4.5	71.10	3
C5.0	83.30	2

4.2.6 Comparison by Another Methodology

Finally, we specify all experiments by the average of 30 runs of randomly choosing two-thirds of the data as a training set and the remainder as the test set to test Promoter using our three algorithms and also randomizing permutation of data set to each partition to choose the best accuracy. Here, we compare with PEBLS, C4.5, and SNM [11], [12]. From Table XVIII, we can see that the performance of k -NN (MVDM) is better than others.

TABLE XVIII
ACCURACIES OF PROMOTER DATA SETS ON DIFFERENT
ALGORITHMS BY ANOTHER METHODOLOGY

	Promoter	Rank
C4.5	74.30	6
SNM (Kibler)	91.40	3
PEBLS	89.40	4
SNM	93.34	2
FSNM	89.26	5
3-NN (MVDM)	96.61	1

4.3 Summary

In Sec. 4.2.1, we first list the information gain and gain ratio of the data set that we want to test. We can boost our accuracies by adding the information gain weighting method. In Sec. 4.2.2, we compare our three algorithms (k -NN (MVDM), SNM, and FSNM) with and without information gain weighting by leave-one-out strategy and find out that k -NN (MVDM) is better than the other two algorithms. In Sec. 4.2.3, we provide comparisons with the variance of difference ratio between SNM and FSNM find the variance of FSNM is larger than the variance of SNM greatly but it is surprising that FSNM's performance approximates to the performance of SNM. In Sec. 4.2.4, we use leave-one-out methodology and provide comparisons with Kasif, Salzberg, Waltz, Rachlin, and Aha [37]. Our performances are all better than theirs. In Sec. 4.2.5, we compare with Rampone [44] by another prediction methodology called ten-fold cross-validation and find out that only the k -NN is better than others. Moreover, we compare with C4.5 and C5.0 and our performance are superior to other algorithms. Finally, in Sec. 4.2.6, ours is compared with Domingos and Pazzani [45] by the average of 30 runs of randomly choosing two-thirds of the data as a training set and the remainder as the test set and find that k -NN (MVDM) is still only the best in our experiments and SNM approximates to PEBS and SNM [11], [12].

Chapter 5. Conclusion

In this thesis, we proposed a nearest neighbor algorithm (IBL) and used sophisticated coding and weighting method in order to classify the data with symbolic domains. In direct comparisons on some famous data sets by different testing methodologies, our k -NN (MVDM) performed better than back propagation, ID3, KBANN, and so on.

In view of prototypes, we proposed a symbolic nearest mean classifier whose prototypes are learned by modifying the minimum distance classifier to solve the symbolic domains, attribute weighting, and learn a prototype to each class. Furthermore, we consider all the contributions of prototypes to each class and design a fuzzy prototype to be the mean to each class. Both of algorithms can be improved by the weighting method. We provide comparisons with other algorithms by distinct prediction methodologies and show our implementations performed as well (or better than) C4.5, C5.0, PEBLS, and BAYES, etc. In addition, nearest neighbor offers clear advantages in that it is much faster to train and its representation relatively easy to interpret. No one yet knows how to interpret the networks of weights learned by neural nets. Decision trees are somewhat easier to interpret, but it is hard to predict the impact of a new example on the structure of the tree. Sometimes one new example makes no difference at all, and at other times it may radically change a large portion of the tree. On the other hand, neural nets have a fixed size, and decision trees tend to be quite small, and in this respect both methods compress the data in a way that

nearest neighbor does not. In addition, classification time is fast (dependent only on the depth of the net or tree, not on the size of the input). Based on classification accuracy, though, it is not clear that other learning techniques have an advantage over nearest-neighbor methods.

With respect to nearest neighbor learning, we have shown how weighting exemplars can improve accuracy by information gain (IG) weight really a probability-weighted average of the informativity of the different values of the feature and can reduce the impact of unreliable examples. The nearest neighbor algorithm is one of the simplest learning methods known, and yet no other algorithm has been shown to outperform it consistently. Taken together, these results indicate that continued research on extending and improving nearest neighbor learning algorithms should prove fruitful.



References

- [1] S. Salzberg, *Learning with Nested Generalized Exemplars*. Norwell, MA: Kluwer Academic Publishers, 1990.
- [2] T.M. Cover and P.E. Hart “Nearest neighbor pattern classification,” *IEEE Trans. Inform. Theory*, vol. 13, pp. 21–27, 1967.
- [3] D. Aha, “Incremental, instance-based learning of independent and graded concept descriptions,” in *Proc. of the Sixth International Workshop on Machine Learning*, 1989. pp. 387–391.
- [4] D. Aha and D. Kibler, “Noise-tolerant instance-based learning algorithms,” in *Proc. 11th Int. Joint Conf. Artificial Intelligence*, 1989. pp. 794–799.
- [5] S. Salzberg, “Nested Hyper-rectangles for Exemplar-based Learning,” in K.P. Jantke ed. *Analogical and Inductive Inference: International Workshop AII*, 1989, pp. 184–201.
- [6] S. Cost and S. Salzberg, “Exemplar-based Learning to Predict Protein Folding,” in *Proc. of the Symposium on Computer Applications to Medical Care*, 1990.
- [7] G. Towell, J. Shavlik, and M. Noordewier “Refinement of approximate domain theories by knowledge-based neural networks,” *Proc. 8th National Conf. Artificial Intelligence*, 1990, pp. 861–866.
- [8] S. Cost and S. Salzberg, “A weighted nearest neighbor algorithm for learning with symbolic features,” *Machine Learning*, vol. 10, pp. 57–78, 1993.
- [9] C. Stanfill and D. Waltz, “Toward memory-based reasoning,” *Communications of the ACM*, vol. 29, pp. 1213–1228, 1986.

- [10] S. Salzberg, *Learning with Nested Generalized Exemplars*. Norwell, MA: Kluwer Academic Publishers, 1990.
- [11] P. Datta, D. F. Kibler, "Symbolic Nearest Mean Classifiers," in *Proc. AAAI, IAAI*, 1997, pp. 82–87.
- [12] P. Datta, D. F. Kibler, "Learning Symbolic Prototypes," in *Proc. ICML*, 1997, pp. 75–82.
- [13] R. Duda and P. Hart, *Pattern classification and scene analysis*. New York: John Wiley & Sons, 1973.
- [14] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, pp. 37–66, 1991.
- [15] J. Zhang, "Selecting typical instances in instance-based learning," in *Proc. 9th. Int. Machine Learning Conf.* 1992, pp. 470–479.
- [16] D. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," in *Proc. 11th Int. Machine Learning Conf.* 1994, pp. 293–301.
- [17] P. Datta and D. Kibler (1995) "Learning Prototypical Concept Descriptions," in *Proc. 12th Int. Machine Learning Conf.* 1995, pp. 158–166.
- [18] C. Cardie, "Automating Feature Set Selection for Case-Based Learning of Linguistic Knowledge," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996, pp. 113–126.
- [19] R. J. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
- [20] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

- [21] W. Daelemans and A. van den Bosch, "Generalization performance of backpropagation learning on a syllabification task," in *M. Drossaers and A. Nijholt (Eds.), Proc. of the 3rd Twente Workshop on Language Technology*. 1992, pp. 27–37.
- [22] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, pp. 349–370, 1987.
- [23] H. I. Witten and A. B. MacDonald, "Using concept learning for knowledge acquisition," *International Journal of Man-Machine Studies*, vol. 27, pp. 349–370, 1988.
- [24] C. Harley and R. Reynolds, "Analysis of E. Coli Promoter Sequences," *Nucleic Acids Research*, vol. 15, pp. 2343–2361, 1987.
- [25] G. Towell, J. Shavlik and M. Noordewier, "Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks," in *Proc. of the 8th National Conf. on Artificial Intelligence*, 1990, pp. 861–866.
- [26] C. M. O'Neill, "Escherichia coli promoters: Consensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organization.," *Journal of Biological Chemistry*, no. 264, pp. 5522–5530, 1989.
- [27] C. M. O'Neill and F. Chiafari, "Escherichia Coli promoters II. A spacing-class dependent promoter search protocol," *Journal of Biological Chemistry*, no. 264, pp. 5531–5534, 1989.
- [28] J. Ortega, "On the Informativeness of the DNA Promoter Sequences Domain Theory" (*Research Note*), vol. 2, pp. 361–367, 1995.
- [29] K. D. Hawley and R. W. McClure, "Compilation and analysis of Escherichia Coli promoter DNA sequences," *Nucleic Acids Research*, vol. 11, pp. 2237–2255, 1983.

- [30] T. Record. Personal communication. 1989.
- [31] S. Brunak, J. Engelbrecht, and S. Knudsen “Prediction of the human mRNA donor and acceptor sites from the DNA Sequence,” *J.Mol.Biol.*, 220, pp. 49–65, 1991.
- [32] 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*, vol. 3, 1991.
- [33] G. G. Towell, J. W. Shavlik and M. W. Craven, “Constructive Induction in Knowledge-Based Neural Networks,” in *Proc. of the 8th International Machine Learning Workshop*, 1991, pp. 213–217.
- [34] G. G. Towell, “Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction,” PhD Thesis, University of Wisconsin – Madison, 1991.
- [35] G. G. Towell and J. W. Shavlik, 1992; “Interpretation of Artificial Neural Networks: Mapping Knowledge-based Neural Networks into Rules,” In *Advances in Neural Information Processing Systems*, vol. 4, 1992.
- [36] D. J. Watson, H. H. Hopkins, W. J. Roberts, A. J. Steitz, and M. A. Weiner, *The Molecular Biology of the Gene. Benjamin-Cummings*, Menlo Park, CA, 1987.
- [37] S. Kasif, S. Salzberg, D. L. Waltz, J. Rachlin, D. Aha, “A Probabilistic Framework for Memory-Based Reasoning,” *Artificial Intelligence*, 104(1-2), pp. 287–311, 1998.
- [38] G. G. Towell, M. W. Craven and J. W. Shavlik “Constructive Induction in Knowledge-Based Neural Networks,” in *Proc of the 8th International Machine Learning Workshop*, 1991, pp. 213-217.

- [39] “Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences,” in *Proc. of the conf. on Advances in neural information processing systems*, 1990, pp. 530–536.
- [40] O. M. Noordewier, G. G. Towell and W. J. Shavlik, “Training knowledge-based neural networks to recognize genes in DNA sequences,” In *Advances in Neural Information Processing Systems*, vol. 3, 1991.
- [41] J. R. Quinlan, “Induction of Decision Trees,” *Machine Learning*, vol. 1, pp. 81–106, 1986.
- [42] Shavlik, J. W., R. J. Mooney, and G. G. Towell, “Symbolic and Neural Learning Algorithms,” An Experimental Comparison. *Machine Learning*, vol. 6, pp. 111–143, 1991.
- [43] Aguilar-Ruiz, J.S., Riquelme, J.C., Toro, M., “Evolutionary Learning of Hierarchical Decision Rules,” *IEEE Systems, Man and Cybernetics, Part B*, vol. 33, pp. 324 – 331, 2003.
- [44] S. Rampone, “Recognition of Splice-Junctions on DNA Sequences by BRAIN learning algorithm,” *Bioinformatics*, vol. 14, pp. 676–684, 1998.
- [45] P. Domingos and M. Pazzani, “Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier,” *Machine Learning*, vol. 29, pp. 103–130, 1997.