EFFICIENT CASE-BASED REASONING THROUGH FEATURE WEIGHTING, AND ITS APPLICATION IN PROTEIN CRYSTALLOGRAPHY

A Dissertation

by

KRESHNA GOPAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Computer Science

EFFICIENT CASE-BASED REASONING THROUGH FEATURE WEIGHTING, AND ITS APPLICATION IN PROTEIN CRYSTALLOGRAPHY

A Dissertation

by

KRESHNA GOPAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Thomas R. Ioerger
Committee Members,	James C. Sacchettini
	Jianer Chen
	Nancy M. Amato
Head of Department,	Valerie E. Taylor

August 2007

Major Subject: Computer Science

ABSTRACT

Efficient Case-Based Reasoning through Feature Weighting, and Its Application in Protein Crystallography. (August 2007)

Kreshna Gopal, B.Tech., Indian Institute of Technology, Kanpur;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Thomas R. Ioerger

Data preprocessing is critical for machine learning, data mining, and pattern recognition. In particular, selecting relevant and non-redundant features in high-dimensional data is important to efficiently construct models that accurately describe the data. In this work, I present SLIDER, an algorithm that weights features to reflect relevance in determining similarity between instances. Accurate weighting of features improves the similarity measure, which is useful in learning algorithms like nearest neighbor and case-based reasoning. SLIDER performs a greedy search for optimum weights in an exponentially large space of weight vectors. Exhaustive search being intractable, the algorithm reduces the search space by focusing on pivotal weights at which representative instances are equidistant to truly similar and different instances in Euclidean space. SLIDER then evaluates those weights heuristically, based on effectiveness in properly ranking pre-determined matches of a set of cases, relative to mismatches.

I analytically show that by choosing feature weights that minimize the mean rank of matches relative to mismatches, the separation between the distributions of Euclidean distances for matches and mismatches is increased. This leads to a better distance metric, and consequently increases the probability of retrieving true matches from a database. I also discuss how SLIDER is used to improve the efficiency and effectiveness of case retrieval in a case-based reasoning system that automatically interprets electron density maps to determine the three-dimensional structures of proteins. Electron density patterns for regions in a protein are represented by numerical features, which are used in a

distance metric to efficiently retrieve matching patterns by searching a large database. These pre-selected cases are then evaluated by more expensive methods to identify truly good matches – this strategy speeds up the retrieval of matching density regions, thereby enabling fast and accurate protein model-building. This two-phase case retrieval approach is potentially useful in many case-based reasoning systems, especially those with computationally expensive case matching and large case libraries.

ACKNOWLEDGMENTS

My most earnest acknowledgment must go to Thomas Ioerger, my long-time teacher, advisor, and chair of my M.S. and Ph.D. committees. His classes on artificial intelligence, machine learning, and intelligent agents were enlightening. I am immensely indebted to him for his great ideas, continual support, and valuable lessons in developing my research and writing skills. His guidance and commitment to science have been inspirational to me.

I am also very thankful to the other members of my committee, James Sacchettini, Jianer Chen, and Nancy Amato. It has been a privilege working with them. Their numerous comments and suggestions on my research were very helpful. Dr. Sacchettini's insights on crystallography and biochemistry were invaluable. I owe much of my understanding of computer algorithms to the outstanding classes taught by Dr. Chen and Dr. Amato.

I would also like to thank my colleagues in the TEXTAL group, especially Tod Romo, Erik McKee, and Jacob Smith, for the numerous informative discussions on computational crystallography.

It has been a great pleasure working with the faculty, staff, and students in the Department of Computer Science at Texas A&M University. In particular, I am very grateful to Bart Childs and Donald Friesen for their critical support in the advancement of my graduate studies. Also, I immensely enjoyed interacting with students in the various classes I taught. Finally, I would like to thank my friends Jeff Beckmann, Lydia Tapia, and Carlos Monroy for the enjoyable coffee and lunch breaks.

TABLE OF CONTENTS

CHAPTER Page

Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

LIST OF TABLES

CHAPTER I
INTRODUCTION

1.1. Feature Relevance

A central problem in machine learning is that irrelevant features (or attributes) tend to mask relevant ones, leading to inefficient learning and inaccurate predictions (Blum and Langley, 1997). Feature selection is thus receiving much attention, especially with the emergence of very large (and rich) data sets in applications like information retrieval, document classification, microarray analysis, proteomics, etc. (Liu, 2005). By eliminating irrelevant and redundant attributes, we can significantly reduce the volume of data needed for learning, and thereby improve the efficiency and robustness of learning, enhance prediction accuracy, augment comprehensibility of models built, and in certain cases, make data acquisition less expensive.

The importance of feature selection cannot be overemphasized. No learning algorithm can compensate for poor feature selection, like predicting gas mileage based on the color of a car. Feature selection has become indispensable for large, real-world data sets, sometimes involving as many as 100,000 features (in document classification or genomics, for example). Forman (2005) shows that how the performance of the naïve Bayes classifier degrades rapidly with addition of more than 50 features in the field of text classification. Even robust learning algorithms like support vector machines (Vapnik, 1998) will perform significantly better with irrelevant features discarded. Almuallim and Dietterich (1994) formally analyze the inductive bias that prefers hypotheses definable over as few features as possible in Boolean domains. They establish a lower limit for the number training examples required to ensure PAC-learning (Valiant, 1984) a concept in terms of the number of relevant and irrelevant features.

This dissertation follows the style of *Bioinformatics*.

Features are widely used to compare entities they are expected to describe. Cases or instances are typically compared by a similarity (or distance) function based on numerical features that describe the relevant aspects of the instances. Defining a suitable feature-based measure of similarity is a fundamental requirement in pattern recognition (Duda *et al*., 2001), instance-based learning (Aha, 1990), case-based reasoning (Kolodner, 1993; Leake, 1996), and other machine learning techniques (Mitchell, 1997). Typically, a preliminary set of many potentially useful features is defined first, and a subset of these features is then automatically selected.

The central difficulty in automated feature selection is the intractability of exhaustive search (there are 2^n possible subsets of n features). There are many algorithms that have been proposed for feature selection (Subramanian *et al*., 1997). These algorithms are commonly categorized into two major groups: *filters* and *wrappers*. This distinction is based on how feature subsets are evaluated. Filter methods (Kira and Rendell, 1992) perform the evaluation by using some properties of the features involved, such as correlations, information gain, dependencies, separability, etc. Wrapper methods (John *et al*., 1994) use part of the data sample to iteratively evaluate subsets of selected features by running the induction program itself, based on techniques such as cross-validation.

The feature selection problem can also be thought of as a *heuristic search* over a space of states, each of which represents a subset of features (Blum and Langley, 1997). Another approach to determine feature relevance is to apply a weighting function to features, which effectively assigns *degrees* of relevance to features (Littlestone, 1992).

1.2. Feature Weighting with SLIDER

In this work, we present a feature weighting algorithm called SLIDER (Gopal *et al*., 2005b; Gopal *et al*., 2004c), in which numerical weights are assigned to features to reflect their relevance in comparing instances (using a distance metric). Feature weighting is a generalization of feature selection. The latter can be viewed as feature weighting with only two possible weights, say 0 and 1. As we shall see later, our feature

weighting algorithm effectively does a lot of feature selection as well, by assigning weights equal to 0, or negligibly small. We argue that feature weighting is a refinement of feature selection, and can potentially improve over feature selection algorithms, at least in certain domains. We later provide empirical results to support this claim.

Like many feature selection and weighting algorithms, SLIDER starts with an initial set of weights, and iteratively selects new weights for evaluation, then does the evaluation, and retains the weights if they are the best ones found so far. It stops when finding even better weights seems unlikely, or is computationally too expensive. The critical decisions that SLIDER makes in every iteration are: (1) selecting a new weight vector for evaluation – out of an exponentially large set of options, and (2) evaluating the weight vector to determine if it outperforms the current best one.

To evaluate or compare two weight vectors, one common (and intuitive) strategy is to run the induction task itself on a known data set, using the different weight sets. This is effectively a wrapper method, which may be expensive (as is typical of wrappers, which must repeatedly run the underlying induction algorithm). SLIDER uses a simpler (and less expensive) strategy to evaluate a set of weights. It is a filter approach that uses a heuristic to assess how the feature-based distance measure ranks pre-determined matches (of training instances) relative to a set of mismatches:

The heuristic that SLIDER uses to evaluate a set of weights is as follows: given a training instance, we look at how well the distance metric (such as weighted Euclidean distance) ranks an instance known to be similar to the training instance, relative to a set of known different ones. This is done for a set of training examples, and the average rank of the similar instances is a measure of how good the weight vector is.

Exhaustive search through a space of weights is intractable. Therefore, given a true match and a true mismatch for an example, SLIDER focuses on only those weights which cause the example to be equidistant to its match and its mismatch in Euclidean space. These “crossover” weights are the ones that will influence the accuracy of ranking, since the evaluation based on ranking of matches is more likely to change at the crossover weights. This makes the search very effective i.e. by limiting the space of

weights to be searched, and identifying only the weights that are more likely to make a significant difference, the efficiency and effectiveness of learning are largely ensured.

We emphasize the fact that SLIDER tries to maximize the *number of instances* (from the training set) for which true matches are closer to training examples than mismatches in weighted Euclidean space. An alternative approach would be finding weights such the *aggregate distance* between instances and their matches is smaller than that between the instances and their mismatches. As we shall discuss later, this alternative formulation can be fairly efficiently solved by linear programming. However, we empirically show that SLIDER outperforms the linear programming approach. The objective function that SLIDER tries to optimize is a much harder problem (NP-hard, in fact), which justifies the use of a heuristic function (based on ranking) to guide the search for optimal weights.

Next we now discuss how SLIDER is used to improve the efficiency of case retrieval, based on a two-phase retrieval strategy that we propose for expensive case-based reasoning systems.

1.3. Two-Phase Case Retrieval

Many case-based reasoning systems need a large database of cases for coverage of a wide variety of problem instances. But large databases may cause degradation in efficiency, especially if the case matching function to determine similarity is computationally expensive (Smyth and Cunningham, 1996). Thus, we propose the use of an approximate, inexpensive, feature-based distance metric (like Euclidean distance) to filter a small number, say k, of potential matches, using the nearest neighbor rule (Fix and Hodges, 1951) – a more accurate matching function is then used to do the final ranking.

The feature-based similarity metric is expected to approximate a correct, objective, and usually expensive matching method. By weighting the features with SLIDER, the accuracy of the similarity metric is improved, and hence fewer potential matches needs to be filtered to ensure retrieval of true matches.

This two-phase method for case retrieval has been previously proposed, in different flavors and application domains. For example, Forbus *et al*. (2001) propose MAC/FAC (for "many are called but few are chosen"), a general strategy for efficient, similarity-based retrieval. Branting and Aha (1995) propose similar stratified or hierarchical case-based reasoning methods in the planning domain. Other similar applications include feature-based recognition of side chain contact environments (Mooney *et al*., 2005) and information retrieval (Jones *et al*., 2000).

In this work, we provide a theoretical analysis to relate the impact of feature weighting by SLIDER on the number of true matches in the top k ranked cases. We analytically show that by minimizing the mean rank of matches (for a training set), we are separating the distributions of distances between training cases and their matches as well as mismatches, such that the distance metric is effectively improved. This leads to an enrichment of the top k cases with more matches, and thus, the probability of getting a true match ranked below k is increased.

1.4. Efficient Case Retrieval: an Application in Protein Crystallography

We describe the use of SLIDER and the two-phase case retrieval method in a crystallographic protein model-building program called TEXTAL (Ioerger and Sacchettini, 2003). TEXTAL uses artificial intelligence and pattern recognition to automatically interpret *electron density maps* of proteins to determine their three-dimensional (3D) molecular structures. TEXTAL uses a database of about 50,000 spherical regions of electron density patterns and their structures (from previously solved maps) to interpret regions in a new electron density map (for a protein whose structure is not known). Given an electron density pattern in a spherical region of the unknown structure, we first select 400 density patterns out of the 50,000 cases in the database, using a weighted distance metric based on 76 features (determined by domain experts) that describe the electron density i.e. less than 1% of the database is filtered. The filtered

cases are then evaluated by a more expensive method (density correlation) to determine truly matching structural fragments.

The weights of the 76 features in TEXTAL are determined by SLIDER. It should be noted that a good match need not be the absolute best one according to the objective metric. It can be any of the top few matches (based on a tolerance on how high we wish the objective evaluation to be to qualify as a match). Given a query pattern, our aim is to obtain as many good matches as possible (anywhere) in the top k, since the expensive objective will re-rank the top k matches, and identify the truly good ones. The effectiveness of this case retrieval system largely hinges on the ability of SLIDER to assign appropriate weights to the features.

SLIDER and the case retrieval method proved to be important in the effectiveness of TEXTAL, which is used in crystallography laboratories around the world. By assigning feature weights judiciously, the distance metric is improved, and matches are efficiently retrieved from a large database.

1.5. Dissertation Outline

The rest of this document is organized as follows:

- Chapter II discusses the feature selection and weighting problem in general, and summarizes the main related work in the literature on artificial intelligence and machine learning. We also provide a brief overview of the two machine learning paradigms used in this work, namely case-based reasoning and nearest neighbor learning.
- Chapter III provides more details on the application domain. First, we discuss the principles and challenges of X-ray crystallography methods to determine protein structures, and we summarize other work related to automated interpretation of electron density maps. The TEXTAL system is then described in its entirety. This will show how the case-based reasoning component is related to the other parts of the system, and will provide the context and motivation for efficient case retrieval

and feature weighting. We also describe the features that experts defined in this domain, and discuss the rationale behind these choices.

- Chapter IV describes the SLIDER algorithm in details. We also discuss a two-phase case retrieval strategy that sets the context for SLIDER. We also present a feature weighting approach based on linear programming, which is closely related to SLIDER.
- Chapter V provides a probabilistic analysis to support the heuristic used to evaluate weights in SLIDER, and shows how the method leads to an increased probability of retrieving a true match from a database.
- Chapter VI presents empirical results in the protein crystallography domain from a machine learning perspective. The experimental setup is first described and then we provide an empirical analysis of SLIDER and the two-phase case retrieval strategy. We also compare SLIDER to other well-known feature selection and weighting algorithms.
- Chapter VII presents more empirical results with emphasis on the domain of protein crystallography. We discuss the contribution of the methods we propose to the field of automated electron density map interpretation.
- Chapter VIII concludes this work by summarizing its salient features, discussing the limitations of the proposed methods, and describing prospects for future work.

CHAPTER II
FEATURE SELECTION AND WEIGHTING

2.1. Feature Relevance

Automated reasoning and learning systems need information to work effectively. But too much information may cause accuracy and efficiency to degrade. Thus determining what information is relevant is an important endeavor in machine learning and data mining – not only it simplifies the problem, but also avoids wastage of computational effort. The human brain, it is believed, spends about 90% of its effort in discarding inputs obtained through the senses (Subramanian *et al*., 1997). Assessment of relevance in learning systems can be made for various types of information – a training example, a proposition, an inference rule, a feature, past experience (in case-based reasoning), etc. Any of these entities of information can be generally defined as irrelevant for a task if the correct output does not change by a significant amount if the entity is changed; otherwise the information entity is defined as relevant (Galles and Pearl, 1997). This definition of relevance is by no means comprehensive. In fact, a general framework for reasoning about relevance is hard to formulate. For instance, Kohavi and John (1997) show that determining the relevance of features in an induction task cannot be made independently of the induction algorithm. In (John *et al*., 1994), the difficulties of defining relevance are highlighted, and different degrees of relevance (weak and strong) are proposed.

Relevance of features is widely studied in machine learning and data mining tasks (Blum and Langley, 1997; Pyle, 1999) like pattern classification (Jain and Zongker, 1997), instance-based learning (Aha, 1990), and case-based reasoning (Riesbeck and Schank, 1989). In these applications, patterns or examples are typically compared to detect similarities. Potentially useful features are generally defined by an expert, or extracted by automated techniques (Liu and Motoda, 1998), and a subset of these features is automatically selected (or highly weighted), based on their relevance to the task at hand (Aha, 1998). This reduction in dimensionality generally speeds up the

learning algorithm, improves predictive accuracy, and makes the results more comprehensible. Irrelevant features tend to mislead pattern matching; the problem is particularly acute in nearest neighbor methods, where irrelevant features can seriously hamper learning (Langley and Iba, 1993). For instance, Forman (2005) shows a decrease of over 10% in classification accuracy with irrelevant features in a text classification domain, using the naïve Bayes classifier.

Determining an optimal subset of features is intractable, since there are 2^n possible subsets of n features. Many feature selection problems have been shown to be NP-hard (Blum and Rivest, 1992). A dimensionality in the order of a hundred is considered large, and an exhaustive search of feature subset space is computationally prohibitive. However, in recent applications there are datasets with tens to hundreds of thousands features available. These trends pose new challenges to scale up and cope with the *curse of dimensionality*.

Another recent trend is the increasing need and use of feature selection for *clustering* with unlabeled data i.e. unsupervised learning (Dy and Brodley, 2004), as opposed to feature selection for *classification* with labeled data (supervised learning).

Feature selection has been widely applied to a variety of applications. In text classification (Forman, 2003), free text documents are categorized in to pre-defined groups. Documents are usually represented by a "bag-of-words", with typically tens of thousands words (or features). Given the huge volume of documents available online, such applications are gaining significant practical importance. Another *information retrieval* area that is benefiting from feature selection is content-based image retrieval (Rui *et al*., 1999; Yavlinsky *et al*., 2004), where large collections of images are handled based on visual features rather than text-based annotations.

The rapid growth of genomic databases is also presenting many opportunities (and challenges) for feature selection. For instance, gene expression microarray experiments (Berens *et al*., 2005) handle the expression levels of thousands or tens of thousands of genes. Thus, dimensionality reduction by filtering relevant genes is necessary for

effective data mining. The problem in this area is particularly challenging because the number of samples or experiments (i.e. training instances) is often very limited.

2.2. Filters and Wrappers

In general, feature selection methods proposed in the machine learning literature can be categorized into two major groups:

1. *Filter* methods try to build classifiers that take into account some properties of the features involved, such as correlations, dependencies, separability, information gain, etc. (Liu and Motoda, 1998). They evaluate the goodness of a feature subset (or a single feature) by looking at the intrinsic characteristics of the training data, independently of the induction algorithm (like classification). The feature selection is done before the induction step; thus irrelevant features are filtered out before induction occurs. For instance, in RELIEF, Kira and Rendell (1992) use a statistical measure of relevance, based on similar and different instances, where similarity and difference is measured by Euclidean distance.
2. *Wrapper* methods use part of the data to iteratively evaluate the subset of selected features using performance on the induction algorithm for evaluation; this is done by techniques such as cross-validation. In wrapper methods, features are selected by taking the bias of the induction algorithm into account (John *et al*., 1994). The advantage of wrapper methods is that the induction method is expected to provide a better estimate of accuracy, as compared to filter-based methods that have imperfect inductive biases. The disadvantage of wrappers is their high computational cost.

Wrappers solve the real problem many times over to search for the relevant features, and can thus be very time-consuming. Filters are usually much faster since they try to solve a simpler version of the problem; but this surrogate may not be an appropriate measure of true performance. There are several *hybrid* algorithms that have been proposed to combine the two models, for instance by pre-determining candidate feature subsets using a fast and independent performance measure (i.e. a filter approach), and

making the final selection of the best subset using the induction algorithm, as in a wrapper (Das, 2001).

2.3. Feature Selection as a Heuristic Search

The feature selection problem can also be thought of as a *heuristic search* over a space of states, each of which represents a subset of features (Blum and Langley, 1997). In this search paradigm, the following need to be defined: (1) the starting state, (2) the organization of the search, or how to generate feature subsets, (3) how to evaluate alternative feature subsets, and (4) when to stop. For example, in *forward selection*, we start with an empty feature set and iteratively add features. In *backward elimination*, we start with all features, and then remove one feature at a time. The search can also start from both ends, with features added and removed simultaneously (i.e. *bidirectional*). It is possible to get trapped in local optima; this can potentially be addressed by starting from a randomly generated state, or introducing stochasticity through techniques like *random-start hill climbing* or *simulated annealing* (Russel and Norvig, 1995). In (Devijver and Kittler, 1982) and (Liu and Yu, 2005), several methods to organize the search and reduce the search space are presented.

2.4. Feature Weighting

In feature weighting, degrees of perceived relevance are assigned to features. Feature selection is a specific case of feature weighting, where only two alternative weights (say 0 and 1) are used. Most feature weighting methods employ some variation of *gradient descent*, in which all the weights are updated simultaneously e.g. the perceptron update rule (Rosenblatt, 1958), least-mean squares (Widrow and Hoff, 1960), and neural networks (Baluja and Pomerleau, 1997). For instance, in Winnow (Littlestone, 1992), weights are updated in a multiplicative manner, rather than an additive one as in a perceptron. Feature weighting has also been implemented as a filter method, for example by using conditional probability distributions (Stanfill, 1987), or information-theoretic metrics (Daelemans *et al*., 1994). Finally, a wrapper approach to feature weighting can

also be employed, like in DIET (Kohavi *et al*., 1997), where instead of feature subsets, a space of discrete space weights is searched.

Weighting of features when computing distance corresponds to stretching the axes in the Euclidean space, such that the axes for more relevant features are lengthened, and shortened for axes of less relevant ones. A major disadvantage of this increase in the number of possibilities of defining distance is a higher risk of *overfitting* i.e. the training data is modeled too well, such that the performance over new problems is poor.

Blum and Langley (1997) argue that feature selection is most natural when the result is expected to be understood and interpreted by humans, or fed into another algorithm. Feature weighting, on the other hand, are generally purely motivated to enhance the performance of the induction algorithm. But it has also been argued that there may be little benefit in increasing the number of possible weights beyond two (0 and 1). More fine-grained weighting may degrade performance due to overfitting (Kohavi *et al*., 1997).

Another approach is discriminate among the contribution of various features is through *feature ranking*, for instance by using metrics like the Pearson correlation coefficient, or Fisher's criterion (the ratio of the between-class variance to the within-class variance). Many algorithms adopt this approach to feature selection because of its simplicity, scalability, and good predictive accuracy in real-world datasets (Guyon and Elisseeff, 2003; Bekkerman *et al*., 2003; Weston *et al*., 2003).

2.5. Context-Sensitivity and Feature Interaction

A different type of criterion for determining feature or attribute relevance is sensitivity to the context. Different features may be relevant for different instances, making attribute relevance a function of the instance and sensitive to the location in the feature space. This motivates local feature weighting methods (Domingos, 1997; Howe and Cardie, 1997; Grenier *et al*., 1997). Nonetheless, in this work we assume that relevance of features is global, independent of the instance.

Another facet of the feature weighting problem is feature interaction. Sometimes information is shared among attributes, and one attribute is effectively meaningful when considered in conjunction with other attributes (Ioerger, 1999; Jakulin and Bratko, 2004). Thus, an attribute may appear irrelevant when analyzed independently, but its relevance manifests itself when combined with other attributes.

2.6. Classification vs. Ranking

Learning a weighted distance metric for the purpose of nearest neighbor classification has been widely studied and successfully applied (Short and Fukunaga, 1980; Stanfill and Waltz, 1986; Paredes and Vidal, 2006; Kohavi *et al*., 1997). Nonetheless, we often wish to retrieve instances, such as relevant documents from the web, or potentially useful planning solutions from a plan library. In these applications, there is no explicit classification, and similarity is usually measured by a continuous metric – the objective is to rank, rather than classify instances. In this work, we focus on finding relevant features for the purpose of ranking and retrieval, rather than classification.

The optimization of ranking is a widely studied problem. In (Burges *et al*., 2005) a gradient descent method called RankNet is proposed to rank search results from the Web. Joachims (2002) uses a support vector machine approach for learning retrieval functions of search engines, utilizing clickthrough data for training. RankBoost is an algorithm that combines a collection of ranking or preference functions based on a boosting approach to learning (Freund *et al*., 2003). In (Cohen *et al*., 1997) and (Dwork *et al*., 2001), other methods are proposed to combine multiple preference functions to create an ordering, and applied to Web searches.

2.7. Previous Work Related to SLIDER

SLIDER's strategy to optimize the weighted Euclidean distance for the purpose of case ranking and retrieval can be related to research in the following two areas: feature selection, and learning how to rank instances.

We earlier presented various approaches to feature selection, such as filters and wrappers. In particular, RELIEF (Kira and Rendell, 1992; Kokonenko, 1994; Sun and Li, 2006) adopts a filter approach to select features. Somewhat like SLIDER, RELIEF uses training data that consists of triplets of an instance X, its near-hit, and its near-miss (i.e. two instances in the close neighborhood of X in Euclidean space, where the near-hit is in the same class as X, whereas the near-miss is in a different class). RELIEF uses the training data to iteratively compute a feature weight vector representing the relevance of the features, and a feature is selected if its relevance value is above a threshold. Given each training triplet, the relevance value for each feature is adjusted by an amount directly proportional to square of the distance (based on that feature) between X and its near-hit and near-miss. In contrast, SLIDER does not adjust weights based on an aggregate of distance values, but rather on the number of times a training instance is closer to its match in Euclidean space, as compared to a mismatch.

There are other methods that, like SLIDER, concentrate on learning to order (rather than classify). For instance, Cohen, Schapire and Singer (1997) address the problem of finding an ordering that agrees with a known preference judgment or feedback (which is analogous to our strategy of looking at known matches and mismatches). They first learn a binary preference function, which returns a numerical measure of how certain one instance should be ranked before another. Then, the learned preference function is used to order a set of instances such that the total ordering agrees, as much as possible, with all pairwise preference judgments (based on the learned preference function). The problem of finding a total order that agrees best with the preference function is shown to be NP-hard. We should point out that in SLIDER, the ultimate aim is not a total order of instances – rather it is the more modest objective of ranking the (rare) matches as best

possible. Nonetheless, in our attempt to optimize feature weights, we try to maximize the number of instances (in a training set) with correct pairwise preference judgments.

In the next two sections, we provide a brief overview of the two machine learning techniques that underlie our feature weighting algorithm, namely case-based reasoning and nearest neighbor learning. In this work, weighted features are used to measure similarity between instances through nearest neighbor learning. This enables efficient case retrieval in case-based reasoning systems that involve large databases.

2.8. Case-Based Reasoning

Case-based reasoning, or CBR (Riesbeck and Schank, 1989; Kolodner, 1993; Leake, 1996) is a problem solving paradigm where domain expertise is captured in a database of *specific* knowledge of concrete past situations (or cases), instead of general rules that relate problems and their solutions. Each case typically contains descriptions of the problem and its solution; the knowledge and reasoning that were needed to solve past problems are not recorded, but are implicit in the solutions.

CBR is a type of *instance-based learning* (Aha, 1990) where a general, explicit description of the target function is avoided. This approach is also referred to a *lazy learning* (Aha, 1997), since all inference (such as classification) is deferred to the time when a new problem is encountered. The advantage of this kind of delayed learning is that instead of formulating solutions for the entire space of problems, reasoning is done locally (and differently) for every new problem. But the major disadvantage is that all computation is done at the time when a problem needs to be solved, and not when cases themselves are encountered.

Thus case retrieval can be computationally very expensive in CBR systems, especially when case matching is expensive and the database of solved cases is large. Typically, large case libraries are necessary for good problem coverage and quality solutions (Smyth and Cunningham, 1996). A lot of CBR research is devoted to the organization and indexing of the case library to make case retrieval as quick and accurate as possible.

A CBR approach to solve a new problem typically involves the following main processes:

1. *Case matching and retrieval*: The new problem is compared to the library of past cases, and the most similar case (or cases) is retrieved. A set of relevant problem descriptors needs to be defined to match cases. A similarity measure is typically used to compare cases, and some sort of similarity threshold is needed to select the best cases.
2. *Case reuse*: The information and knowledge in cases retrieved are used to solve the new problem.
3. *Case revision*: If necessary, the solution is revised and adapted. Revising the solution generated by the reuse process is necessary when the solution proves incorrect. This also provides an opportunity to learn from failures (Hammond, 1989).
4. *Case retention*: The new problem and its solution are (optionally) retained as a new case, depending on how useful it is expected to be in solving future problems. This step involves deciding what information to retain, how to retain it, and how it should be indexed for future retrieval.

2.9. Nearest Neighbor Learning

Nearest neighbor (NN) learning is a simple form of instance-based, lazy learning that has been thoroughly studied and applied (Cover and Hart, 1967; Fix and Hodges, 1951). The nearest neighbor algorithm represents all instances or cases as points in the *n*-dimensional space, where each of the *n* dimensions correspond to a numerical feature that describe the instance i.e. any instance in the space is represented by a vector of *n* features. The nearest neighbors of an instance are those that are closest to the instance in the space, based on a distance measure such as Manhattan, Euclidean (or Minkowski distance, in general), Mahanalobis, and other more sophisticated measures (such as those based on statistical properties of the data).

Nearest neighbor learning can be used to classify instances into one of a finite set of classes i.e. the target function is discrete-valued. This function can be real-valued as well, where the true difference between two instances is in terms of a continuous value.

It may be misleading to consider only the single closest neighbor (as in 1-NN) because the chances of error may be very high. Typically we consider *k* nearest neighbors (*k*-NN). These *k* neighbors may have the same weight, or can be weighted differently, for example by assigning greater weight to closer neighbors. If weights are to be assigned, then all examples can be considered (Shepard, 1968) – examples far way will have practically no influence in classification or predicting the target function.

Nearest neighbor algorithms have proven to be effective for many practical problems. It is robust to noise, especially since averaging over *k* neighbors reduces the impact of noisy examples. Also, it handles large training sets well. Nonetheless, being a lazy learner that delays all computation until a new instance is encountered, the efficiency of learning can be an issue, which may require better indexing methods, such as *kd-trees* (Bentley, 1975).

An important problem in nearest neighbor methods is that the distance between instances is based on all features. While other learning methods (such as decision trees or rule-based systems) narrow down the set of features when formulating the hypothesis, nearest neighbor learning generally does not discriminate between relevant and irrelevant features. Irrelevant features are effectively forms of noise, and be can be very misleading in the measurement of similarity between instances. As the number of features (irrelevant ones, in particular) increases, the *curse of dimensionality* becomes very serious.

CHAPTER III
X-RAY PROTEIN CRYSTALLOGRAPHY

3.1. Proteins

Proteins are large and complex macromolecules that are essential to the chemical processes in living systems. For example, enzymes are proteins that are responsible for catalyzing the thousands of chemical reactions in the cell. *Lysozyme* is an example of an enzyme that helps fight bacterial infections. Proteins also include hormones that regulate metabolism e.g. *insulin* stimulates the uptake of excess glucose by liver cells for conversion to glycogen. Other functions of proteins include transportation (e.g. *hemoglobin* found in red blood cells carry oxygen from the lung to the tissues), mechanical work (e.g. *actin* and *myosin*, which enable contraction in the muscle), etc. Proteins also play signaling, regulatory, and immune-response roles in cells.

Proteins are made up of *amino acids* (also called *residues*) that are linked through covalent chemical linkages known as *peptide* bonds. The amino acids form linear polymeric structures called *polypeptide* chains. Typically these chains contain 100 to 1000 amino acids, arranged in a specific order for a given protein. The average number of residues in natural proteins is about 300. There are twenty unique amino acids that are commonly found in nature. Every amino acid has the same generic structure (Figure 3.1): There is a central carbon atom (called Cα) that is bonded to a hydrogen atom, a carboxyl group, an amino group, and a side chain or "R" group. Thus the α carbon atom in amino acids is *chiral* (except when the R group is a hydrogen); all amino acids found in proteins occur in the L-configuration about the chiral carbon atom (Murray *et al*., 2000). The R group gives an amino acid its particular characteristics. The physicochemical properties of a protein are determined by the analogous properties of the amino acids in it.

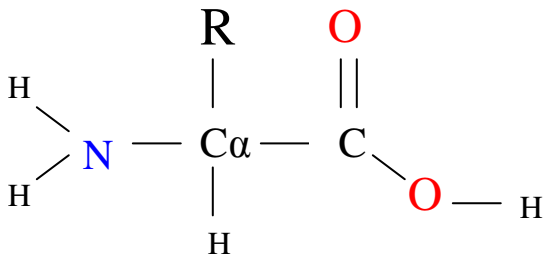

Fig. 3.1. Generic structure of an amino acid. The α Carbon atom (Cα) is linked to a hydrogen atom, an amino group (-NH_2), a carboxyl group (-COOH), and a side chain (or "R" group).

In the polypeptide chain of a protein, the carboxyl group of one amino acid links with the amino group of the next amino acid in the protein sequence. A protein is typically viewed as a *main chain* or *backbone* (which is the chain of Cα atoms, connected through another carbon and an oxygen), with side chains hanging off the Cα atoms.

The side chains vary in their complexity and properties. For example, the side chain of *glycine* is simply a hydrogen, whereas the side chain of *tryptophan* is aromatic (i.e. it contains the benzene ring). Amino acids are classified by the chemical nature of their side chains. One useful classification is in terms of polarity and affinity to water: the polar amino acids have side chains that interact with water (hydrophilic), while those of non-polar amino acids do not (hydrophobic).

The term *primary structure* denotes the precise linear sequence of amino acids that constitutes the polypeptide chain of the protein molecule. The interaction of sequential amino acid subunits results in *secondary structures*, such as part of a chain twisted into a linear helix (called the α-helix). Another common secondary structure consists of two or more sequences within the same protein that are arranged adjacently and in parallel, but with alternating orientation such that hydrogen bonds can form between the two strands. These are called β-pleated sheets. The side chains in a β sheet structure may also be arranged such that the adjacent side chains on one side of the sheet are mostly hydrophobic, while those adjacent to each other on the alternate side of the sheet are

mostly hydrophilic. Some sequences involved in a β sheet take a *hairpin turn* in orientation, when traced along the backbone.

Every protein has a characteristic 3D shape, known as the *tertiary structure* (Figure 3.2). Fibrous proteins, such as collagen, consist of roughly parallel polypeptide chains forming fibers or sheets, and are usually not soluble in water. Globular proteins tend to be tightly and extensively folded into an ellipsoid or sphere. Two or more chains that are linked by weak forces and that behave mostly as a single structural and functional entity are said to exhibit *quaternary structure*.

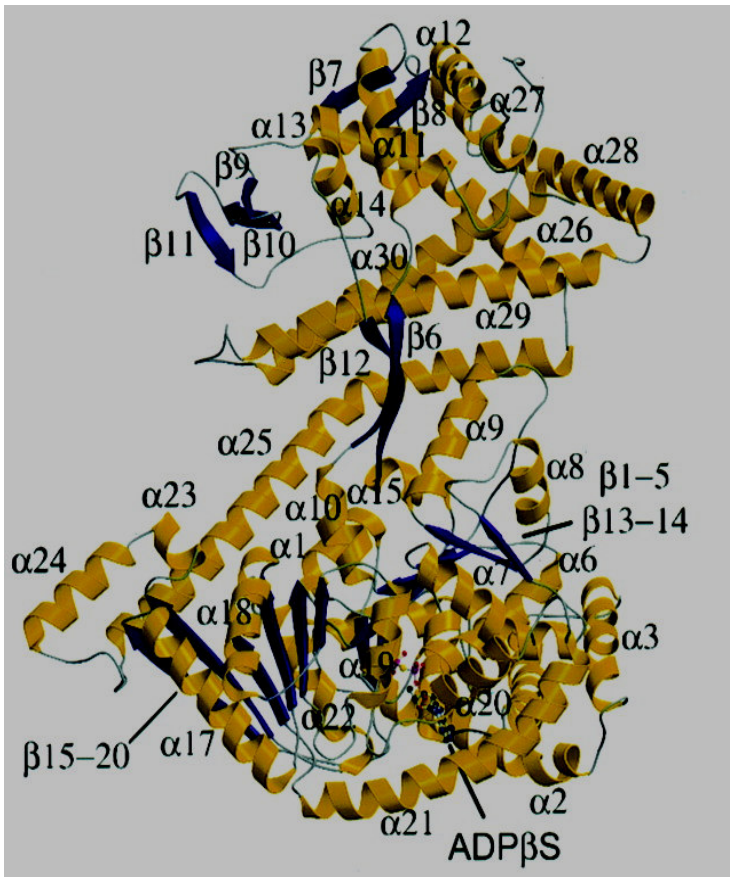

Fig. 3.2. Secondary and tertiary structures in proteins. Shown is the cartoon of *mycobacterium tuberculosis SecA*, a protein whose structure was solved in the Sacchettini Lab at Texas A&M University.

Proteins may also shift among several similar structures in performing their biological function. These tertiary or quaternary structures are usually referred to as *conformations*, and transitions among them are called conformational changes**.**

The precise 3D shape of a protein molecule is referred to as its native state; the protein generally needs to be in its native structure to perform its biological function. A protein usually folds into its native state based only on the sequence of amino acids. Sometimes special molecules called *chaperones* help proteins fold into the correct 3D shape, or conformation. In a seminal study, Anfinsen and his collaborators (1954) showed that the sequence of amino acids in a polypeptide chain determines the folding pattern. That is, the intricate process of protein folding could be completely explained by the physical and chemical interactions among the residues.

If the 3D structure of a protein is altered (by changing the temperature, *p*H or salt concentration for instance), the protein is said to be denatured, which usually results in a loss of biological activity. Protein misfolding can cause diseases like Alzheimer's, cystic fibrosis, mad cow disease, and many types of cancer.

Diseases are also caused by mutations in a person's DNA, since these errors often change the amino acid composition of proteins encoded by genes. For instance, cells with altered enzymes may lose the ability to catalyze certain chemical reactions, or cells with a mutant receptor may fail to bind to that receptor's ligand.

3.2. Structural Genomics

Determination of the native 3D structure of proteins, and of how folding (and misfolding) occurs are important endeavors. Knowledge of a protein structure is essential to fully understand how the protein functions, and how to change the function by protein engineering. The structure helps understand the protein's role in diseases, and the design of drugs (for example, inhibitors).

In the past few years, the genomic sequence databases have grown phenomenally. The entire genomes of the human, plus those of various organisms are now known and accessible. Keeping up the protein structure determination rate with this growth of

genomic information has become a major challenge. In fact, the ratio of solved crystal structures to the number of discovered proteins is about 0.15 (Tsigelny, 2002). This wealth of sequence information can serve as the foundation to developing a comprehensive view of the protein structure universe. S*tructural genomics* (Burley, 1999) is a worldwide initiative aimed at determining a large number of protein structures mainly by high-throughput X-ray crystallography and Nuclear Magnetic Resonance (NMR) spectroscopy methods. The initiative is expected to yield a large number of experimental protein structures (tens of thousands) and an even larger number of calculated comparative protein structure models (millions). This will have tremendous impact on all areas of biological science, including human health and diseases, natural ecosystems, plant genetics, etc.

3.3. X-ray Crystallography

X-ray crystallography is the most widely used technique to accurately determine the structure of proteins and other macromolecules. It is based on the fact that X-rays can be diffracted by crystals. X-rays are scattered by the electrons around atoms, and this scattering from periodic arrangements of atoms in a crystal results in diffraction patterns. These patterns are detected and used to reconstruct the electron density, from which the macromolecular model (i.e. atoms and their coordinates) can be determined. X-ray crystallography usually produces accurate molecular structures, from global folds to atomic-level bonding details.

Crystallographic structure determination involves many steps: first the protein has to be isolated, purified, and crystallized. After crystallization, X-rays are shone through the crystal and diffraction data (intensities of diffraction spots) are collected. The diffraction pattern can in principle be used to reconstruct a map of the electron density around the molecule by inverse Fourier transform, although phases for the structure factors (Fourier coefficients) have to be estimated. This is necessary because the diffraction spots contain information only about the amplitudes of diffracted waves; the phase information, which is also required for calculating the map, is lost. Approximate phase information can be

obtained by a variety of experimental techniques, including *multi-wavelength anomalous diffraction*, *multiple isomorphous replacement*, and *molecular replacement* (McRee, 1999a).

The sample of diffraction spots at which intensities can be collected is limited, which constrains the degree to which atoms can be distinguished from one another. This imposes limits on the *resolution* of the map, measured in Å (or Angstrom, where 1 Å = 10^{-10}m). The resolution is determined by a variety of experimental factors. At 4 Å, the backbone may appear connected, but side chains might not be very distinguishable. At 3 Å, it may be possible to discriminate a few residues. In 2 Å maps, all residues usually appear quite distinct, and at 1 Å, we can even see the density around individual atoms. In the majority of cases, data can be collected only at medium resolution. Thus, the major focus (and challenge) of automated map interpretation is for the 2-3 Å resolution range.

There have been many recent improvements in many areas of X-ray crystallography. For instance, developments of gene technology allow expression of large amounts of proteins, which enables trying a larger variety of crystal growth conditions; this improves the chances of obtaining suitable crystals. The process can be further speeded up by the use of crystallization robots. Other significant advances include new X-ray detectors (like electronic area detectors) and X-ray sources (such as high energy electron or position synchrotrons). The use of more intense X-ray radiation allows for rapid data collection, and thus less damage to the crystal, which can therefore be used for longer periods and in smaller sizes.

However, the model-building step in protein crystallography remains a major bottleneck. Manual interpretation of maps is time-consuming and error-prone, especially if the data quality is poor. The automation of map interpretation has proven to be difficult because it requires extensive domain knowledge and experience.

3.4. Automated Electron Density Map Interpretation

The final step in protein crystallography is model-building, or determining coordinates of atoms from an electron density map. Model-building is typically a two-stage process. First, the path of the polypeptide chain through the density is determined. Then, amino acids are fitted into the density map; each amino acid has several rotational degrees of freedom and can adopt various conformations. The directionality of the chain must also be determined. Computer graphics programs are widely used to visualize and manipulate the model as well as the density in 3D. Contoured meshes are used to portray the density at various levels of detail (Figure 3.3). Fitting of maps is a decision-making process that must take into account factors like the quality of the electron density, stereochemistry of amino acids, recognition of secondary structures, etc.

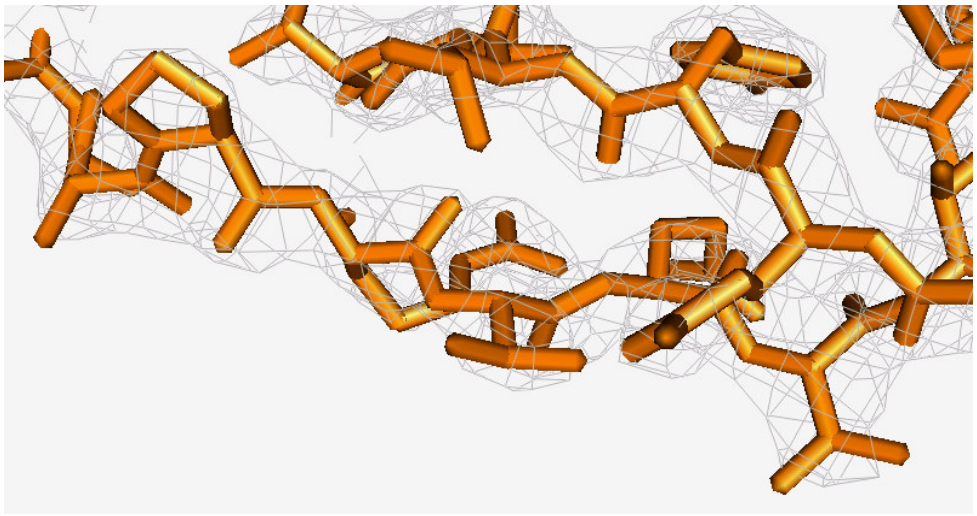

Fig. 3.3. Example of electron density around a fragment of a protein. The fragment shown consists of two strands of a β-sheet in *exocytosis-sensitive phosphoprotein*. The electron density map has been calculated from the solved structure at 2.8 Å. This image was generated with PyMOL (W.L. DeLano, http://www.pymol.org).

Once a preliminary structure has been built, it can often be used to obtain better phase information and generate an improved map, which can then be re-interpreted. This

process can go through many cycles, and it may take weeks or sometimes months of effort for an expert crystallographer to produce a refined structure, even with the help of molecular 3D visualization programs. The difficulty of manual structure determination depends on factors like the size of the structure, resolution of the data, complexity of the molecular packing, etc. There can be many sources of errors and noise, which distort the electron density map, making interpretation difficult (Richardson and Richardson, 1985). There is also a subjective component to model-building (Mowbray, 1999; Branden and Jones, 1990); decisions of an expert are often based on what seems most reasonable in specific situations, based on background knowledge and experience.

Various tools and techniques have been proposed for automated map interpretation: treating model-building and phase refinement as one unified procedure using free atom insertion in ARP/wARP (Perrakis *et al*., 1999), fitting α-helices and β-sheets, followed by local sequence assignment and extension through loops in MAID (Levitt, 2001), template matching and iterative fragment extension in RESOLVE (Terwilliger, 2002), Cα tracing, fuzzy logic sequence assignment and real-space torsion angle refinement in X-AUTOFIT (Oldfield, 2003), artificial intelligence and expert systems (Feigenbaum *et al*., 1977; Terry, 1983), molecular-scene analysis (Leherte *et al*., 1997), using templates from the Protein Data Bank (Jones *et al*., 1991), template convolution and other FFT-based approaches (Kleywegt and Jones, 1997), using a database of protein domains in DADI (Diller *et al*., 1999), artificial intelligence and pattern recognition techniques in TEXTAL (Ioerger and Sacchettini, 2003), etc. Many of these approaches require user-intervention or work well only with high quality data. TEXTAL, however, has been designed to be fully automated, and to work with medium quality data (around 2.8 Å resolution). Most maps are, in fact, noisy and fall in the low-medium resolution category due to difficulties in protein crystallization and other limitations of the data collection methods.

3.5. Database Approaches in Structural Bioinformatics

The determination of the 3D structure of proteins is an important and challenging problem, especially with the rapid growth of protein sequence data derived from large-scale DNA sequencing efforts, such as the Human Genome Project (Collins *et al*., 2003). Despite the advances in structural genomics (Chandonia and Brenner, 2006), the output of experimentally determined protein structures (typically by X-ray crystallography or NMR spectroscopy) is lagging far behind the output of protein sequences.

A number of computational approaches have also been proposed for protein structure prediction from the sequence (Baker and Sali, 2001). For instance, *ab initio* techniques, such as molecular dynamics, tries to solve a structure from basic principles, such as finding the minimum energy configuration that a protein folds into. But these methods typically require vast computational resources, and have thus been applied to very small proteins. Comparative techniques (such as homology modeling and threading) use solved structures as templates to predict new structures (Bowie *et al*., 1991; Zhang and Skolnick, 2005). Despite advances in these techniques, protein structures are still largely solved by time-consuming and expensive experimental methods: around 90% of the structures available in the Protein Data Bank have been determined by X-ray crystallography. Roughly 9% of the known protein structures have been obtained by NMR.

The arguments for database approaches in structural bioinformatics are compelling: although there are a huge number of actual proteins (millions), there is a limited set of structural motifs that exist in all proteins (thousands). A significant portion of new protein structures contain folds that are related to those seen before. Furthermore, the underlying difficulty of protein structure determination or prediction is the astronomically large number of possible conformations a polypeptide chain can fold into, given that an unfolded polypeptide chain has a very large number of degrees of freedom (Levinthal, 1968). Thus, exploiting existing solutions is a natural approach to solve protein structures. This is the basis of comparative (or homology) modeling, and related

methods such as sequence comparison (Altschul *et al*., 1997) and Hidden Markov Models (Krogh *et al*., 1994).

In crystallographic model-building, there are a number of database approaches that have been successfully used. One of the first applications of structural databases was described by Jones and Thirup (1986). They observed that some structural components could be constructed from other similar ones (such as the two turns in *retinol*-binding protein that could be reconstructed from only three other protein structures). They went on to create a database consisting of a small number (initially 37) of well-refined high-resolution protein structures that could be used to construct new protein models using crystallographic data, NMR data, or homology modeling (Jones and Thirup, 1986).

Another example is ESSENS (Kleywegt and Jones, 1997), a routine that interprets electron density maps by recognizing given templates at each point of the map through an exhaustive six-dimensional search in real space. This idea was extended by Cowtan (1998), who used FFT methods to improve the search in reciprocal space in the FFFEAR program.

The idea behind DADI (Database Assisted Density Interpretation) is similar. DADI interprets electron density maps through the use of a small database of protein domains (Diller *et al*., 1999). The rationale behind DADI's approach is to exploit redundancy in protein domains by first working with entire domains, then with the secondary structure elements of these domains, and finally with individual residues of the secondary structure.

RESOLVE (Terwilliger, 2003) is another program that interprets electron density maps by using existing solved structures. First, secondary structures (α-helices and β-strands) are located and constructed from templates derived from refined protein structures. Then, fragment libraries consisting of sequences of three amino acids are used to extend the initially determined helices and strands.

McRee (1999b) also proposed methods to build the main chain and side chains by fitting an electron density map with fragments from a main chain library, and a rotamer library of side chains. A more recent system is Automatic Crystallographic Map

Interpreter (ACMI), which uses a probabilistic model known as a Markov field to represent the protein (Dimaio *et al*., 2006). Residues are modeled as nodes in a graph, while edges model pairwise structural interactions. ACMI uses a library of templates (of 5 connected residues) to match side chains locally.

TEXTAL (Ioerger and Sacchettini, 2003) uses of database of 200 proteins (electron density maps and their structures) to build side chains in a new map by finding matching (local) density patterns, and retrieving their corresponding structures. It should be noted that TEXTAL finds matching side chains locally, independent of neighboring side chains, and of the sequence. This is different from other database approaches, like RESOLVE and ACMI, where alignment with the sequence is factored in when matching templates are selected. In TEXTAL, there is an independent sequence alignment step that is performed *after* building a preliminary model.

3.6. Overview of the TEXTAL System

TEXTAL is a program that uses artificial intelligence and pattern recognition techniques to fully automate electron density map interpretation, thereby saving considerable effort and time required by human experts to solve a protein structure. TEXTAL takes a real-space pattern recognition approach to model-building. It has been designed to be robust to noise, and has been optimized for medium resolution X-ray diffraction data (in the 2.4 to 3.0 Å range).

TEXTAL tries to mimic the typical strategy employed by human crystallographers when they interpret electron density maps. It adopts a divide-and-conquer, multi-stage approach to the problem, involving three main steps:

First, it builds a set of chains of Cα atoms representing the backbone. (Recall that Cα atoms are the connection points along the backbone where the side chains are attached). This is done by using a neural network to predict the positions of the Cα atoms that lie along the backbone trace (medial axis of the density contours), and connecting them to form chains of Cαs.

Second, side chains are fitted into the density based on pattern recognition of the local density around the Cα atoms. A case-based reasoning approach is used for fitting side chains, where we match the density regions around Cαs with instances in a database of regions, retrieve corresponding residue structures (i.e. atomic coordinates) that best fit the density, and concatenate them to build a complete structure.

Third, the structure is refined through post-processing routines, such as aligning the sequence built with the true sequence, and improving the fit to the density by real-space refinement. The model obtained can then be manually improved by the crystallographer, and used to obtain better phases. An improved map can thus be generated, and input back to TEXTAL.

Figure 3.4 shows the overall architecture of the TEXTAL system. TEXTAL is available to the community of crystallographers in three ways: (1) as downloadable binary distributions (http://textal.tamu.edu), (2) through a web-based interface called WebTex (Gopal *et al*., 2006a; http://textal.tamu.edu), and (3) as the model-building component of the PHENIX (Python-based Hierarchical ENvironment for Integrated Xtallography) system, a comprehensive software package for automated X-ray crystal structure determination (Adams *et al*., 2002; Adams *et al*., 2004; http://www.phenix-online.org).

In the next few sections, we provide a brief description of the system, with emphasis on issues related to case-based reasoning and feature relevance. For a more detailed discussion on TEXTAL and its sub-systems, refer to previous work (Holton *et al*., 2002; Ioerger and Sacchettini, 2002; Ioerger and Sacchettini, 2003; Gopal *et al*., 2003; Gopal *et al*., 2005a; Romo *et al*., 2005; Gopal *et al*., 2006b).

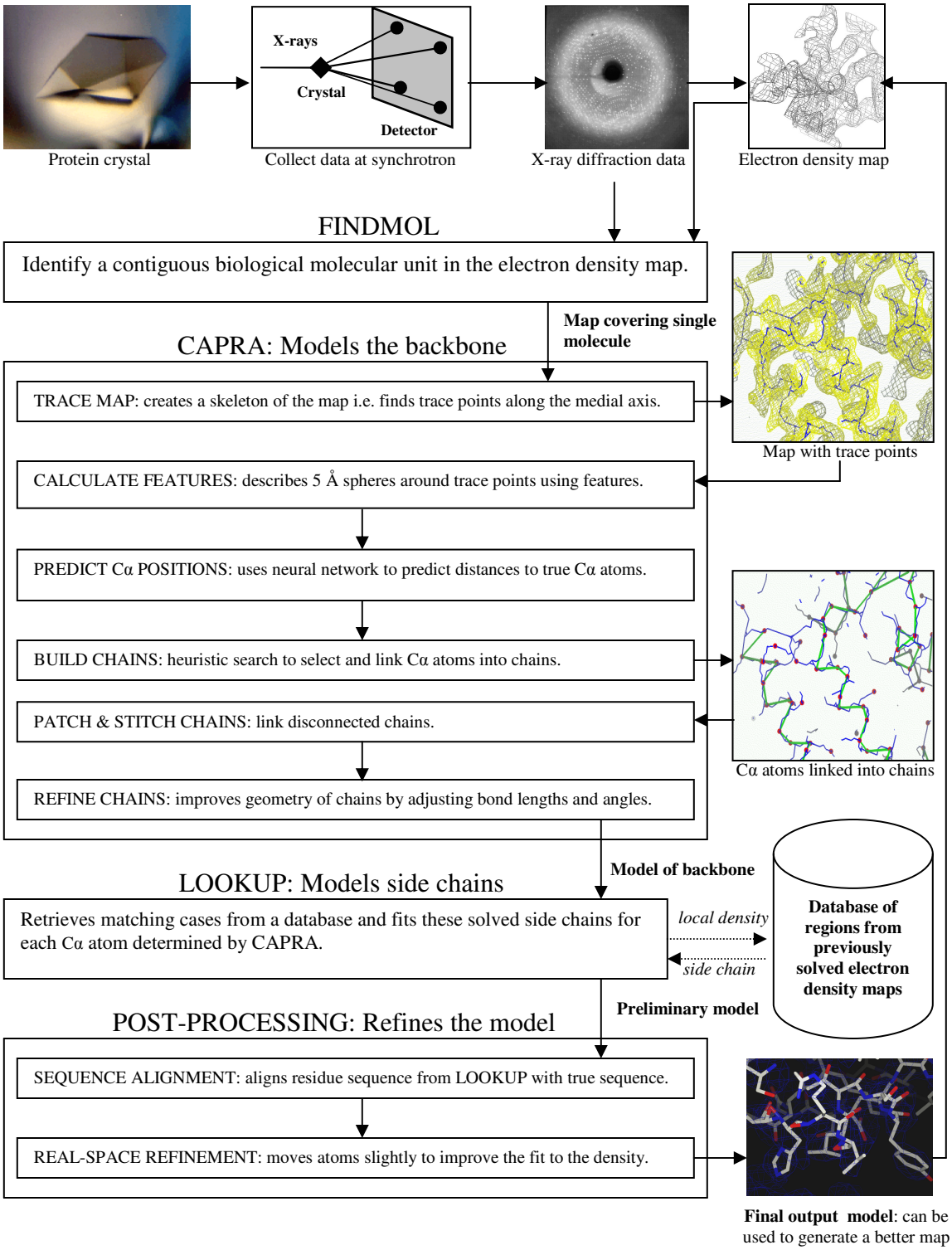

Fig. 3.4. Architecture of the TEXTAL system. There are four main sub-systems: FINDMOL, CAPRA, LOOKUP, and POST-PROCESSING.

3.7. Building the Backbone

The determination of the backbone is done by a system called CAPRA, or C-Alpha Pattern Recognition Algorithm (Ioerger and Sacchettini, 2002). The input to CAPRA is either an electron density map, or X-ray diffraction data (reflection file), which is converted into a map at a resolution TEXTAL is designed to work best (i.e. 2.8 Å). As shown in the architecture of the TEXTAL system (Figure 3.4), CAPRA comprises of the following steps:

- TRACE MAP: Given an electron density map, this routine creates a chain of grid points along the medial axis of the contours of the map. This is analogous to skeletonization programs (Greer, 1985; Swanson, 1994). The trace points represent the shape of the density contours in a compact form.
- CALCULATE FEATURES: This module takes a map as input, and computes numerical features of spherical regions defined around each of the trace points from TRACE MAP. These features are subsequently used to determine the positions of Cα atoms, and to model side chains.
- PREDICT Cα POSITIONS: To determine the 3D coordinates of Cα atoms, a traditional feed-forward neural network is used to predict the distance of various candidate positions (along the trace of the density) to the nearest true Cα, and to select the ones that are predicted to be closest (Ioerger and Sacchettini, 2002). The objective of the neural network is to learn the relationship between characteristics of electron density patterns around a coordinate and its proximity to a Cα atom. We use the 19 features (defined at 3 and 4 Å) to characterize the local density; these features are input to the network, which uses one layer of 20 hidden units with sigmoid thresholds, and outputs the predicted distance to a true Cα atom. The network is trained with a set of coordinates in maps of solved proteins with known distances to true Cαs, and the network weights are optimized using backpropagation (Hinton, 1989).
- BUILD CHAINS: An approach based on artificial intelligence is used to link the Cα atoms (as predicted by the neural network described above) into backbone chains.

The primary criterion is based on connectivity in the density map, although there are often many alternative branches, creating ambiguity. Linking Cα atoms into chains is a combinatorial search problem; whenever possible, an exhaustive search is done to create a solution that maximizes chain length. When a complete search becomes intractable, TEXTAL uses a heuristic function to guide the search for the best way to connect Cα atoms, based on criteria that favor better adherence to stereochemical constraints and secondary structures. These heuristics and decision criteria try to capture the type of reasoning that experienced crystallographers employ, such as following apparent α-helices and β-strands. It should be emphasized that automation of this process is particularly challenging because noisy data can be easily misleading, such as breaks in backbone connectivity or close contacts between side chains. A more thorough discussion on the methods used to build the backbone can be found in (Ioerger and Sacchettini, 2002).

- PATCH & STITCH CHAINS: These are backbone improvement steps that follow the initial construction of the backbone chains. The PATCH CHAINS module connects chains together in regions where density is weak by adjusting the contour level. The STITCH CHAINS module attempts to further connect different chains, especially in regions of weak density e.g. where the backbone makes a loop. A case-based reasoning approach is employed to "stitch" chains together. Regions of the structure that probably should have been connected (typically at close extremities of different chains) are identified and a database of protein structure fragments, (constructed from about 100 PDB files) is searched to find the most plausible fragment that could connect them. The case matching is done by superposing all chain fragments (of 7 to 11 consecutive Cα atoms) from the database with the region under consideration, and computing the root mean square deviation. If the deviation is small enough, and the electron density in the region is adequately high, then stitching is justified, which may entail adding new Cα atoms, guided by the retrieved case. These approaches are necessary to deal with noise in real-world diffraction datasets.

- REFINE CHAINS: This module improves the geometry of the Cα chains with respect to typical atomic bond lengths and angles. The distance between consecutive Cα atoms (as predicted by the neural network) tend to vary widely between 2.5 and 5.0 Å. A refinement procedure is applied to the chains to adjust the distance between adjacent Cαs to about 3.8 Å, as it is the case in proteins. The routine also reduces other defects in the backbone, like implausible bond angles.

3.8. Building Side Chains

Placement of side chains is done by a sub-system called LOOKUP. The program takes a set of Cα chains and an electron density map as inputs, and uses case-based reasoning and nearest neighbor learning to effectively and efficiently retrieve, from a database, spherical regions (of 5 Å radius) that are structurally similar to regions from the unsolved map. The regions centered around Cα atoms in the backbone model produced by CAPRA are compared to a large database of about 50,000 regions from 200 maps of proteins (for which the local structures of the regions are known and cover a very wide range of structural motifs in proteins). The corresponding local structures are retrieved and assembled together to produce a preliminary model, which can be further refined by post-processing routines. Matching regions are found by an efficient case retrieval system that uses a similarity metric based on 76 numerical features that locally characterize the spherical regions (the focus of this work).

Given an unsolved spherical query pattern of electron density, its distance from every case in the database can be determined, and the most similar (smallest distance) can be returned as the best match. In TEXTAL, the similarity (or distance) between two regions is measured by *density correlation*, a metric that involves the computation of the optimal superposition between two patterns. Since the number of possible 3D rotations is very large, the computation of density correlation is computationally expensive, which we cannot afford to run over the whole database. Thus, we use an approximate, inexpensive, feature-based distance metric to select a small subset of k potential matches, and the density correlation procedure then makes the final ranking. In previous work (Gopal *et*

al., 2004b), we evaluated and compared various feature-based distance metrics for this approach.

It should be noted that a good feature-based match need not be the absolute best one according to the objective metric (density correlation). It can be the top few matches (based on a tolerance on how high we wish the density correlation value to be for the region to qualify as a match). Given a query pattern, our aim is to try to get as many potentially good matches (anywhere) in the top *k*, since the expensive objective will be employed to re-rank the top *k* matches and identify the truly good ones (Gopal *et al*., 2004a).

Recently a simplex optimization (Romo *et al*., 2006; Nelder and Mead, 1965) routine was incorporated into LOOKUP. It iteratively optimizes the placement of the Cα atoms and the side chain orientation relative to local density correlation. The method proved to be robust, and improved the modeling in terms of both the identity of the residues built, and the geometry of the model.

3.9. Post-Processing

POST-PROCESSING routines refine the initial model built by LOOKUP. There are two main routines in this sub-system: (i) *Sequence alignment*, where the sequence of residues in the initial model produced by LOOKUP is aligned with the known sequence of amino acids in the protein, based on a dynamic programming approach proposed by Smith and Waterman (1981). This enables another round of LOOKUP to make corrections in the amino acid identities initially determined. (ii) *Real-space refinement*, where slight adjustments in the positions of atoms are made to better fit the density (Diamond, 1971).

3.10. Other TEXTAL Modules

There are additional stand-alone modules that complement the three main stages (i.e. building the backbone, placing side chains, and post-processing). The four main modules in this category are:

1. FINDMOL identifies a contiguous biological molecular unit of the protein in an electron density map (McKee *et al*., 2005). The boundaries of the repeating asymmetric unit often cut the molecule into multiple fragments, which makes map interpretation difficult. FINDMOL can identify a contiguous region of density in which to build by using a combination of clustering and symmetry operations.

2. A pattern recognition approach is used to automatically detect *disulfide bridges* in electron density maps (Ioerger, 2005). A disulfide bridge is a covalent bond between the *sulfur* atoms of two *cysteine* residues from different parts of the polypeptide chain. The residues with disulfide bridges can be located anywhere in the chain, and this cross-linking contributes to the stability of the protein. Disulfide bridges occur in roughly one out of every four proteins; localizing them in an electron density map can facilitate model-building, especially since the presence of a disulfide bridge reveals the position of cysteine residues. Disulfide bridges are detected by the following method: First, local spherical regions in the electron density map are characterized by nineteen numerical features calculated at four different radii (the same features used for building side chains). Then a linear discriminant model is applied to estimate resemblance of the local density pattern to a disulfide bridge, based on a training set with known disulfide and non-disulfide examples. The training cases are used to determine the parameters of the linear discriminant. In particular, the Fisher linear discriminant model is used to optimally maximize class separation, while minimizing variance within each class. This classification method projects the high-dimensional data onto an optimal line in feature-space, along which classification is performed, using a single threshold to distinguish between the two classes.

3. A routine to identify *non-crystallographic symmetry* (i.e. symmetry that exists locally within the asymmetric unit of the crystal) has recently been added (Pai *et al*.,

2006). This is achieved by analyzing similarities in density patterns between regions of the map.

4. Finally, a method has been developed to improve the performance of LOOKUP and sequence alignment. It involves exploiting information on *selenium* sites from *seleno-methionine* multi-wavelength anomalous diffraction (MAD) experiments to enhance side chain building and sequence alignment through identification of methionine residues immediately after CAPRA has identified Cα atoms and built chains.

3.11. Quality of Models Built by TEXTAL

The payoff of TEXTAL is mostly in terms of time saved to solve a structure. While a crystallographer may spend several days and sometimes weeks of painstaking effort to interpret a single map, TEXTAL produces a solution in a couple of hours, without human intervention. Even if the model produced by TEXTAL is only partially accurate, it provides a reasonable initial solution, which can be manually refined by the crystallographer to produce a more accurate and complete model.

The quality of output produced by TEXTAL depends on the size and complexity of the structure, and the quality of the data. TEXTAL and its sub-systems have been designed to work for a wide variety of proteins, of different sizes, with different structural components. TEXTAL usually outputs a reasonable model even with average quality data (i.e. around 3 Å resolution). Table 3.1 shows results for five representative proteins of various sizes, with maps spanning a wide range of resolution (from 1.8 Å to 2.8 Å). The proteins are *epsin* (Hyman *et al*., 2000), *tryparedoxin-I* (Alphey *et al*., 1999), *antitrypsin* (Kim *et al*., 2001), *nsf-d2* (Yu *et al*., 1998), and *penicillopepsin* (James and Sielecki, 1983). Typically CAPRA builds about 80-90% of the backbone, with less than 1 Å root mean square distance error. (For perspective, the average distance between consecutive Cα atoms in proteins is 3.8 Å). TEXTAL usually predicts more than 50% of the side chains with the correct identity. In cases where TEXTAL cannot find the exact amino acid, it typically places one that is structurally similar to the correct one. The model produced by TEXTAL can be manually improved, or used to generate better

phase information and create a better electron density map, which can be fed back into TEXTAL for subsequent model-building. For an average-sized protein (300 residues), TEXTAL's processing time is about 2 hours. Figure 3.5 shows examples of models built by TEXTAL from experimental data, and compare them to the true structures.

Table 3.1. Quality of models that TEXTAL outputs

Protein name	No. of residues	Resolution of data (Å)	Percentage built	RMS error (Å)	Correctly predicted residues (%)
Epsin	149	1.8	99	0.87	91
Tryparedoxin-I	147	2.0	96	0.86	93
Antitrypsin	394	2.1	81	1.12	52
Nsf-d2	273	2.4	98	1.10	90
Penicillopepsin	323	2.8	92	0.87	78

The table shows the quality of representative models built by TEXTAL, given data at five different resolutions. TEXTAL typically builds around 90% of the model in the 1.8-3.0 Å resolution range. The root mean square (RMS) errors shown are for all atoms in the models. The RMS error for Cα atoms only is less that 1.0 Å. The last column shows the percentage of residues whose identity is correctly determined, after corrections with sequence alignment.

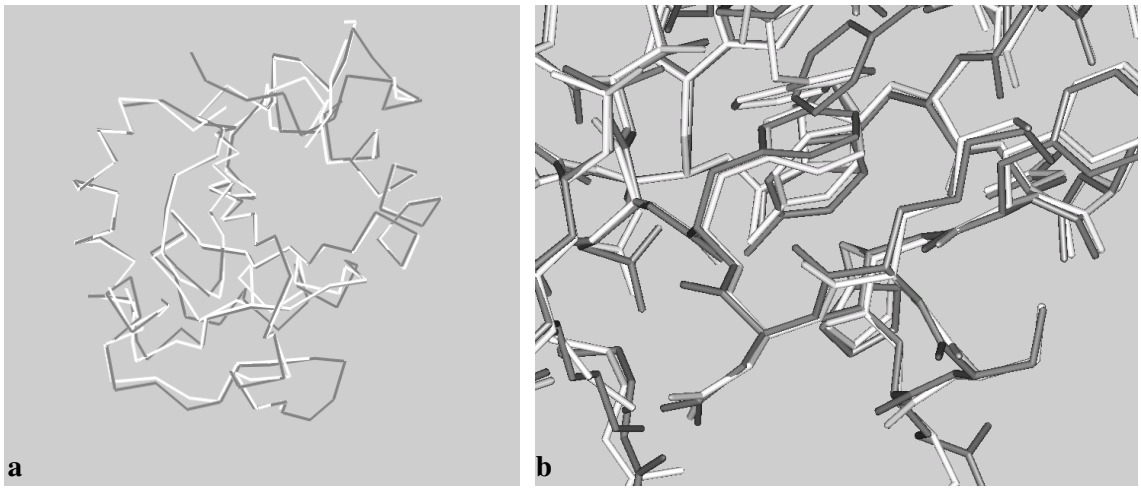

Fig. 3.5. Models of *tryparedoxin-I*, a monomer of 147 residues. The manually built and refined model is shown in gray; the model built by TEXTAL is shown in white. TEXTAL builds 96% of the structure. Figure 3.5(a) shows how TEXTAL builds the α-helices, β-sheets, and loops fairly accurately. TEXTAL correctly identifies 93% of the residues; the placement of residues is also accurate, as shown in Figure 3.5(b). The root mean square error over all atoms is 0.86 Å. These images were generated with PyMOL (W.L. DeLano, http://www.pymol.org).

3.12. Features in TEXTAL

The fundamental pattern recognition approach in TEXTAL is based on extracting numerical features that attempt to capture relevant information about local electron density for various purposes (such as identifying Cα atoms, comparing side chains, or detecting disulfide bridges). The features were derived manually based on knowledge about crystallography, and have the important property of being *rotation-invariant* (since the regions that we want to compare can occur in any 3D orientation). Nineteen features have been defined in TEXTAL (Table 3.2); the features are calculated at different radii. For instance, to model side chains, we calculate the features over spheres of size 3, 4, 5, and 6 Å. This is necessary since amino acids vary in shape and size, and each feature captures slightly different information at different sizes.

The features are categorized into four classes that capture different types of information about density patterns:

1. *Statistical* features like mean, standard deviation, skewness, and kurtosis of electron density distribution for a set of grid points in the spherical region.
2. Features based on *moments of inertia*, which gives the distribution of density in three dimensions. The primary moment lies along the path around which the density is most widely distributed; the secondary and tertiary moments are orthogonal to the primary moment (and to each other) and describe directions in space that have narrower density distributions. The magnitudes of the three moments of inertia are taken as features. The various ratios of eigenvalues for the three mutually perpendicular moments of inertia are also defined as features.
3. A feature that captures how *symmetric* or balanced the region is, based on the distance from the center of the sphere to its center of mass.
4. Features that capture information about the *shape* of the pattern: typically an amino acid has three "spokes" emanating from its Cα (one to the side chain and two to the main chain in opposite directions). These spokes are identified, and various features are calculated based on the angles between these spokes. We specifically look at the minimum, median and maximum angle among the spokes. The three spokes are defined as vectors from the center to the surface of the sphere with maximum radial sum, where the radial sum is calculated as the sum of the densities evaluated at points sampled evenly along the spoke. Computation of all possible spoke directions is too expensive; thus a finite number of trial spokes are sampled, and the best triplet of spokes is then computed. Besides the minimum, median and maximum spoke angles, other features include the sum of spoke angles, radial sum for each spoke, and the area of the triangle formed by the endpoints of the three spokes.

Table 3.2. Definitions of features used in TEXTAL

Feature class	*Description of feature*	*Method of computation (ρ_i is the electron density value at the i^{th} of n grid points in a region)*
Statistical	Mean	$(1/n)\Sigma\ \rho_i$
	Standard deviation	$[(1/n)\Sigma\ (\rho_i-\rho)^2]^{1/2}$
	Skewness	$[(1/n)\Sigma\ (\rho_i-\rho)^3]^{1/3}$
	Kurtosis	$[(1/n)\Sigma\ (\rho_i-\rho)^4]^{1/4}$
Moments of Inertia	Magnitude of primary moment Magnitude of secondary moment Magnitude of tertiary moment Ratio of primary to secondary moment Ratio of primary to tertiary moment Ratio of secondary to tertiary moment	Compute inertia matrix, diagonalize & sort eigenvalues.
Symmetry	Distance from center of sphere to center of mass	$\lvert\langle x_c,y_c,z_c\rangle\rvert$, where $x_c = (1/n)\Sigma x_i\rho_i$, $y_c = (1/n)\Sigma y_i\rho_i$, $z_c = (1/n)\Sigma z_i\rho_i$
Shape	Minimum angle between spokes Maximum angle between spokes Median angle between spokes Sum of spoke angles Radial sum of first spoke Radial sum of second spoke Radial sum of third spoke Spoke triangle area	Find three "spokes" i.e. three distinct vectors with highest density summation, and compute information like minimum, maximum, median, sum of angles, etc.

This table defines the features used to describe spherical electron density patterns in TEXTAL. The features are grouped into four classes; each feature has four versions for different radii of the sphere (3, 4, 5 and 6 Å).

3.13. Feature-Based Approaches in Structural Bioinformatics

The use of features to capture localized information on protein structures is quite common in structural bioinformatics. For instance, S-BLEST (Mooney *et al*., 2005) annotates protein structures with information on the environment of proteins. The information is derived from a database of annotated amino acids, and the comparison is done based on *k*-nearest neighbor search, using Manhattan distance on 66 features (each at four radii).

There are strong similarities between the approaches in S-BLEST and TEXTAL in terms of features. Nonetheless, S-BLEST's features are different from TEXTAL's in various respects: first, they are centered around the Cβ atoms, whereas in TEXTAL, they are centered on Cα aroms. The radii used in S-BLEST are also different from those in TEXTAL (they range from 1.875 to 7.5 Å, whereas in TEXTAL they are from 3 to 6 Å).

More fundamentally, features in TEXTAL capture structural properties in electron density patterns around Cα atoms, whereas in S-BLEST, they describe geometric as well as functional properties of residues in actual solved structures. The features in S-BLEST are used in a non-weighted Manhattan distance to determine similarity – it may be helpful to use SILDER to weight the features, and thereby improve the distance metric used by S-BLEST.

Another type of local information that is captured in electron density maps is *critical points* (Leherte *et al*., 1997), which are points that correspond to local maxima of electron density local features in electron density maps. In a crystallographic threading system subsequently proposed by Ableson and Glasgow (1999), features on the environment of critical points (like maximum peak, distance to solvent, size, volume, mass, etc.) are analyzed and combined with information on the sequence to construct a protein model.

The Buccaneer program proposed by Cowtan (2006) repeatedly applies an electron density likelihood function to identify Cα positions in a noisy electron density map. The density likelihood function is used to recognize characteristic features corresponding to a

Cα position in a sphere of density whose radius is 4 Å. This approach is related to that of CAPRA, but unlike CAPRA, the features used in Buccaneer are not rotation-invariant.

CHAPTER IV
SLIDER

In this chapter, we first provide a formal and general definition of the case-based reasoning system and case retrieval strategy that we propose. We then describe the main features of the SLIDER algorithm, and discuss its significance in efficient case retrieval. We also present an alternative approach to feature weighting based on linear programming – this approach is closely related to SLIDER, but is nonetheless fundamentally different, especially in terms of computational complexity.

4.1. A Two-Phase Case Retrieval Strategy

Consider a case-based reasoning system with a database $\mathbb{D}$ consisting of N cases, and a set of query instances Q. We here assume that $\mathbb{D}$ and Q are drawn from the same distribution of instances, each of which is represented by a set F of numerical features. We also define θ, a finite set of possible similarity judgments between two cases. For example, in the binary case, $\theta = \{\textit{similar}, \textit{different}\}$ i.e. two cases can be either similar or different (in TEXTAL, we use the electron density correlation measure to determine the true similarity between two density regions on a continuous scale – a correlation above a certain threshold is taken to imply similarity, and below implies dissimilarity). We define a retrieval function $\mathcal{R}$ that maps a $\langle \textit{query}, \textit{case} \rangle$ onto a similarity value in θ i.e. $\mathcal{R}$: $Q \times \mathbb{D} \rightarrow \theta$. In this analysis, we consider similarity as dichotomous, for simplicity – that is, a case is either similar or different to a query.

Now given a query instance q, let the set of cases in $\mathbb{D}$ that are truly similar to q be S. Let there be N_S such cases i.e. $|S| = N_S$. Let the set of cases different from q be D, where $|D| = N_D = N - N_S$. We define the distance between q and case x in $\mathbb{D}$ by the weighted Euclidean metric:

$$d_F(q,x) = \sqrt{\sum_{i=1}^{|F|} w_i (q_i - x_i)^2} \tag{4.1}$$

where x_i and q_i are the i^{th} feature of x and q respectively, and w_i is the weight of that feature. This distance is a surrogate measure of true similarity, where a smaller distance implies higher similarity. We compute $d_F(q,x)$ for all x's in $\mathbb{D}$ and rank the x's according to their distances to q in increasing order. Our aim in the proposed filtering method is to rank as many truly similar cases as possible in the top k ranked instances, where $k \ll N$ (e.g. $k/N \approx 0.01$). If the features are properly selected (or weighted), the distance measure should reflect true dissimilarity better; this will enrich the top k cases with more matches, and increase the expected probability of selecting at least one true match. It will also imply affordance of a lower k, and hence more efficient retrieval. Without optimized weights, we would have to set k higher to ensure retrieval of enough matches. The choice of k depends on domain-dependent factors like how expensive case matching is, and how much time performance are we willing to sacrifice for a desired level of accuracy (Gopal *et al*., 2004a). In TEXTAL, k is set to 400, for a database of 50,000 cases i.e. only 0.8% of the database is filtered, and further evaluated by a more expensive comparison (density correlation).

4.2. The SLIDER Algorithm

We now provide a more detailed explanation of how weights are tuned by SLIDER. We first consider two-component mixtures (i.e. involving two features, where their weights sum up to 1) and then extend it to an arbitrary number of features. The distance metric we use is weighted Euclidean. Nonetheless this method can be applied to Minkowski distances in general (Gopal *et al*., 2005b).

The weighted Euclidean distance between two instances x and y, using two features i and j (with weights w_i and w_j respectively, where $w_i + w_j = 1$) is defined as:

$$d_{\{i,j\}}(x,y) = \sqrt{w_i \left(x_i - y_i\right)^2 + w_j \left(x_j - y_j\right)^2} \tag{4.2}$$

We can drop the square root in (4.2), since it is a monotonic transformation, and we use distances as a relative measure (their absolute values are not intrinsically meaningful). Thus $d_{\{i,j\}}(x,y)$ can be re-defined as:

$$\begin{aligned} d_{\{i,j\}}(x,y) &= w_i\left(x_i - y_i\right)^2 + w_j\left(x_j - y_j\right)^2 \\ &= (1-w)\left(x_i - y_i\right)^2 + w\left(x_j - y_j\right)^2 \end{aligned} \tag{4.3}$$

where w is set to w_j, the weight of feature j. Consider an instance x that has y as its closest neighbor according to feature i, and z as its closest neighbor according to feature j i.e. the nearest neighbor of x is y when $w = 0$, and it is z when $w = 1$ (w is the weight of feature j). If w "slides" from 0 to 1, then there is a weight w_c at which $d_{\{i,j\}}(x,y) = d_{\{i,j\}}(x,z)$; this point is called a "crossover". By expanding $d_{\{i,j\}}(x,y) = d_{\{i,j\}}(x,z)$, we get:

$$(1-w)\left(x_i - y_i\right)^2 + w\left(x_j - y_j\right)^2 = (1-w)\left(x_i - z_i\right)^2 + w\left(x_j - z_j\right)^2 \tag{4.4}$$

Solving for w, and setting it to w_c, we get:

$$w_c = \frac{\left(x_i - z_i\right)^2 - \left(x_i - y_i\right)^2}{\left(x_j - y_j\right)^2 - \left(x_i - y_i\right) + \left(x_i - z_i\right)^2 - \left(x_j - z_j\right)^2} \tag{4.5}$$

In other words, w_c is a weight where there is a net increase (or decrease) in accuracy, depending on which of y and z is truly closer to x. The concept of a crossover point is illustrated in Figure 4.1. When there is an increase in accuracy (i.e. the match becomes closer to x than the mismatch, for all weights above w_c), it is referred to as "positive crossover", and "negative crossover" otherwise. We can find the crossover weights for a training set of 3-tuples, and choose the optimum weight w^* that represents the best compromise between positive and negative crossover weights.

It should be noted that not all 3-tuples of instances will have a crossover for a given pair of features. In fact, there will not be a crossover point if, for all values of w, the distance between x and its match is always larger (or smaller) than the distance between x and its mismatch (i.e. the lines representing distances $d_{\{i,j\}}(x,y)$ and $d_{\{i,j\}}(x,z)$ in Figure 4.1 do not intersect).

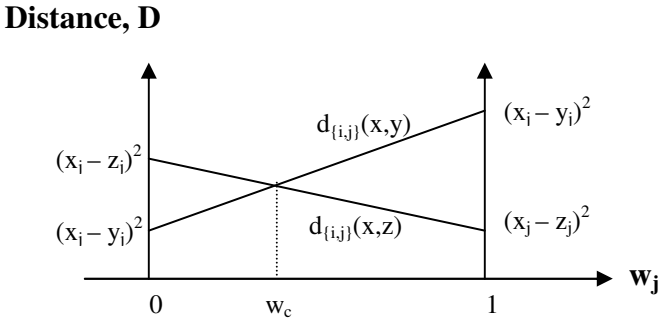

Fig. 4.1. Finding a "crossover" weight. As the weight of feature j, w_j, slides from 0 to 1, the weighted Euclidean distance between x and y [$d_{\{i,j\}}(x,y)$] changes linearly from lesser to greater than that between x and z [$d_{\{i,j\}}(x,z)$]. The crossover occurs at w_c i.e. there is a change in accuracy of prediction at w_c, depending on whether y or z is truly more similar to x.

Crossover points can also be determined by considering two subsets of features (instead of just two features). Consider two feature subsets A and B, with corresponding Euclidean distances d_A and d_B respectively. A composite metric, d_{A+B}, can be defined as $d_{A+B}(x,y) = wd_A(x,y) + (1 - w)d_B(x,y)$. As w slides from 0 to 1, it may cause a switch of neighbors, as described earlier. Thus, w can be used to determine the new weight vector that increases accuracy, based on crossover points. In SLIDER, we randomly choose one feature (singleton set A) and evaluate it against all remaining features (set B).

SLIDER starts by assigning the same weight to all features i.e. $w_i = 1/|F|$, $1 \leq j \leq |F|$. It then uses a hill-climbing approach by iteratively choosing a (random) feature, adjusting its weight to make the distance metric more accurate, and stopping when there is no net increase in the quality of the weights. Now we describe how, in each iteration of the algorithm, (1) the "optimum" weight w^* (the best crossover weight, over a set of examples) is determined, and (2) we decide whether the quality of the weight set has improved or not.

The optimum weight w^* with respect to a set of examples in a given iteration is determined as follows. We randomly choose a feature and find all crossover points from the set of training examples T by sliding the weight of that feature against those of all other features combined. Each 3-tuple may have a crossover, and our goal is to find the

weight that maximizes the difference between positive and negative crossovers. Any crossover point w_c divides the line segment [0,1] (that represents the weight of feature *j*, which can range from 0 and 1) into two line segments: [0,w_c] and [w_c,1]. One line segment is "positive" in the sense that any weight of feature *j* in that segment will yield a distance metric that correctly places the match of *x* closer to *x* than the mismatch. The other segment is "negative" in that the metric does the opposite. If for each crossover point we increase the score of the positive segment by 1, then there is one or more line segment(s) with a maximum score. We define the optimum weight *w** (of the chosen feature *j*) as the midpoint of any one of the segments with the maximum score.

We now describe how *w** is computed more formally. Given a feature *j* (randomly chosen in each iteration of SLIDER), a training example $t = \langle x, x^+, x^- \rangle \in T$ (where *x* is an example feature vector, x^+ is a match of *x*, and x^- is a mismatch of *x*), and a corresponding crossover weight w_c (of feature *j*), we first define the composite distances between *x* and x^+, and that between *x* and x^-:

$$d_F(x, x^+) = w_j d_{\{j\}}(x, x^+) + (1 - w_j) d_{\{1,..,j-1,j+1,..,|F|\}}(x, x^+) \tag{4.6}$$

$$d_F(x, x^-) = w_j d_{\{j\}}(x, x^-) + (1 - w_j) d_{\{1,..,j-1,j+1,..,|F|\}}(x, x^-) \tag{4.7}$$

Note that $d_F(x, x^+) = d_F(x, x^-)$ when $w_j = w_c$. Now we define a step function Γ to capture the notion of positive and negative line segments (for feature *j*), given a training example *t*:

$$\Gamma(w_j, t) = 1 \quad \text{for all } w_j \text{ where } d_F(x, x^+) < d_F(x, x^-),$$
$$0 \quad \text{otherwise.} \tag{4.8}$$

If we find w_j's with the maximum $\Gamma(w_j, t)$ summed over all training examples i.e. $\text{argmax}_{w_j} \sum_{t \in T} \Gamma(w_j, t)$, we shall get at least one interval of w_j with that maximum value. The midpoint of that interval is assigned to *w**.

We illustrate how *w** is determined by an example. Consider two positive crossover points (0.2, 0.7), and two negative crossover points (0.3, 0.5). For a positive crossover c_+,

the line segment $[c_+,1]$ is the positive side, and for a negative crossover $c\_$, the segment $[0,c\_]$ is positive. Figure 4.2 shows how the four positive line segments overlap. The optimum resulting line segment is [0.2,0.3]. Any point in that segment will give the best ranking; we choose the midpoint of that segment as the optimum weight i.e. $w^* = 0.25$.

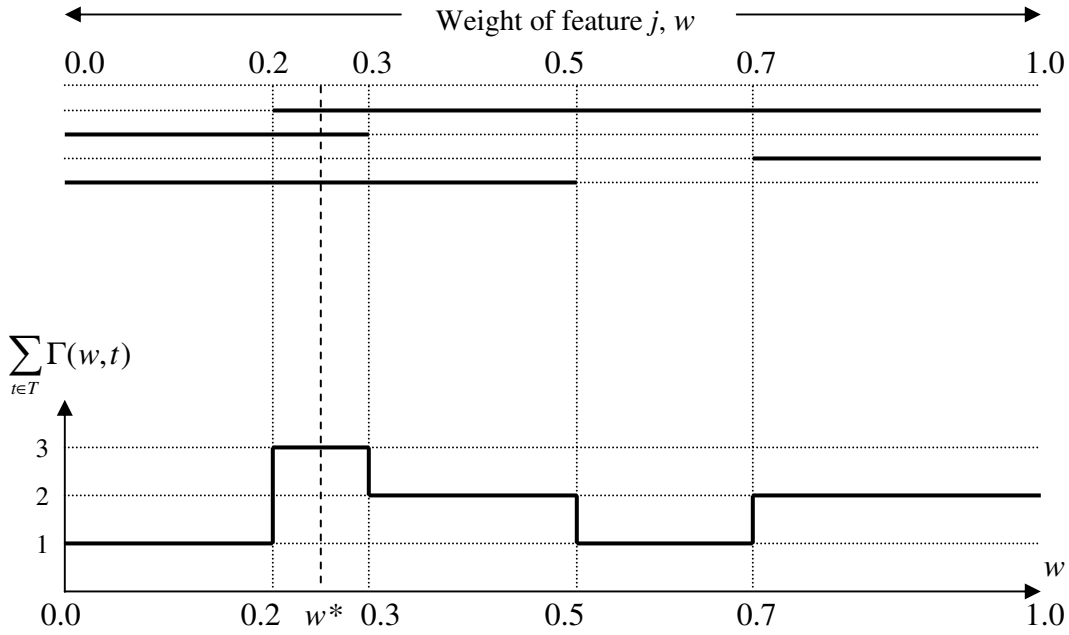

Fig. 4.2. Finding the optimum weight w^*. Given two positive crossovers (0.2 and 0.7) and two negative crossovers (0.3 and 0.5), the four corresponding positive line segments are shown (in thick lines) at the top. This splits [0, 1] into 5 segments, and the segment [0.2, 0.3] has the highest score (of 3) in terms of the sum of $\Gamma(w,t)$ over all training examples t; the scores (plotted at the bottom) are the number of positive line segments that intersect that range. We return the midpoint 0.25 of [0.2, 0.3] as w^*.

To evaluate the chosen weight vector, we compute how well the corresponding weighted Euclidean distance ranks matches relative to mismatches. We use an independent validation set V of instances (different from the training set T to find crossover weights), and for each instance in V we find a true match from our database $\mathbb{D}$. Our goal is to estimate the average rank of the match against any sample of mismatches

drawn from $\mathbb{D}$. Given a weight vector $\vec{w}$ that we want to evaluate, we define the rank $R\left(v,\vec{w}\right)$ for an instance v in V as the average rank over n samples $\mathbf{\Phi = \{\ \Phi_1, \Phi_2, \ldots, \Phi_n\}}$ of mismatching instances from $\mathbb{D}$ i.e.

$$R\left(v,\vec{w}\right)=\frac{1}{\left|\Phi_i\right|}\sum_{i=1}^{n} rank\left(v,\Phi_i,\vec{w}\right) \tag{4.9}$$

where $rank(v,\mathbf{\Phi_i},\vec{w})$ is the rank of the match of validation instance v relative to all instances in $\mathbf{\Phi_i}$, using the underlying metric with weight vector $\vec{w}$. Note that lower rank implies more similar to the query instance (i.e. the match should ideally have rank = 1). For practical purposes, we estimate R using a randomly drawn sample of 100 mismatches (singleton set $\mathbf{\Phi}$) from our database of 50,000 instances. (The larger the sample, the more accurate the estimate will be; nonetheless, computation of density correlation being expensive, we can afford only a limited number of mismatches.) Then we compute the average $\overline{R}$ over all instances v in V i.e.

$$\overline{R}\left(V,\vec{w}\right)=\frac{1}{\left|V\right|}\sum_{v\in V} R(v,\vec{w}) \tag{4.10}$$

The pseudo-code for the SLIDER algorithm is given in Figure 4.3.

Inputs: 1. Training set T of ‹*instance*, *match*, *mismatch*› 3-tuples (all drawn from $\mathbb{D}$);
2. Validation set V of instances; for each instance in V, one match and 100 mismatches, drawn from $\mathbb{D}$;
3. Set F of features.

Output: Optimized weight vector $\overrightarrow{w^*}$ = ‹w_1*, w_2*, …, $w_{|F|}$*›

Initialize weights of all F features uniformly i.e. $w_i = 1/|F|$, $1 \leq i \leq |F|$.

repeat
  Select feature f randomly.
  Set w_f to 0 and adjust other weights proportionally so that they add up to 1.
  Find all crossover points from T by sliding w_f from 0 to 1.
  Find "optimum" weight of f, w_f* i.e. the midpoint of line segment with best score (Fig. 4.2).
  Set w_f to w_f* and other weights are proportionally decreased to keep sum of weights 1.
  Compute mean rank of matches for new weight vector, $\overline{R}\left(V, \overrightarrow{w}\right)$ using (4.10)
  if $\overline{R}\left(V, \overrightarrow{w}\right)$ decreases (improves) *then* best weights $\overrightarrow{w^*}$ is set to $\overrightarrow{w}$
until $\overline{R}\left(V, \overrightarrow{w}\right)$ reaches a plateau *or* number of iterations exceeds a threshold.
return $\overrightarrow{w^*}$

Fig. 4.3. The SLIDER algorithm.

4.3. A Linear Programming Approach to Feature Weighting

In this section, we present an alternative approach to feature weighting based on linear programming (for comparison). Although SLIDER bears some resemblance to linear programming, there are nonetheless some key differences between the two. Note that SLIDER aims at maximizing the *number* of training 3-tuples $\left\langle x, x^+, x^- \right\rangle$ where the weighted Euclidean distance between instance x and x^+ (where x^+ is truly similar to x) is smaller than the distance between x and x^- (where x^- is truly different from x). In contrast, the linear programming approach tries to optimize the aggregate difference in

the *absolute* weighted Euclidean distances themselves; we now describe this formulation of the problem in more precise terms.

Given a training set $T = \{\langle x_1, x_1^+, x_1^- \rangle, ..., \langle x_m, x_m^+, x_m^- \rangle\}$ of m such 3-tuples (similarity and difference are defined as in SLIDER, i.e. based on density correlation), the linear programming method tries to weight features such that for each 3-tuple, the weighted Euclidean distance between x_i and x_i^- is larger than that between x_i and x_i^+. If we take the difference of the square of the Euclidean distances, the same inequality will hold. We define this difference, Ω_i, in (squared) distances as follows, where $x_{i,j}$ is the j^{th} feature value for x_i:

$$\Omega_i = \sum_{j=1}^{|F|} w_j (x_{i,j} - x_{i,j}^-)^2 - \sum_{j=1}^{|F|} w_j (x_{i,j} - x_{i,j}^+)^2 \tag{4.11}$$

This can be re-written as:

$$\Omega_i = \sum_{j=1}^{|F|} w_j \Delta_{i,j} \tag{4.12}$$

where

$$\Delta_{i,j} = (x_{i,j} - x_{i,j}^-)^2 - (x_{i,j} - x_{i,j}^+)^2 \tag{4.13}$$

Now we wish to find w_j's, such that $\Omega_i \geq 0$ for every instance $\langle x_i, x_i^+, x_i^- \rangle$ in T (or as many as possible). It is very unlikely that this constraint can be satisfied simultaneously for all instances in T. Thus, we use slack variables σ_1, σ_2, …, σ_m (one variable for each of the m 3-tuples in T) that are constrained to be positive, and that make up for Ω_i whenever Ω_i is negative. Thus, we define the feature weighting problem as the following optimization in linear programming: Find w_j's that *minimize* $\sigma_1 + \sigma_2 + \ldots + \sigma_m$, subject to the following constraints:

i. $0 \leq w_j \leq 1$ for each feature j, and $\sum_{j=1}^{|F|} w_j = 1$;

ii. $\sum_{j=1}^{|F|} w_j \Delta_{i,j} + \sigma_i \geq 0$ for each 3-tuple $\langle x_i, x_i^+, x_i^- \rangle$ in T;

iii. $\sigma_i \geq 0$ for each 3-tuple $\left\langle x_i, x_i^+, x_i^- \right\rangle$ in T.

This optimization problem is different from the (harder) problem of maximizing the number of cases where the distance between an instance and its match is greater than the distance between the instance and its mismatch. The two are closely related, but the linear programming formulation is much more tractable – the worst-case complexity of linear programming is polynomial, and there are several algorithms that are known to perform well both in theory and practice (Karmakar, 1984). In contrast, the former optimization problem is a *constraint satisfaction problem* (CSP), which is known to be NP-hard (Ceberio and Kreinovich, 2005). The NP-hardness of the problem of maximizing the maximum number of instances that are closer to their matches than their mismatches motivates the need for heuristics such as the one used in SLIDER.

CHAPTER V
RELATIONSHIP BETWEEN IMPROVING RANKING AND THE PROBABILITY OF SUCCESSFUL RETRIEVAL

In this chapter, we develop a theoretical foundation for SLIDER. We show how the ranking-based heuristic used to evaluate weight vectors leads to a greater probability of retrieving a match in the two-phase filtering method employed for case retrieval. (Note: This chapter will refer to many terms and equations defined in Chapter IV).

5.1. Probabilistic Analysis of the Relationship between Distance and Ranking

Before evaluating the impact of feature weighting on the effectiveness of case retrieval, we first establish the relationship between the real-valued Euclidean distances for matches and the ranks of the matches (relative to mismatches).

Given a query q and a set of weights, let the weighted Euclidean distances between q and cases in $\mathbb{D}$ that are truly similar to q (i.e. set S) have a probability density function P_S. Also, let the distances between q and cases that are different from q (i.e. set D) have a probability density function P_D. We expect the mean of P_S to be smaller than the mean of P_D (Figure 5.1).

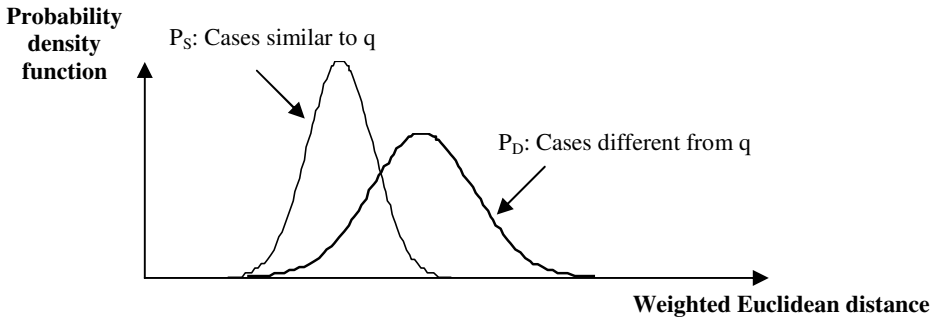

Fig. 5.1. Probability density functions of Euclidean distance for similar and different cases. Given a query instance q, we show typical probability distributions of distance for cases similar to q (P_S), and cases different from q (P_D). We expect that the mean of P_S to be smaller than the mean of P_D.

Now each case similar to q in $\mathbb{D}$ is (independently) ranked relative to the different ones i.e. the rank of the match ranges from 0 to N_D (matches are not ranked relative to each other). It can be shown that the mean rank $\hat{r}$ of the true matches of q is given by the following:

$$\hat{r} = N_D \int_{x=0}^{\infty} \int_{y=0}^{x} P_S(x) P_D(y) dy dx \tag{5.1}$$

The rationale for this double integral is as follows: $P_S(x)$ is the probability that the Euclidean distance between q and a match is x. The (inner) integral over y of the $P_D(y)$ component is the fraction of cases in $\mathbb{D}$ different from q that have a distance smaller than x. It is the area under P_D from 0 to x, or the cumulative distribution at x (we set the lower limits to 0 since the Euclidean distance is always 0 or greater). If we sum this fraction [weighted by $P_S(x)$] for all x's, and multiply by N_D, we get the average number of mismatches that have a Euclidean distance smaller than the match i.e. we get the average rank of matches.

Consider the following three special cases:

Case 1: The distance between q and any match is smaller than the distance between q and any mismatch. This scenario (Figure 5.2) is one where the distance metric perfectly

distinguishes similar cases from different ones. Every match gets a perfect rank (i.e. zero) relative to mismatches [since the area under P_D (from 0 to x) is zero for every x in $P_S(x)$]; the mean rank of matches is thus expected to be zero, which is what (5.1) yields for this special case.

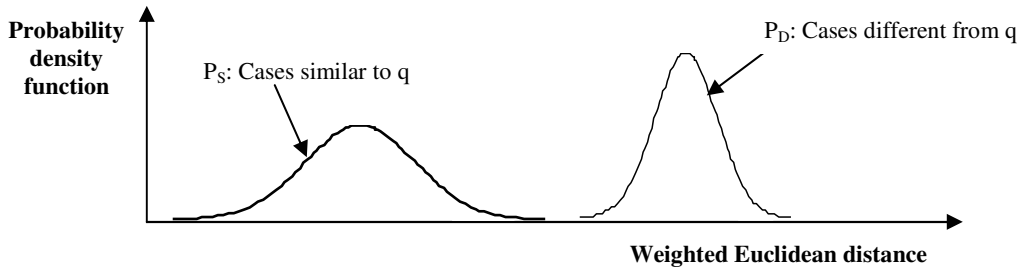

Fig. 5.2. Special case of probability density functions. The Euclidean distance between query case q and x, for all x's similar to and different from q – in this scenario the distance metric perfectly discriminates similar cases from different ones.

Case 2: The distance between q and any match is larger than the distance between q and any mismatch. This is an odd scenario (Figure 5.3) where the distance metric is completely misleading in case matching. The mean rank of matches here is N_D, which is the worst possible. It can be easily verified from (5.1) that $\hat{r}$ is indeed equal to N_D for this special case.

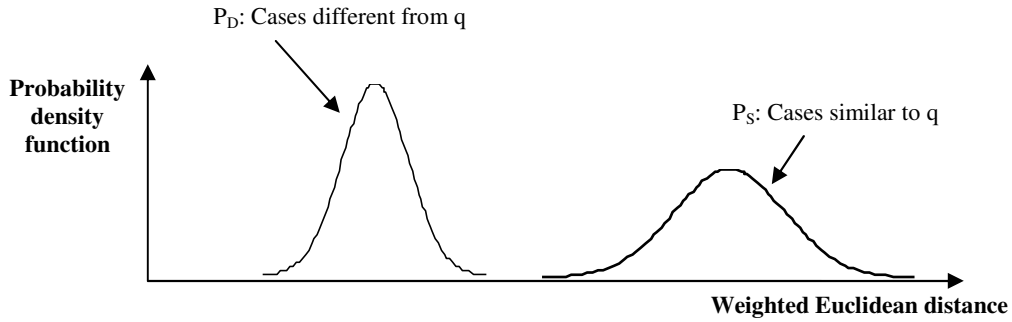

Fig. 5.3. Special case of distance distributions where the distance metric completely misleads similarity assessment. This is the worst possible scenario, where every match is ranked last when compared to mismatches.

Case 3: The probability density functions of distances to cases similar to and different from q are the same i.e. $P_S = P_D$. That is, the features and the distance metric contain no information about what constitutes similarity (or difference). In the absence of any discriminating ability (by assigning distances randomly, for instance), the distance metric would rank a match anywhere from 0 to N_D with equal likelihood. Thus, one would expect the average rank to be $N_D/2$. We now show that $\hat{r}$ is indeed equal to $N_D/2$ when $P_S = P_D$. We first re-write (5.1) as follows:

$$\hat{r} = N_D \int_{x=0}^{\infty} P_S(x) C_D(x) dx \tag{5.2}$$

where $C_D(x)$ is the cumulative distribution function of distance to mismatches at x. Let $C_S(x)$ be the cumulative distribution function of distance to matches at x i.e.

$$\frac{d}{dx} C_S(x) = P_S(x) \tag{5.3}$$

From (5.2) and (5.3) we get

$$\hat{r} = N_D \int_{x=0}^{\infty} \frac{d}{dx}\left[C_S(x)\right] C_D(x) dx \tag{5.4}$$

Now if $P_S = P_D$, then $C_S = C_D$, and therefore

$$\hat{r} = N_D \int_{x=0}^{\infty} \frac{d}{dx}\left[C_S(x)\right] C_S(x)\,dx \tag{5.5}$$

We know that

$$\tfrac{d}{dx} y^2 = 2y\frac{dy}{dx} \tag{5.6}$$

Also, $C_S(0) = 0$, and $C_S(\infty) = 1$. We use these facts to derive the expected result:

$$\hat{r} = \frac{N_D}{2} \int_{x=0}^{\infty} \frac{d}{dx}\left[C_S(x) * C_S(x)\right] dx$$

$$= \frac{N_D}{2}\left[C_S(x) * C_S(x)\right]_{x=0}^{x=\infty} \tag{5.7}$$

$$= \frac{N_D}{2}$$

More generally, we expect the two distributions to overlap partially, and SLIDER tries to push them further apart (i.e. move P_D to the right of P_S) to make $\hat{r}$ approach 0. In the next section, we show that the way SLIDER updates weights has this effect on the probability density functions.

5.2. The Impact of Optimizing the Mean Rank of Matches

We now discuss the impact of SLIDER's strategy to select weights that minimize the mean rank of matches. SLIDER chooses weights that improve (decrease) the average rank of matches (for a training set) relative to mismatches. Recall that the algorithm works as follows: First *m* training examples are randomly selected; for each example, all similar cases as well as different ones are pre-determined, using the true similarity measure. SLIDER greedily searches a space of weight vectors and selects those weights that cause a decrease in the mean rank $\overline{R}$ of matches [$\overline{R}$ is defined in (4.10)]. The value of $\overline{R}$ is essentially an empirical estimate of $\hat{r}$ as defined in (5.1), averaged over a sample of *m* training cases i.e.

$$\overline{R} = \frac{1}{m} \sum_{i=1}^{m} \hat{r}_i \tag{5.8}$$

It should be pointed out that the computation of $\overline{R}$ can be very expensive, since it implies exhaustively searching the database (many times) to find all true matches. Thus, in SLIDER we use an approximation of $\overline{R}$, where we rank only one match of a query, relative to a fixed number (say 100) of mismatches sampled from the background distribution.

We hypothesize that a decrease in $\overline{R}$ that SLIDER seeks in every iteration reflects (for a typical q in the query space Q) "separation" between P_S and P_D, where P_D shifts relatively further to the right (larger distance), and P_S shifts relatively to the left (Figure 5.4). These shifts are consequences of choosing better weights. That is, in every iteration of the weighting algorithm, $\overline{R}$ is decreased and more information is provided to the distance metric (through the weights) to enable separation (and hence distinction) between similar and different cases.

We emphasize that it is neither necessary, nor sufficient that the Euclidean distances to true matches decrease (and distances to mismatches increase) so as to improve $\overline{R}$. Altering the distance metric (via a change in weights) can make the distance (to a match or mismatch) increase or decrease. What really matters is that there is a relative amount of change between distances to matches and mismatches. In fact, the mean distance of P_S and P_D can both increase or decrease for the method to work, as long as they separate as described above. Assuming that they are roughly Gaussian, the change in $\overline{R}$ will therefore depend on the combined change in the means μ_S and μ_D, standard deviations σ_S and σ_D (of P_S and P_D respectively). For simplicity, we assume that σ_S and σ_D remain more or less invariant.

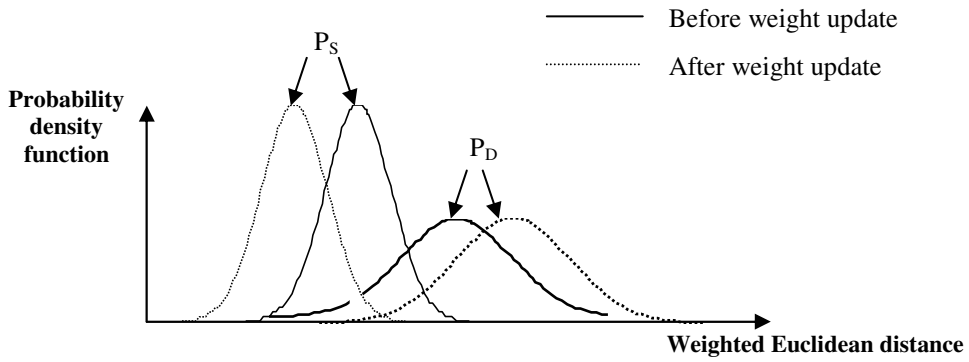

Fig. 5.4. Weight update in SLIDER leads to "separation" between P_S and P_D. We here show a simplified scenario where the distributions are Gaussian and variances do not change when feature weights are updated.

Figure 5.5 shows typical probability density functions of the Euclidean distance from the TEXTAL system. This particular example is for 1000 matches and 1000 mismatches drawn from the TEXTAL database, given a random query electron density pattern. The distance metric uses 76 features that are uniformly weighted. We here define a region to be a match if its electron density correlation with the query exceeds 0.8; it is mismatch if the correlation is lower than 0.6 (density correlation ranges from 0 to 1, where a correlation of 1 implies perfect similarity). We can observe that assuming the distributions to be Gaussian is fairly realistic.

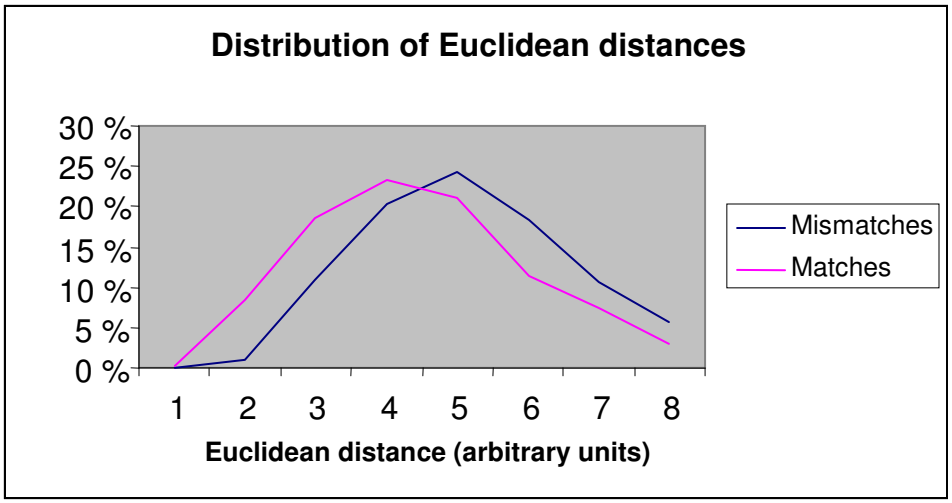

Fig. 5.5. Distribution of Euclidean distances for a random query electron density region. We show the distances for 1000 matches and 1000 mismatches (from the TEXTAL database) for the query region. 76 numerical features are used in the distance metric; the features are uniformly weighted.

We hypothesize that a decrease in $\overline{R}$ (as targeted by SLIDER) represents a decrease in $\hat{r}$ (5.1), which in turn signifies a relative separation between the distributions P_S and P_D. We now show that this separation leads to an increase in accuracy of retrieval (in terms of the probability of getting a match in the top k cases selected from the database). Given a new query case q (that comes from the same distribution where the training examples were drawn from), we define d_k as the k^{th} smallest distance between q and all its mismatches in the database. If we define successful retrieval for q as the ranking of *at least* one match of q below k, then the probability of successful retrieval ($P_{success}$) is given by

$$P_{success} = 1 - [1 - C_S(d_k)]^{N_S} \tag{5.9}$$

where N_S is the number of true matches of q in $\mathbb{D}$. The rationale behind (5.9) is that $P_{success} = 1 - P_{failure}$, where $P_{failure}$ is the probability that not a single match is ranked below k. The latter situation arises when a given match does not have a distance less than

d_k, and this should occur N_S times for retrieval failure. Our goal is to maximize $P_{success}$; a lower bound of $P_{success}$ is $C_S(d_k)$, which applies when there is only match i.e. $N_S = 1$.

Figure 5.6 shows how the cumulative distribution functions C_S and C_D are expected to change, given the relative separation in P_S and P_D, assuming that P_S and P_D are approximately Gaussian, and their variances do not change. We also show the k^{th} smallest Euclidean distance, both before and after the weight update. This is equal to the distance where $C_D = k/N_D$. Figure 5.6 shows how $C_S(d_k)$ increases when weights are modified (we shall later prove that $C_S(d_k)$ indeed increases with weight update). This increase is tantamount to an increase in probability of retrieving a true match from the database ($P_{success}$), as per (5.9).

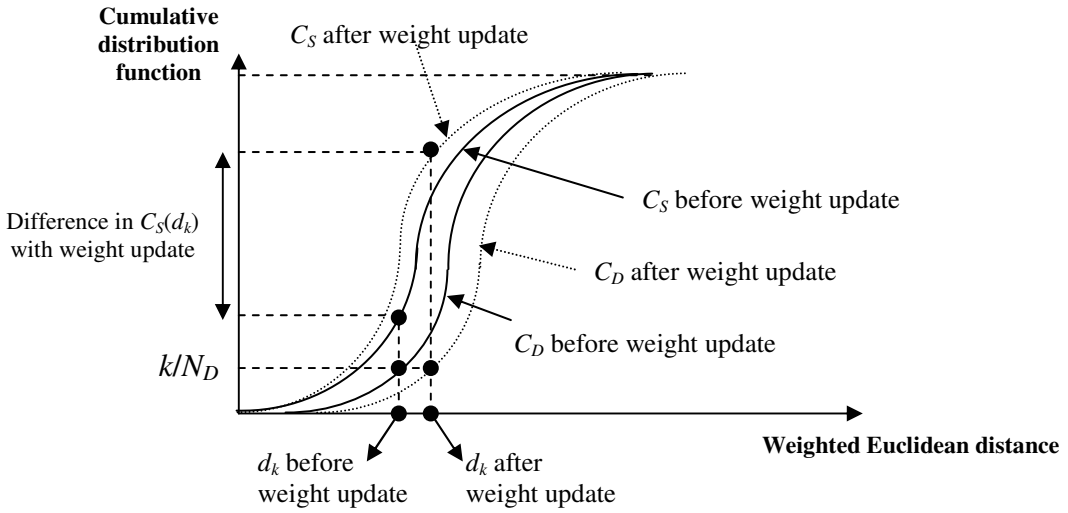

Fig. 5.6. Cumulative distribution functions C_S and C_D before and after updating feature weights. This is a simplified version where P_S and P_D are Gaussian and their variances do not change with weight update. d_k is the k^{th} smallest distance from the query to all its mismatches in the database. The change in weights causes C_S and d_k to change, and leads to an increase in $C_S(d_k)$.

It may seem that $P_{success}$ is independent of $C_D(d_k)$, which is unintuitive, and incorrect. In fact, $C_S(d_k)$ and $C_D(d_k)$ are related as follows:

$$N_S C_S(d_k) + N_D C_D(d_k) = k \tag{5.10}$$

From (5.10), we can see that $C_S(d_k)$ and $C_D(d_k)$ are inversely related – an increase in one automatically entails a decrease in the other. Thus, $P_{success}$ depends on $C_S(d_k)$ as much as $C_D(d_k)$.

It seems intuitive that the number of matches retrieved in top k as well as $P_{success}$ will increase with the weight update, based on the qualitative arguments presented so far (Figure 5.6). But now we provide an analytical basis for this intuitive result. More specifically, we show that a decrease in the mean rank $\overline{R}$ of matches implies an increase in $P_{success}$.

We know that $\overline{R}$ decreases with the update in weights (as targeted by the SLIDER algorithm, shown in Figure 4.3). This is represented as integrals, using (5.2):

$$N_D \int_{x=0}^{\infty} P_S(x) C_D(x) dx \quad > \quad N_D \int_{x=0}^{\infty} \widetilde{P_S}(x) \widetilde{C_D}(x) dx \tag{5.11}$$

where $P_S(x)$ and $\widetilde{P_S}(x)$ are the probability density functions for similar cases before and after the weight update respectively. $C_D(x)$ and $\widetilde{C_D}(x)$ are the cumulative distribution functions for different cases before and after the weight update respectively. (The "~" symbol is used to denote means and standard deviations *after* the weight update). If $P_S(x)$ and $P_D(x)$ are assumed to be Gaussian before and after the weights are changed, then (5.11) can be expanded to the following:

$$\int_{x=0}^{\infty} \frac{e^{-(x-\mu_S)^2/\left(2\sigma_S^2\right)}}{\sigma_S\sqrt{2\pi}} \int_{y=0}^{x} \frac{e^{-(y-\mu_D)^2/\left(2\sigma_D^2\right)}}{\sigma_D\sqrt{2\pi}} dydx \; > \; \int_{x=-\infty}^{\infty} \frac{e^{-(x-\widetilde{\mu_S})^2/\left(2\widetilde{\sigma_S}^2\right)}}{\widetilde{\sigma_S}\sqrt{2\pi}} \int_{y=0}^{x} \frac{e^{-(y-\widetilde{\mu_D})^2/\left(2\widetilde{\sigma_D}^2\right)}}{\widetilde{\sigma_D}\sqrt{2\pi}} dydx \tag{5.12}$$

Given that there is no analytical solution to the cumulative distribution of the normal density function, we make the following affine transformations that will put the functions in the same frame of reference and enable comparison between the distributions before and after updating the weights. The transformations essentially reduce P_D and $\widetilde{P_D}$ to the standard normal distribution (with mean = 0, and standard

deviation = 1). The location and scale parameters of P_S and $\widetilde{P_S}$ have to be transformed in the same proportions to maintain the inequality in (5.12). Let $z=(x-\mu_D)/\sigma_D$ and $w=(y-\mu_D)/\sigma_D$; we express (5.12) in terms of z and w by first substituting x by $(z\sigma_D)+\mu_D$, y by $(w\sigma_D)+\mu_D$, dx by $\sigma_D dz$ and dy by $\sigma_D dw$ on the LHS of (5.12). We also adjust the limits of the integration accordingly. If we make the corresponding transformations on the RHS of (5.12) as well, we obtain

$$\int_{z=-\infty}^{\infty} \frac{e^{-(z\sigma_D+\mu_D-\mu_S)^2/\left(2\sigma_S{}^2\right)}}{\sigma_S\sqrt{2\pi}} \int_{w=-\infty}^{z} \frac{e^{-w^2}}{\sigma_D\sqrt{2\pi}}\left(\sigma_D dw\right)\left(\sigma_D dz\right)$$

$$> \int_{z=-\infty}^{\infty} \frac{e^{-(z\widetilde{\sigma_D}+\widetilde{\mu_D}-\widetilde{\mu_S})^2/\left(2\widetilde{\sigma_S}^2\right)}}{\widetilde{\sigma_S}\sqrt{2\pi}} \int_{w=-\infty}^{z} \frac{e^{-w^2}}{\widetilde{\sigma_D}\sqrt{2\pi}}\left(\widetilde{\sigma_D} dw\right)\left(\widetilde{\sigma_D} dz\right) \qquad (5.13)$$

We convert (5.13) into

$$\int_{z=-\infty}^{\infty} \frac{e^{-\left(z-\frac{\mu_S-\mu_D}{\sigma_D}\right)^2/\left[2(\sigma_S/\sigma_D)^2\right]}}{(\sigma_S/\sigma_D)\sqrt{2\pi}} \int_{w=-\infty}^{z} \frac{e^{-w^2}}{\sqrt{2\pi}} dwdz$$

$$> \int_{z=-\infty}^{\infty} \frac{e^{-\left(z-\frac{\widetilde{\mu_S}-\widetilde{\mu_D}}{\widetilde{\sigma_D}}\right)^2/\left[2\left(\widetilde{\sigma_S}/\widetilde{\sigma_D}\right)^2\right]}}{\left(\widetilde{\sigma_S}/\widetilde{\sigma_D}\right)\sqrt{2\pi}} \int_{w=-\infty}^{z} \frac{e^{-w^2}}{\sqrt{2\pi}} dwdz \qquad (5.14)$$

This can be simplified into

$$\int_{z=-\infty}^{\infty} \underline{P_S}(z) \int_{w=-\infty}^{z} N(w;0,1)dwdz \;>\; \int_{z=-\infty}^{\infty} \underline{\widetilde{P_S}}(z) \int_{w=-\infty}^{z} N(w;0,1)dwdz \qquad (5.15)$$

where $N(w;0,1)$ is the standard normal function (with variable w). $\underline{P_S}(z)$ and $\underline{\widetilde{P_S}}(z)$ are the transformed probability density functions of distances to matches before and after weight update (the affine transformation is denoted by the underline) i.e.

$$\underline{P_S}(z) = \frac{e^{-\left(z-\frac{\mu_S-\mu_D}{\sigma_D}\right)^2/\left[2(\sigma_S/\sigma_D)^2\right]}}{(\sigma_S/\sigma_D)\sqrt{2\pi}} \qquad (5.16)$$

$$\underline{\widetilde{P_S}}(z) = \frac{e^{-\left(z-\frac{\widetilde{\mu_S}-\widetilde{\mu_D}}{\widetilde{\sigma_D}}\right)^2 / \left[2(\widetilde{\sigma_S}/\widetilde{\sigma_D})^2\right]}}{(\widetilde{\sigma_S}/\widetilde{\sigma_D})\sqrt{2\pi}} \tag{5.17}$$

It can be observed that $\underline{P_S}$ and $\underline{\widetilde{P_S}}$ are themselves normal distributions with means and variances as follows:

$$\underline{P_S}(z) = N\left(z; \frac{\mu_S - \mu_D}{\sigma_D}, (\sigma_S/\sigma_D)^2\right) \tag{5.18}$$

and

$$\underline{\widetilde{P_S}}(z) = N\left(z; \frac{\widetilde{\mu_S} - \widetilde{\mu_D}}{\widetilde{\sigma_D}}, (\widetilde{\sigma_S}/\widetilde{\sigma_D})^2\right) \tag{5.19}$$

With the affine transformation, the probability density functions in Figure 5.4 are transformed as shown in Figure 5.7 (if we assume the variances do not change with weight update i.e. $\sigma_S = \widetilde{\sigma_S}$ and $\sigma_D = \widetilde{\sigma_D}$). We observe that the mean of the affine transformed density function for matches changes from $\frac{\mu_S - \mu_D}{\sigma_D}$ to $\frac{\widetilde{\mu_S} - \widetilde{\mu_D}}{\widetilde{\sigma_D}}$ with the update in weights.

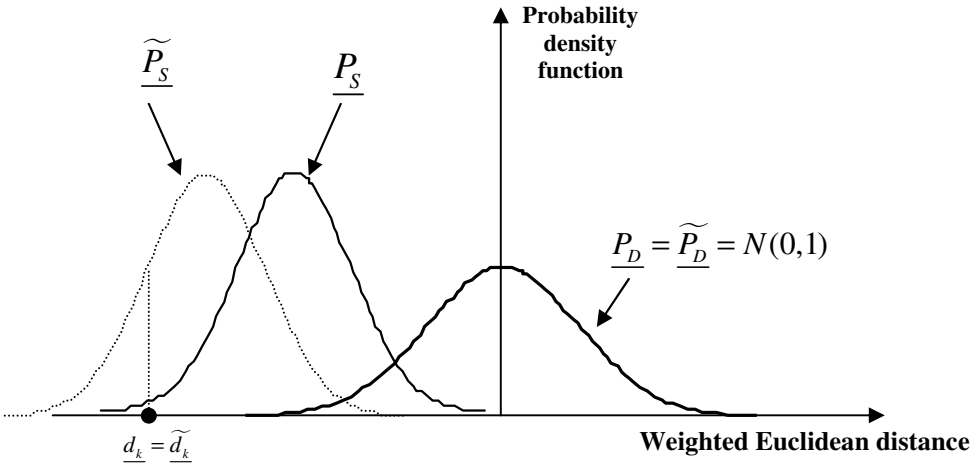

Fig. 5.7. Probability density functions after an affine transformation. If the underlying distributions are normal and their variances remain unchanged with weight update, then P_D and $\widetilde{P_D}$ become the standard normal distribution after the affine transformation described above. P_S and $\widetilde{P_S}$ are proportionally adjusted; after the transformation, they become normal distributions with means and variances as defined in (5.18) and (5.19).

From (5.15), we can deduce that for all *x*,

$$\int_{z=-\infty}^{x} \underline{P_S}(z)dz < \int_{z=-\infty}^{x} \underline{\widetilde{P_S}}(z)dz \tag{5.20}$$

This step can be formally proved as follows: Given our assumption that the variances of the distributions are equal, $\underline{\widetilde{P_S}}$ is a translation of $\underline{P_S}$ along the weight axis i.e. $\underline{\widetilde{P_S}}(z) = \underline{P_S}(z+\xi)$ for some constant ξ. We need to show that $\xi > 0$, which would lead to (5.20). To prove that $\xi > 0$, we re-write (5.15) as follows:

$$\int_{z=-\infty}^{\infty} \underline{P_S}(z)C_S(z)dz > \int_{z=-\infty}^{\infty} \underline{\widetilde{P_S}}(z)C_S(z)dz, \text{ where } C_S(z) = \int_{w=-\infty}^{z} N(w;0,1)dw \tag{5.21}$$

Substituting $\underline{\widetilde{P_S}}(z)$ by $\underline{P_S}(z+\xi)$, we get

$$\int_{z=-\infty}^{\infty} \underline{P_S}(z)C_S(z)dz > \int_{z=-\infty}^{\infty} \underline{P_S}(z+\xi)C_S(z)dz \tag{5.22}$$

The translation factor ξ can be applied to $C_S(z)$ instead of $\underline{P_S}(z)$ (in the opposite direction), without changing the inequality (5.22) i.e.

$$\int_{z=-\infty}^{\infty} \underline{P_S}(z)C_S(z)dz \quad > \quad \int_{z=-\infty}^{\infty} \underline{P_S}(z)C_S(z-\xi)dz \tag{5.23}$$

i.e.

$$\int_{z=-\infty}^{\infty} \underline{P_S}(z)\left[C_S(z)-C_S(z-\xi)\right]dz > 0 \tag{5.24}$$

Now since $\underline{P_S}$ is always non-negative, and $C_S(z)$ increases monotonically with z, we conclude that $C_S(z)-C_S(z-\xi) > 0$ i.e. $\xi > 0$. So, given the assumption that the variance of $\widetilde{\underline{P_S}}$ and the variance of $\underline{P_S}$ are equal, we infer (5.20) i.e. $\underline{P_S}$ is to the right of $\widetilde{\underline{P_S}}$, or $\xi > 0$.

Furthermore, the mean of $\underline{P_S}(z)$ is greater than the mean of $\widetilde{\underline{P_S}}(z)$. This can be written as the following [using (5.18) and (5.19)]:

$$\frac{\mu_S - \mu_D}{\sigma_D} \quad > \quad \frac{\widetilde{\mu_S} - \widetilde{\mu_D}}{\widetilde{\sigma_D}} \tag{5.25}$$

We observe that the changes in the distributions of distances that is expected from feature weight update do not require that the absolute mean μ_S of distances for matches to decrease, or the absolute mean μ_D of distances for mismatches to increase. What is necessary is that the relative difference between the means decreases [i.e. $\mu_S - \mu_D > \widetilde{\mu_S} - \widetilde{\mu_D}$, from (5.25)], assuming that the variances do not change.

We are particularly interested in how $C_S(d_k)$ (the cumulative distribution function of matches at the distance cutoff d_k corresponding to the k^{th} smallest distance) changes, since we recall from (5.9) that $P_{success}$ increases monotonically with $C_S(d_k)$. We note that d_k and $\widetilde{d_k}$ are the Euclidean distances at which the cumulative distribution functions equal k/N_D (Figure 5.6). $\underline{d_k}$ and $\widetilde{\underline{d_k}}$ are defined as follows:

$$\underline{d_k} = \frac{d_k - \mu_D}{\sigma_D} \qquad \text{and} \qquad \underline{\widetilde{d_k}} = \frac{\widetilde{d_k} - \widetilde{\mu}_D}{\widetilde{\sigma}_D} \tag{5.26}$$

The transformed probability density functions of mismatches before and after weight update are the same; they are, in fact, the standard normal distribution, as shown in Figure 5.7 i.e. $\underline{P_D} = \underline{\widetilde{P_D}} = N(0,1)$. It follows that $\underline{d_k} = \underline{\widetilde{d_k}}$, since the weighted Euclidean distances at cumulative probability = k/N_D would the same for $\underline{P_D}$ and $\underline{\widetilde{P_D}}$. So, given that $\underline{d_k} = \underline{\widetilde{d_k}}$, the following inequality holds [using (5.20)]:

$$\int_{z=-\infty}^{d_k} \underline{P_S}(z)dz < \int_{z=-\infty}^{\widetilde{d_k}} \underline{\widetilde{P_S}}(z)dz \tag{5.27}$$

In other words, the affine transformed cumulative function at d_k increases with a change in weights i.e. $\underline{\widetilde{C_S}}\left(\underline{\widetilde{d_k}}\right) > \underline{C_S}\left(\underline{d_k}\right)$. This implies an increase in $P_{success}$ [from (5.9)] i.e. an increase in the expected probability of getting at least one true match in the k filtered cases.

CHAPTER VI
EMPIRICAL RESULTS

In this chapter, we present empirical results of SLIDER from the protein crystallography domain, based on the TEXTAL system that interprets electron density maps to determine the 3D structures of proteins. The results are presented here from a machine learning perspective; in the next chapter we discuss the significance of the results in protein crystallography.

First, we analyze the impact of weighting these 76 features in the two-phase case retrieval strategy. We also compare SLIDER's performance to that of other feature selection and weighting algorithms. We also evaluate the effectiveness of the filtering approach to case retrieval in TEXTAL, using various approximate similarity measures.

6.1. Experimental Setup for SLIDER

In this evaluation, we use SLIDER to optimize weights for three different weighted distance metrics: Manhattan, Euclidean, and Minkowski distance of order 3 (or L_3). We closely look at which of the 76 features get how much weight, and analyze the dependence of weighting on the distance metric.

We also compare three different weighting schemes: (1) continuous weights, where we use the actual weights determined by SLIDER in the distance metric; (2) binary weights, where we convert the weights obtained from SLIDER to either 0 or 1. We use a threshold weight for that purpose – any weight below that threshold is converted into a 0, and any weight above the threshold is converted into a 1. Finally, for the purpose of comparison, we use uniform weights i.e. all features are weighted equally.

We assess SLIDER's performance on "ideal" protein maps i.e. maps that have been artificially generated by back-transforming from their correct structures at 2.8 Å. (In the next chapter, we discuss the impact of SLIDER for solving real, experimentally determined maps.) Analysis of the performance of SLIDER on ideal maps is insightful

since it reduces the confusion that noise, variance in resolution, and phase error in real maps may introduce.

We use a database of ideal maps for our case-based reasoning approach. Ideal maps are more suitable than real maps, since the latter usually have a lot of variance, and noise, and tend to confuse the pattern recognition. Ideal maps computed at 2.8 Å leads to better case matching and retrieval in solving real maps that are also re-computed at 2.8 Å. In general, we use ideal maps for the various machine learning tasks in TEXTAL, such as in training sets for the neural network to determine Cα positions, or for tuning feature weights in SLIDER.

Our evaluation of SLIDER on ideal maps involves four sets of data: (1) A training set of 3-tuples, each with an instance (of density pattern), a match and a mismatch to determine crossover points and tune the weights. In Figure 4.3, this set is referred to as the training set *T*. (2) Another training set of examples, each with an instance, a match and a set of mismatches, for the purpose of evaluating weight vectors in the heuristic search for optimum weights. In Figure 4.3, this set is referred to as the validation set *V*. (3) A database from where matches are actually retrieved. (4) An independent test set.

These four sets were constructed using maps from independent sets of proteins in PDBSelect (Hobohm *et al*., 1992), a subset of the PDB database (Berman *et al*., 1992). As mentioned earlier, the maps were artificially generated at 2.8 Å from their correct structures, and the regions were defined as 5 Å spheres centered on Cα atoms.

In this experiment, the training set *T* (used to tune the weights) consisted of 200 3-tuples of density regions (drawn randomly from 48 proteins in PDBSelect). For each example, a match and a mismatch were pre-determined by the calculation of density correlation. An instance is defined as a match (mismatch) if its density correlation to the corresponding example is above (below) 0.7, a threshold above which local patterns of density tend to look similar.

For the validation set *V* (used to evaluate weight vectors), we chose 100 density regions obtained from 51 proteins in PDBSelect (independently of the proteins used for tuning the weights). For each region, we exhaustively searched a database of about

30,000 regions (obtained from yet another 137 proteins in PDBselect) to find their true, objective matches (based on density correlation). We then used the weighted feature-based distance metrics to rank all the 30,000 regions according to similarity, and find out how many truly good matches are found in the top *k* by the feature-based metric.

As discussed earlier, one need not be searching for absolute best match, since the second, third, and other subsequent best matches will often be adequate. So we use a more relaxed notion of a *good match*: Given an example *x*, a database $\mathbb{D}$ and an objective similarity metric *obj*, an example *y* in $\mathbb{D}$ is said to be good match of *x* if

$$\frac{obj(x,b) - obj(x,y)}{obj(x,b)} < \delta \tag{6.1}$$

where δ ($\delta \geq 0$) is a tolerance, and *b* is the absolute best match of *x* in $\mathbb{D}$ i.e. $b = \underset{d \in \mathbb{D}}{\text{argmax}}\ obj(x,d)$.

We here assume that $obj > 0$, and it increases with similarity. This definition accepts multiple hits (in the database) as reasonable matches if they have a similarity within some threshold of the best possible score. For any given tolerance δ, let the number of good matches be λ. In this domain, the objective similarity metric *obj* is density correlation between two spherical regions, which involves an expensive rotational search for the best possible superposition. Figure 6.1 shows how the average number of matches λ (over the test set of 100 regions) varies with tolerance δ, using a database of about 30,000 instances. Domain experts found that a tolerance of about 0.02 is reasonable to produce useful models of side chains. (Note that density correlation ranges from 0 to 1; if the absolute best match has a correlation of 0.90, then with a tolerance of 0.01, all cases with correlation between 0.89 and 0.90 are considered to be similar – all others are different).

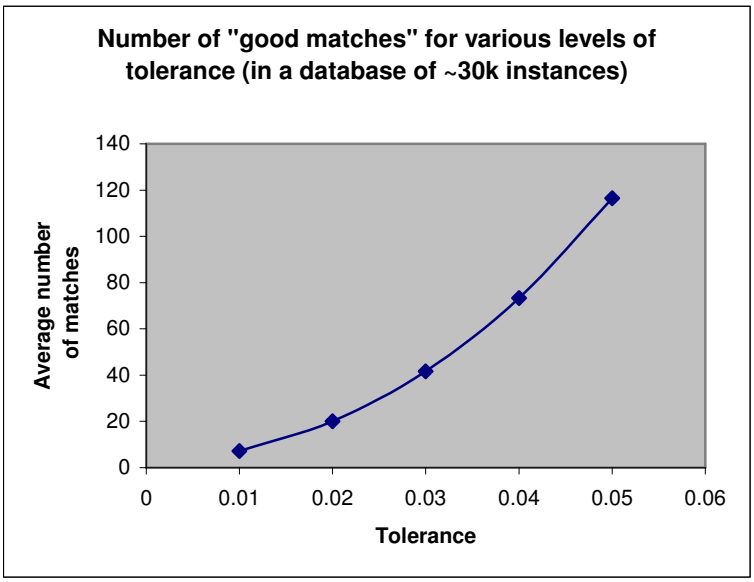

Fig. 6.1. The number of "good matches" λ grows exponentially with the tolerance δ.

6.2. Analysis of the Weights Assigned by SLIDER

Since SLIDER is greedy and non-deterministic, we run it ten times for each of the three distance metrics (Manhattan, Euclidean, and L_3). The average weight for each of the 76 features was calculated and proportionally adjusted so that all weights sum up to 1. For all the three distance metrics, between 23 and 31 features (out of 76) were selected (we take a feature to be selected if its weight is greater than 0.01). Moreover, there is strong tendency to choose the same features, and even weigh them similarly. There are 34 features for which all three metrics have yielded zero weight. Table 6.1 shows all features that were found to be irrelevant by SLIDER for all three metrics. We emphasize that these features need not be irrelevant in absolute terms; they might just be useless in the context of other features deemed relevant, especially in situations where some features are redundant. Table 6.2 shows those features that were found relevant for *any* of the three metrics – the average weights over ten runs are also shown, with their standard errors.

It should be noted that when different SLIDER runs "select" different features, the features selected may actually be quite similar, in two ways: (1) The features could be closely related e.g. standard deviation, skewness, and kurtosis, or the three moments of inertia; (2) They could be the same feature (like mean density) but calculated over different radii, especially those radii that are close to each other.

Figure 6.2a tries to capture this concordance in returned weights by first sorting the features based on radius and then listing them in a particular order (such that related features are as close together as possible). Figure 6.2b groups the features the other way round i.e. first it lists all features (in the same order as in Figure 6.2a), and then sorts them based on radius, in ascending order. The weights have been linearly graded on a five-level scale, where darker shade implies higher weight.

Table 6.1. Irrelevant features

Name (and radii in Å) of features with weight = 0
Mean (3, 4)
Standard deviation (4, 5, 6)
Skewness (4, 5, 6)
Kurtosis (5)
Primary moment of inertia (6)
Secondary moment of inertia (3, 4, 5, 6)
Tertiary moment of inertia (4)
Ratio of primary to secondary moment of inertia (5, 6)
Ratio of primary to tertiary moment of inertia (6)
Ratio of secondary to tertiary moment of inertia (5, 6)
Distance to center of mass (6)
Maximum angle between spokes (3, 6)
Minimum angle between spokes (4, 6)
Median angle between spokes (6)
Radial sum of first spoke (3, 4, 5, 6)
Radial sum of second spoke (3)
Radial sum of third spoke (4, 5, 6)

The list of 34 features found irrelevant by SLIDER for all 3 metrics.

Table 6.2. Relevant features

Feature name (radius in Å)	Manhattan	Euclidean	L_3
Mean (5)	.046 ± .015	.028 ± .008	.036 ± .012
Mean (6)	.020 ± .011	.026 ± .007	.000 ± .002
Standard deviation (3)	.000 ± .008	.042 ± .007	.000 ± .005
Skewness (3)	.017 ± .007	.030 ± .006	.000 ± .006
Kurtosis (3)	.033 ± .007	.000 ± .005	.000 ± .004
Kurtosis (4)	.016 ± .006	.000 ± .005	.000 ± .008
Kurtosis (6)	.000 ± .004	.000 ± .004	.026 ± .009
Primary moment of inertia (3)	.021 ± .005	.000 ± .006	.000 ± .009
Primary moment of inertia (4)	.000 ± .006	.031 ± .008	.024 ± .008
Primary moment of inertia (5)	.000 ± .000	.000 ± .000	.024 ± .012
Tertiary moment of inertia (3)	.000 ± .000	.000 ± .007	.023 ± .009
Tertiary moment of inertia (5)	.017 ± .009	.029 ± .009	.025 ± .010
Tertiary moment of inertia (6)	.000 ± .006	.023 ± .006	.025 ± .006
Ratio of primary to secondary MOI (3)	.029 ± .006	.056 ± .004	.062 ± .009
Ratio of primary to secondary MOI (4)	.014 ± .005	.023 ± .003	.000 ± .002
Ratio of primary to tertiary MOI (3)	.000 ± .002	.030 ± .008	.000 ± .004
Ratio of primary to tertiary MOI (4)	.026 ± .008	.024 ± .006	.000 ± .005
Ratio of primary to tertiary MOI (5)	.018 ± .007	.000 ± .006	.022 ± .005
Ratio of secondary to tertiary MOI (3)	.087 ± .011	.040 ± .007	.067 ± .015
Ratio of secondary to tertiary MOI (4)	.035 ± .007	.036 ± .007	.029 ± .005
Distance to center of mass (3)	.109 ± .005	.114 ± .006	.116 ± .012
Distance to center of mass (4)	.017 ± .006	.026 ± .007	.024 ± .006
Distance to center of mass (5)	.021 ± .007	.028 ± .005	.000 ± .006
Maximum angle between spokes (4)	.032 ± .007	.031 ± .005	.000 ± .004
Maximum angle between spokes (5)	.022 ± .007	.049 ± .007	.055 ± .013
Median angle between spokes (3)	.000 ± .003	.000 ± .005	.030 ± .005
Median angle between spokes (4)	.026 ± .006	.026 ± .006	.000 ± .003
Median angle between spokes (5)	.034 ± .005	.031 ± .005	.041 ± .007
Minimum angle between spokes (3)	.024 ± .006	.000 ± .004	.000 ± .003
Minimum angle between spokes (5)	.026 ± .006	.044 ± .004	.026 ± .007
Sum of spoke angles (3)	.000 ± .000	.000 ± .003	.026 ± .008
Sum of spoke angles (4)	.018 ± .006	.000 ± .006	.000 ± .006
Sum of spoke angles (5)	.088 ± .019	.054 ± .013	.106 ± .017
Sum of spoke angles (6)	.039 ± .006	.000 ± .006	.044 ± .014
Radial sum of second spoke (4)	.000 ± .000	.000 ± .004	.025 ± .011
Radial sum of second spoke (5)	.020 ± .009	.000 ± .005	.000 ± .001
Radial sum of second spoke (6)	.015 ± .007	.000 ± .002	.000 ± .000
Radial sum of third spoke (5)	.000 ± .006	.025 ± .009	.000 ± .000
Spoke triangle area (3)	.037 ± .010	.034 ± .006	.000 ± .006
Spoke triangle area (4)	.024 ± .005	.034 ± .006	.045 ± .012
Spoke triangle area (5)	.050 ± .016	.062 ± .013	.097 ± .015
Spoke triangle area (6)	.018 ± .007	.025 ± .005	.000 ± .002

List of features found relevant (i.e. non-zero weights) for *any* of the three Minkowski distance metrics. The weights are averaged over multiple runs. The standard errors are also shown.

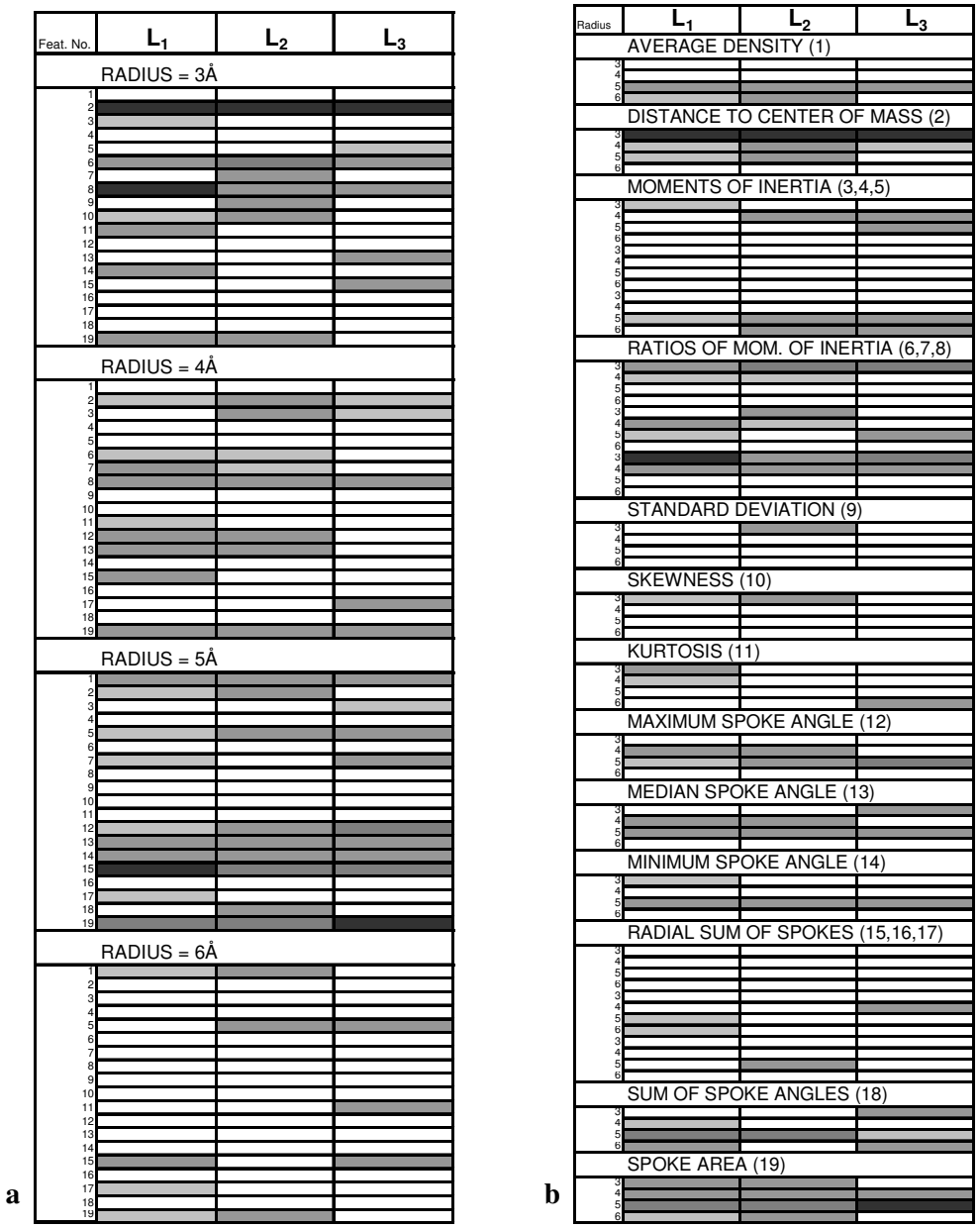

Fig. 6.2. The relative weights of 76 features returned by SLIDER for L_1 (Manhattan), L_2 (Euclidean), and L_3. In Fig. 6.2(a), the features are first sorted by radius, and for each radius, 19 features are listed in a specific order. In Fig. 6.2(b), the 19 features are first listed in the same order as in Fig. 6.2(a), and then sorted by radius. Darker shade implies higher weight. The white cells represent features with zero weight.

We make the following remarks regarding the feature weights computed by SLIDER:

- The consistency in features selected (and weighted) across the three metrics shows that the algorithm converges. But the risk of local minima still exists; this is partially addressed by the randomized choice of a feature in each iteration. The major cause of local minima is the fact that the weight of only one feature is greedily adjusted at a time.
- The absolute moments of inertia seem irrelevant individually, but their ratios provide more information related to the shape of the density pattern (e.g. spherical, ellipsoidal, etc.). This exemplifies the feature interaction problem (Jakulin and Bratko, 2004; Ioerger, 1999), where several features may not appear relevant on an individual basis, but when looked at in combination, they contribute significantly to the description of the pattern.
- The strong similarity of weights across the three Minkowski metrics is largely expected. Some weights are relevant, irrespective of the underlying metric. For example, the distance between the center of sphere and its center of mass at 3Å is weighted highly for all three metrics. Nonetheless, there are differences, and interestingly, these differences do capture the sensitivity of "optimum" weights to the metric being used. An important point to note is that, given a distance metric, using continuous weights does not improve pattern matching (as compared to binary weights) if the weights used are those optimized for another metric e.g. the weights determined by SLIDER for Euclidean will not make continuous weights outperform binary weights for Manhattan.

6.3. SLIDER's Impact on Retrieval for Ideal Maps

In this section, we analyze the contribution of SLIDER in retrieving matches from a database, given 100 test cases (electron density patterns) from ideal maps i.e. maps that have been calculated (back-transformed) from known protein structures.

Figures 6.3-6.5 plot the number of good matches (averaged over the test set) that the three Minkowski metrics obtain (from the database of 30,000 regions) in the top k, for various values of tolerance (at a constant $k = 500$).

Figures 6.6-6.8 plot the average number of good matches in the top k for the three metrics, this time varying k (keeping tolerance fixed at .02). We can observe that both feature selection (binary weights) and feature weighting (continuous weights) improve over uniform weights (all 76 features equally weighted). Furthermore, continuous weights are better than binary weights for all three metrics.

Figures 6.3-6.5 show that as tolerance increases, more matches are found, but at a higher rate for the weighted metrics compared to the non-weighted one. Of course, it can be argued that only one good match needs to be found, which can be achieved by setting k sufficiently high. But the improved retrieval allows us to use a lower k, and hence improve efficiency by reducing the database search time.

Figures 6.6-6.8 examine how performance scales with k, at constant tolerance. The results again show that using weights determined by SLIDER gives the best performance, since it needs the smallest value of k for the same number of hits (good matches).

Figure 6.9 shows the effectiveness of case retrieval in a different way: the number of instances retrieved in the top 500 that have the same amino acid identity as that of the test region. We can see that the weighted schemes help in getting more side chains of similar identity in the top 500. Similar results are obtained for other values of k (not shown). Selecting as many correct residues as possible in the top k is helpful in the sequence alignment step of TEXTAL – it is often rewarding to look at several top matches (rather than just the first), and see a better alignment with the sequence can be obtained with the second, third, or subsequent matches. Note, however, that even regions with the correct amino acid identity are not necessarily good matches, as they might represent a different side chain conformation. Conversely, sometimes residues of alternative amino acid identity can be good matches, due to structural similarity (e.g. between *valine* and *threonine*, or *glutamate* and *glutamine*).

Figure 6.10 compares the effectiveness of retrieval of the Euclidean metric to the *Mahalabonis distance* (Duda *et al*., 2001), which is a distance metric that takes into account some statistical properties of the data. The Mahalanobis distance is based on the correlations between the feature values; it effectively selects or weights features based on feature covariances. From Figure 6.10, it can be observed that, while the Mahalabonis metric outperforms the non-weighted Euclidean measure, both binary and continuous weighting schemes based on SLIDER outperform the Mahalabonis distance.

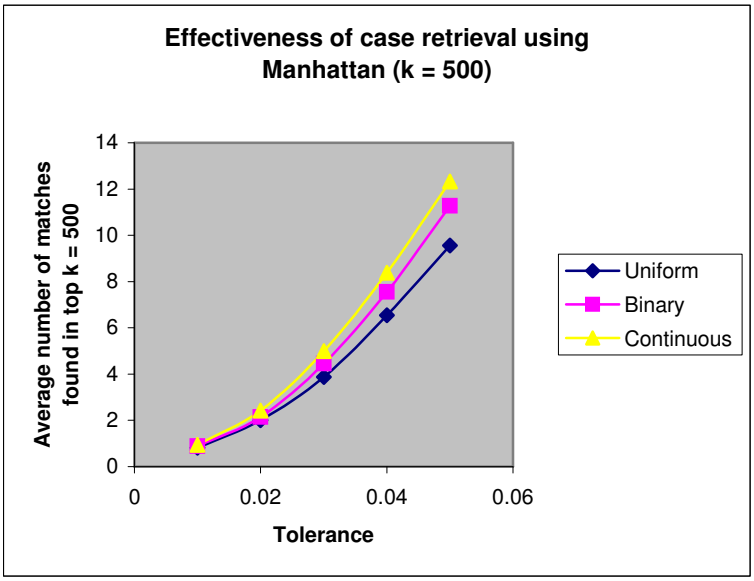

Fig. 6.3. Weighted Manhattan metrics find more matches than non-weighted Manhattan distance. The results shown are for top k = 500 from a database of ~30,000 regions (for various levels of tolerance). Also, continuous weights outperform binary weights.

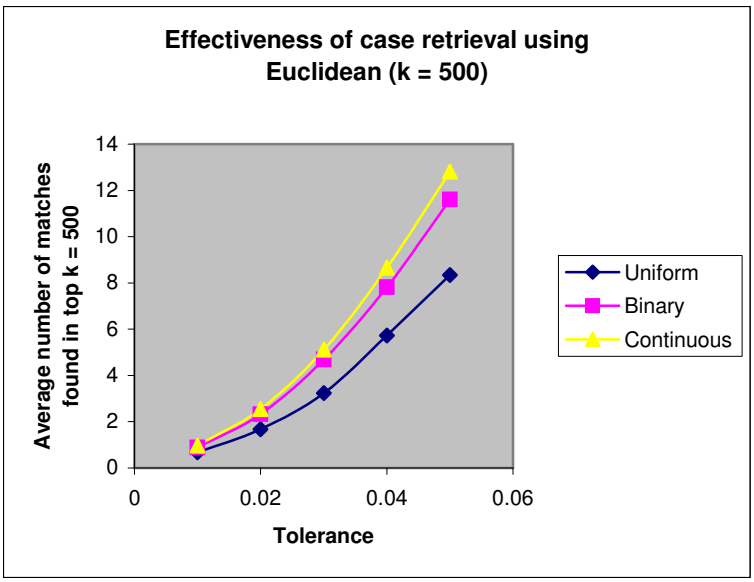

Fig. 6.4. Continuous weights are more effective than binary weights (using Euclidean distance) in retrieval of matching electron density patterns from a database. Both weighted metrics outperform the non-weighted (uniform weights) metric.

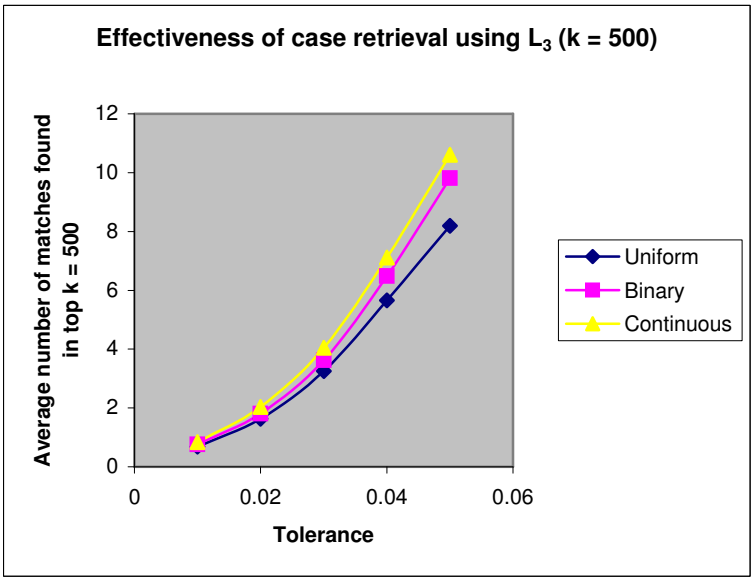

Fig. 6.5. Weighted L_3 metrics outperform the non-weighted (uniform) L_3 metric. Continuous weights outperform binary weights.

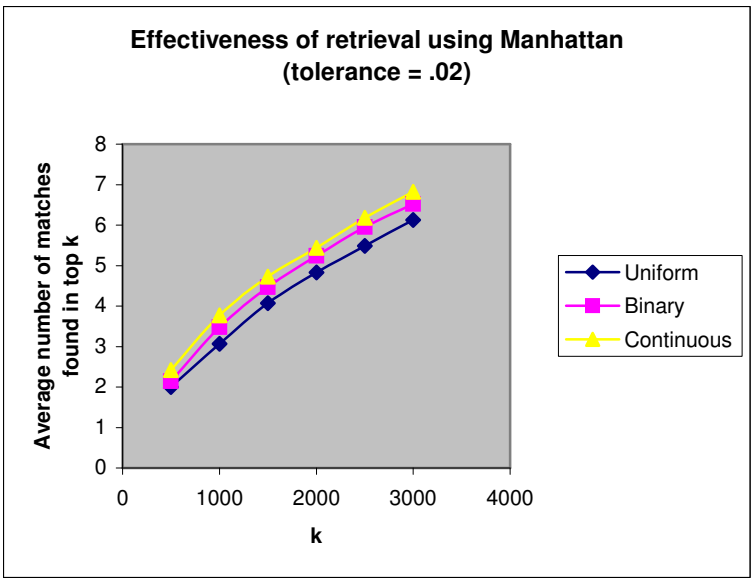

Fig. 6.6. Weighted Manhattan metrics find more matches than the non-weighted Manhattan metric. The results shown are for various values of k from a database of ~30,000 regions (tolerance = .02). Also, continuous weights outperform binary weights.

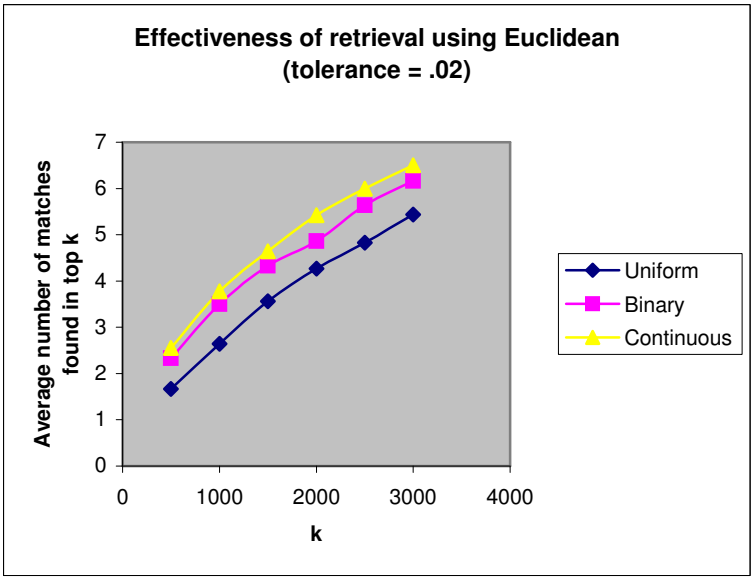

Fig. 6.7. Continuous weights are more effective than binary weights (using Euclidean distance) in retrieval of matching electron density patterns. The database has about ~30,000 regions. The results shown are for k ranging from 1000 to 3000. Both weighted metrics outperform the non-weighted (uniform weights) metric.

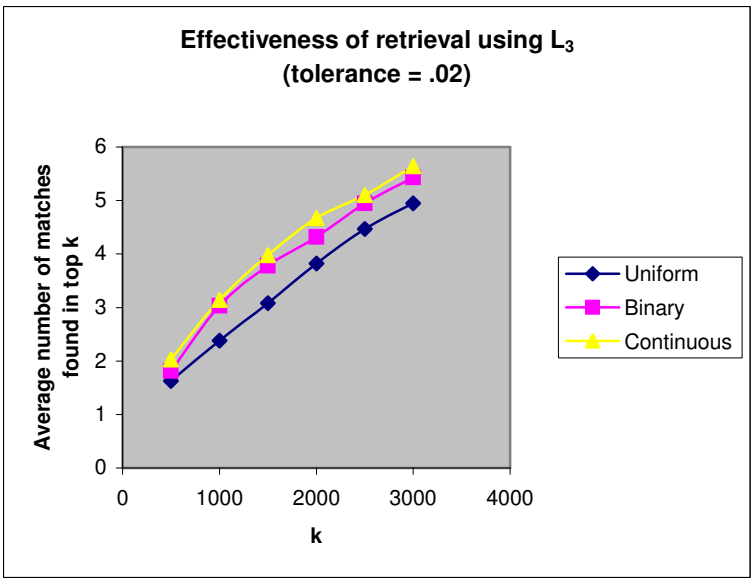

Fig. 6.8. Weighted L_3 metrics outperform the non-weighted (uniform) L_3 metric. Continuous weights outperform binary weights.

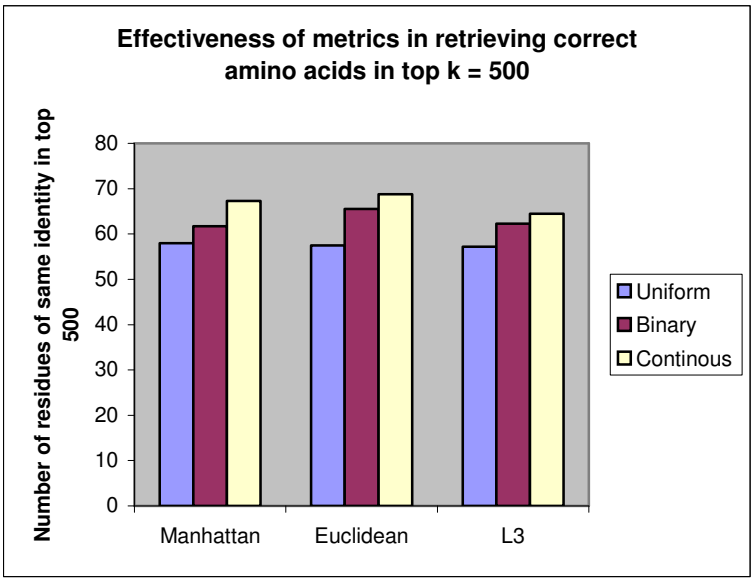

Fig. 6.9. Weighted metrics retrieve more matches with the same residue identity than non-weighted metrics. The results shown are for k = 500. Similar results are obtained with other values of k.

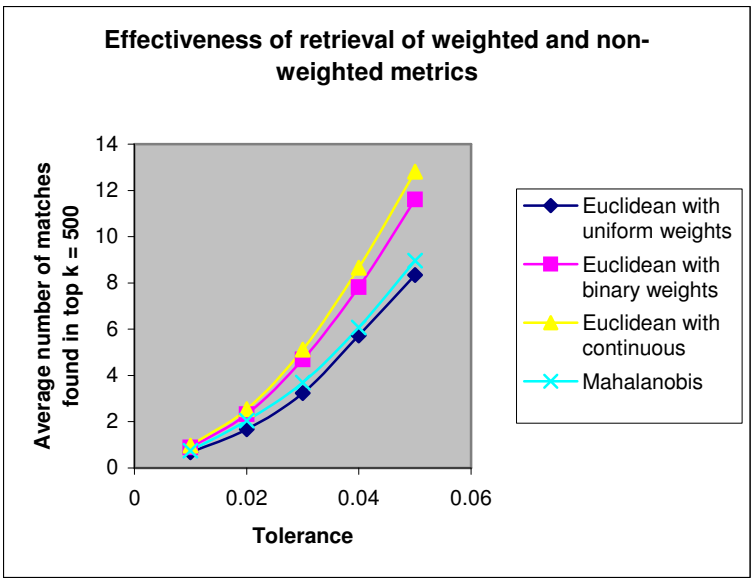

Fig. 6.10. Effectiveness of various weighted and non-weighted metrics. Mahalanobis distance outperforms the non-weighted (uniform) Euclidean metric in obtaining matches in the top k = 500 for various levels of tolerance. But weighted Euclidean distances (using weights determined by SLIDER) are more effective than the Mahalanobis metric in case matching and retrieval.

6.4. Comparing SLIDER to Other Feature Selection and Weighting Algorithms

In this section, our main aim is to compare the performance of SLIDER to that of other feature selection and weighting algorithms. In this experiment, we do an analysis similar to that in section 6.3, but with the actual database that TEXTAL currently uses (containing about 50,000 instances).

Figure 6.11 illustrates how the weighted Euclidean metric (with weights determined by SLIDER) performs compared to the non-weighted (uniform) Euclidean distance in the retrieval of matches in the top k = 400 cases. Again, we compare the two metrics for various tolerances in defining similarity, using density correlation. With feature weights determined by SLIDER, about twice as many matches are found at each level of tolerance (compared to uniform weights). Figure 6.12 shows the effectiveness of weighting in a slightly different way – at a fixed tolerance (of 0.02), we show how many

truly similar cases are placed in the top k, for varying values of k. Again, by using SLIDER weights, the number of matches retrieved is roughly doubled.

In Figure 6.13, we broaden the comparison (with the same experimental data) and examine how SLIDER performs compared to other standard feature selection and weighting methods. The actual induction algorithm (retrieval from the database) is too expensive and impractical to be used to evaluate and select weights, as in typical wrappers. So, in all the algorithms we use the mean rank of matches [$\overline{R}$, as defined in (4.10)] to tune the weights. Figure 6.13 shows retrieval accuracy for the weighted Euclidean distance metric using SLIDER weights, uniform weights (i.e. non-weighted), and weights determined by the following algorithms:

1. DIET (Kohavi *et al*., 1997) is a wrapper approach that searches a space of discrete weights. It uses a set of p+1 possible weights: 0, 1/p, 2/p, …, (p -1)/p, 1 (the results shown are based on p = 10). The operators in this heuristic search replace the current weight of a feature by either the next larger or smaller value in the allowed set (unless the minimum or maximum has been reached) based on improvement in ranking.
2. Sequential forward and backward selection (SFS and SBS) start from an empty and full set of features respectively, and greedily add or remove one feature at a time, depending on which addition or deletion improves ranking the most (Kittler, 1978).
3. The linear programming (LP) method (described in section 4.3) that approaches the feature weighting problem in a way that is related to SLIDER, but with some fundamental differences. SLIDER aims at maximizing the *number* of training instances where the weighted Euclidean distance between an instance and its mismatch is greater than the distance between the instance and its match. In contrast, the LP approach tries to optimize the aggregate difference in the (squared) weighted Euclidean distances themselves. The GNU Linear Programming Kit (http://www.gnu.org/software/glpk/glpk.html) was used to solve the optimization problem.

Figure 6.13 shows that SLIDER weights outperform weights determined by all other methods, including the LP approach. This can be attributed to the way SLIDER chooses candidate weights for evaluation, based on *crossover* points, as discussed in Chapter IV.

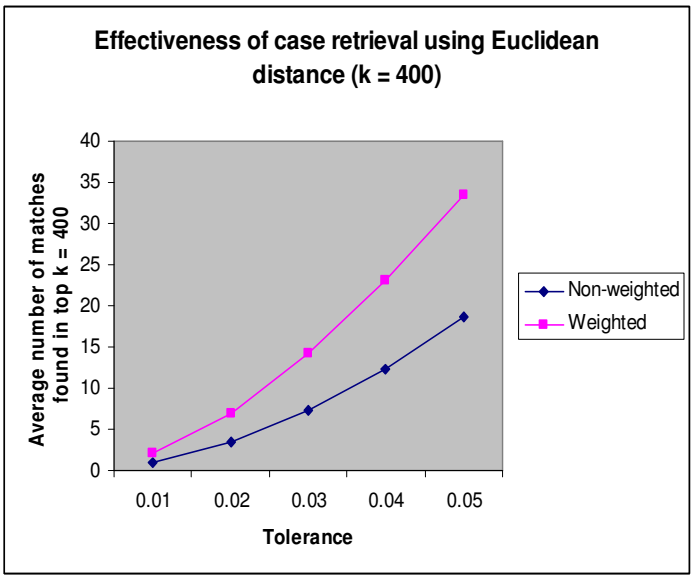

Fig. 6.11. Weighted Euclidean distance places more matches in the top 400 cases than non-weighted Euclidean distance. This is true for various values of tolerance, which determines how lenient we are in defining similarity, based on density correlation – with higher tolerance, more cases will qualify as being similar.

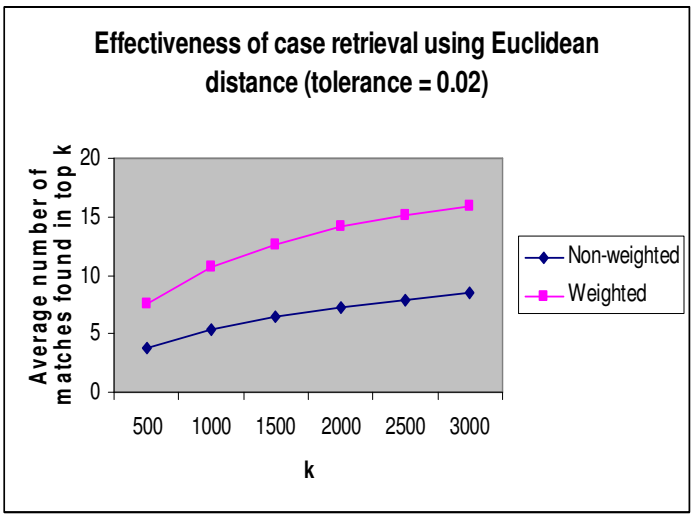

Fig. 6.12. Weighted Euclidean distance places more matches in the top k cases than non-weighted Euclidean. This is true for various values of k (at a fixed tolerance in the evaluation of similarity based on density correlation).

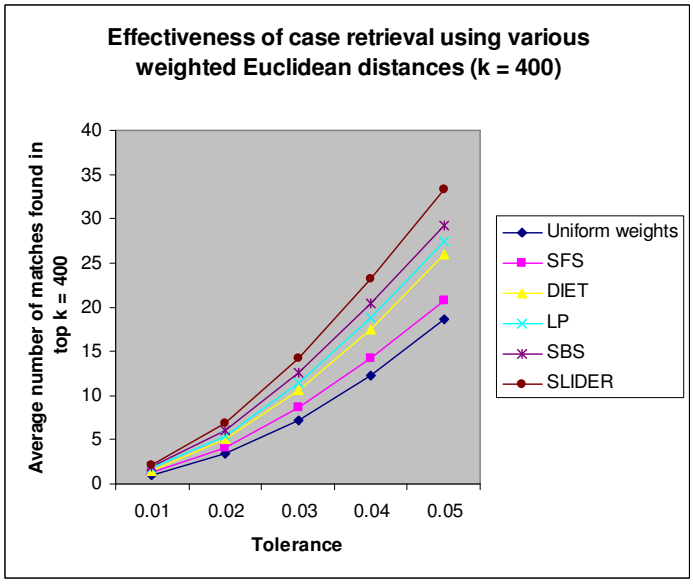

Fig. 6.13. SLIDER is more effective in weighting features (such that true matches are highly ranked and retrieved from the TEXTAL database) as compared to other standard feature weighting and selection algorithms.

CHAPTER VII
DATABASE RETRIEVAL AND FEATURE WEIGHTING: APPLICATION TO PROTEIN CRYSTALLOGRAPHY

In the previous chapter, we presented empirical results (in protein crystallography) from a machine learning perspective, with minimal analysis and discussion relevant to the domain. In this chapter, we discuss the significance of the methods we developed (for efficient case retrieval and feature weighting) in enhancing protein model-building through interpretation of electron density maps. We also relate the challenges and methods in our work to other work done in protein crystallography (and structural bioinformatics, in general), particularly those that are based on exploiting information available in databases, and the use of features.

7.1. Size of the TEXTAL Database

Accurate and fast automated protein model-building are critical to high-throughput structural genomics pipelines. After data collection, crystallographers usually build protein models iteratively and incrementally, using a number of tools to improve the map and the model. They have to experiment with many possible scenarios, and this can be very time-consuming. So, it is of practical importance for automated electron map interpretation methods to solve structures quickly (in less than a few hours). Database approaches are usually expensive, because a large number of instances are required for adequate coverage of all possible variations of the problem. In TEXTAL, we use a large library of side chains that covers all the 20 amino acids, for different conformations, and in various orientations. To achieve a reasonable tradeoff between efficiency of model-building, and good coverage of density patterns (and hence quality of solutions), we adopt the two-phase strategy discussed earlier: given a local density region in the input map, we quickly compare the region to every instance in the database (about 50,000 local spherical regions from 200 proteins) using an efficient feature-based metric. This

enables us filter the most promising k candidates, which are then analyzed by a more expensive method (density correlation) to find the best match.

We empirically observed that a database of about 50,000 regions is suitable for this approach. Figure 7.1 shows how TEXTAL performs on test maps, using databases of different sizes (with the same distribution of residues) and a constant $k = 400$. We show the percentages of structurally similar residues that TEXTAL found for the test proteins. (The 20 amino acids are divided into 10 groups, where amino acids within a group are structurally similar e.g. *alanine* and *glycine*, or *valine* and *threonine*; some groups have only one amino acid, such as *proline* and *isoleucine*.) We also observed that the gain in performance beyond 50,000 is not significant.

Our set of test cases includes real (experimentally determined) maps for the following proteins: (i) *CzrA* (*chromosome-determined zinc-responsible operon A*) is a dimer consisting of 96 amino acids in four α-helices (Eicken *et al*., 2003); (ii) *IF-5A* (*translation initiation factor 5a*) consists of a pair of β-barrel domains; it has 137 amino acids (Peat *et al*., 1998); (iii) *MVK* (*mevalonate kinase*) is a medium-sized protein with 317 amino acids, including both α and β secondary structures (Yang *et al*., 2002); (iv) *PCA* (*mycolic acid cyclopropane synthase*) has 262 amino acids, and is made up of both α-helices and β-sheets (Huang *et al*., 2002).

Note that a database of 50,000 regions seems large, because there are only a limited number of rotamers for each amino acid. Nonetheless, we need more instances, because TEXTAL may not find the right rotamer from the library, since the rotamer may be in a different orientation relative to the density region in the input map. Furthermore, a large database will lead to more robust modeling because input maps are typically at low resolution, noisy, and distorted. The two-phase approach that we propose makes a large library affordable, because the filtering process is very efficient (since it involves computing Euclidean distances). The effectiveness of the filtering depends on a good distance metric (based on relevant and properly weighted features) that enrich the pre-selected k examples with as many true matches as possible. This is the motivation to weigh the features to reflect their relevance in comparing instances.

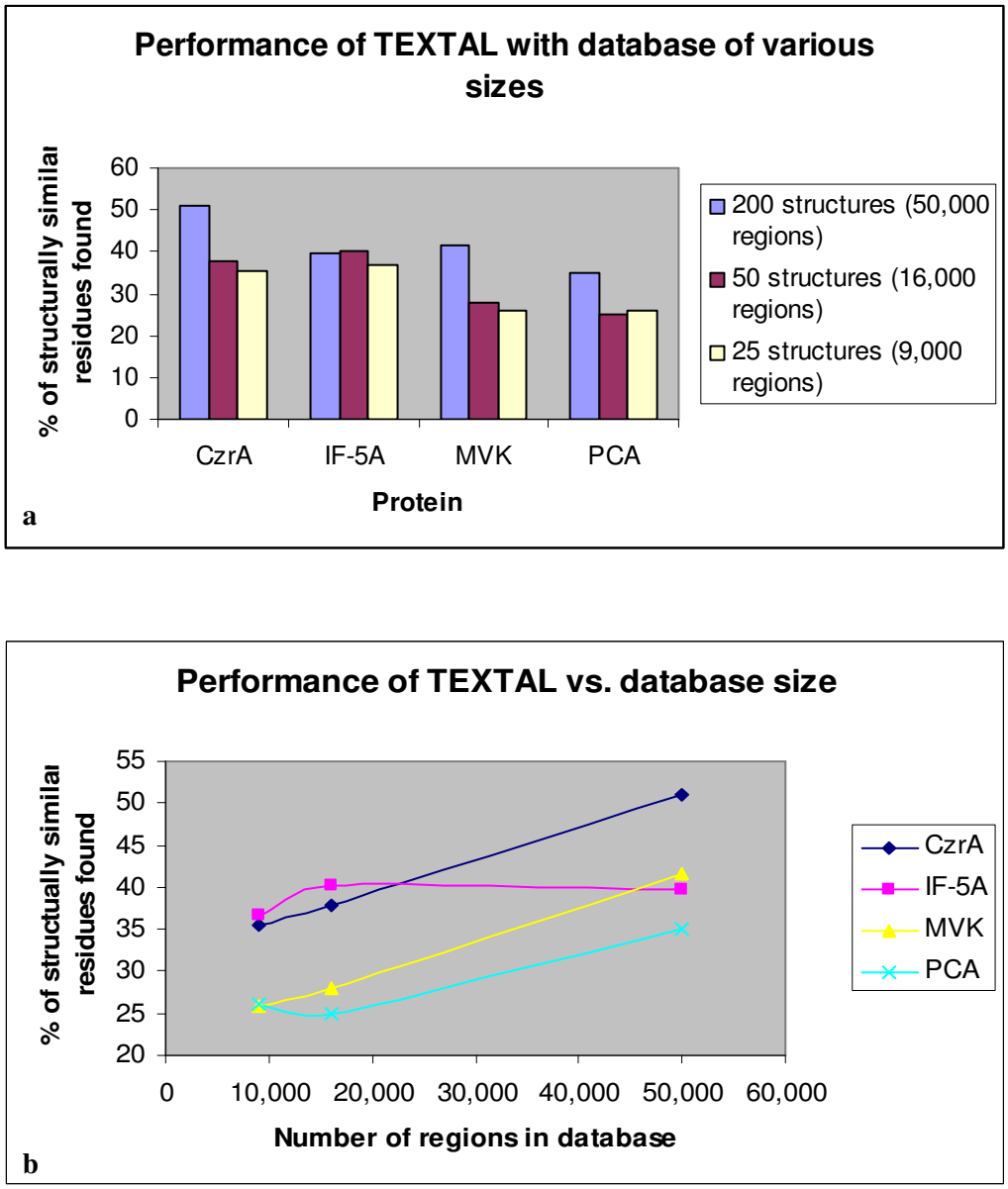

Fig. 7.1. Quality of models built by TEXTAL with databases of various sizes.

7.2. Composition of the TEXTAL Database

The database that TEXTAL uses is made of maps that are computed (back-transformed at 2.8 Å) from manually built and refined structures from the PDB. It can be argued that the database should consist of maps at various resolutions and phase error,

since real maps that we need to solve will vary in terms of resolution, phase error, and other types of noise. This hypothesis was tested by using a database of real maps. The results are shown in Figure 7.2, where we compare the performance of TEXTAL using two databases of similar sizes (and with similar proportions of amino acids): one with "ideal" maps, and the other with real maps. It can be observed that TEXTAL performs better with the database of ideal maps. We argue that although the input map will invariably be noisy, a database of real maps (at different resolutions and with noise) will tend to confuse pattern recognition. By keeping the database uniform and "clean", there seems to be a better chance of distinguishing patterns in a real map. We also tried introducing controlled (uniform) error in the database (such as a phase error of 40°), but this did not improve the recognition of density patterns.

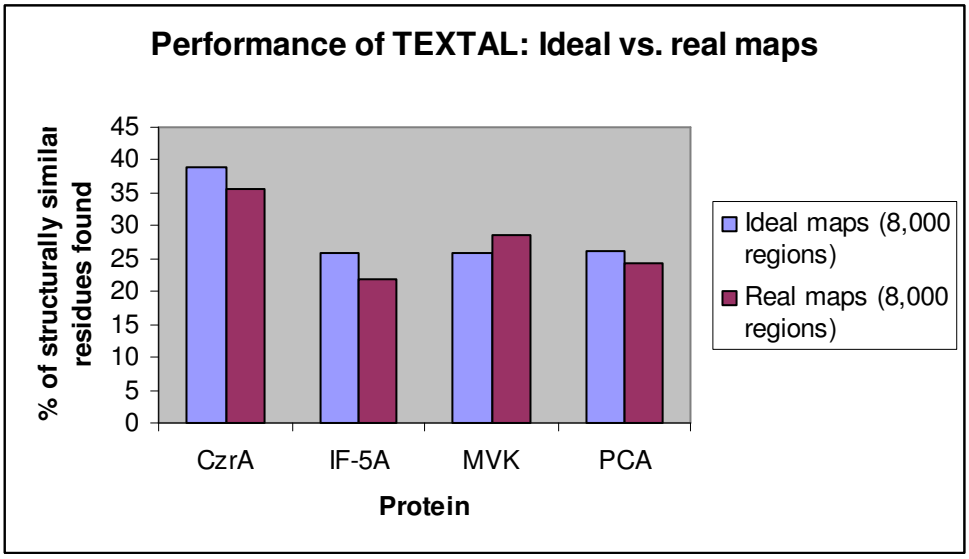

Fig. 7.2. Quality of models built by TEXTAL with databases of ideal and real maps. Using real maps does not improve the performance of TEXTAL.

One variation that led to significant improvement is uniform scaling of the electron density maps i.e. modifying the density values such that maps can be meaningfully

compared. The scaling of density is similar to a normalization procedure, where the mean density is zero, and the standard deviation is 1.0. In TEXTAL, we first reset too high and too low density values, because they most likely are due to solvent and other forms of noise, rather than protein. Thus, two threshold density values θ_1 and θ_2 are determined for a given map, such that the highest 20% of density points lie above θ_1 and the lowest 20% of density points lie below θ_2. Given a point with density ρ, the scaled density ρ' is given by $\rho' = [\rho - (\theta_1 + \theta_2)/2]/[(\theta_1 - \theta_2)/2]$. The same scaling is done for the input map as well.

Not surprisingly, this led to more accurate comparison between density patterns, and hence better modeling of side chains. Figure 7.3 shows the percentage of amino acids for which a structurally similar amino acid is retrieved by TEXTAL, using a non-scaled and a scaled database.

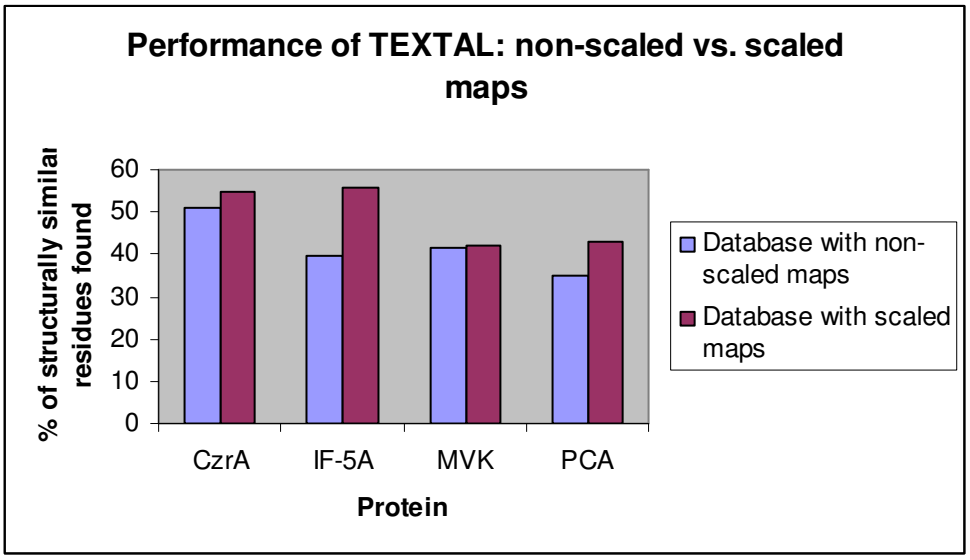

Fig. 7.3. Quality of models built by TEXTAL with databases of maps that are scaled and those that are not scaled. Scaling of the database maps (and the input map) leads to significant improvement in the performance of TEXTAL.

7.3. Distribution of Amino Acids

Since TEXTAL's database constitutes of artificially created maps from manually determined and refined proteins, the distribution of the twenty amino acids in the database will more or less reflect the distribution that is found in naturally occurring proteins (Figure 7.4).

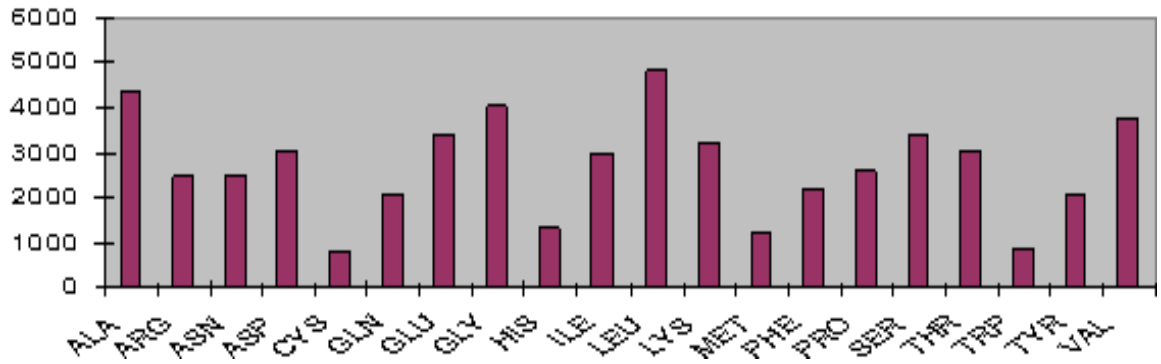

Fig. 7.4. Distribution of amino acids in TEXTAL's database.

It can be argued that the database should be biased so that it keeps only limited number of the possible rotamers of each residue. For instance, since there is one rotamer of *alanine*, having multiple instances of alanines is redundant. With this approach, a smaller database will be needed. There are several rotamer libraries that are publicly available (Lovell *et al*., 2000; Dunbrack, 2002), and used in crystallographic protein modeling programs (McRee, 1999b; Perrakis *et al*., 1999).

Nonetheless, TEXTAL uses a very large database that covers all rotamers that occur in real structures – the rationale behind this is the fact that TEXTAL is designed for low resolution, noisy maps, and variations due to noise may not be adequately covered by a small library of rotamers. Another form of noise is the error in finding the coordinates of Cα atoms. TEXTAL uses a neural network approach (Ioerger and Sacchettini, 2002) to find Cα coordinates, and they have limited accuracy (usually, a root mean square error of

less than 1 Å), which can easily mislead pattern matching for side chains. Also, TEXTAL attempts to get only structurally similar residues in a preliminary phase of side chain placement. Subsequent sequence alignment and real-space refinement steps correct the amino acid identities and fit them better into the density. Furthermore, the two-phase approach used in TEXTAL makes the use of a large database with a lot of redundancy affordable.

7.4. Definition of a True Match for a Side Chain

Ideally, given a electron density region around a Cα, TEXTAL should retrieve the right rotamer of the right amino acid that the region contains. But in practice, the situation is more complicated. As mentioned earlier, one of the problems is that the Cα coordinates determined from electron density map may not be accurate. They may be off the correct Cα by a certain distance. More importantly, there may be wrong insertions and deletions of Cα atoms, owing to noise like broken or weak density. Furthermore, we try to find a match for a spherical sphere of density, where we look at the density patterns at different radii (3, 4, 5, and 6 Å). Amino acids vary in shape and size; at a smaller radius (like 3 Å), we may not capture enough information about the density corresponding to a (large) residue. On the other hand, with a large radius (6 Å), we may have noise from density patterns for neighboring residues.

These complications make prediction of amino acid identity difficult. We used a classifier based on support vector machines (Vapnik, 1998) to predict the exact residue identity, using the 76 features described earlier. We trained 20 two-class classifiers to predict whether a region contains a given amino acid or not i.e. one classifier for each amino acid. The classifiers were trained with 1000 positive and 1000 negative examples for each amino acid. We used the Gist SVM classifier (Pavlidis *et al*., 2004) for these experiments.

As expected, the SVM classifier performed reasonably well with precise Cα coordinates, but the performance degraded with inaccurate Cα predictions. We analyzed how difficult it is to predict the identities of various amino acids.

Figures 7.5 and 7.6 show results obtained when we use an SVM classifier that, given a pattern for an amino acid (correctly centered around its true Cα) and an amino acid identity X, it predicts whether the pattern is for X or not. Figure 7.5 gives the sensitivity of prediction in descending order for all amino acids, and Figure 7.6 gives the specificity of prediction in descending order. The sensitivity and specificity are related to Receiver Operating Characteristic (ROC) analysis. If TP, TN, FP, FN are rates for true positives, true negatives, false positives, and false negatives respectively, then sensitivity [TP/(TP+FN)] can be thought of as the likelihood of spotting a positive case when presented with one. Specificity [TN/(TN+FP)] can be thought of as the likelihood of spotting a negative case when presented with one. We observe that there is significant consistency in the ranks of the amino acids in terms of sensitivity and specificity. We also observe that amino acids that are small and spherical in shape tend to be most easily recognizable (e.g. *alanine*, *glycine*, and *proline*). *Proline* is particular distinct with its ring that contains a nitrogen atom. On the other hand, "long" amino acids (such as *arginine*, *lysine*, and *leucine*) tend to be least recognizable, probably because we look at features covering spherical regions of density. The presence of a large sulfur atom in a *cysteine* and *methionine* tends to make them somewhat more distinct. Amino acids with benzene rings can also be recognized with relative ease – it is interesting that the three aromatic amino acids (*tyrosine*, *tryptophan*, and *phenylalaline*) are ranked next to each other in the middle of the sensitivity chart.

We note that there is significant (negative) correlation between number of rotamers for an amino acid and the sensitivity (as well as specificity) of the accuracy with which it can be recognized. Table 7.1 provides the number of rotamers for each amino acid (Lovell *et al*., 2000), and Figures 7.7 and 7.8 are scatter plots of number of rotamers versus sensitivity and specificity respectively. The linear coefficient correlation for sensitivity vs. number of rotamers is -0.55, and for specificity vs. number of rotamers, it is -0.53. This clearly shows that it is easier to recognize amino acids with fewer rotamers from the features that we use. This result is intuitive since more rotamers effectively

makes pattern recognition more confusing, since one amino acid can be represented by more feature vectors, even if we assume that there is no noise in the features.

Note that the above SVM-based predictions assume accurate Cα positions. With real, noisy maps, the problem becomes much harder. Given that TEXTAL places amino acids for medium-low resolution maps (and therefore imprecise Cα coordinates), we adopt a strategy of finding structurally similar residues in an as robust way as possible (i.e. insensitive to orientation, translation, and noise). Once structurally similar residues are found, we improve on the accuracy in terms of residue identity by aligning with the true sequence, substituting some residues (based on a substitution matrix that captures the types of amino acid prediction errors that TEXTAL makes), and doing a real-space refinement of the model.

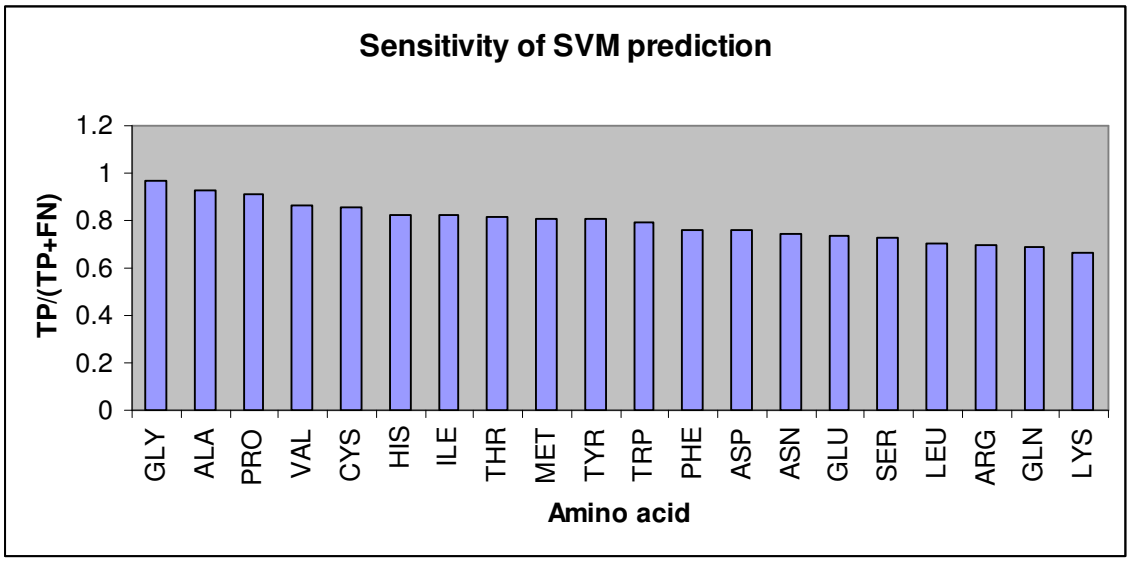

Fig. 7.5. Sensitivity of prediction of amino acid type using an SVM classifier.

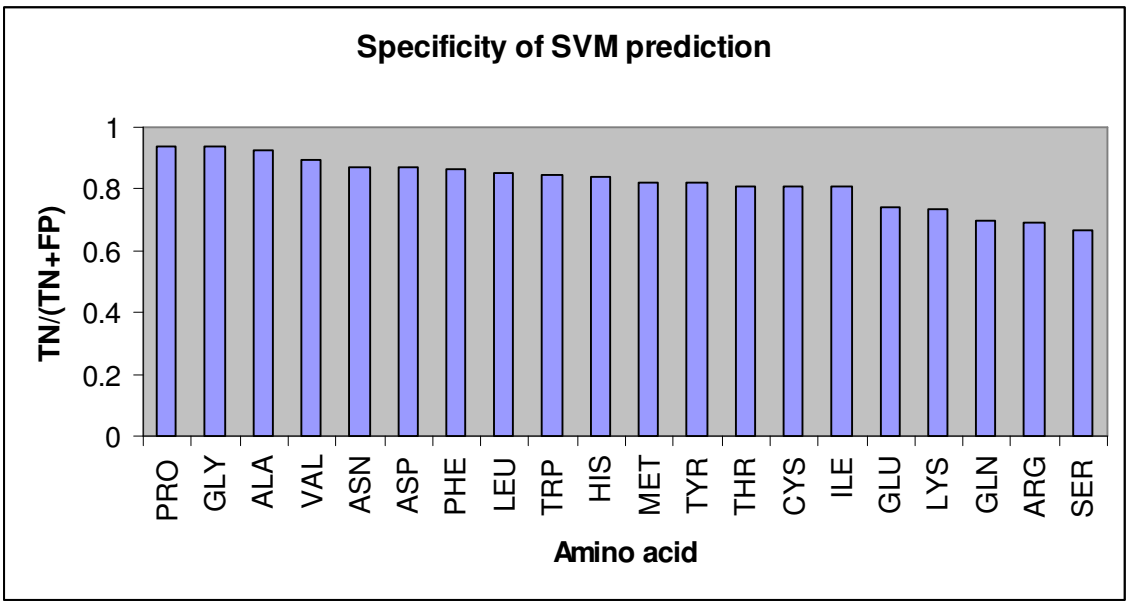

Fig. 7.6. Specificity of prediction of amino acid type using an SVM classifier.

Table 7.1. Number of rotamers for each amino acid

Amino acid	Number of rotamers
ALA	1
ARG	34
ASN	7
ASP	5
CYS	3
GLN	9
GLU	8
GLY	1
HIS	8
ILE	7
LEU	5
LYS	27
MET	13
PHE	4
PRO	3
SER	3
THR	3
TRP	7
TYR	4
VAL	3

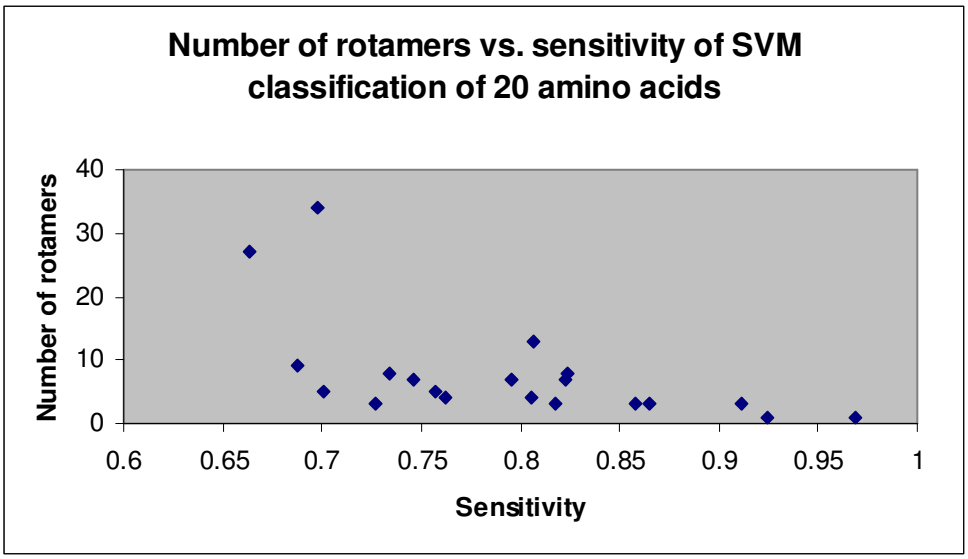

Fig. 7.7. Scatter plot of the number of rotamers vs. the sensitivity of pattern recognition for the 20 amino acids. The linear correlation coefficient is -0.55.

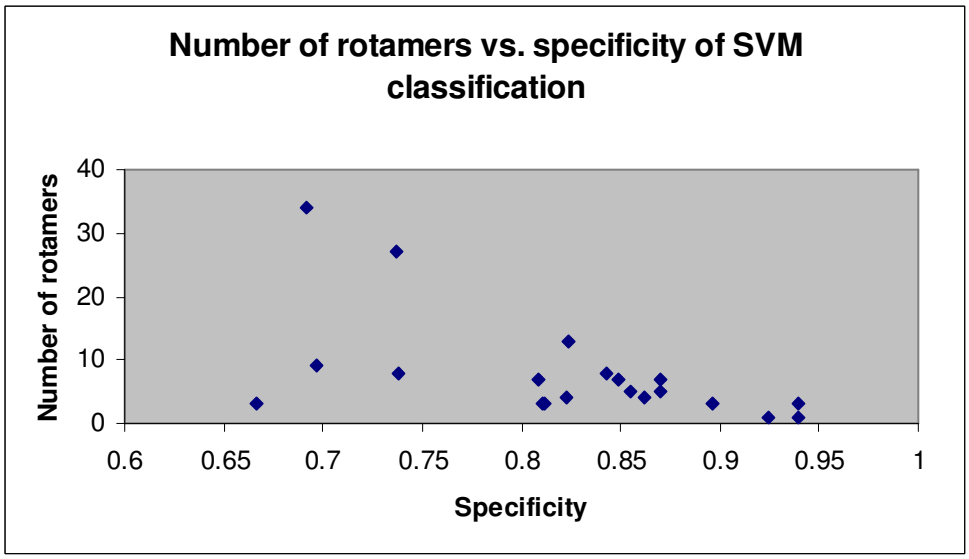

Fig. 7.8. Scatter plot of the number of rotamers vs. the specificity of pattern recognition for the 20 amino acids. The linear correlation coefficient is -0.53.

7.5. Density Correlation and Feature-Based Distance

If finding the true amino acid identity is difficult, we need another metric to find the (structural) similarity between two regions of density. We use a measure that superposes two density regions (one from the unsolved protein, and the other from the map library), and evaluate the *density correlation* to find out how similar they are. This density correlation coefficient (CC) is expensive to compute, and it is impractical to compare each region of the input to every region in our database. That is why we filter a small number of cases from the database using feature-based distance measure. In this section, we analyze the relationship with these two metrics. In Figure 7.9, we show the distribution of CC for 1000 pairs of regions randomly chosen from the database. Figure 7.10 shows the typical distribution of CC for the top 400 cases filtered from the database, given a random query region. We can see the difference in the distribution, which shows the ability of the feature-based metric to filter regions with high density correlation values.

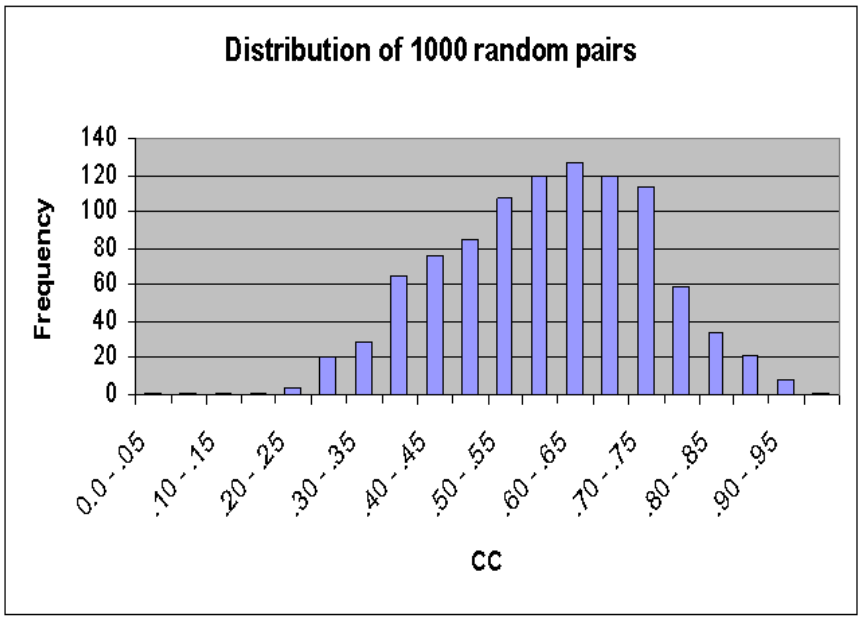

Fig. 7.9. Distribution of density correlation coefficient (CC) for 1000 random pairs from the TEXTAL database.

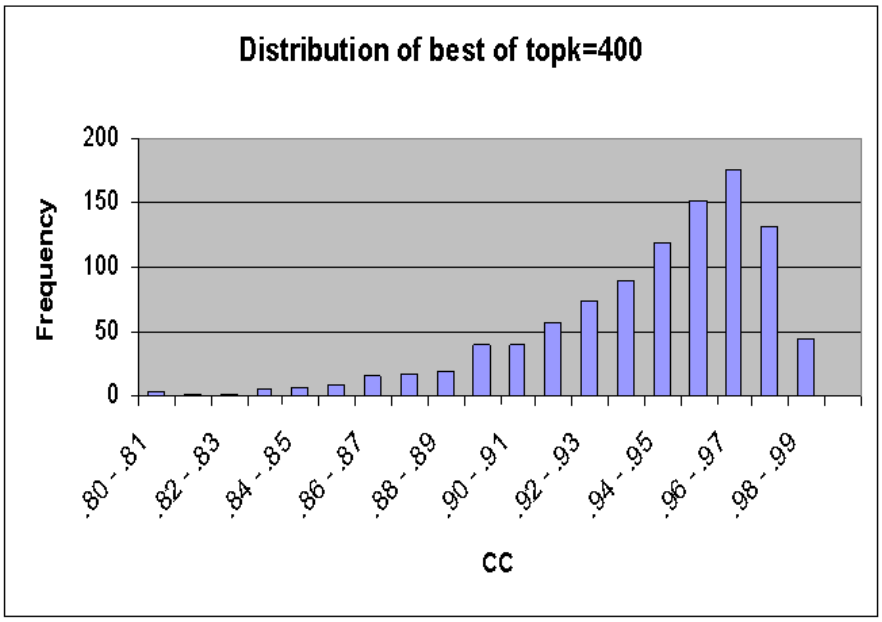

Fig. 7.10. Distribution of density correlation coefficient (CC) between a random region and the top 400 cases filtered (using a feature-based distance) from the TEXTAL database.

One important question is what *k* should be set to. Our aim is to set *k* the lowest possible (so that as few density correlations values as possible have to be computed), without sacrificing the quality of solutions obtained. Figures 7.11a and 7.11b show how the average density correlation coefficient (CC) of all side chains for two test proteins solved by TEXTAL (*CzrA* and *IF-5A* respectively) varies with various top *k* cases filtered. We observe that the gain in CC is significant till the top 2,000 to 3,000 are pre-selected, which is close to theoretically determined results (Gopal *et al*., 2004a)

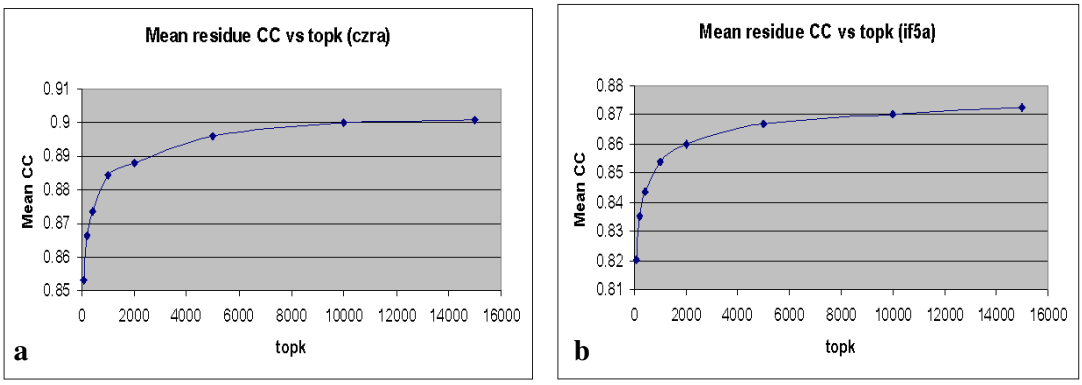

Fig. 7.11. The mean density correlation coefficient (CC) increases with top *k* cases pre-selected. CC reaches a plateau with *k* at about 3000 for both *CzrA* (Fig. 7.11(a))and *IF-5A* (Fig. 7.11(b)).

Table 7.2 analyzes the choice of *k* from a slightly different perspective. For one representative of each amino acid, we first determine its truly best match (highest density correlation) found in the top $k = 400$, and $k = 100$. We then compare this best match found in the top *k* to all other instances in the database, and rank it based on density correlation. We call this rank *absolute rank*. Table 7.2 shows the absolute ranks i.e. the rank of the best match found in the top *k* based on density correlation, relative to all instances in the database. It can be observed that with $k = 400$, we find a match ranked fairly close to the absolute best one. With $k = 100$, the match found are often poorly ranked ones.

Table 7.2. Absolute rank of matches filtered by Euclidean distance

Amino acid	*Absolute rank* of best match found in top k	
	k = 400	k = 100
ALA	34	150
ARG	87	768
ASN	9	12
ASP	8	8
CYS	4	4
GLN	7	7
GLU	12	12
GLY	10	17
HIS	32	43
ILE	36	354
LEU	2	22
LYS	1	1
MET	5	5
PHE	16	20
PRO	27	354
SER	6	6
THR	4	32
TRP	2	2
TYR	1	30
VAL	5	8

For one representative instance of each amino acid, we first determine its truly best match (highest density correlation) found in the top k = 400, and k = 100 (pre-selected by Euclidean distance). We then compare this best match found in the top k to all other instances in the database, and rank it based on density correlation. This rank is referred to as *absolute rank*. With k = 400, significantly better instances (lower rank) are retrieved as compared to k = 100.

7.6. SLIDER's Impact on Retrieval for Real Maps

In this section, we evaluate the contribution of SLIDER in solving real maps. We run TEXTAL (including the sequence alignment and real-space refinement steps) on the four test cases mentioned in section 7.1, using Euclidean distance with uniform weights as well as with weights determined by SLIDER. The distance metric is invoked in the LOOKUP system, which performs the database search. The other components (CAPRA and POST-PROCESSING) do not depend on the distance metric. As mentioned earlier, the database we use in TEXTAL is made up of ideal (back-transformed) maps. The training set that SLIDER uses to determine the weights is also generated from ideal maps. Experimental maps are noisy and have high variance in terms of quality and resolution; thus we achieve better case retrieval and tuning of weights if ideal maps are used for training and in the database.

Table 7.3 shows the percentage of amino acids that were correctly determined by TEXTAL on our test cases, using the weighted and non-weighted Euclidean measure, setting k to 100. Given an unsolved query region, the top 100 potential matches are retrieved from a database (of about 50,000 regions) by the Euclidean distance metric, and the final selection is done by choosing the one with the highest density correlation with the query region. We can observe that feature weighting by SLIDER seems to contribute little to the performance of TEXTAL for the first two (smaller) proteins (*CzrA* and *IF-5A*). Nonetheless, in the last two cases (*MVK* and *PCA*), the percentage of amino acids correctly identified increases significantly when SLIDER weights are used. The wide variation in the performance of TEXTAL (and the contribution of SLIDER) on different real maps can most likely be attributed to differences in qualities of the maps. The accuracy with which CAPRA places the Cα atoms is also influenced by the quality of the map, and the performance of LOOKUP is sensitive to that of CAPRA. Thus the "interpretability" of maps varies widely, depending on resolution and degree of phase error.

We emphasize that the improvement in accuracy in predicting residue identities is not necessarily the direct contribution of better feature weighting – its contribution is often

indirect, in the sense that by helping in the retrieval of slightly better matches, SLIDER helps the sequence alignment routine (Romo *et al*., 2006) recognize protein fragments, and make more appropriate corrections in amino acid identities.

Table 7.3. Performance of TEXTAL, with and without SLIDER weights

Protein	No of residues	% of residues correctly identified by TEXTAL	
		Euclidean distance with uniform weights	Euclidean distance with SLIDER weights
CzrA	96	98.9	95.6
IF-5A	137	78.1	79.7
MVK	317	25.4	54.7
PCA	262	23.9	57.7

SLIDER makes little contribution in model-building of *CzrA* and *IF-5A*, but contributes significantly in the cases of *MVK* and *PCA*, which are larger proteins.

7.7. Interpretation of Feature Weights

In this previous chapter we discussed feature weighting from the perspective of efficiency of case retrieval. But do the features determined as relevant by SLIDER meet the expectations of a crystallographer? In this section, we analyze feature relevance from the perspective of the domain.

The descriptions of the features are provided in section 3.12. As pointed out earlier, although SLIDER may select/weight features differently in different runs, the different features selected may actually be quite similar e.g. closely related features, like standard deviation and skewness. Also, there may be more redundancy in terms of the same feature at close radius values e.g. average density at 3 and 4 Å. Thus, the features may not be relevant or irrelevant in absolute terms, but they might seem useful or useless in the context of other features selected or rejected. Table 6.1 lists the features that were found irrelevant for all the three metrics analyzed (L_1, L_2, and L_3). So these features can

be considered to be largely irrelevant. Table 6.2 lists the features that were found relevant for *any* of the three metrics. Not all of them are highly relevant; the weight also matters – the differences in feature relevance based on weights are captured in Figure 6.2.

We first discuss the impact of radius on relevance (irrespective of the feature identities), and then we analyze the features individually. Table 7.4 shows the sum of weights (determined by SLIDER) for each radius, irrespective of the nature of the feature. We can observe significant similarity of weights for the three metrics that use the weights (weighted Manhattan, Euclidean, and L_3). Furthermore, we can note that the total weights for radii 3 Å and 5 Å are the highest, and comparable to each other, and the total weights for radius 6 Å is significantly lower. The 3D spherical patterns are expected to cover amino acids of various shapes and sizes, which justifies the choice of feature values at different radii. A 3 Å radius sphere centered on a Cα atom is expected to contain significant information about the side chain; but this information may not be adequate to recognize the side chain, especially for large amino acids that may not be totally encapsulated in the 3 Å sphere. But at 6 Å, we face the problem of having noise due to density of neighboring residues or long-range contacts, and hence we expect their relevance to be lower; this trend is captured by our weight optimization algorithm.

Table 7.4. Feature weights at different radii

Radius in Å ($1 Å = 10^{-10}m$)	**Sum of weights for three Minkowski metrics**		
	Manhattan	**Euclidean**	**L_3**
3	.36	.35	.32
4	.21	.23	.15
5	.34	.35	.43
6	.09	.07	.10

Sum of all feature weights determined by SLIDER at four different radii. There are 19 features for each radius.

We now look at the various features individually, and discuss the significance of the observed results:

- Mean density: it is irrelevant at small radii (3 and 4 Å), but relevant at large radii (5 and 6 Å). This seems intuitive. The average densities at a small radius would tend to be the same for all amino acids – they become distinguishable only with larger radius.
- Standard deviation in density: it is irrelevant for almost all radii (it has a very small weight at 3 Å only). This is also expected; there seems to be little compelling argument for the density value to have different variances for different residues, especially when the density has been scaled.
- Skewness: The observations are the same as for the standard deviation i.e. irrelevant at almost all radii. The same explanation probably holds too.
- Kurtosis: This feature has a small weight at three different radii. This is somewhat surprising, especially given the observation about standard deviation and skewness. The kurtosis is a measure of "peakedness". The peakedness is expected to be large at larger radii (and this is actually observed) – it is arguably capturing some differences among residues, or it maybe a case of data overfitting.
- Moments of inertia: In general, the magnitudes of the three moments of inertia are largely found irrelevant by SLIDER, especially the secondary moment. Nonetheless the ratios of the different pairs of moments of inertia are found relevant. Recall that the primary moment of inertia is along the path with the highest distribution of density – the other two moments of inertia are orthogonal to the primary moment (and to each other). The primary moment would probably capture some information about long residues (like *arginine* and *leucine*). But the magnitude of moments along the two other directions are not expected to contain much information, since they are orthogonal to the primary moment. This is consistent with the empirical results. But, why are the ratios generally found relevant? The ratio arguably capture how stretched out it is in one particular direction. For example, like *arginine*, *histidine* also stretches along one direction, but the latter also has high density along

other directions (due to its ring). Thus the primary moment may not distinguish them apart, whereas the ratio of moments will.

- The distance from the center of a region to its center of mass: as expected, this is a highly relevant feature (large weights), except at a radius of 6 Å. This feature tries to capture how symmetric or balanced the region is, and this is expected to vary significantly across amino acids. But at 6 Å, noise from neighboring features is expected to really distort this feature value.
- Features based on "spokes": by and large, these features are found to be highly relevant, especially the sum of spoke angles and spoke triangle area (the latter two are relevant for all radii and have relatively large weights). This is not surprising, since this feature set is the only one that tries to capture information based on the typical shape of amino acids (three spokes emanating from the Cα). Three spokes are calculated along arbitrary directions of high density – this is probably why these features are more relevant compared to the moments of inertia, which are computed at strictly orthogonal directions.

CHAPTER VIII
CONCLUSION

8.1. Feature Weighting and SLIDER

In this work, we discuss the importance of feature selection and weighting in machine learning and pattern recognition. We describe the SLIDER algorithm, and show that the feature weights it determines lead to an improvement in weighted distance metrics, like Euclidean or Manhattan. This in turn results in an increase in the number of true matches occurring in the top ranked cases retrieved from a large database. Thus, the probability of retrieving a match from the database is increased.

We propose a two-phase strategy for efficient case retrieval, where we approximate an objective, expensive similarity metric with a fast, feature-based similarity measure, and use the latter as a filter of probably good matches, based on *k*-nearest neighbor learning. With this approach, case-based reasoning systems can afford large databases as well as improve on time performance.

SLIDER is a filter method that avoids searching a large space of possible weight vectors. Instead, the evaluation is performed at weight values that matter i.e. at specific "crossover" weights, where there is a change in accuracy of ranking of matches relative to mismatches. Furthermore, locating these weight values can be done efficiently, since it involves solving linear equations applicable to many metrics (like the Euclidean distance). The benefits of restricting the number of weights searched and used for nearest neighbor classification are emphasized in (Kohavi *et al*., 1997); they also argue that there are probably no benefits in using weights beyond two possible values (0 and 1). Nonetheless, SILDER does manage to compute finer weight values that improve case matching and retrieval.

SLIDER was used to optimize weights for three different Minkowski distance metrics, and proved to be successful in improving pattern matching and retrieval for each of the three metrics, in the context of the case-based reasoning and nearest neighbor strategies to efficiently retrieve matches. The weights determined by SLIDER were largely similar

for the various metrics. Nonetheless, the slight differences were significant in capturing the sensitivity of relevance to the distance metric being used. We argue that the relevance of features in describing a pattern is not absolute; it depends on how the features are used to determine similarity, especially since similarity itself is often a fuzzy concept, with multiple ways of determining it.

We also present a theoretical framework based on probability to justify the heuristic SLIDER used to evaluate weights. We argued that by using a training set to minimize the mean rank of true matches relative to mismatches, we are causing a separation between the distribution of Euclidean distances for matches and the distribution of Euclidean distances for mismatches. This separation is a measure of how well the distance metric is able to distinguish matches from mismatches. We analytically showed that if the mean rank of matches is decreased as a consequence of the weight update by SLIDER, this implies a separation between the Euclidean distance distributions for matches and mismatches, and hence a higher probability of retrieving a true match. This provides a theoretical justification to SLIDER's strategy of minimizing ranking of matches by adjusting weights through a search of a space of candidate weight vectors. Our analysis also sheds light on the statistical relationship between objective (numerical) evaluations of a set of instances and the ranking of the instances based on the evaluations. This is potentially useful in many other applications, such as ranking of web pages, document/image retrieval, proteomics, and other applications that involve large and rich databases.

We also discuss the significant contribution of SLIDER in making case retrieval more efficient and effective in the TEXTAL system (for automated electron density map interpretation in protein crystallography). We made an empirical comparison between SLIDER and other feature selection and weighting algorithms, including a method based on linear programming that is closely related to SLIDER.

The feature weighting and the related case matching and retrieval methods that we propose have been motivated by the exigencies of the electron density map interpretation problem. Nonetheless, the techniques employed are general and potentially useful in

other domains, especially those with high-dimensional and noisy data, expensive case matching, and large databases.

8.2. Application to Protein Crystallography

We also discuss the contribution of the methods proposed (for efficient case retrieval and feature weighting) to the challenging problem of automated electron density map interpretation in protein crystallography. The techniques developed have been used in TEXTAL, a system that takes a real-space pattern recognition approach to solve electron density maps. TEXTAL is a case-based reasoning system that exploits information on existing density maps and their corresponding 3D structures to solve new maps.

We relate the proposed methods to previous work on database and feature-based approaches in structural bioinformatics. We discuss various key practical issues in TEXTAL, which may be useful in other bioinformatics applications. What should be the size and composition of the database? We empirically determined a "saturated" database size, beyond which there is no marked improvement. We argue that large databases are necessary for such complex and noisy domains, despite the fact the number of possible rotamers is limited. This is where appropriate feature weighting is instrumental, since it makes the database retrieval efficient.

We also argue that the database should be made of ideal (noise-free) electron density maps, as opposed to real maps (from experimental data). This may be counter-intuitive, since ideal maps may not capture information generally found in real maps. But with ideal maps, pattern recognition is found to be more effective. What is more important is proper "feature engineering" – by scaling the maps properly (to make comparisons meaningful) and normalizing the features appropriately, recognition of patterns becomes much easier.

Another issue we dealt with is the definition of a true match for side chains. How do we define a true match? Since many amino acids are structurally similar, and there are errors in the Cα coordinates as well noise in the input map, we found it better to adopt a more relaxed version of identity – we try to find structurally similar residues rather than

the "obvious" similarity metric i.e. amino acid identity. (We subsequently correct amino acid identity using sequence alignment.)

We show the difficulties of recognizing various residues based on identity, even with state-of-the-art classification algorithms like support vector machines. We also look at the relationships among various similarity metrics: amino acid identity, density correlation, and feature-based metrics. This provides convincing arguments to our approach, especially if the parameters are set properly – such as an appropriate k (the number of cases to be efficiently filtered from the database).

In recognizing density patterns in a large map, we look at local, spherical regions. One important issue is the need to look at density regions at different radii – because of noise, different shapes and size of residues, and the imperfections in Cα predictions. The usefulness of this approach was clear when weights were analyzed at different radii. Also, we interpret the weights of the features determined by SLIDER based on domain knowledge, and saw that they are quite consistent with what is expected.

8.3. Limitations and Future Work

There is considerable scope to improve and extend this work, and address its various limitations. In particular, we plan to investigate the following:

- SLIDER can be extended to optimize more than one weight at a time, based on the same geometric principles in higher dimensions. It can be shown that each ‹*instance, match, mismatch*› 3-tuple represents a line in a two-dimensional (2D) space of two features, with one side representing an improvement in accuracy, and the other side a loss in accuracy. If many such 3-tuples are considered, the problem can be reduced to finding an optimal 2D region (a convex polygon, actually) that represents optimum pairs of weights for these two features. This will make the algorithm less greedy. The algorithm can be further extended to even higher dimensions.
- SLIDER is currently limited to distance metrics for which crossovers weights can be calculated by solving simple linear equations, like for Minkowski metrics. This may not be possible for other similarity or distance measures, like those based on

probabilistic and statistical methods (Aksoy and Haralick, 2001; Kontkanen *et al*., 1997). We plan to investigate approaches where crossover points for such metrics can be efficiently determined (by binary search over the space of weights, for instance).

- One aspect that necessitates closer scrutiny is the definition of match and mismatch to assess if the updated weights improve accuracy. We use a simple strategy where two patterns are said to match/mismatch if their density correlation in above/below a threshold. We observed that the final set weights returned by SLIDER is sensitive to this threshold. What would be an appropriate threshold, and how can it be determined? Or is there a better way of assessing similarity in this context? Should we use "perfect" matches/mismatches in our training set, or do we need to allow for near-matches/near-mismatches as well, which may enable us capture finer information that is required to confidently say how different two instances are?
- In this work, we assume that feature weights are globally optimum, independent of the case. This may not be true all the time. If some features are deemed more important than others for comparing some cases, it does not necessarily imply that the same features will be equally important in comparing other cases (Domingos, 1997; Howe and Cardie, 1997; Greiner *et al*., 1997).
- More generally, we are looking into other strategies to weight features, including methods based on Single Value Decomposition (SVD) and Principal Component Analysis (PCA).
- The ranking-based heuristic we use to evaluate weights in SLIDER necessitates an exhaustive search of the database to find truly similar and different cases for a training set. This may be too expensive. Also, in some applications, the number of positive and negative examples may be scarce. In our empirical investigation, given a training instance, we search for only one match and a set of mismatches (say 100) to measure the accuracy of ranking; this is only an approximation of the theoretical version of the heuristic.

- The assumption that two cases can be either similar or different may, in many domains, be an over-simplification. Similarity and difference can be a matter of degree. In fact, we observed that optimal weights determined by SLIDER in the protein crystallography domain vary with the definition of true similarity between electron density patterns.
- Finally, we recognize that a global value of *k* has its limitations. For some cases that have a large number of good matches, a relatively low *k* should catch a match with high probability, whereas more difficult cases may require comparison with more potential matches for effective retrieval. Context-sensitive determination of *k* is yet another worthwhile future work.

REFERENCES

Ableson,A. and Glasgow,J.I. (1999) Crystallographic threading. *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology,* pp. 2-9.

Adams,P.D., Gopal,K., Grosse-Kunstleve,R.W., Hung,L.W., Ioerger,T.R., McCoy,A.J., Moriarty,N.W., Pai,R.K., Read,R.J., Romo,T.D., Sacchettini,J.C., Sauter,N.K., Storoni,L.C. and Terwilliger,T.C. (2004) Recent developments in the PHENIX software for automated crystallographic structure determination. *Journal of Synchrotron Radiation,* **11**, 53-55.

Adams,P.D., Grosse-Kunstleve,R.W., Hung,L.W., Ioerger,T.R., McCoy,A.J., Moriarty, N.W., Read,R.J., Sacchettini,J.C. and Terwilliger,T.C. (2002) PHENIX: building new software for automated crystallographic structure determination. *Acta Crystallographica*, **D58**, 1948-1954.

Aha,D.W. (1990) A study of instance-based algorithms for supervised learning tasks: mathematical, empirical, and psychological observations. Ph.D. Dissertation, University of California, Irvine.

Aha,D.W. (1997) Editorial. *Artificial Intelligence Review*, **11**(1-5), 1-6.

Aha,D.W. (1998) Feature weighting for lazy learning algorithms. In Liu,H. and Motoda,H. (eds), *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer, Boston, MA.

Aksoy,S. and Haralick,R.M. (2001) Probabilistic vs. geometric similarity measures for image retrieval. *Proceedings of Conference on Computer Vision and Pattern Recognition*, pp. 112-128.

Almuallim,H. and Dietterich,T.G. (1994) Learning Boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, **69**, 279-305.

Alphey,M.S., Leonard,G.A., Gourley,D.G., Tetaud,E., Fairlamb,A.H., Hunter,W.N. (1999) The high resolution crystal structure of recombinant *Crithidia Fasciculata Tryparedoxin-I. Journal of Biological Chemistry*, **274**, 25613-25622.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W., Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, **25**(17), 3389-3402.

Anfinsen,C.B., Redfield,R.R., Choate,W.L., Page,J. and Carroll,W.R. (1954) Studies on the gross structure, cross-linkages, and terminal sequences in ribonuclease. *Journal of Biological Chemistry*, **207**(1), 201-210.

Baker,D. and Sali,A. (2001) Protein structure prediction and structural genomics. *Science*, **294**, 93-96.

Baluja,S. and Pomerleau,D. (1997) Dynamic relevance: vision-based focus of attention using artificial neural networks. *Artificial Intelligence*, **97**, 381-395.

Bekkerman,R., El-Yaniv,R., Tishby,N. and Winter,Y. (2003) Distributional word clusters vs. words for categorization. *Journal of Machine Learning Research*, **3**, 1183-1208.

Bentley,J.L. (1975) Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 509-517.

Berens,M., Liu,H. and Yu,Lei. (2005) Fostering biological relevance in feature selection for microarray data. *IEEE Intelligent Systems*, **20**(6), 71-73.

Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat, T.N., Weissig,H., Shindyalov, I.N. and Bourne,P.E. (1992) The Protein Data Bank. *Nucleic Acids Research*, **28**, 235-242.

Bowie,J.U., Luthy,R. and Eisenberg,D. (1991) A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, **253**(5016), 164-70.

Blum,A.L. and Langley,P. (1997) Selection of relevant features and examples in machine learning. *Artificial Intelligence*, **97**, 245-271.

Blum,A.L. and Rivest,R.L. (1992) Training a 3-node neural networks in NP-complete. *Neural Networks*, **5**, 117-127.

Branden,C.I. and Jones,T.A. (1990) Between objectivity and subjectivity. *Nature*, **343**, 687-689.

Branting,L.K. and Aha,D.W. (1995) Stratified case-based reasoning: reusing hierarchical problem solving episodes. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 384-390.

Burges,C., Shaked,T., Renshaw,E., Lazier,A., Deeds,M., Hamilton,N. and Hullender,G. (2005) Learning to rank using gradient descent. *Proceedings of the Twenty-second International Conference on Machine Learning*, pp. 89-96.

Burley,S.K., Almo,S.C., Bonanno,J.B., Capel,M., Chance,M.R., Gaasterland,T., Lin,D., Sali,A., Studier,W. and Swaminathian,S. (1999) Structural genomics: beyond the human genome project, *Nature Genetics*, **232**, 151-157.

Ceberio,M. and Kreinovich,V. (2005) Towards an optimal approach to soft constraint problems. *Proceedings of the Seventeenth World Congress Scientific Computation, Applied Mathematics and Simulation* (http://www.cs.utep.edu/vladik/2004/tr04-32b.pdf).

Chandonia,J.M. and Brenner,S.E. (2006) The impact of structural genomics: expectations and outcomes, *Science*, **311**, 347-51.

Cohen,W.W., Schapire,R.E. and Singer,Y. (1997) Learning to order things. *Advances in Neural Processing Systems*, **10**, 451-457.

Collins,F.S., Green,E.D., Guttmacher,A.E. and Guyer,M.S. (2003) A Vision for the Future of Genomics Research: a blueprint for the genomic era. *Nature*, **422**, 835-847.

Cover,T.M. and Hart,P.E. (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **13**(1), 21-27.

Cowtan,K. (1998) Modified phased translation functions and their application to molecular fragment location. *Acta Crystallographica*, **D54**, 750-756.

Cowtan,K. (2006) The "Buccaneer" protein model building software, *CCP4 Newsletter* **44**.

Daelemans,W. Gillis,S. and Durieux,G. (1994) The acquisition of stress: a data-oriented approach. *Computational Linguistics*, **20**(3), 421-451.

Das,S. (2001) Filters, wrappers and a boosting-based hybrid for feature selection. *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 74-81.

Devijver,P.A. and Kittler,J. (1982) *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, London, UK.

Diamond,R. (1971) A real-space refinement procedure for proteins. *Acta Crystallographica*, **A27**, 436-452.

Diller,D.J., Redinbo,M.R., Pohl,E. and Hol,W.G.J. (1999) A database method for automated electron density map interpretation in protein crystallography. *Proteins: Structure, Function, and Genetics*, **36**, 526-541.

Dimaio,F., Shavlik,J. and Phillips,G.N. (2006) A probabilistic approach to protein backbone tracing in electron density maps. *Bioinformatics*, **22**(14), e81-89.

Domingos,P. (1997) Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, **11**, 227-253.

Duda,R.O., Hart,P.E. and Stork,D.G. (2001) *Pattern Classification*. John Wiley and Sons Inc., New York, NY.

Dunbrack,R.L. (2002) Rotamer libraries in the 21st century. *Structural Biology*, **12**(4), 431-40.

Dwork,C., Kumar,R., Naor,M. and Sivakumar,D. (2001) Rank aggregation methods for the Web. *Proceedings of the Tenth International Conference on World Wide Web*, pp. 613-622.

Dy,J.G. and Brodley,C.E. (2004) Feature selection for unsupervised learning. *Journal of Machine Learning Research*, **5**, 845-889.

Eicken,C., Pennella,M.A., Chen,X., Koshlap,K.M., VanZile,M.L., Sacchettini,J.C. and Giedroc,D.P. (2003) A metal-ligand-mediated intersubunit allosteric switch in related SmtB/ArsR zinc sensor proteins. *Journal of Molecular Biology*, **333**(4), 683-695.

Feigenbaum,E.A., Engelmore,R.S. and Johnson,C.K. (1977) A correlation between crystallographic computing and artificial intelligence research. *Acta Crystallographica*, **A33**, 13-18.

Fix,E. and Hodges,J. (1951) Discriminatory analysis, nonparametric discrimination: consistency properties. *Technical Report 4*, USAF School of Aviation Medicine.

Forbus,K., Gentner,D. and Law,K. (2001) MAC/FAC: a model of similarity-based retrieval. *Cognitive Science*, **19**(2), 141-205.

Forman,G. (2003) An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, **3**, 1289-1305.

Forman,G. (2005) Feature selection: we've barely scratched the surface. *IEEE Intelligent Systems*, **20**(6), 74-76.

Freund,Y., Iyer,R., Schapire,R.E. and Singer,Y. (2003) An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, **4**, 933-969.

Galles,D. and Pearl,J. (1997) Axioms of causal relevance. *Artificial Intelligence*, **97**, 9-43.

Gopal,K., McKee,E.W., Romo,T.D., Pai,R., Smith,J.N., Sacchettini,J.C. and Ioerger,T.R. (2006a). Crystallographic protein model-building on the web. *Bioinformatics*, doi: 10.1093/bioinformatics/btl584.

Gopal,K., Pai,R., Ioerger,T.R., Romo,T.D. and Sacchettini,J.C. (2003) TEXTAL: artificial intelligence techniques for automated protein structure determination. *Proceedings of the Eighth Conference on Innovative Applications of Artificial Intelligence*, pp. 93-100.

Gopal,K., Romo,T.D., McKee,E.W., Childs,K.C., Kanbi,L., Pai,R., Smith,J.N., Sacchettini,J.C. and Ioerger,T.R. (2005a). TEXTAL: automated crystallographic

protein structure determination. *Proceedings of the Innovative Applications of Artificial Intelligence conference*, pp. 1483-1490.

Gopal,K., Romo,T.D., McKee,E.W., Pai,R., Smith,J.N., Sacchettini,J.C. and Ioerger,T.R. (2006b). TEXTAL: crystallographic protein model-building using AI and pattern recognition. *AI Magazine*, **27**(3), 15-24.

Gopal,K., Romo,T.D., Sacchettini,J.C. and Ioerger,T.R. (2004a) Efficient retrieval of electron density patterns for modeling proteins by X-ray crystallography. *Proceedings of the International Conference on Machine Learning and Applications*, pp. 380-387.

Gopal,K, Romo,T.D., Sacchettini,J.C. and Ioerger,T.R. (2004b) Evaluation of geometric & probabilistic measures of similarity to retrieve electron density patterns for protein structure determination. *Proceedings of the International Conference on Artificial Intelligence*, pp. 427-432.

Gopal,K., Romo,T.D., Sacchettini,J.C. and Ioerger,T.R. (2004c). Weighting features to recognize 3D patterns of electron density in X-ray protein crystallography. *Proceedings of the Computational Systems Bioinformatics*, pp. 255-265.

Gopal,K., Romo,T.D., Sacchettini,J.C. and Ioerger,T.R. (2005b) Determining relevant features to recognize electron density patterns in X-ray protein crystallography. *Journal of Bioinformatics & Computational Biology*, **3**(3), 645-676.

Greer,J. (1985) Computer skeletonization and automatic electron density map analysis. *Methods in Enzymology*, **115**, 206-224.

Greiner,R., Grove, A.J. and Kogan,A. (1997) Knowing what doesn't matter: exploiting the omission of irrelevant data. *Artificial Intelligence*, **97**, 345-380.

Guyon,I. and Elisseeff,A. (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157-1182.

Hammond,K.J. (1989) Case-Based Planning: Viewing Planning as a Memory Task. Academic Press, Boston, MA.

Hinton,G.E. (1989) Connectionist learning procedures. *Artificial Intelligence*, **40**, 185-234.

Hobohm,U., Scharf,M., Schneider,R. and Sander,C. (1992) Selection of a representative set of structures from the Brookhaven Protein Data Bank. *Protein Science*, **1**, 409-417.

Holton,T.R., Christopher,J.A., Ioerger,T.R. and Sacchettini,J.C. (2002) Determining protein structure from electron density maps using pattern matching. *Acta Crystallographica*, **D46**, 722-734.

Howe,N. and Cardie,C. (1997) Examining locally varying weights for nearest neighbor algorithms. *Proceedings of the Second International Conference on Case-Based Reasoning*, pp. 455-466.

Huang,C.C., Smith,C.V., Glickman,M.S., Jacobs,W.R Jr. and Sacchettini, J.C. (2002) Crystal structures of mycolic acid cyclopropane synthases from mycobacterium tuberculosis. *Journal of Biological Chemistry*, **277**, 11559-11569.

Hyman,J.H., Chen,J., Decamilli,P. and Brunger,A.T. (2000) Epsin1 undergoes nucleocytosolic shuttling and its ENTH domain, structurally similar to Armadillo and HEAT repeats, interacts with the transcription factor PLZF. *Journal of Cell Biology*, **149**, 537-545.

Ioerger,T.R. (1999) Detecting feature interactions from accuracies of random feature subsets. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pp. 49-54.

Ioerger,T.R. (2005) Automated detection of disulfide bridges in electron density maps using linear discriminant analysis. *Journal of Applied Crystallography*, **38**(1), 121-125.

Ioerger,T.R. and Sacchettini,J.C. (2002) Automatic modeling of protein backbones in electron-density maps via prediction of Cα coordinates. *Acta Crystallographica*, **D5**, 2043-2054.

Ioerger,T.R. and Sacchettini,J.C. (2003) The TEXTAL system: artificial intelligence techniques for automated protein model building. In Sweet,R.M. and Carter,C.W. (eds), *Methods in Enzymology*, **374**, 244-270.

Jain,A. and Zongker,D. (1997) Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(2), 153-158.

Jakulin,A. and Bratko,I. (2004) Testing the significance of attribute interactions. *Proceedings of the Twenty-first International Conference on Machine Learning*, pp. 409-416.

James,M.N.G. and Sielecki,A.R. (1983) Structure and refinement of Penicillopepsin at 1.8 Angstroms resolution. *Journal of Molecular Biology*, **163**, 299-361.

Joachims,T. (2002) Optimizing search engines using clickthrough data. *Proceedings of the Eighth ACM Conference on Knowledge Discovery and Data Mining*, pp. 133-142.

John,G., Kohavi,R. and Pfleger,K. (1994) Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 121-129.

Jones,K.S., Walker,S. and Robertson,S.E. (2000) A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, **36**, 779-840.

Jones,T.A. and Thirup,S. (1986) Using known substructures in protein model building and crystallography. *European Molecular Biology Organization (EMBO) Journal*, **5**, 819-822.

Jones,T.A., Zou,J.Y. and Cowtan,S.W. (1991) Improved methods for building models in electron density maps and the location of errors in these models. *Acta Crystallographica*, **A47**, 110-119.

Karmakar,N. (1984) A new polynomial-time algorithm for linear programming. *Combinatorica*, **4**, 373-395.

Kim,S., Woo,J., Seo,E.J., Yu,M. and Ryu,S. (2001) A 2.1 Å resolution structure of an uncleaved alpha(1)-antitrypsin shows variability of the reactive center and other loops. *Journal of Molecular Biology*, **306**, 109-119.

Kira,K. and Rendell,L.A. (1992) A practical approach to feature selection. *Proceedings of the Ninth International Conference on Machine Learning*, pp. 249-256.

Kleywegt,G.J. and Jones,T.A. (1997) Template convolution to enhance or detect structural features in macromolecular electron density maps. *Acta Crystallographica*, **D53**, 179-185.

Kittler,J. (1978) Features set search algorithms. In Chen,C.H. (ed), *Pattern Recognition and Signal Processing*, pp. 41-60. Sijthoff and Noordhoff, Netherlands.

Kohavi,R. and John,G.H. (1997) Wrappers for feature subset selection. *Artificial Intelligence*, **97**, 273-324.

Kohavi,R., Langley,P. and Yun,Y. (1997) The utility of feature weighting in nearest-neighbor algorithms. *Lecture Notes in Computer Science*, **1224**, 85-92.

Kokonenko,I. (1994) Estimating attributes: analysis and extensions of RELIEF. *Proceedings of the European Conference on Machine Learning*, pp. 171-182.

Kolodner,J. (1993) *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA.

Kontkanen,P., Myllymaki,P., Silander,T. and Tirri,H. (1997) A Bayesian approach for retrieving relevant cases. In Smith,P. (ed), *Artificial Intelligence Applications*, (*Proceedings of EXPERSYS-97 conference*), pp. 67-72.

Krogh,A., Brown,M., Mian,I.S., Sjolander,K. and Haussler,D. (1994) Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, **235**, 1501-1531.

Langley,P. and Iba,W. (1993) Average-case analysis of a nearest neighbor algorithm. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 889-894.

Leake,D.B. (1996) *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. MIT Press, Cambridge, MA.

Leherte,L., Glasgow,J.I., Fortier,S., Baxter,K. and Steeg,E. (1997) Analysis of three-dimensional protein images. *Journal of Artificial Intelligence Research*, **7**, 125-159.

Levinthal,C. (1968) Are there pathways for protein folding? *Journal de Chimie Physique et de Physico-Chimie Biologique*, **65**, 44.

Levitt,D.G. (2001) A new software routine that automates the fitting of protein X-ray crystallographic electron density maps. *Acta Crystallographica*, **D57**, 1013-1019.

Littlestone,N. (1992) Learning quickly when irrelevant attributes abound: a new linear threshold algorithm. *Machine Learning*, **8**, 293-321.

Liu,H. (2005) Evolving feature selection. *IEEE Intelligent Systems*, **20**(6), 59-63.

Liu,H. and Motoda,H. (eds) (1998) Feature Extraction, Construction, and Selection: A Data Mining Perspective. Kluwer, Boston, MA.

Liu,H. and Yu,L. (2005) Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, **17**(4), 491-502.

Lovell,S.C., Word,J.M, Richardson,J.S. and Richardson,D.C. (2000) The Penultimate Rotamer Library, *Proteins: Structure Function and Genetics* **40**, 389-408.

McKee,E.W., Kanbi,L.D., Childs,K.L., Grosse-Kunstleve,R.W., Adams,P.D., Sacchettini,J.C. and Ioerger,T.R. (2005) FINDMOL: automated identification of macromolecules in electron-density maps. *Acta Crystallographica*, **D61**, 1514-1520.

McRee,D.E. (1999a) *Practical Protein Crystallography*. Academic Press, San Diego, CA.

McRee,D.E. (1999b) XtalView/Xfit – a Versatile Program for Manipulating Atomic Coordinates and Electron Density, Journal of Structural Biology **125**, 156-165.

Mitchell,T.M. (1997) *Machine Learning*. McGraw-Hill, Boston, MA.

Mooney,S.D., Liang,M.H., DeConde,R. and Altman,R.B. (2005) Structural characterization of proteins using residue environments. *Proteins: Structure, Function and Bioinformatics*, **61**(4), 741-747.

Mowbray,S.L., Helgstrand,C., Sigrell,J.A., Cameron,A.D. and Jones,T.A. (1999) Errors and reproducibility in electron-density map interpretation. *Acta Crystallographica*, **D55**, 1309-1319.

Murray,R.K., Gramner,D.K., Mayes,P. and Rodwell,V. (2000) *Harper's Biochemistry*. Appleton and Lange, Stanford, CA.

Nelder,J.A. and Mead,R. (1965) A simplex method for function minimization. Computer Journal, **7**, 308-313.

Oldfield,T.J. (2003) Automated tracing of electron density maps of proteins. *Acta Crystallographica*, **D59**, 483-491.

Pai,R., Sacchettini,J.C. and Ioerger,T.R. (2006) Identifying non-crystallographic symmetry in protein electron-density maps: a feature-based approach. *Acta Crystallographica*, **D62**, 1012-1021.

Paredes,R. and Vidal,E. (2006) Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(7), 1100-1110.

Pavlidis,P., Wapinski,I. and Noble,W.S. (2004) Support vector machine classification on the web, *Bioinformatics* **20**(4), 586-587.

Peat,T.S., Newman.J., Waldo,G.S., Berendzen,J. and Terwilliger,T.C. (1998) Structure of translation initiation factor 5A from Pyrobaculum Aerophilum at 1.75 Å resolution. *Structure*, **6**, 1207-1215.

Perrakis,A., Morris,R. and Lamzin,V. (1999) Automated protein model-building combined with iterative structure refinement. *Nature Structural Biology*, **6**, 458-463.

Pyle,D. (1999) *Data Preparation for Data Mining*. Morgan Kaufmann Publishers.

Richardson,J.S. and Richardson,D.C. (1985) Interpretation of electron density maps. *Methods in Enzymology*, **115**, 189-206.

Riesbeck,C. and Schank,R. (1989) *Inside Case-Based Reasoning*. Lawrence Erlbaum, Hillsdale, NJ.

Romo,T.D., Gopal,K., McKee,E.W., Kanbi,L., Pai,R., Smith,J.N., Sacchettini,J.C. and Ioerger,T.R. (2005) TEXTAL: AI-based structural determination for X-ray protein crystallography. *IEEE Intelligent Systems*, **20**(6), 59-63.

Romo,T.D., Sacchettini,J.C. and Ioerger,T.R. (2006) Improving amino acid identification, fit and C-alpha prediction using the simplex method in automated model building. *Acta Crystallohraphica* D62, 1401-1406.

Rosenblatt,F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**(6), 386-408.

Rui,Y., Huang,T.S. and Chang,S. (1999) Image retrieval: current techniques, promising directions and open issues. *Visual Communication and Image Representation*, **10**(4), 39-62.

Russel,J.R. and Norvig,P. (1995) *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Sadle River, NJ.

Shepard,D. (1968) A two-dimensional function for irregularly spaced data. *Proceedings of the Twenty-third ACM National Conference*, pp. 517-524.

Short,R. and Fukunaga,K. (1980) A new nearest neighbor distance measure. *Proceedings of the Fifth IEEE International Conference on Pattern Recognition*, pp. 81-86.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195-197.

Smyth,B. and Cunningham,P. (1996) The utility problem analyzed: a case-based reasoning perspective. *Proceedings of the Third European Workshop on Case-Based Reasoning*, pp. 392-399.

Stanfill,C.W. (1987) Memory-based reasoning applied to English pronunciation. *Proceedings of the American Association for Artificial Intelligence Conference*, pp. 577-581.

Stanfill,C.W. and Waltz,D. (1986) Toward memory-based reasoning. *Communications of the ACM*, **29**, 1213-1228.

Subramanian,D., Greiner,R. and Pearl,J. (1997) The relevance of relevance. *Artificial Intelligence*, **97**, 1-5.

Sun,Y. and Li,J. (2006) Iterative RELIEF for feature weighting. *Proceedings of the Twenty-third International Conference on Machine Learning*, pp. 913-920.

Swanson,S.M. (1994) Core tracing: depicting connections between features in electron density. *Acta Crystallographica*, **D50**, 695-708.

Terry,A. (1983) The CRYSALIS project: hierarchical control of production systems. *Technical Report HPP-83-19*, Stanford University.

Terwilliger,T.C. (2002) Automated side-chain model-building and sequence assignment by template matching. *Acta Crystallographica*, **D59**, 45-49.

Tsigelny,I. (ed) (2002) *Protein Structure Determination: Bioinformatic Approach*. International University Line, La Jolla, CA.

Valiant,L. (1984) A theory of the learnable. *Communications of the ACM*, **27**(11), 1134-1142.

Vapnik,V.N. (1998) *Statistical Learning Theory. Adaptive and Learning Systems for Signal Processing, Communications, and Control*. Wiley, New York, NY.

Weston,J., Elisseff,A., Schoelkopft,B. and Tipping,M. (2003) Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, **3**, 1439-1461.

Widrow,B. and Hoff,M.E. (1960) Adaptive switching circuits. *IRE WESCON Convention Record*, 96-104.

Yang,D., Shipman,L.W., Roessner,C.A., Scott,I.A. and Sacchettini,J.C. (2002) Structure of the Methanococcus jannaschii mevalonate kinase, a member of the GHMP kinase superfamily. *Journal of Biological Chemistry*, **277**(11), 9462-9467.

Yavlinsky,A., Pickering,M.J., Heesch,D. and Ruger,S. (2004) A comparative study of evidence combination strategies. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1040-1043.

Yu,R.C., Hanson,P.I., Jahn,R. and Brunger,A.T. (1998) Structure of the ATP-dependent oligomerization domain of *N-ethylmaleimide* sensitive factor complexed with ATP. *Nature Structural Biology*, **5**, 803-811.

Zhang,Y and Skolnick,J. (2005) The protein structure prediction problem could be solved using the current PDB library. *Proceedings of the National Academy of Sciences, USA*, **102**(4), 1029-34.

VITA

Name: Kreshna Gopal

Address: Department of Computer Science
Texas A&M University
301 H.R. Bright Building
College Station, TX 77843-3112
USA

Email Address: kgopal@cs.tamu.edu

Education: B.Tech., Computer Science, Indian Institute of Technology, Kanpur, 1990
M.S., Computer Science, Texas A&M University, 2000
Ph.D., Computer Science, Texas A&M University, 2007