This is to certify that the

dissertation entitled

ENHANCING PATTERN RECOGNITION USING EVOLUTIONARY COMPUTATION FOR FEATURE SELECTION AND EXTRACTION WITH APPLICATION TO THE BIOCHEMISTRY OF PROTEIN-WATER BINDING

presented by

Michael L. Raymer

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Computer Science and Engineering

Major professor

Date 8/23/00

Enhancing Pattern Recognition Using Evolutionary Computation for Feature Selection and Extraction with Application to the Biochemistry of Protein-Water Binding

By

Michael L. Raymer

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering

August 23, 2000

E

Co

A

Sta

tive se

have a

ture e

order

allow

volve li

useful

do not

new fea

a new a

Abstract

Enhancing Pattern Recognition Using Evolutionary Computation for Feature Selection and Extraction with Application to the Biochemistry of Protein-Water Binding

By

Michael L. Raymer

Statistical pattern recognition techniques classify objects in terms of a representative set of features. The selection and quality of the features representing each object have a considerable bearing on the success of subsequent pattern classification. Feature extraction is the process of deriving new features from the original features in order to reduce the cost of feature measurement, increase classifier efficiency, and allow higher classification accuracy. Many current feature extraction techniques involve linear transformations of the original features to produce new features. While useful for data visualization and increasing classification efficiency, these techniques do not necessarily reduce the number of features that must be measured since each new feature may be a linear combination of some or all of the original features. Here a new approach is presented in which feature selection, feature extraction, and classi-

fier tra

metho

outperf

minimi

racy. T

junctio

hybrid

and cla

ability t

mining

nique is

of poten

water-bi

ical feat

binding.

fier training are performed simultaneously using evolutionary computing (EC). This method is tested in conjunction with a k-nearest-neighbors classifier, and shown to outperform other current methods for feature selection and extraction in terms of minimizing the number of features employed while maximizing classification accuracy. Two new classifiers based on the naïve Bayes classifier are developed in conjunction with the EC feature selection and extraction technique, and the resulting hybrid classifiers are shown to yield further improvements in feature subset parsimony and classification accuracy. A key advantage to the methods presented here is the ability to examine the set of linear feature weights produced by EC to perform data mining and exploratory data analysis. The EC feature selection and extraction technique is applied to an important and difficult problem in biochemistry—classification of potential protein-water binding sites. The resulting classifier is able to identify water-binding sites with $\sim$68% accuracy, and identifies a set of physical and chemical features that correspond well with the results of other studies of protein-water binding.

For

For my family, who never failed to believe in me, and especially for my father, who taught me the importance of learning.

First ar

Kuhn f

good fo

you bot

I would

Erik Go

as well

I gra

tion. M

Recogni

In a

GARAG

Victor M

venkatac

Wulfeku

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors, Dr. Bill Punch and Dr. Leslie Kuhn for their guidance, support, encouragement, and patience. It has been my good fortune to work with two such outstanding scientists and individuals. Thank you both for inspiring me to strive to excel, and for providing examples of excellence. I would also like to thank the other two members of my graduate committee, Dr. Erik Goodman and Dr. Rich Enbody, for their guidance and direction over the years, as well as their excellent suggestions for improvement of this dissertation.

I gratefully acknowledge the financial support of the National Science Foundation. Much of this work was supported by the NSF grant "Resolving Protein-Water Recognition".

In addition, I would like to thank my colleagues and friends from the MSU GARAGe, past and present, including Owen Matthews, Terry Dexter, Jason Greanya, Victor Miagkikh, Oleg Lukibanov, Iliana Martinez-Bermoudes, Lakshman Thiruvenkatachari, Michael Resendez and others. In particular, many thanks to Marilyn Wulfekuhler, Boris Gelfand, Arnold Patton, and Mike Boers for their ideas, construc-

tive cr

Lik

Analys

Vishal

an enj

special

their n

Tha

intellec

place to

Brehob

Fina

Laura—

courage

Withou

this far.

are.

tive criticisms, moral support, and friendship.

Likewise, I am grateful to the many members of the MSU Protein Structural Analysis and Design lab, and especially Sridhar Venkataraman, Carrie Barkham, Vishal Thakkar, Rajesh Korde, Trevor Barkham, and Maria Zavodszky, for creating an enjoyable research environment, and for their many contributions to this work. A special thanks to Paul Sanschagrin, Volker Schnecke and Brandon Hespenheide for their many contributions and friendship over the years.

Thank you to my friends and colleagues at MSU who have provided technical, intellectual, and emotional support, as well as making East Lansing an enjoyable place to spend the.past seven years: Travis Doom, Jen White, Mark Brehob, Katrina Brehob and Dave Paoletti.

Finally, I would like to extend my heartfelt gratitude to my family. Mom, Ed, Laura—there is nothing in my life that has provided me with more motivation, encouragement, and support than your belief in me. Above all, I thank my wife, Delia. Without your support, understanding, and encouragement I would never have come this far. I continue to aspire to become as remarkable a scientist and person as you are.

LIST O

LIST O

Table of Contents

List of Tables

3.5 Ap

3.6 Res

3.7 Bal

3.8 F

3.9 R

4.1 P

4.2 P

2.1 A ge

2.2 A th

2.3 A ge

2.4 A *d*

2.5 Effe

3.1 A C

3.2 An

3.3 An

3.4 Effe

A tv

List of Figures

3.6 Div

4.1 The

4.2 An

5.1 A p

5.2 Con

5.3 Ave

5.4 Relat
b
p
c
t
s
i
s
s

Cha

Intr

1.1

This the
tern rec
for simul
describe
feature s
classifica
sifier are
methods.
and extra
produce c
it is show
data mini

Chapter 1

Introduction

1.1 Overview—Contributions of the Thesis

This thesis describes several contributions to the state of the art of computational pattern recognition, evolutionary computation (EC), and biochemistry. A novel method for simultaneous feature selection and extraction using evolutionary computation is described. This method compares favorably with the best current algorithms for feature subset selection in terms of minimizing the number of features selected and classification accuracy. In addition, several new classifiers based on the Bayes classifier are developed and tested alongside the k-nearest-neighbors classifier and other methods, and are shown to perform well in conjunction with the EC feature selection and extraction methods developed here. The ability of the hybrid EC-classifiers to produce consistent feature subsets across various experiments is demonstrated, and it is shown that this capability can be useful in the analysis of classifier results for data mining and analysis. Finally, application of these methods to the biochemical

problem

that can

curacy of

prediction

not exhib

Analysis

the deter

sifiers des

incorpora

Kuhn, 2

software

ture dete

1.2

1.2.1

The field

many ar

employed

written.

in medic

problem of understanding protein-water interactions demonstrates two new classifiers that can predict water binding and conservation among protein structures. The accuracy of these classifiers is shown to compare well with other current methods for prediction of protein solvation. Furthermore, the new classifiers developed here do not exhibit a tendency to overpredict solvation, as is common among other methods. Analysis of the experimental results of these classifiers has provided new insights into the determinants of water binding to proteins. The weighted k-nearest-neighbor classifiers described here, with feature weights determined by GA optimization, has been incorporated into a new algorithm for identifying promising drug leads (Schnecke and Kuhn, 2000), and is currently being evaluated for incorporation into a well-known software package, XtalView, for identifying water binding sites during protein structure determination (McRee, 1992).

1.2 Motivation

1.2.1 Evolutionary Computation for Feature Selection and Extraction

The field of computational pattern recognition has produced useful applications in many areas of science and engineering. Pattern recognition techniques have been employed with great success, enabling computer systems to recognize typed, handwritten, and spoken language, to analyze satellite imagery, to aid in decision making in medicine and finance, and to perform numerous other difficult classification tasks.

A broad
works, a
problems
and techn
mary goa
system, f
disease, v
one of 10
perform
ple objec
training s
as a list
features
OCR syst
in a chara

In des
in advanc
redundan
ing featur
difficulty.

Featu

best

A broad range of methods, including statistical classifiers, decision trees, neural networks, and fuzzy logic have been applied to pattern recognition and classification problems. (Duda and Hart, 1973). Yet despite this wide variation in problem areas and techniques, several features are common to most classification systems. The primary goal of a classifier is to categorize objects or concepts into groups. A diagnosis system, for example, might classify patients as "at-risk" or "not-at-risk" for a certain disease, while OCR systems typically classify typewritten characters as belonging to one of 100 or so groups, one for each possible printed ASCII character. In order to perform this categorization, a classifier is typically trained by providing it with sample objects (or concepts) for which the correct classification is already known. These training samples, as well as the objects to be classified later, are generally represented as a list of features. For a diagnosis system, a patient might be represented by such features as height, weight, age, sex, and the results of various clinical tests. For an OCR system, features might include the length of the longest straight line segment in a character, or the presence or absence of a closed loop.

In designing a classification system for a particular task, it is often not known in advance which features will prove useful for classification, and which will contain redundant or even misleading information. Several areas of investigation, including feature subset selection and feature extraction, have come about to address this difficulty.

Feature selection is the problem of finding the subset of all the available features that best satisfies some classification-related criteria. Usually the goal is to find a

small

impos

sets gr

that m

investig

to find

Feat

original

cars fro

of the a

mass) v

Feat

recognit

vides a

equipme

tational

of featur

the inpu

In ad

conjuncti

fiers to b

trends in

small subset of features that provides good classification accuracy. It is generally impossible to exhaustively search all feature subsets, since the number of such subsets grows rapidly as d increases. For a fixed subset size, n, the number of subsets that must be searched is $\binom{n}{d}$. When n is not fixed, there are 2^d possible subsets to investigate. As a result, heuristic and suboptimal search methods are often employed to find a feature subset that performs well, but cannot be guaranteed to be optimal.

Feature extraction goes a step further and computes new features based on the original feature set. For example, if a classifier were attempting to distinguish sports cars from other vehicle types, and engine horsepower and total vehicle mass were two of the available features, a feature extraction method might find that (**horsepower** $\div$ **mass**) was a useful new feature, computed from the original features.

Feature selection and extraction can provide a number of benefits to a pattern recognition system. Limiting the number of features to be considered generally provides a reduction in the overall cost of a classifier by reducing the time, effort, and equipment involved in feature measurement and processing. Furthermore, the computational efficiency of many classification methods is improved by lowering the number of features to consider. Even the accuracy of a classifier can be improved by limiting the input data to a minimal set of salient features.

In addition to these direct benefits, the use of feature selection and extraction in conjunction with traditional pattern recognition can allow many traditional classifiers to be employed for data mining—the identification and analysis of interesting trends in large data sets. By identifying the particular features that are useful for

sepa tin

in u lers

data Wi

the owi

all a eas

from arg

This c

evol ion

of n ura

tech que

have been

of p oble

1996 Fog

algo thm

class iers

the aye

The esul

reco nitio

meth ds

mini g c

impo tan

sites f w

separating natural classes in large data sets, feature selection and extraction can aid in understanding the interrelationships between the features and the classes in the data. With the low cost and relatively high speed of modern magnetic storage, and the growing use of computational techniques for collection and analysis of data from all areas of science, engineering, and industry, the idea of mining interesting data from large data sets has become a topic of much experimentation and study recently.

This dissertation describes a feature selection and extraction method based on evolutionary computing, a parameter optimization method based on the dynamics of natural selection and the Darwinian model of evolution. The two evolutionary techniques employed, genetic algorithms (GAs) and evolutionary programming (EP), have been shown to be effective search and optimization methods for a broad array of problems (Fogel, 1998; Holland, 1975; Goldberg, 1989; Bäck and Schwefel, 1993, 1996; Fogel *et al.*, 1966; Koza, 1992). The EC-based feature selection and extraction algorithms developed here are utilized to optimize the performance of several different classifiers, including traditional algorithms such as nearest-neighbor classification and the Bayes classifier, and several novel methods derived from the Bayes classifier. The resulting hybrid algorithms are shown to outperform various traditional pattern recognition techniques alone and in conjunction with commonly-used feature selection methods on a variety of artificial and real-world data sets. In addition, the data-mining capabilities of the EC-hybrid classifiers are explored in the analysis of an important problem from biochemistry—understanding the features determining the sites of water binding at protein surfaces.

1.2.2

Extensive
portant a
Much of
their stru
drug desi
tion of pr
play an i
molecule
design, p
and func
protein c

The h
structure
sites and
extractio
nants of
the inter

1.2.2 Understanding Protein-Water Interactions

Extensive application of the EC-hybrid classifiers was made in the study of an important and difficult problem in biochemistry—understanding protein–water binding. Much of the current research in biochemistry is targeted at understanding proteins—their structure, function, and interactions with other molecules. Most pharmaceutical drug design efforts are directed at producing drugs that bind to and inhibit the function of proteins. It is well known that the water molecules that surround most proteins play an important role in protein folding, structural stability, and binding to other molecules, such as drugs. Unfortunately, correct treatment of water molecules in drug design, protein folding studies, and many other areas of study in protein structure and function has proven to be a difficult task, because the state of hydration of the protein changes during these processes.

The hybrid EC-classifiers described here have been applied to a database of protein structures to produce a classifier that is capable of predicting protein water-binding sites and their conservation upon binding other molecules. The feature selection and extraction capabilities of this classifier have been employed to analyze the determinants of water binding, which has contributed to a more complete understanding of the interactions between proteins, the molecules they bind, and water molecules.

Cha

Bac

2.1

2.1.1

Modern
previous
simulate
broadly
pattern
our inte
by anal
processe
As such
benefits

Chapter 2

Background and Synopsis of Literature

2.1 Pattern Classification

2.1.1 The Pattern Recognition Task

Modern advances in computer hardware and software technology have allowed many previously intractable problems from various fields of scientific inquiry to be analyzed, simulated, or otherwise addressed using computational techniques. One of the most broadly applicable and potentially profitable of these techniques is computational pattern recognition. For humans, the ability to identify patterns is vital to many of our intellectual faculties, including: our ability to abstract from examples, to reason by analogy, and to learn by induction. Yet the actual mechanisms by which these processes are effected remains a subject for debate and empirical experimentation. As such, advances in the realm of computational pattern recognition bear potential benefits in many areas, including improvement of data analysis methods for the phys-

ical sci

unders

Wh

niques

it is po

classifi

system

Subje

Figure

In th

The tas

the obje

of this o

For phys

various p

observat

ical sciences, advancement of machine perception and vision, and eventually further understanding of human perception and cognition.

While the large quantity and variety of pattern recognition algorithms and techniques prevent the construction of a universal model for all pattern recognition tasks, it is possible to consider a simple model which captures the essential features of many classification systems. Figure 2.1 illustrates a model of a generic pattern recognition system.

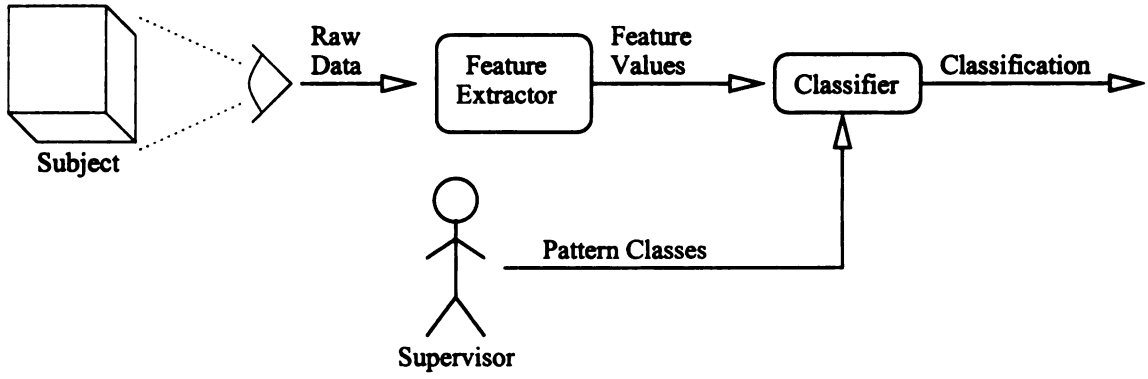


Figure 2.1: A general model for classical supervised pattern classification systems.

In this model, the object of classification may be a physical object or a concept. The task is to assign the object to one of several prespecified categories. Initially, the object is observed, yielding a collection of unprocessed, raw data. The nature of this observation can vary greatly according to the type of object being classified. For physical objects, cameras, sensors, or other transducers are often used to record various physical and visual properties of the object. For classification of concepts, 'observation" may be more abstract. For a medical diagnosis system, for example,

the

prov

clas

prop

a ca

aver

I

mon

valu

sible

of it

is oft

to as

N

ing s

of th

inform

the d

and t

not b

the us

the data is often a combination of clinical test results and a description of symptoms provided by the patient. While it is possible to use all of the unprocessed data to classify the object, it is usually beneficial to reduce this raw data to a set of features or properties related to the classification categories. Visual information collected with a camera, for example, might be summarized into a set of features describing the average color, general shape, and approximate size of the classification object.

In general, the goal of feature selection is to reduce the input data to a parsimonious and salient representation for the classification subject: a vector of feature values. This feature vector then serves as the input for the classifier, which is responsible for determining which category each subject belongs to according to the values of its features. The feature vector associated with a particular classification subject is often called a *pattern*, and the category associated with the subject may be referred to as that pattern's *class*.

Many classical pattern recognition methods are operated in two stages. The *training* stage is distinguished by the presence of a supervisory agent that has knowledge of the true class for each pattern. During training, the supervisor provides this class information to the classifier as each pattern is observed, allowing the classifier to learn the distinctions among classes. During the *testing* stage, the class input is removed, and the performance of the classifier is observed. The training and testing stages need not be completely disjoint. In some handwriting recognition systems, for example, the user can correct misclassified letters while the system is in normal operation.

2.1

Con

the

func

$P(\omega$

info

$P(\omega$

best

any

prob

obta

com

Onc

mat

the

2.1.2 The Bayes Classifier

Consider the task of assigning a sample to one of C classes, $\{\omega_1, \omega_2, ...\omega_C\}$, based on the d-dimensional observed feature vector $\vec{x}$. Let $p(\vec{x}|\omega_i)$ be the probability density function for the feature vector, $\vec{x}$, when the true class of the sample is ω_i. Also, let $P(\omega_i)$ be the relative frequency of occurrence class ω_i in the samples. If no feature information is available, the probability that a new sample will be of class ω_i is $P(\omega_i)$—this probability is referred to as the *a priori* or prior probability, and is the best estimate that a sample will belong to a particular class prior to observation of any feature values. Once the feature values are obtained, we can combine the prior probability with the class-conditional probability for the feature vector, $p(\vec{x}|\omega_i)$, to obtain the *a posteriori* probability that a pattern belongs to a particular class. This combination is done using Bayes rule (Bayes, 1763):

$$P(\omega_j|\vec{x}) = \frac{p(\vec{x}|\omega_j)P(\omega_j)}{\sum_{i=1}^{C} p(\vec{x}|\omega_i)P(\omega_i)} \tag{2.1}$$

Once the posterior probability is obtained for each class, classification is a simple matter of assigning the pattern to the class with the highest posterior probability—the resulting decision rule is Bayes decision rule:

given $\vec{x}$, decide ω_i if

$$P(\omega_i|\vec{x}) \quad > P(\omega_j|\vec{x}) \quad \forall j$$

When

probab

in the

1973. I

teriori

trainin

from t

param

distrib

trainin

distrib

class.

of esti

classif

tion c

a hist

of the

as the

tion I

histo

O

the q

When the class-conditional probability density for the feature vector and the prior probabilities for each class are known, the Bayes classifier can be shown to be optimal in the sense that no other decision rule will yield a lower error rate (Duda and Hart, 1973, pp. 10–17). Of course, these probability distributions (both *a priori* and *a posteriori*) are rarely known during classifier design, and must instead be estimated from training data. Class-conditional probabilities for the feature values can be estimated from the training data using either a parametric or a non-parametric approach. A parametric method assumes that the feature values follow a particular probability distribution for each class and estimate the parameters for the distribution from the training data. For example, a common parametric method is to assume a Gaussian distribution of the feature values, and then estimate the parameters μ_i and σ_i for each class, ω_i, from the training data. The parametric method can simplify the problem of estimating the class-conditional feature distributions, but can also result in poor classification if the assumed probability distribution does not fit the actual distribution of feature values. A non-parametric approach usually involves construction of a histogram from the training data to approximate the class-conditional distribution of the feature values. Several decisions in the construction of this histogram, such as the number of bins and the size of each, are critical in obtaining good classification performance. The lack of a standard procedure for constructing a distribution histogram is a primary weakness in the non-parametric sampling approach.

Once the distribution of the feature values has been approximated for each class, the question remains how to combine the individual class-conditional probability den-

sity fu

ity der

assume

Th

to perf

is not

probab

literatu

relative

prior p

for all

2.1.3

Estimat

sificatio

design a

values,

way, no

k-neares

sity functions for each feature, $p(x_1|\omega_i), p(x_2|\omega_i)...p(x_d|\omega_i)$ to determine the probability density function for the entire feature vector: $p(\vec{x}|\omega_i)$. A common method is to assume that the feature values are statistically independent:

$$p(\vec{x}|\omega_i) = p(x_1|\omega_i) \times p(x_2|\omega_i) \times ... \times p(x_d|\omega_i) \tag{2.2}$$

The resulting classifier, often called the *naïve Bayes classifier* has been shown to perform well on a variety of data sets, even when the independence assumption is not strictly satisfied (Domingos and Pazzani, 1996). The selection of the prior probabilities for the various categories has been the subject of a substantial body of literature (Jaynes, 1968). One of the most common methods is to simply estimate the relative frequency for each class from the training data and use these values for the prior probabilities. An alternate method is to simply assume equal prior probabilities for all categories by setting $P(\omega_i) = \frac{1}{C}$, $i = 1...C$.

2.1.3 Nearest Neighbors Classification

Estimation of the class-conditional distribution of the feature values in Bayesian classification is a potential source of classification error. An alternative approach is to design a classifier that does not require an estimate of the distribution of the feature values, but instead memorizes the feature values for every training sample. In this way, no training information is lost in parameter estimation. One such classifier is the k-nearest-neighbors (knn) classifier (Cover and Hart, 1967). The operation of the knn

classif

are st

consid

corres

space.

demo

Figur
plott
test s

A

and

of the

predi

singl

the n

[1]
is de
of th

classifier during training is quite simple—the feature values for each training sample are stored in their entirety. To classify a new testing sample, the training samples are considered in a d-dimensional space, where each of the orthogonal coordinate axes corresponds to one classification feature. The test sample is plotted into this feature-space, and the k nearest neighbors, for some small constant k, are found. Figure 2.2 demonstrates the behavior of a 3-class, 5-nearest-neighbors classifier where $d = 2$.

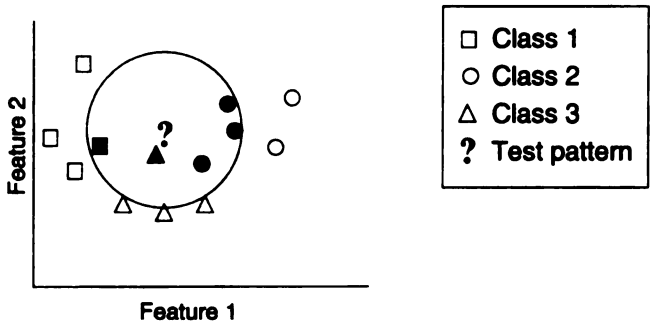


Figure 2.2: A three class, 5-nearest-neighbors classifier. The training samples are plotted in a two-dimensional feature space. Three of the five nearest neighbors of the test sample are of class 2, so the test sample will be classified as belonging to class 2.

Any distance metric can be used to calculate the neighbors—Euclidean distance and Mahalanobis distance[1] are the two metrics most commonly employed. The classes of these k neighbors are tallied, and the most commonly represented class becomes the predicted class for the test sample. Several tie breaking schemes are possible when a single majority class is not present among the near neighbors. The most common, and the method used here, is to choose the class of the closest neighbor. Other methods

[1]Euclidean distance is defined as $d^2 = (\vec{x}_2 - \vec{x}_1)^{\mathrm{T}} (\vec{x}_2 - \vec{x}_1)$. Mahalanobis distance is defined as $d^2 = (\vec{x}_2 - \vec{x}_1)^{\mathrm{T}} \Sigma^{-1} (\vec{x}_2 - \vec{x}_1)$, where Σ is the $d \times d$ covariance matrix of the d-dimensional training data.

incl

mos

mis

dist

sho

of t

bou

dis

an

sig

de

(G

tes

as

pe

tra

ne

include scaling each vote by the distance from the current test sample, choosing the most common class from the training data, and choosing the class with the greatest misclassification cost. Feature values are usually normalized prior to computing the distances to prevent the scale of each feature from biasing the prediction. It can be shown that the asymptotic error probability (that is, the error rate as the number of training samples approaches infinity) of the nearest neighbor classification rule is bounded by twice the Bayes error probability(Duda and Hart, 1973, p. 101); i.e.

$$\lim_{n\to\infty} P_e(\text{Bayes}) \leq \lim_{n\to\infty} P_e(\text{NN}) \leq 2 * \lim_{n\to\infty} P_e(\text{Bayes}) \tag{2.3}$$

A drawback to the knn classifier is the computational expense of computing the distance between the testing sample and each of the training samples. A branch-and-bound approach to near-neighbor search (Fukunaga and Narendra, 1975) can significantly reduce this overhead for large training sets.

Many variants of the knn classifier have been proposed and explored. The condensed nearest neighbor rule (Hart, 1968) and the reduced nearest neighbor rule (Gates, 1972) seek to reduce the number of training patterns that must be stored and tested for each near-neighbor search, while maintaining the same decision boundaries as the nearest neighbor rule. The reduced nearest neighbor rule, in particular, expends considerable computation effort to find the minimal consistent subset of the training patters in order to reduce the computation time needed later to find near neighbors during on-line classification. The edited nearest neighbor rule (Wilson,

1972)

sampl

nated

neighb

Fo

from t

confid

can re

The u

Th

param

using

condi

sumpt

traini

densit

distril

The n

1962)

tions

the de

metho

1972) smoothes the decision surface by removing "outlier" training samples—that is, samples belonging to a certain class that fall in an area of the feature space dominated by another class. The resulting decision rule is less sensitive than the nearest neighbor rule to erroneous feature measurement and anomalous training data.

For values of k larger than 1, a measure of classification confidence can be obtained from the number of nearest neighbors belonging to the majority class. This measure of confidence can be used to define a reject option for the k nearest neighbor rule, which can result in a reduction in the error rate for those samples that are not rejected. The use of majority votes as a confidence measure is discussed further in Chapter 3.

The well-known statistical classifiers can be partitioned into parametric and non-parametric methods. As described previously, the Bayes classifier can be implemented using either approach. The parametric methods assume that the form of the class-conditional density function of the features is known in advance. A common assumption is that the feature values follow a multivariate Gaussian distribution. The training data are then used to estimate the parameters of these class-conditional densities (e.g. the mean vector, μ, and the covariance matrix, σ, for the Gaussian distribution) and these estimated densities are are then used to classify new patterns. The non-parametric methods, including Parzen window density estimation (Parzen, 1962) and the nearest-neighbor methods (Cover and Hart, 1967), make no assumptions about the class-conditional distribution of feature values. Rather, the form of the density function is either fitted to the training data, as in the Parzen window method, or dealt with implicitly, as in the nearest neighbor rule.

2.

2.

Th

pro

cla

des

rel

to

fea

tim

of

eac

nei

cor

spr

du

fea

rec

is

19

2.2 Feature Selection and Extraction

2.2.1 Feature Costs and the Curse of Dimensionality

The selection of features to measure and include in the feature vector can have a profound impact on the accuracy of the resulting classifier, regardless of what specific classification rule is implemented. A common approach is to have human experts describe as many features as possible that are readily measurable and likely to be related to the classification categories. Unfortunately, there are several disadvantages to evaluating a profuse number of features in classification. First, each additional feature to be considered often incurs an additional cost in terms of measurement time, equipment costs, and storage space. In addition, the computational complexity of classification grows with each additional feature. For some classifiers the cost of each additional feature in computational complexity can be significant. The nearest-neighbor classification, for example, has a complexity that is $O(n^2)$, where n is the combined size of the training and testing data sets. In addition, the inclusion of spurious features (features unrelated to the classification categories) is likely to reduce classification accuracy. In fact, it is sometimes the case that the inclusion of features that do, in fact, contain information relevant to classification can result in reduced accuracy when the number of training samples is fixed. This phenomenon is sometimes referred to as *the curse of dimensionality* (Jain and Chandrasekaran, 1982).

At the root of the problem of high-dimensionality lies the fact that in real-world

classifiers the class-conditional densities are not known, and must be estimated from a finite number of training samples. This fact is easily seen when considering a classification task involving two classes, where the feature vector for each class follows a multivariate Gaussian distribution. As each new feature, x_i, is added to the pattern vector, additional information is made available for classification, as long as the mean of the feature, μ_i is not identical to that of some other feature. However, in addition to providing new information, the inclusion of the new feature introduces additional parameters which must be estimated to characterize the class-conditional distribution of the feature vector. In the case of a Gaussian distribution, an additional mean value is added to $\vec{\mu}$ and the covariance matrix σ changes from a $d \times d$ matrix to a $(d+1) \times (d+1)$ matrix. Since the covariance matrix is symmetric, the number of parameters which must be estimated increases by $(d+2)$ for each additional feature. Clearly, the geometric growth in the number of parameters to be estimated has the potential to deteriorate the value of additional features to the classifier, even when the new features contain additional information. This effect was illustrated for a specific two-class problem with Gaussian distribution of feature values by Trunk (1979).

2.2.2 Feature Selection

A number of techniques have been developed to address the problem of dimensionality, including feature selection and feature extraction. The main purpose of feature selection is to reduce the number of features used in classification while maintaining an acceptable classification accuracy. Less discriminatory features are eliminated,

leaving a subset of the original features which retains sufficient information to discriminate well among classes. For most problems the brute-force approach is prohibitively expensive in terms of computation time. Cover and Campenhout (1977) have shown that to find an optimal subset of size n from the original d features, it is necessary to evaluate all $\binom{d}{n}$ possible subsets when the statistical dependencies among the features are not known. Furthermore, when the size of the feature subset is not specified in advance, each of the (2^d) subsets of the original d features must be evaluated. In the special case where the addition of a new feature always improves performance, it is possible to significantly reduce the number of subsets that must be evaluated using a branch and bound search technique (Narendra and Fukunaga, 1977). Unfortunately, this sort of monotonic decrease in the error rate as new features are added is often not found in real-world classification problems due to the effects of the curse of dimensionality and finite training sample sizes.

Various heuristic methods have been proposed to search for near-optimal feature subsets. Sequential methods involve the addition or removal of a single feature at each step. Sequential forward selection (Whitney, 1971) begins with the single feature which provides the best classification performance. At each iteration, the feature which provides the best performance in combination with the current subset of features is added, until a subset of the desired size is generated. Once a feature is added, it cannot be removed. Sequential backwards selection operates similarly, but begins with all features included, removing a single feature at each iteration. "Plus l – take away r" selection combines these two methods by alternately enlarging and reducing

the feature subset repeatedly. The sequential floating forward selection algorithm (SFFS) of Pudil *et al.* (1994) is a further generalization of the plus l, take away r methods, where l and r are not fixed, but rather are allowed to "float" to approximate the optimal solution as much as possible. The algorithm basically executes as follows:

- Let $\mathbf{X}_k = x_1 ... x_k$ be a set of k features
- Let $J(\cdot)$ be a criterion function (e.g. classification accuracy)
- Let the significance of the feature x_j in the set $\mathbf{X}_k$, be $S_{k-1}(x_j) = J(\mathbf{X}_k) - J(\mathbf{X}_k - x_j)$
- Let the significance of the feature $x_j \notin \mathbf{X}_k$, be $S_{k+1}(x_j) = J(\mathbf{X}_k + x_j) - J(\mathbf{X}_k)$
- Beginning with an initial set of k features, $\mathbf{X}_k$...
    1. Using the sequential forward selection method, find $\mathbf{X}_{k+1}$.
    2. Find the least significant feature in $\mathbf{X}_{k+1}$. If x_{k+1} is the least significant feature, then set $k = k + 1$ and go to step 1.
    3. If $x_r, 1 \leq r \leq k$ is the least significant feature in $\mathbf{X}_{k+1}$, then exclude x_r from $\mathbf{X}_{k+1}$ to form a new feature set $\mathbf{X}'_k$.
    4. If $J(\mathbf{X}'_k) > J(\mathbf{X}_k)$:
        - If $k = 2$ then set $\mathbf{X}_k = \mathbf{X}'_k$ and go to step 1.
        - Otherwise continue conditional exclusion by returning to step 2.

In a recent study of current feature selection techniques, Jain and Zongker (1997) evaluated the performance of fifteen feature selection algorithms in terms of classifica-

tio

Th

for

and

SF

beh

clas

add

gan

beh

the

niqu

a nu

the i

asso

have

gene

tradi

probl

classi

on a

tion error and run time on a 2-class, 20-dimensional, multivariate Gaussian data set. Their findings demonstrated that the SFFS algorithm dominated the other methods for this data, obtaining feature selection results comparable to the optimal branch-and-bound algorithm while requiring less computation time. Further tests of the SFFS technique in combination with a knn classifier were conducted to observe the behavior of the knn classification rule as additional features were provided to the classifier. The results showed that classification accuracy was initially improved as additional features were introduced, but eventually reached a maximal value and began to decline with the introduction of further features. Because of this unimodal behavior, selection of an optimal number of features is straightforward when using the SFFS method in combination with a knn classifier.

When classification is being performed using neural networks, node pruning techniques can be used for dimensionality reduction (Mao *et al.*, 1994). After training for a number of epochs, nodes are removed from the neural network in such a manner that the increase in squared error is minimized. When an input node is pruned, the feature associated with that done is no longer considered by the classifier. Similar methods have been employed in the use of fuzzy systems for pattern recognition through the generation of fuzzy if-then rules (Nozaki *et al.*, 1996; Ishibuchi *et al.*, 1995). Some traditional pattern classification techniques, while not specifically addressed to the problem of dimensionality reduction, can provide feature selection capability. Tree classifiers (Quinlan, 1986b), for example, typically partition the training data based on a single feature at each tree node. If a particular feature is not tested at any

node of the decision tree, it is effectively eliminated from classification. Additionally, simplification of the final tree can provide further feature selection (Quinlan, 1987).

2.2.3 Feature Extraction

Feature extraction, a superset of feature selection, involves transforming the original set of features to provide a new set of features, where the transformed feature set usually consists of fewer features than the original set. While both linear and non-linear transformations have been explored, most of the classical feature extraction techniques involve linear transformations of the original features. Formally, the objective for linear feature extraction techniques can be stated as follows:

> Given an $n \times d$ pattern matrix $\mathcal{A}$ (n points in a d-dimensional space), derive an $n \times m$ pattern matrix $\mathcal{B}$, $m < d$, where $\mathcal{B} = \mathcal{A}\mathcal{H}$ and $\mathcal{H}$ is a $d \times m$ transformation matrix.

According to this formalization, many common methods for linear feature extraction can be specified according to the method of deriving the transformation matrix, $\mathcal{H}$. For unsupervised linear feature extraction, the most common technique is principal component analysis (Duda and Hart, 1973). For this method, the columns of $\mathcal{H}$ consist of the eigenvectors of the $d \times d$ covariance matrix of the given patterns. It can be shown that the new features produced by principal component analysis are uncorrelated and maximize the variance retained from the original feature set (Duda

and Hart, 1973). The corresponding supervised technique is linear discriminant analysis. In this case, the columns of $\mathcal{H}$ are the eigenvectors corresponding to the nonzero eigenvalues of the matrix $\mathcal{S}_W^{-1}\mathcal{S}_B$, where $\mathcal{S}_W$ is the within-class scatter matrix and $\mathcal{S}_B$ is the between-class scatter matrix for the given set of patterns. Deriving $\mathcal{H}$ in this way maximizes the separation between class means relative to the covariance of the classes (Duda and Hart, 1973). In the general case, the matrix $\mathcal{H}$ is chosen to maximize some criteria, typically related to class separation or classification accuracy for a specific classifier. In this view, feature selection is a special case of linear feature extraction, where the off-diagonal entries of $\mathcal{H}$ are zero, and the diagonal entries are either zero or one.

Feature selection and extraction can enhance pattern recognition in a number of different ways. As discussed previously, these techniques can reduce the cost of feature measurement and storage, as well as providing improved classification accuracy. Additionally, feature selection and extraction can help to reveal the relationships between the features available to a classifier and the classification categories. By identifying minimal sets of features that are sufficient for accurate classification, feature selection and extraction can aid researchers in understanding which features are related to the categories in the data. This function is closely related to the problem of data mining, and can be a useful tool for analysis of large data sets.

2.3 Evolutionary Computation

Dimensionality reduction is well suited to formulation as an optimization problem, thus making it available to the diverse array of computational techniques that have been developed and explored in this area. Some of these methods have been developed by observation and modeling of the process of Darwinian evolution and natural selection. Such methods, collectively termed evolutionary computation, have been developed and studied from various viewpoints, leading to a number of different techniques and specific algorithms (Fraser, 1957; Crosby, 1967; Bremermann *et al.*, 1966; Reed *et al.*, 1967; Fogel, 1998).

2.3.1 Genetic Algorithms

Genetic Algorithms comprise a subset of evolutionary computation focusing on the application of selection, mutation, and recombination to a population of competing problem solutions (Holland, 1975; Goldberg, 1989). Figure 2.3 shows a general model of a simple GA. Problem solutions are encoded as strings of information, or chromosomes. A population of competing solutions is maintained and sorted according to the application-specific fitness of each solution. Each generation, a stochastic selection process is employed to choose individuals to advance to the next generation. Individuals with higher fitness values are more likely to be chosen, but the stochastic nature of the selection process allows for the selection of any individual in the population. Various selection methods have been employed, including fitness proportionate

selection (Holland, 1975), tournament selection, rank selection, and others. From the individuals that do advance to the next generation, some are chosen at random as parent individuals for recombination. The process of recombination combines traits of the parent individuals to form a new child individual. Again, various techniques are possible, including one- and two-point crossover, uniform crossover, and others (Goldberg, 1989; Mitchell, 1996). It has been proposed that the learning power of the genetic algorithm is largely derived from the interaction of recombination and selection, which together isolate "building blocks", or short motifs common to highly fit chromosomes. Such motifs can then combine to form new chromosomes of even higher fitness (Holland, 1975; Goldberg, 1989). Finally, random mutation is employed to introduce a controlled amount of noise into the chromosomes, which in turn maintains diversity in the population and helps to avoid premature convergence of the population at local extrema. Genetic algorithms have been shown to be a useful tool in computer optimization problems from a broad range of scientific disciplines.

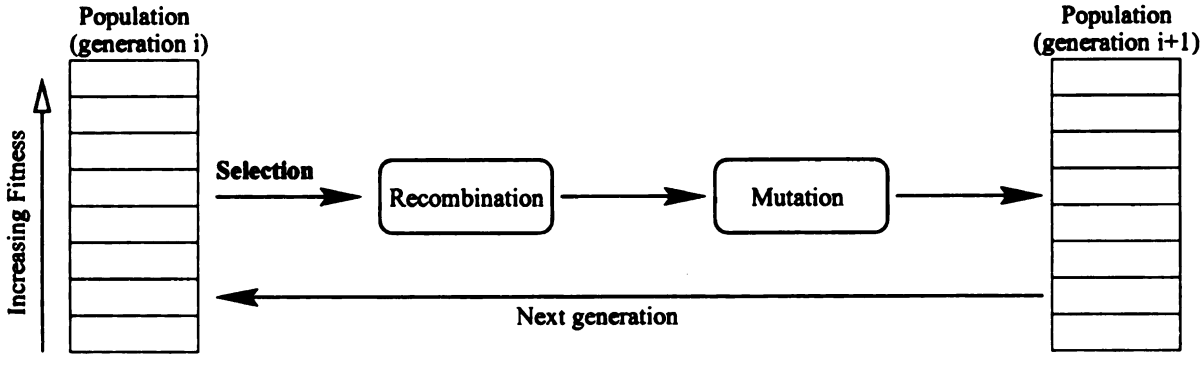


Figure 2.3: A general model of a simple Genetic Algorithm.

2.3.2 Evolution Strategies and Evolutionary Programming

The popularity of the GA as an optimization tool is largely due to its ability to avoid local optima by maintaining a diverse population of competing problem solutions. Unfortunately, the mechanisms of recombination and selection are not the most appropriate tools for fine-tuning nearly-optimal problem solutions. As a result, genetic algorithms are more powerful for the early stages of optimization, when general areas of high fitness are sought, than for later stages where solutions in these areas are being refined toward a global optimum. Some researchers have concluded that GAs are, in fact, not an appropriate technique for real-valued function optimization (De Jong, 1992).

Evolutionary programming (Fogel *et al.*, 1966), and evolution strategies (ES; Rechenberg, 1973; Schwefel, 1977), two subsets of the EC family of methods, are often considered more suited to real-valued function optimization and refinement of nearly-optimal problem solutions. Like GAs, EP and ES algorithms are parallel, iterative optimizers. Also like GAs, EP and ES maintain a population of competing problem solutions which are evaluated in terms of a fitness function and subjected to a selection function for inclusion in successive generations. Unlike GAs, however, EP and ES do not emphasize recombination as a primary method for optimization. Instead, these techniques generally rely on various forms of mutation as the key means of learning. For EP, which is often employed for real-valued function optimization, one of the most common mutation operators involves the addition of a Gaussian deviate to the current value of a field on the chromosome. The Gaussian deviate is

generally chosen from a distribution with a mean of zero and a variance that is either computed based on the degree of convergence of the current population or stored and evolved with each individual chromosome (Bäck and Schwefel, 1996). The specific selection method used by EP is slightly different from those commonly used in GAs as well. A typical method is to compete each individual in a fixed number of "tournaments", in which its fitness is compared with a random member of the population to determine a winner. After each individual has competed in the requisite number of tournaments, the entire population is sorted according to the number of tournament wins, and the upper half of this sorted population is allowed to advance to the next generation. As with GA selection methods, many variations on this basic method have been explored.

2.4 Evolutionary Computation in Feature Selection and Extraction

A direct approach to using GAs for feature selection was introduced by Siedlecki and Sklansky (1989). In their work, a GA is used to find an optimal binary vector, where each bit is associated with a feature (Figure 2.4). If the i^{th} bit of this vector equals 1, then the i^{th} feature is allowed to participate in classification; if the bit is a 0, then the corresponding feature does not participate. Each resulting subset of features is evaluated according to its classification accuracy on a set of testing data using a nearest-neighbor classifier.

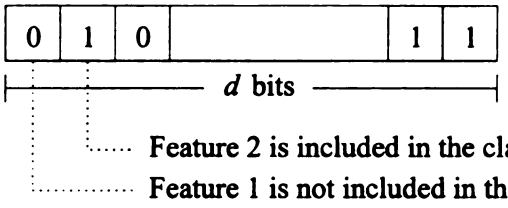


Figure 2.4: A d-dimensional binary vector, comprising a single member of the GA population for GA-based feature selection.

This technique was later expanded to allow linear feature extraction, by Punch *et al.* (1993) and independently by Kelly and Davis (1991). The single bit associated with each feature is expanded to a real-valued coefficient, allowing independent linear scaling of each feature, while maintaining the ability to remove features from consideration by assigning a weight of zero. Given a set of feature vectors of the form $\mathbf{X} = \{x_1, x_2, ...x_d\}$, the GA produces a transformed set of vectors of the form $\mathbf{X}' = \{w_1x_1, w_2x_2...w_dx_d\}$ where w_i is a weight associated with feature i. Each feature value is first normalized, then scaled by the associated weight prior to training, testing, and classification. This linear scaling of features prior to classification allows a classifier to discriminate more finely along feature axes with larger scale factors. A knn classifier is used to evaluate each set of feature weights. The effects of linear feature weighting on the knn classification rule are visualized in Figure 2.5. Patterns plotted in feature space are spread out along feature axes with higher weight values, and compressed along features with lower weight values. The value of k for the knn classifier is fixed and determined empirically prior to feature extraction.

In a similar approach, Yang and Honavar (1998) use a simple GA for feature subset

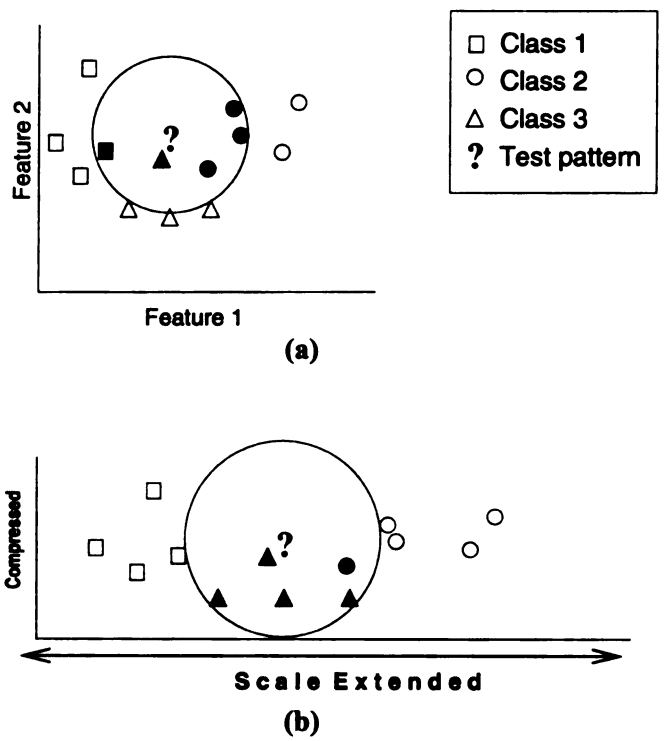


Figure 2.5: Effect of scaling feature axes on k-nearest-neighbor classification. (a) original data; (b) scaled data. Extension of the scale of the horizontal axis increases the distance between patterns which differ in feature 1, allowing the knn to discriminate more finely along this dimension. Here, the prediction of the unknown changes from class 2 to class 3 as a result of scaling.

selection in conjunction with DistAI, a neural network-based pattern classifier (Yang *et al.*, 1998). As in other GA-based feature selectors, a simple binary representation was used where each bit corresponds to a single feature. The use of the GA for feature subset selection improved the accuracy of the DistAI classifier for nearly all of the data sets explored, while simultaneously reducing the number of features considered. Their hybrid classifier, GADistAI, outperformed a number of modern classification methods on the various data sets presented.

Vafaie and De Jong (1998) describe a hybrid technique in which EC methods are employed for both feature selection and extraction[2] in conjunction with the C4.5 decision tree classifier system (Quinlan, 1986a). Again, a binary representation is used for feature subset selection using traditional GA techniques. In this system, however, the features seen by the classifier are functions of the original features composed of simple arithmetic operations. For example, one such feature might be $\{(F1 - F2) \times (F2 - F4)\}$, where $F1, F2$, and $F4$ represent values from the original feature set. These functions for constructing new features are represented as trees, in much the same way as commonly seen in the genetic programming (GP; Koza, 1992) literature. GP subtree crossover is used for recombination of these function trees, while traditional GA-style mutation and crossover are used for the feature selection chromosome. In this system, feature selection and extraction are not simultaneous. Rather, feature selection, feature extraction, and classifier construction are performed serially each generation. The resulting system was shown to outperform the decision

[2]The authors use the term "feature construction".

tree classifier using the original feature set, while reducing the number of features considerably, for human facial image recognition. Additionally, the hybrid system was shown to outperform several contemporary classifiers on three diverse data sets.

Ch

Fea

the

Co

Lea

3.1

3.1.

The

use

the

sets.

valu

Chapter 3

Feature Selection and Extraction using the K-Nearest-Neighbors Classifier in Combination with Evolution-based Learning

3.1 Methods

3.1.1 Branch and Bound Near-Neighbor Searching

The k-nearest-neighbors classifier has several features that make it a good choice for use with feature selection and extraction algorithms. The nonparametric nature of the nearest neighbors rule allows the classifier to be applied to a broad range of data sets, including those where the form of the class-conditional distribution of feature values is completely unknown. When used with a Euclidean distance metric, the knn

decision rule is sensitive to scaling of the feature values, which forms the basis for the feature extraction technique applied here. Many classifiers based on the class-conditional distribution of feature values, including the Bayes classifier, are invariant to scaling of the feature values.

The primary drawback of the knn classifier is the computational complexity of the near neighbor search. The search for the neighbors of a single test pattern takes $O(n * d)$ time where n is the number of training samples, and d is the number of features being considered. Thus, the evaluation of a test set of m samples takes $O(n*m*d)$ time, which can quickly become prohibitively expensive for large training and testing set size and large numbers of features. To reduce the computational cost of the near neighbor search, the branch and bound search algorithm of Fukunaga and Narendra (1975) was incorporated into the knn classifier. The resulting branch and bound knn (bbknn) classifier scales more efficiently with the number of samples in the knn training set.

3.1.2 The Hybrid GA/knn Classifier

The basis for the hybrid nearest neighbor classification technique is the weighted nearest neighbors classifier of Punch *et al.* (1993). As described in Section 2.4, a genetic algorithm is used to optimize a vector of weight values, $\{w_1, w_2, ...w_d\}$, where each weight is used as a linear scaling factor for one of the features considered by the classifier. The feature values are normalized over a common range prior to scaling. For the work described here the normalized feature values ranged from 1.0 to 10.0.

The formalism for linear feature extraction described in Section 2.2.3 states that the objective is a $d \times m$ transformation matrix, $\mathcal{H}$, where d is the number of original features and m is the number of transformed features. The GA-weighted knn classifier fits this framework, where the GA produces the diagonal elements of the transformation matrix, $\mathcal{H}$, and all the off-diagonal elements are zero. Figure 3.1 shows a schematic overview of a GA-based linear feature extractor in combination with a knn classifier. For the sake of efficiency, only the weight vector itself (that is, the diagonal elements of the transformation matrix, $\mathcal{H}$) need be passed to the knn. Since the value of k (the number of nearest neighbors to consider) is closely related to the scaling of the feature dimensions, the value of k was also included on the GA chromosome for co-optimization with the weight vector.

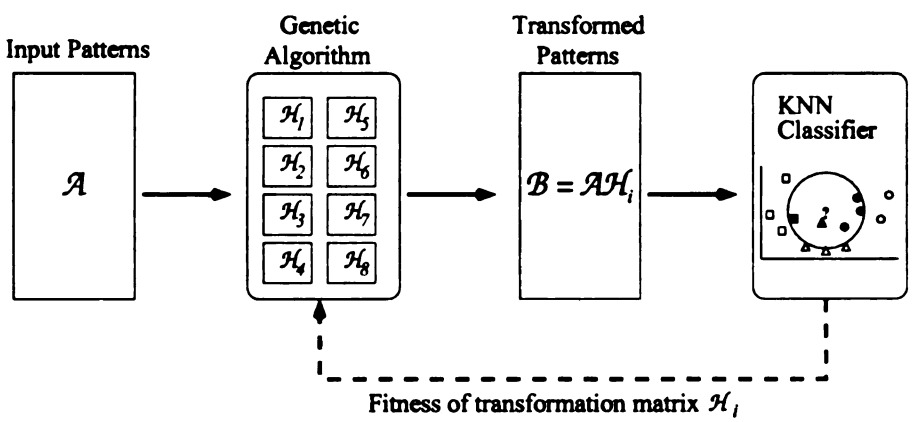


Figure 3.1: A GA-based feature extractor using an objective function based on classification accuracy. Each transformation matrix from the GA is used to transform the input patterns, which are then passed to a knn classifier. The fitness of a particular transformation matrix is based on the classification accuracy of the knn using the transformed patterns.

The C

As eac

(or in

the kr

seeks

in det

be co

Feat

Mea

The GA Cost Function

As each weight vector and k value is sent to the knn classifier for evaluation, a cost (or inverse fitness) score is computed, based primarily on the accuracy obtained by the knn in classifying a set of samples of known class. Since the genetic algorithm seeks to minimize the cost score, the formulation of the cost function is a key element in determining the quality of the resulting classifier. As such, several trade-offs must be considered in the design of the objective function:

Feature subset size versus classification accuracy — Up to a certain point, inclusion of meaningful features will generally increase the accuracy of a classifier. Thus, reducing the size of the feature set and obtaining classification accuracy are often mutually exclusive goals. The importance of feature set reduction relative to overall accuracy can be controlled through the fitness function.

Measures of classification accuracy — All types of classification errors do not necessarily incur equal costs. In a two-class system, for example, the cost of misclassification for one class might differ from the other. The most commonly employed measure of accuracy is the classification rate, usually defined as the number of correct classifications divided by the number of test samples. This simple metric may not suffice for problems with differing objectives. Consider a test set consisting of n samples, $\{\vec{x}_1, \vec{x}_2, ...\vec{x}_n\}$, belonging to c classes. For a given classifier let P_i be the number of patterns from the test set that were correctly predicted to belong to class i. Let U_i be the number of patterns falsely

predicted as class i (that is, predicted as class i, but observed to belong to class $j, i \neq j$). Similarly, let O_i be the number of patterns belonging to class i, but mispredicted as belonging to some other class; and let N_i be the number of patterns in the testing set neither observed nor predicted as belonging to class i. Using these four variables, we can establish several measures of classification accuracy that emphasize different aspects of the classification task.

- Classification rate: $Acc = \frac{\sum_{i=1}^{c} P_i}{n}$

  This common measure of accuracy emphasizes correct prediction of as many testing samples as possible, regardless of their class distribution. While overall accuracy is generally desirable, it can be a somewhat short-sighted measure of classifier quality when used alone. For example, for a two-class problem where the test set consists of many samples from class 1 and only a few from class 2, a good classification rate might be achieved by always predicting class 1. Class balance metrics can be combined with classification rate to mitigate this effect. Classification rate is also often expressed as error rate, $Err = 1.0 - Acc$.

- Average class accuracy: $ACA = \sum_{i=1}^{c} \left(\frac{P_i}{O_i + P_i} \right) / c$

  Average class accuracy helps to insure class balance by computing the classification rate for each class, and then taking the mean over all classes.

- Class balance: $Bal = \max_{i=1}^{c} \left(\frac{P_i}{O_i + P_i} \right) - \min_{j=1}^{c} \left(\frac{P_j}{O_j + P_j} \right)$

  This direct measure of class balance can be combined with the classification rate to penalize for class bias.

- Prediction-based Accuracy: $PBA(i) = \frac{P_i}{U_i + P_i}$

  Sometimes called "hit rate", PBA is defined on a per-class basis. For a given class, i, $PBA(i)$ is the proportion of all the predictions for class i that are correct. For example, if 100 samples are predicted to belong to class 1, and 85 of them are observed to actually belong to class one, then $PBA(1) = 0.85$. This measure explicitly penalizes for overprediction of each class, while ACA penalizes for underprediction. Average prediction-based accuracy ($APBA$) can be computed for the entire testing set: $APBA(i) = \sum_{i=1}^{c} \left(\frac{P_i}{U_i+P_i}\right) / c$

- Matthews coefficient (Matthews, 1975):

  $$C_m(i) = \frac{P_i * N_i - U_i * O_i}{(P_i + U_i) * (P_i + O_i) * (N_i + U_i) * (N_i + O_i)}$$

  Like PBA, the Matthews coefficient is defined on a per-class basis. The Matthews coefficients for each class can be averaged to produce an overall measure of accuracy and class balance. When the testing set is extremely unbalanced, C_m can have small values even when the accuracy for each class is high. C_m for a particular class is undefined when none of the training samples are predicted to belong to that class.

Objective function smoothness – For the knn classifier an objective function based largely on classification accuracy can have an unnecessarily rough, stepwise character. By smoothing this function we can provide more fine-grained feedback to the GA, enabling more efficient optimization. One way to achieve this is to consider the individual near neighbors and their classes. Consider a

two-class *knn* classifier with $k = 7$, and a weight vector that results in misclassification of a particular test sample. Further suppose that the test sample belongs to class two, but all the near neighbors of the test sample are of class one when scaled by the weight vector. If the weight vector undergoes mutation or crossover and the resulting set of near neighbors contains two members of class two, then the classifier is closer to a correct classification of this test sample, but the fitness value of the weight vector does not change because the sample is still misclassified. By adding a penalty for each incorrect near-neighbor "vote" to the cost function, we can reward an individual weight vector each time an additional vote is cast for the correct class. This additional feedback to the GA can guide the search toward new correct classifications, providing a more efficient search.

Coefficients are associated with each term in the GA cost function that allow control of each run. The following cost function is computed by the knn classifier for each individual (consisting of a weight vector and k value):

$$\begin{aligned} cost(\vec{w}, k) &= C_{acc} \times Err(\vec{w}, k) \\ &+ C_{pars} \times nonzero(\vec{w}) \\ &+ C_{vote} \times incorrect\_votes(\vec{w}, k) \\ &+ C_{bal} \times Bal(\vec{w}, k) \end{aligned} \tag{3.1}$$

Where Err is the error rate, as defined above; $nonzero(\vec{w})$ is the number of nonzero weights in the weight vector $\vec{w}$; $incorrect\_votes(\vec{w}, k)$ is the total number of incorrect near neighbor votes cast during classification with the weight vector $\vec{w}$ and neighbor count k; and Bal is the balance function, also defined above. Additionally, C_{acc}, C_{pars}[1], C_{vote}, and C_{bal} are coefficients for each of these terms, respectively. The coefficients determine the relative contribution of each part of the fitness function in guiding the GA search for the optimal weight vector and k value. The values for the cost function coefficients were determined empirically in a set of initial experiments for each data set. Typical values for these coefficients are given in Table 3.1. Deviations from these values for particular GA runs or data sets will be specifically noted in subsequent discussion.

Table 3.1: Typical values for the GA cost function coefficients.

Coefficient	Typical value
C_{acc}	20.0
C_{pars}	1.0
C_{vote}	2.0
C_{bal}	10.0

Representation Issues and Masking

The representation of the feature weights and k value on the chromosome is fairly direct—a 32 bit integer is used to represent each weight value. The resulting gene

[1]Pars is short for parsimony, indicating the importance of the *nonzero* term in obtaining a minimal feature set.

on

valu

is r

50.

the

elim

each

selv

Sinc

valu

value

featu

zero.

meth

porat

the fe

The v

above

mask

In the

each fe

on the GA chromosome yields an unsigned value over the range $[0, 2^{32} - 1]$. This value is then scaled by division to yield a real value over $[0.0, 100.0]$. The value of k is represented as a 6-bit unsigned integer, but is constrained to values between 1 and 50. For two-class problems, the k value is set to $k = (k_{chrom} * 2) + 1$, where k_{chrom} is the k-value from the GA chromosome. This constrains the value of k to odd integers, eliminating the need for a tie-breaking scheme in the knn classifier.

While the cost function encourages parsimony by penalizing a weight vector for each nonzero feature weight, a simple real-valued representation for the weights themselves does not provide an easy means for the GA to reduce feature weights to zero. Since the GA mutation operator tends to produce a small change in a single weight value, numerous mutations of the same feature weight are often required to yield a value at or near zero. Several methods were tested to aid the search for a minimal feature set, including reducing weight values below a predefined threshold value to zero, and including a penalty term in the cost function for higher weight values. The method that proved most effective, however, was a hybrid representation that incorporates both the GA feature selection technique of Siedlecki and Sklansky (1989) and the feature weighting techniques of Punch *et al.* (1993) and Kelly and Davis (1991). The weights and k value are represented directly on the chromosome, as described above. Additionally, a mask field is assigned to each feature. The contents of the mask field determine whether the feature is included in the classifier (see Figure 3.2). In the initial implementation, a single mask bit was stored on the chromosome for each feature. If the value of this bit was 1, then the feature was weighted and included

in classification. If, on the other hand, the mask bit for a feature was set to 0, then the feature weight was treated effectively as zero, eliminating the feature from consideration by the classifier. Since the masking fields comprised a very small section of the chromosome relative to the feature weights and k value, the number of mask bits associated with each feature was later increased to five. This increase had the effect of increasing the probability that a single bit mutation in a random location would affect the masking region of the chromosome. The interpretation of multiple mask bits is a simple generalization of the single bit case. When the majority of the mask bit values for a field are 1, then the field is weighted and included in classification. Otherwise, the field weight is reduced to 0, removing the feature from consideration by the knn. The number of mask bits is always odd so there is no possibility of a tie. Figure 3.2 shows a typical GA chromosome for the hybrid GA/knn classifier with masking.

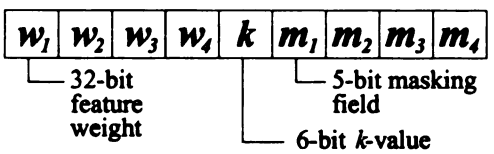


Figure 3.2: An example of a GA/knn chromosome with masking for a 4-dimensional feature set.

The use of a masking field on the chromosome allows a more efficient search for a minimal subset of features that provides good classification accuracy, while simultaneously searching for the optimal weights for the non-masked features. A single mutation can allow a feature to be tentatively included or removed from a

specific feature set without the loss of the partially-optimized weight value for that feature. Without masking, a weight value must be reduced to zero (or below a fixed threshold) to remove a feature from the current feature set. If the feature is later reintroduced, its weight value must be re-optimized from scratch.

GA Optimization Details

Several GA engines were employed to search for the optimal set of feature weights and k value, including the GAUCSD algorithm (Schraudolph and Grefenstette, 1992) and GALOPPs (Goodman, 1996). GA run parameters were determined empirically via a set of initial experiments. Table 3.2 summarizes the GA parameters for a typical run. Where specific runs have parameters that differ from these values in the subsequent discussion, this fact will be explicitly noted. Most runs used fitness proportionate selection, which requires an explicit scaling of the fitness of each individual in the GA population prior to selection. The specific scaling method used has a significant impact on the selection pressure applied by the GA during the subsequent search. For runs described here, the sigma scaling method of the GAUCSD algorithm was used. Several sigma scaling factors were tested, with the most common value being 3.0, as noted in the table. Further details on sigma scaling and the significance of the sigma scaling factor can be found in Schraudolph and Grefenstette (1992).

3.1

Evol

and

1991;

lem.

were a

to tha

optimi

ture, ar

for each

some we

k value

by GA-

dichotor

As n

valued f

Table 3.2: Typical values for the GA run parameters.

GA Run Parameter	Typcial value
Population size	200
Crossover rate	0.8 per individual
Mutation rate	0.001 per bit
Max number of generations	200
Sigma scaling factor	3.00

3.1.3 EP Optimization with the Knn Classifier

Evolutionary programming was also tested as a method for optimizing feature weights and k value for the knn classifier. An EP framework based on SGA-C (Smith *et al.*, 1991; Goldberg, 1989) was adapted to suit the feature selection and extraction problem. In order to facilitate efficient sampling of feature subsets, the usual EP operators were augmented with a recombination operator, and a hybrid representation similar to that of the masking GA/knn was used for the EP chromosome. As in the GA optimizer, the EP chromosome consisted of a single real-valued weight for each feature, an integer value for k, and a masking field consisting of one or more mask bits for each feature. Unlike the GA, however, the distinct regions of the EP chromosome were subjected to different operators. The real-valued feature weights and the k value underwent only Gaussian mutation, while the masking fields were modified by GA-style bitwise mutation and single-point crossover. Figure 3.3 illustrates the dichotomy of the chromosome used for EP/GA hybrid optimization.

As mentioned in Section 2.3.2, EP is commonly employed for the tuning of real-valued function parameters, but lacks a standard mechanism for dealing with binary

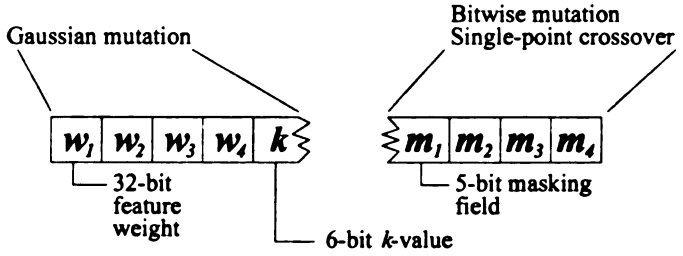


Figure 3.3: An example of an EP/GA hybrid chromosome with masking for a 4-dimensional feature set.

fields such as the masking region of the chromosome. The objective of the EP/GA hybrid approach was to explore the real-valued parameter optimization capability of the EP for tuning the feature weights, while exploiting the known ability of traditional GA operators to search binary alphabets (Holland, 1975; Goldberg, 1989) in optimizing the mask bits.

The EP framework employed, like most EP-based real-valued parameter optimizers, utilizes a mutation operator that varies each parameter according to a Gaussian distribution. The standard deviation for this distribution is an adaptive parameter in the sense that it is stored with the individual as a vector of σ-values, one for each parameter. Each parameter, x_i, is mutated to produce x_i' as follows:

$$x_i' = x_i + N(0, \sigma_i) \tag{3.2}$$

The standard deviations themselves are randomly perturbed according to a log-normal distribution to promote a diversity of individuals with differing optimization strategies within the population. Each generation, the vector of standard deviations

for each

where .

i^{th} elem

of para

parame

suggest

The

Table 3

Table 3
parame

3.1.4

For m

classif

for each individual is updated according to:

$$\sigma_i' = \sigma_i \cdot exp(\tau \cdot N(0,1) + \tau' \cdot N_i(0,1)) \tag{3.3}$$

where $N(\mu, \sigma)$ is a single normally distributed random variable, and $N_i(0,1)$ is the i^{th} element of a vector of standard normal deviates of a length equal to the number of parameters to mutate (Angeline, 1995). The values of τ and τ' are compile time parameters of the EP engine. Here, these parameter are set according to guidelines suggested by Bäck and Schwefel (1996).

The rest of the EP run parameters, like those of the GA, were tuned empirically. Table 3.3 shows the run parameters for a typical EP/knn experiment.

Table 3.3: Run parameters for a typical EP/knn experiment. d is the number of parameters to be optimized (i.e. the dimensionality of the problem).

EP Run Parameter	**Typical value**
Population size	200
Mask field crossover rate	0.8 per individual
Mask field mutation rate	0.001 per bit
Max number of generations	200
τ'	$\left(\sqrt{2d}\right)^{-1}$
τ	$\left(\sqrt{2\sqrt{d}}\right)^{-1}$

3.1.4 Bayes and KNN Classifiers

For many experiments, results of an unweighted knn classifier and a naïve Bayes classifier are presented for comparison. The naïve Bayes classifier, described in Sec-

tion 2.1.2, is nonparametric in the sense that the class conditional distributions of the feature values are estimated by constructing a histogram for each feature based on the training data. The bin size of this histogram is a run-time parameter of the classifier. Unless otherwise noted, 20 bins for each feature were used for the experiments described here. In addition, a Gaussian smoothing factor is applied in order to mitigate sampling anomalies that might introduce classification bias. Given a feature value, x_i, for feature i, and a class ω_j, then let $b_{\omega_j}(x_i)$ be the bin that x_i occupies in the histogram for class ω_j. When the Gaussian smoothing is applied, the effective marginal probability $p(x_i|\omega_j)$ depends on the histogram value of bin $b_{\omega_j}(x_i)$, as well as the histogram values of neighboring bins. Let $h_{\omega_j}(b_{\omega_j}(x_i))$ be the histogram value for bin $b_{\omega_j}(x_i)$—that is, the proportion of the training samples of class ω_j that have values for feature i that belong in the bin $b_{\omega_j}(x_i)$—then the effective marginal probability for feature value x_i is:

$$p(x_i|\omega_j) = \sum_{k=-\sigma}^{+\sigma} \Big(G(k,\sigma) \times h_{\omega_j}(b_{\omega_j}(x_i) + k) \Big) \tag{3.4}$$

where $G(k,\sigma)$ is the mass density function for the Gaussian distribution at $\mu = 0.0$, with variance σ^2:

$$G(k,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{3.5}$$

The value of σ, also a run-time parameter, determines the number of bins that will contribute to each effective marginal probability value. Figure 3.4 illustrates the effect of Gaussian smoothing on the effective marginal probability for a particular

feature value.

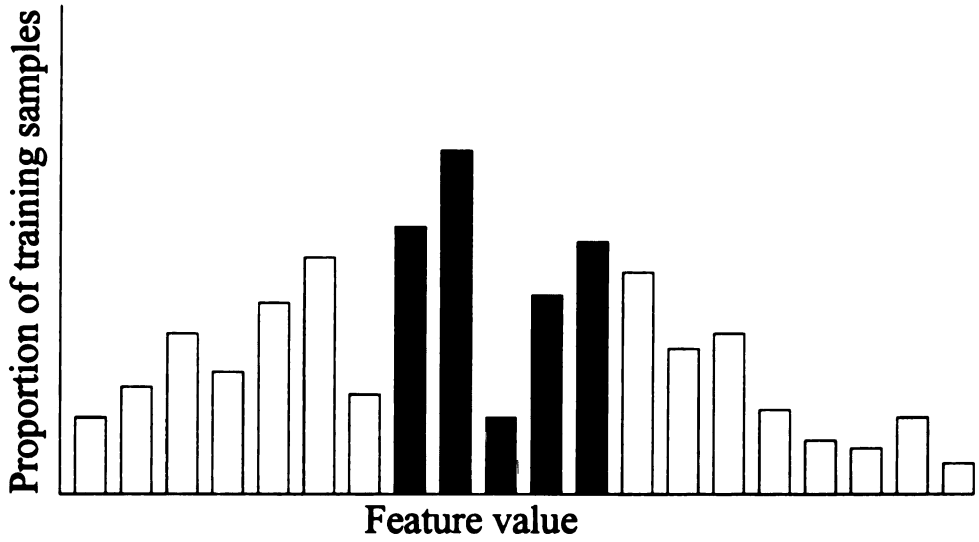


Figure 3.4: Effects of Gaussian smoothing on the computation of effective marginal probabilities. Assuming that the current feature value falls in the center bin (black rectangle), and assuming $\sigma = 2$, then the two surrounding bins on either side (grey rectangles) also contribute to the effective marginal probability for the current feature.

The unweighted knn classifier, also employed for comparison with the hybrid classification results, uses a Euclidean distance metric to find nearest neighbors. When there is a tie in determining the most frequent class among the near neighbors, the class of the nearest neighbor is used as a tie breaker.

3.1.5 Data Sets

Artificially Generated Data

The GA/knn and EP/knn hybrid classifiers were trained and tested against a variety of artificially constructed data sets to evaluate their error rates and feature selection

and extraction capability under a variety of classification conditions. These data sets were constructed with a varying number of features of the following types:

Univariate Gaussian: $G(\mu, \sigma)$ — A random Gaussian deviate selected from the distribution with mean μ and variance σ^2.

Uniform: $U(min, max)$ — A random real value selected from a uniform random distribution over the range $[min, max]$.

Uniform integer: $UI(min, max)$ — A random integer selected from a uniform random distribution over the range $[min, max]$.

Computed fields — Field values can be computed from the results of other field values in combination with basic arithmetic operations and/or any of the previously mentioned types of random variables. For example, a data field might consist of twice the value of the first data field, plus a uniform random value from 1 to 10: $2 * f1 + U(1, 10)$.

Based on these basic types of field values, a number of two-class data sets were generated for use in testing various aspects of the GA and EP-based feature selection and extraction methods in combination with the knn classifier. The specific data sets generated include:

Name:	$G_{10,1}$
Number of Features:	10
Class 1 specifier:	$G(5,2)$, $G(5,2)$, ...$G(5,2)$

Class 2 specifier: $G(10,2)$, $G(10,2)$, ...$G(10,2)$

This simple 10-dimensional data set consists of independent Gaussian distributed features. The two classes are generally linearly separable, and weighting of the features is not likely to improve classification accuracy. This data set serves as a baseline for classification performance.

Name: $G_{10,2}$

Number of Features: 10

Class 1 specifier: $G(7,2)$, $G(7,2)$, ...$G(7,2)$

Class 2 specifier: $G(10,2)$, $G(10,2)$, ...$G(10,2)$

This data set is similar in construction to $G_{10,1}$, but the means for the two classes are closer, and thus the data is slightly more overlapped. See Figure 3.5 for a comparison between $G_{10,1}$ and $G_{10,2}$.

Name: G_5

Number of Features: 5

Class 1 specifier: $G(5,2)$, $G(5,2)$, $G(5,2)$, $G(5,2)$, $G(5,2)$

Class 2 specifier: $G(10,2)$, $G(9,2)$, $G(8,2)$, $G(7,2)$, $G(6,2)$

This 5-dimensional data set again consists of independent features selected from Gaussian distributions. However, the distance between the class means is reduced by one for each feature after the first, so feature weighting is more likely to be useful for this data set. Furthermore, since inappropriate feature selection should now prove detrimental to classification accuracy, this is the first data set that will provide a test of the feature selection capability of each method.

Name: G_6

Number of Features: 6

Class 1 specifier: $G(5,2)$, $G(5,2)$, $G(5,2)$, $G(5,2)$, $G(5,2)$, $G(5,2)$

Class 2 specifier: $G(10,2)$, $G(9,2)$, $G(8,2)$, $G(7,2)$, $G(6,2)$, $G(5,2)$

This data set is identical to G_5, but an additional feature, feature 6, is included. The mean and variance of feature 6 are identical for both classes. This feature should provide no useful classification information and is included to further test the feature selection capability of each method.

Name: M_6

Number of Features: 6

Class 1 specifier: $G(5,2)$, $G(2,1)$, $G(5,3)$, $G(2,2)$, $U(1,5)$, $UI(1,5)$

Class 2 specifier: $G(7,2)$, $G(3,1)$, $G(7,3)$, $G(3,2)$, $U(3,7)$, $UI(3,7)$

This is a mixed data set consisting of features drawn from overlapping Gaussian and uniform distributions. Various levels of overlap between classes are exhibited by the different features. Both continuous and discrete uniformly-distributed features are included.

Name: N_5

Number of Features: 5

Class 1 specifier: $G(5,2)$, $G(5,2)+U(1,2)$, $G(5,2)+U(1,3)$, $G(5,2)+U(1,4)$, $G(5,2)+U(1,5)$

Class 2 specifier: $G(7,2)$, $G(7,2)+U(1,2)$, $G(7,2)+U(1,3)$, $G(7,2)+U(1,4)$, $G(7,2)+U(1,5)$

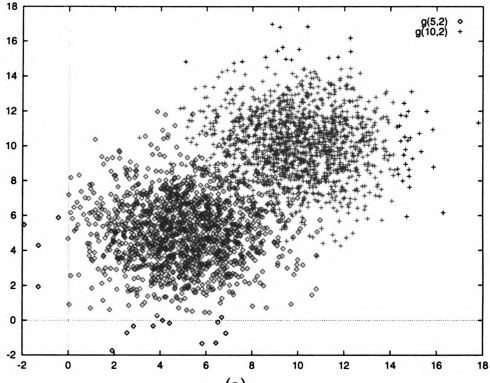
This data set is designed to evaluate the ability of each method to deal with increasing noise in each feature. The first feature consists of a Gaussian random deviate with moderately overlapped distribution between classes. Each subsequent feature is a similar Gaussian deviate, but with an increasing level of uniform random noise added.

Name:	N_{10}
Number of Features:	10
Class 1 specifier:	$G(5,2)$, $G(5,3)$, $G(5,4)$, $G(5,5)$, $G(5,6)$, $G(5,6)$, $G(5,6)$, $G(5,6)$, $G(5,6)$, $G(5,6)$
Class 2 specifier:	$G(7,2)$, $G(7,3)$, $G(7,4)$, $G(7,5)$, $G(7,6)$, $G(7,6)$, $G(7,6)$, $G(7,6)$, $G(7,6)$, $G(7,6)$

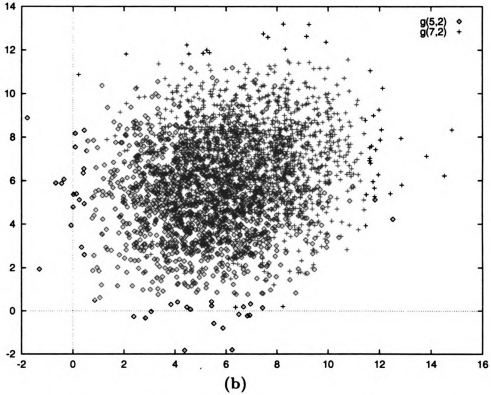
Another data set designed to test the ability of each method to deal with noisy features, this set includes a large number of features with extremely overlapped distribution between the two classes.

Medical, Biochemical and Other Data

Several data sets were selected from the UCI machine learning repository (Blake and Merz, 1998) in order to test the capability of the EC/knn classifier methods on real-world problems, and to facilitate comparison with other EC-based hybrid classification methods. The specific data sets tested were chosen to allow comparison with the GADistAI technique of Yang and Honavar (1998) and the EC feature construction and selection methods of Vafaie and De Jong (1998), and included the following:



(a)



(b)

Figure 3.5: A two-dimensional projection of the data sets $G_{10,1}$ **(a)**, and $G_{10,2}$ **(b)** onto the first two feature axes. $G_{10,2}$ exhibits much more overlap between classes than $G_{10,1}$.

Hepatitis – This data consists of 19 descriptive and clinical test result values for 155 hepatitis patients (Diaconis and Efron, 1983; Cestnik *et al.*, 1987). The two classes, survivors and patients for whom the hepatitis proved terminal, are strongly unbalanced—123 samples belong to the survivor class while 32 belong to the terminal class. The data includes qualitative, as well as both continuous and discrete-valued quantitative features. There are missing values, the number of which varies largely by feature. Many features have no missing values, while others have as many as 67 missing values out of 155 samples. The small sample size and incompleteness of this data set are typical of many medical classification problems.

Pima – Diabetes diagnosis information for native American women of the Pima heritage, aged 21 or over (Smith *et al.*, 1988). This data consists diagnostic information for 768 women; 268 of these patients tested positive for diabetes, while 500 tested negative. Six of the eight features are quantitative and continuous, consisting of various clinical test results. The remaining two features, age in years and number of times pregnant, are quantitative and discrete. There are no missing feature values in the data. The completeness and moderate dimensionality of this data set make it suitable for testing the ability of a classifier and feature extractor to maintain or increase classification accuracy while reducing dimensionality when there are fewer features to work with.

Wine – This data set consists of the results of a chemical analysis of wines derived from three different cultivars (Aeberhard *et al.*, 1992a,b). There are 13 continu-

ous features, with no missing values. There are 59, 71, and 48 members of each of the three classes, respectively. The three classes are nearly linearly separable, and linear discriminant analysis can obtain 98.9% accuracy over all three classes. This data set is thus better for evaluating feature selection capability than classifier accuracy.

Ionosphere – The 34 continuous features in this data set are derived from the signals read by a phased array of 16 high-frequency antennas in Goose Bay, Labrador (Sigillito *et al.*, 1989). These radar signals are designed to recognize structure in the ionosphere. Each reading consists of 17 pulses, with two attributes per pulse resulting in 34 features. There are 351 samples in this data set—225 are considered "good" readings, for which some structure in the ionosphere was detected, while 126 readings showed no structure. There are no missing feature values in this data. This data set was selected for evaluation of feature selection capability for higher-dimensionality data sets.

Two additional data sets, also selected from the UCI repository, were employed by Weiss and Kapouleas (1989, 1990) in a comparative study of classification methods from statistical pattern recognition, neural networks, and machine learning. These two medical data sets, **thyroid** and **appendicitis**, are included here to facilitate comparison with these results. The thyroid data consists of 21 clinical test results for a set of patients tested for thyroid dysfunction (Quinlan *et al.*, 1986)—15 of these features are binary-valued, while the other 6 are continuous. The training data consist of 3772 cases from the year 1985, while the testing data consist of 3428 cases from

the following year. The data are grouped into two classes, consisting of the patients that were/were not diagnosed as having certain categories of hypothyroid disorder. The two classes are highly unbalanced: the training data consist of 3487 negative diagnoses and 284 positive, while the testing data consist of 3177 negative samples and 250 positive. The appendicitis data consists of seven laboratory tests to confirm the diagnosis of acute appendicitis (Marchand *et al.*, 1983). All seven features are continuous. This data set consists of only 106 samples in two classes. 85 patients had confirmed appendicitis while 21 did not.

In addition, extensive experimentation was conducted on a large biochemical data set dealing with the binding of water molecules to protein surfaces. The features and biological significance of this data, as well as the classification and feature selection results for the GA/knn, EP/knn and other classifiers and feature selection and extraction methods are detailed in Chapter 5—Identifying the Determinants of Solvent Binding in Proteins.

3.1.6 Testing and Error Rate Estimation

Various techniques were used to estimate the error rate of the trained EC/knn hybrid classifier. For data sets from the UCI machine learning data repository (i.e. the hepatitis, Pima, wine, and ionosphere data sets) the testing methods were similar to those of Yang and Honavar (1998) in order to facilitate comparison with these results. Several minor differences were introduced to avoid problems with overfitting small training and testing sets. Two types of experiments were conducted on the

UCI data sets, depending on the number of training and testing samples available for each class in the data set. Where ample training and testing data (at least 100 samples) were available for each class, *hold-out tuning* experiments were conducted as follows. First, the available data are partitioned into three sets, training, tuning, and testing, with an equal number of samples of each class in each set. Thus, the size of these three sets is limited by the number of samples in the least numerous class. The first set (training) is used to populate the feature space for the knn classifier. Next, the EC is employed to optimize the k-value and feature weights. As previously described in Section 3.1.2, the EC cost function is based on classification accuracy for a set of samples of known class. The second set of samples (tuning) is used for this purpose. Finally, after the k-value and feature weights have been optimized, the resulting classifier is evaluated using the final data set (testing) to obtain an estimate of the error rate. This estimate is then averaged over 5 independent runs of the EC using the same partitioning into training, testing, and tuning sets. Finally, the entire process is repeated 10 times with 10 different partitionings of the data, resulting in an error rate estimate averaged over 50 EC experiments.

When the number of samples for one or more classes is extremely limited (less than 100 samples), this procedure is modified slightly. In this case, the data are partitioned into only two sets (training and testing), and *leave-one-out tuning* experiments are conducted. In these experiments, the training and tuning sets are identical. When evaluating the EC cost function, each sample is temporarily removed from the training set and evaluated as a tuning set of size one. The cost function is then computed

based on the average classification accuracy over all the training samples. After EC optimization of the feature weights and k-value, the error rate of the resulting classifier is estimated using the testing set, exactly as in the *hold-out tuning* experiments. Again, 5 independent EC runs are performed for each of 10 partitionings of the data, and the error estimates are averaged over all 50 experiments.

This method was simplified for the artificially generated data sets, since the number of potential independent samples is infinite for this data. Here, experiments were conducted for each classifier using large, independent training, tuning, and testing sets (1000 samples each). For GA and EP experiments, 5 independent experiments were conducted and the accuracy for the best result is reported.

Bootstrap Testing

For other data sets, including the appendicitis, thyroid, and protein–water binding data, a variant of the bootstrap test method (Jain *et al.*, 1987; Efron, 1979, 1982) is employed in order to obtain both an error rate estimate and a simple measure of the variance of this estimate. Traditional bootstrap measures require that a classifier be retrained and tested on a number of bootstrap testing sets selected with replacement from the available data (Jain *et al.*, 1987). Since the training of the hybrid EC/knn classifier is an iterative, computationally-intensive, offline process, a modified bootstrap method was employed to obtain an error estimate for the trained and tuned hybrid classifier.

As with the previously described methods, the available data are partitioned into

three sets, training, tuning, and testing. Within each of these sets, the number of samples of each class is equal. The training and tuning sets are used during each EC experiment in the same manner described above: the training set is used to populate the knn feature space, while the tuning set is used to evaluate a particular set of feature weights and k-value in terms of classification error, providing feedback to the EC optimizer. Once the feature weights and k-value are optimized, the testing set is randomly sampled with replacement to form n bootstrap test sets. The accuracy of the classifier (or any other performance measure to be estimated) is evaluated on each of these bootstrap test sets using the final weight set and k value produced by the EC experiment. Finally, the mean and standard deviation of the error rates for all n external test sets are computed and used as an estimator of the true error rate for the optimized classifier.

During the bootstrap tests, the same data set is used for training the knn classifier as during the EC experiment. Since the feature weights, k value, and training data cooperatively define the decision boundary for the final classifier, it would place an undue negative bias on the error rate estimation procedure to test the final classifier using a knn training set other than the one used during the EC experiment that produced that classifier. Figure 3.6 shows a schematic of the data sets used for training, tuning, and testing during a single EC experiment using the bootstrap-based testing method.

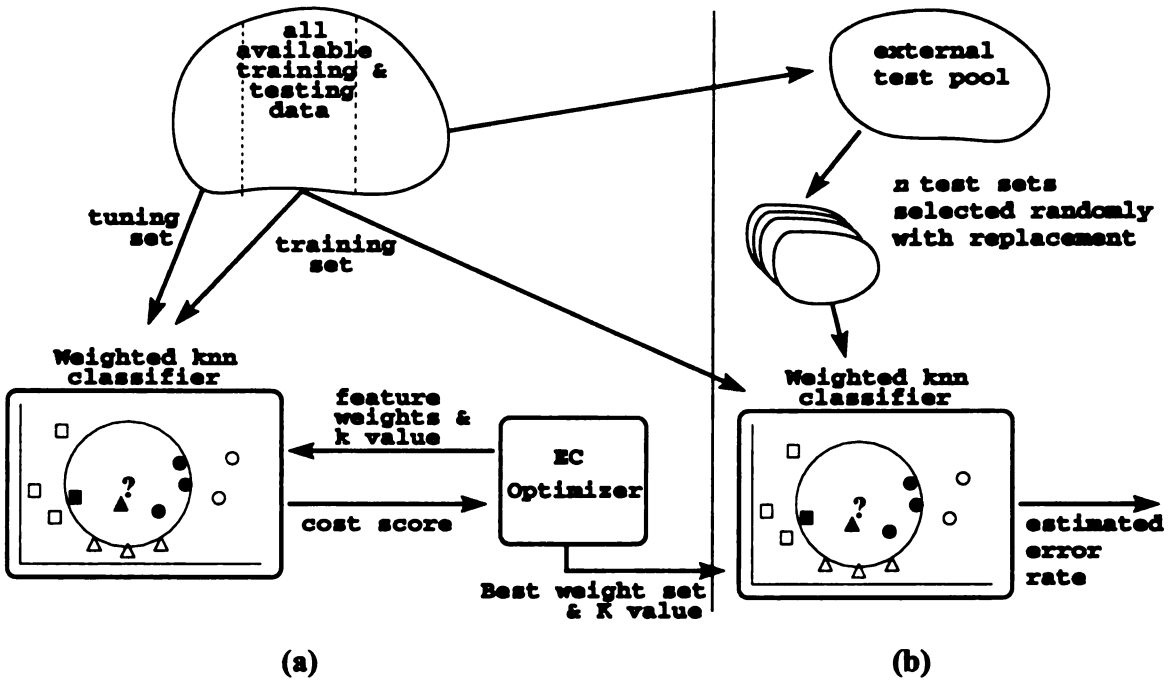


Figure 3.6: Division of training, testing, and tuning data during a single EC experiment (a), and error rate estimation of the tuned classifier (b). The cost score in (a) is based on the performance of the knn on the tuning set for a particular weight set and k value. The same knn training data is used for the entire EC experiment and during external testing.

3.2 Results

3.2.1 Classification of Artificial Data Sets

Independent Gaussian Features

Three of the artificially generated data sets were constructed entirely of features drawn from independent Gaussian distributions, $G_{10,1}$, $G_{10,2}$, and G_5. The set $G_{10,1}$ is linearly separable, and a naïve Bayes classifier is able to achieve 100% prediction accuracy using all 10 features. The set $G_{10,2}$ is not as easily classified, but like $G_{10,1}$, the features are all drawn from identical independent distributions, so there is no feature weighting that can obtain better performance than that of the unweighted classifiers. Like $G_{10,1}$ and $G_{10,2}$, the data set G_5 is not difficult to classify—a naïve Bayes classifier can discriminate between the two classes with $\sim 96\%$ accuracy—but the features are not drawn from identical distributions, so feature weighting can potentially have a positive effect on classification accuracy. For these data, the primary goal is to reduce the number of features used, while reducing classification accuracy as little as possible. For comparison, a naïve Bayes classifier was trained and tested on each data set, as well as a knn classifier using odd values for k from 1 to 100. As described in Section 3.1.6, 5 independent experiments were conducted for each EC-based method, and the best k value and weight set were then tested on independent data. These results are summarized in Table 3.4.

The results for $G_{10,1}$ and $G_{10,2}$ verify the ability of the GA and the EP to perform feature selection. Since all the features in these two data sets are drawn from identical

Table 3.4: Apparent accuracy and bootstrap accuracy rates for the Bayes classifier, the knn classifier, and the EC/knn hybrid classifiers for artificially generated data sets consisting of independent, Gaussian distributed features. For the Bayes classifier, accuracy is shown for re-classifying the training data (**Train**), and for classifying the independent testing data (**Test**). For the knn classifier, experiments were run using odd values of k ranging from 3 to 101 using the same independent training and tuning set for all experiments. The best accuracy on the tuning set (**Tune**) and the corresponding k value are reported. The best k value was then evaluated by reclassifying the training data (**Train**), and by classifying a new, independent test set (**Test**). For the EC/knn hybrid classifiers, five EC experiments were conducted using the same data sets but differing initial random starting conditions, and the best result is reported. The k value for the best result, the accuracy on reclassification of the knn training data (**Train**), the accuracy when reclassifying the EC tuning data (**Tune**), and the accuracy when classifying an independent test set (**Test**) are provided along with the number of non-zero feature weights (**Features**).

$G_{10,1}$	k	Train	Tune	Test	Features
Bayes	—	100	—	100	10
Knn	3	100	100	100	10
GA/knn	3	98.9	100	99.4	4
EP/knn	3	100	100	100	4

$G_{10,2}$	k	Train	Tune	Test	Features
Bayes	—	92.9	—	91.5	10
Knn	67	94.1	94.2	93.3	10
GA/knn	25	94.2	94.8	93.1	10
EP/knn	19	94.6	93.8	93.1	10

G_5	k	Train	Tune	Test	Features
Bayes	—	96.0	—	95.6	5
Knn	79	95.8	97.7	96.5	5
GA/knn	33	94.7	97.2	95.7	3
EP/knn	35	95.8	97.2	96.2	3

distributions, the feature weights produced by the hybrid classifiers are valueless. For $G_{10,1}$, the class conditional distributions of feature values are well enough separated that 6 of the 10 features can be removed without a significant adverse effect on classification accuracy. For $G_{10,2}$, the EC optimizers were unable to remove any features while maintaining maximal classification performance. Results for G_5 are slightly more interesting. Since the distributions for the two classes have different degrees of overlap for each feature, the quality of the feature weighting provided by the EC optimizers should have a bearing on the classification accuracy. For this data, both the GA and the EC were able to remove 2 of 5 features while maintaining classification accuracies near to that of the knn using all 5 features. In all cases, the EP/knn was able to obtain slightly better overall accuracy than the GA/knn, probably due to the EP's ability to fine tune the final feature weights during the later phases of the optimization run.

Mixed Multivariate Data

Several of the artificially-constructed data sets contained a mixture of Gaussian and uniformly distributed random feature values. As described in Section 3.1.5, M_6 consists of Gaussian deviates, discrete uniform random values, and continuous uniform random values. This data set provides a test of the hybrid classifiers' ability to maintain recognition accuracy through feature weighting while reducing the number of features to consider. The set N_5 consists of five features with increasing levels of random noise, and is designed to test the feature selection capability of the two EC

methods, and the effect of feature selection on classification accuracy. The test methods for these data sets were identical to those described for the previous artificially constructed data sets. The results are summarized in Table 3.5.

Table 3.5: Apparent accuracy and bootstrap accuracy rates for the Bayes classifier, the knn classifier, and the EC/knn hybrid classifiers for artificially generated data sets consisting of features with a mixture of random distributions. For the Bayes classifier, accuracy is shown for re-classifying the training data (**Train**), and for classifying the independent testing data (**Test**). For the knn classifier, experiments were run using odd values of k ranging from 3 to 101 using the same independent training and tuning set for all experiments. The best accuracy on the tuning set (**Tune**) and the corresponding k value are reported. The best k value was then evaluated by reclassifying the training data (**Train**), and by classifying a new, independent test set (**Test**). For the EC/knn hybrid classifiers, five EC experiments were conducted using the same data sets but differing initial random starting conditions, and the best result is reported. The k value for the best result, the accuracy on reclassification of the knn training data (**Train**), the accuracy when reclassifying the EC tuning data (**Tune**), and the accuracy when classifying an independent test set (**Test**) are provided along with the number of non-zero feature weights (**Features**).

M_6	k	Train	Tune	Test	Features
Bayes	—	89.5	—	87.7	6
Knn	29	91.6	92.4	91.2	6
GA/knn	19	92.4	93.6	90.8	5
EP/knn	25	90.8	92.2	89.0	4

N_5	k	Train	Tune	Test	Features
Bayes	—	94.4	—	81.6	5
Knn	9	86.5	86.8	82.8	5
GA/knn	11	85.1	86.2	80.9	4
EP/knn	7	85.6	85.8	79.3	4

Again, the hybrid methods were able to reduce the feature set size by one or more features with a minimal reduction in classification accuracy for these data sets. For all data sets discussed thus far, the EP and GA classifiers have produced similar values

for k (having a difference of 6 or less).

3.2.2 Classification of Data from the UCI Repository

Four data sets—**Pima**, **wine**, **ionosphere**, and **hepatitis**—were selected for comparison with the GADistAI iterative feature selection and construction method of Yang and Honavar (1998). As described previously, the testing and error rate estimation for these data sets were performed in a manner similar to that of Yang and Honavar to facilitate direct comparison. Classification and feature selection results for this data, averaged over 50 EC/knn experiments, are summarized in Table 3.6. Unfortunately, the error rates reported for GADistAI were obtained using the same data set used by the GA to tune feature subsets and weights. This is essentially equivalent to the "Train/Tune" accuracy provided in Table 3.6 for the EC-hybrid classifiers. Since no external or bootstrap testing was done, it remains to be seen how the GADistAI algorithm generalizes to new data, and no comparison to the "Test" accuracy obtained by the EC/knn hybrid classifiers can be made.

In all four data sets the bootstrap test accuracy of the EC-hybrid classifiers are comparable to the best accuracy achieved by any classifier. For **wine** and **pima**, the best test results were, in fact, obtained by the EC/knn hybrids. Additionally, both hybrid classifiers were able to achieve a significant reduction in the number of features considered for all four data sets. The GA/knn hybrid classifier used the lowest mean number of features in classification of all the feature selection and classification methods presented. For the four data sets examined here, the GA/knn consistently

Table 3.6: Results of the hybrid EC/knn classifiers and the GADistAI algorithm on various data sets from the UCI Machine Learning data set repository, averaged over 50 runs. **Train/Tune** refers to the accuracy obtained when reclassifying the data used by the EC in tuning (optimizing) feature subsets and weights. **Test** refers to the accuracy obtained on an independent test set for each experiment, disjoint from the training and tuning sets. **Features** is the number of features with nonzero weights in the best performing weight set for each run; the mean value over all 50 runs is shown.

Hepatitis	**Train/Tune**	**Test**	**Features**
Bayes	85.3	65.7	19
Knn	87.1	73.4	19
GADistAI	97.1	—	9.2
GA/knn	86.0	69.6	8.1
EP/knn	87.2	73.1	8.9

Wine	**Train/Tune**	**Test**	**Features**
Bayes	98.8	94.7	13
Knn	94.9	94.3	13
GADistAI	99.4	—	6.7
GA/knn	99.7	94.8	6.0
EP/knn	99.5	93.2	6.2

Ionosphere	**Train/Tune**	**Test**	**Features**
Bayes	93.0	90.1	34
Knn	83.4	93.2	34
GADistAI	98.6	—	17.3
GA/knn	95.0	91.9	8.5
EP/knn	93.2	92.3	13.5

Pima	**Train/Tune**	**Test**	**Features**
Bayes	76.1	64.6	8
Knn	73.5	71.5	8
GADistAI	79.5	—	3.8
GA/knn	80.0	72.1	3.1
EP/knn	79.1	72.9	3.9

reduced the number of features considered to roughly one-half the original number of features.

Another important factor to consider when comparing the results of the EC/knn hybrid classifiers to classical methods and other feature selection and classification techniques is the balance between the prediction accuracies among the various classes. That is, depending on the class distribution of the training data, some classifiers can perform very well on some classes at the expense of others. The cost function that drives the optimization process for the EC/knn hybrid classifiers penalizes for disparity in the prediction accuracies among classes. In some cases, this will cause the EC to reject weight sets that lead to higher overall prediction accuracy in favor of those that lead to a better balance in prediction accuracy among classes. Table 3.7 summarizes the balance in predictive accuracy, as defined in Section 3.1.2, of the various classifiers on the four UCI data sets. The EC/knn hybrid classifiers obtained the best balance in all cases when reclassifying the GA tuning data. For the **Hepatitis** and **Pima** data sets, the GA/knn classifier was able to achieve significantly better balance than the traditional classifiers. The drop in balance between the tuning data and the bootstrap testing data for the **Wine** data set may indicate that some overfitting of the tuning data has occurred. No information regarding balance in predictive accuracy was provided for GADistAI.

Table 3.7: Balance in predictive accuracy among classes for various classifiers. The maximum accuracy, (**Max**), and the minimum accuracy, (**Min**), among the various classes are shown for each classifier, along with the difference between the two values (**Bal**). Lower values for **Bal**, indicating a smaller difference between the minimum and maximum predictive accuracies, are preferred. **Train/Test** results refer to accuracies for reclassifying the training data for the Bayes and Knn classifier, and for reclassifying the EC tuning data set for the GA/knn and EP/knn classifiers. **Test** refers to the accuracy when classifying an independent test set. The knn classifier was tested for odd values of k from 1 to 101, and the best results are shown.

	Train/Tune			Test		
Hepatitis	Min	Max	Bal	Min	Max	Bal
KNN	34.5	97.6	63.1	30.8	95.7	64.8
Bayes	17.5	98.8	81.3	6.7	96.5	89.9
GA/knn	67.3	89.7	22.5	45.4	82.2	36.8
EP/knn	81.9	88.3	6.4	48.7	85.9	37.2

	Train/Tune			Test		
Pima	Min	Max	Bal	Min	Max	Bal
KNN	68.5	78.4	9.9	62.8	73.4	10.6
Bayes	68.1	84.1	16.0	62.0	75.7	13.7
GA/knn	79.8	80.1	0.3	72.1	72.4	0.3
EP/knn	78.9	79.3	0.4	72.5	74.6	2.2

	Train/Tune			Test		
Ionosphere	Min	Max	Bal	Min	Max	Bal
KNN	68.8	97.9	29.1	72.3	97.5	25.2
Bayes	92.4	93.6	1.2	88.6	96.9	8.3
GA/knn	94.6	95.4	0.8	80.7	94.2	13.5
EP/knn	91.5	94.9	3.4	83.6	94.1	10.5

	Train/Tune			Test		
Wine	Min	Max	Bal	Min	Max	Bal
KNN	84.8	100.0	15.3	89.4	100.0	10.6
Bayes	97.8	98.8	1.0	93.2	94.8	1.7
GA/knn	99.3	100.0	0.7	90.6	99.4	8.8
EP/knn	99.1	99.8	0.7	88.3	98.6	10.4

3.2.3 Classification of Medical Data

The **thyroid** and **appendicitis** data sets were selected for comparison with a broader study encompassing several traditional classification methods (Weiss and Kapouleas, 1989, 1990), including the linear and quadratic discriminant functions, the 1-nearest neighbor classifier, a neural network classifier trained using the backpropagation method, the CART tree classifier (Breiman *et al.*, 1984), the naïve Bayes classifier, and a 2nd order Bayes classifier. Error rates for these data sets were evaluated using the bootstrap testing method described in Section 3.1.6. Five independent experiments were conducted, and the best weight set and k value for each experiment were subjected to 100 bootstrap tests. Weiss and Kapouleas report two accuracy estimates for each classifier. The first, training accuracy, is the accuracy obtained when reclassifying the training samples. Since training of the EC/knn hybrid classifiers is a two step process, Weiss' training accuracy is best compared to the accuracy obtained by the hybrid classifier when reclassifying the tuning set. Weiss also reports a less biased estimate of accuracy based on an independent testing set, this measure is best compared to the bootstrap accuracy obtained by the hybrid classifiers. Tables 3.8 and 3.9 compare the results reported by Weiss and Kapouleas with those of the GA/knn hybrid classifier on the **thyroid** and **appendicitis** data sets, respectively.

Comparison with SFFS

The sequential floating forward selection (SFFS) algorithm of Pudil *et al.* (1994) has been shown to be a powerful technique for feature subset selection (Jain and Zongker,

Table 3.8: Results of various classifiers on hypothyroid data, as reported by Weiss, in comparison with that of the EC/knn hybrid classifiers. Train/tune refers to the accuracy obtained when reclassifying the training data, in the case of Weiss' results, or the tuning data, in the case of the EC/knn hybrid classifiers. Testing refers to the accuracy obtained on an independent test set for Weiss, and to the bootstrap accuracy estimate for the hybrid classifiers.

Method	Accuracy (train/tune)	Accuracy (testing)
GA/knn	98.5%	98.4%
Linear Discriminant	93.8%	93.8%
Quadratic Discriminant	89.7%	88.4%
Nearest Neighbor	100%	95.3%
Bayes (independent)	97.1%	96.1%
Bayes (2nd order)	97.7%	92.4%
Neural Net (Back prop)	99.5%	98.5%
Predictive Value Max.	99.8%	99.3%
CART Tree	99.8%	99.4%

Table 3.9: Results of various classifiers on the appendicitis data, as reported by Weiss, in comparison with that of the EC/knn hybrid classifiers. As in the previous table, rain/tune refers to the accuracy obtained when reclassifying the training data, in the case of Weiss' results, or the tuning data, in the case of the EC/knn hybrid classifiers. Testing again refers to the accuracy obtained on an independent test set for Weiss, and to the bootstrap accuracy estimate for the hybrid classifiers.

Method	Accuracy (train/tune)	Accuracy (testing)
GA/knn	90.4%	90.6%
Linear Discriminant	88.7%	86.8%
Quadratic Discriminant	79.3%	73.6%
Nearest Neighbor	100%	82.1%
Bayes (independent)	88.7%	83.0%
Bayes (2nd order)	95.3%	81.1%
Neural Net (Back prop)	90.0%	85.8%
Predictive Value Max.	91.5%	89.6%
CART Tree	90.0%	84.9%

1997). For comparison with the EC feature selection and extraction methods, SFFS was tested in conjunction with a knn classifier using odd values of k between 1 and 101 for the two medical data sets. For the thyroid data, the sequential floating forward selection method achieved good classification results. The best accuracy obtained by the SFFS/knn algorithm during feature selection was 97.99%, using 6 of the 21 available features. A mean bootstrap accuracy of 98.06%, with a standard deviation of 0.6032%, was obtained over 100 bootstrap tests on this feature set. This accuracy is similar to those obtained by the various methods reported by Weiss. The GA feature extractor combined with a knn classifier obtained a similar accuracy, 98.48%, using only 3 of the available 21 features, and a k-value of 87. 100 bootstrap tests for this set of feature weights yielded a mean bootstrap accuracy of 98.40%, with a standard deviation of 0.6256%.

For the appendicitis data, the best result was obtained for $k = 7$. The best predictive accuracy during selection was 88.46% using 3 of the 7 available features. Bootstrap testing for 100 trials using the best feature set found by SFFS yielded a mean predictive accuracy of 91.44% with a standard deviation of 3.94%. The GA feature extractor achieved a slightly higher accuracy than SFFS during extraction: 90.38% using 2 of 7 weighted features and $k = 7$. In bootstrap testing, however, the mean bootstrap accuracy over 100 trials proved to be similar to that of SFFS—90.60% with a standard deviation of 4.21%.

3.2.4 Classification of Protein Solvation Sites

The most extensive set of experiments was conducted on a sizeable set of biochemical data. These data, comprising the physical and chemical environments of 5542 protein-bound water molecules, were examined in light of two distinct objectives. The first goal of these experiments was to obtain good pattern recognition performance on various classes of water molecules, while the second objective was to identify the specific features which were consistently associated with membership in each class. The methods and results for these experiments are discussed in detail in Chapter 5—Identifying the Determinants of Solvent Binding in Proteins.

3.3 Discussion

In all of the data sets examined, the EC/knn hybrid classifiers exhibited an ability to reduce the number of features considered by the classifier significantly, while maintaining or slightly improving predictive accuracy, as compared to a non-weighted knn classifier using all available features. The overall predictive accuracy of the EC/knn classifiers was comparable to those of the best classifier tested for each data set examined, and in many cases the best accuracy was achieved by one of the EC/knn hybrids.

In terms of feature selection, the EC feature selection and extraction techniques equaled or exceeded the ability of other methods examined to reduce the number of features considered by the classifier, while maintaining similar or improved classifica-

tion accuracy. For the medical data sets, the EC methods were able to meet or exceed the accuracy obtained using a knn classifier in conjunction with SFFS (Pudil *et al.*, 1994), which is generally considered the most effective of the commonly-used feature selection methods (Jain and Zongker, 1997), while further reducing the number of features considered.

In addition, the parameterized nature of the cost function which drives EC optimization allows the EC methods the ability to consider factors other than the apparent error rate in tuning the feature set and weights. One such factor is the balance in predictive accuracy among classes. On many data sets where the classical methods result in strong disparity of predictive accuracy among classes, the EC-hybrid methods are able to obtain better balance while maintaining overall accuracy.

Chapter 4

Variations on the Bayes Classifier using Evolutionary Computation-Based Learning

4.1 Methods

4.1.1 Bayesian Discriminant Functions

The Bayesian classifier has a computational advantage over the knn classifier in that the training data are summarized, rather than stored. The comparison of each test sample with every known training sample to find nearest neighbors during knn classification is a computationally expensive process, even when efficient search methods are employed (Fukunaga and Narendra, 1975). In contrast, finding the marginal probability associated with a particular feature value is computationally efficient for both

the parametric and nonparametric forms of the Bayesian classifier. Since EC-based hybrid classifiers require many classifications to be performed during feature selection and extraction,[1] the use of a computationally efficient classifier such as a Bayes classifier is expedient.

Unfortunately, the direct application of the Bayes classifier to the problem is not effective, because the Bayes decision rule is invariant to linear scaling of the feature space. In other words, multiplying the feature values for a given feature by a constant has no effect on the class-conditional probabilities considered by the classifier, as illustrated in Figure 4.1. Direct scaling of the marginal probabilities is also ineffective for the naïve Bayes classifier, since the joint class-conditional probabilities are simply the products of the marginal probability values.

There are, nevertheless, several aspects of the Bayesian classifier that, when optimized, can yield better classification performance. One such area is is the manner in which the marginal probabilities for each feature will be combined into the multivariate class-conditional probability densities. As noted previously (see Section 2.1.2, the most common approach is to assume that all features are independent. For the resulting naïve Bayes classifier, the class-conditional probability is the product of the marginal probabilities for each feature. A more general approach would be to encode the entire $d \times d$ covariance matrix describing the interrelationships between

[1]The actual number of classifications to be performed is typically $O(npt)$ where n is the number of generations for the EC experiment, p is the number of individual solutions in the EC population, and t is the number of samples in the EC tuning set. If all-neighbors search is conducted for each classification, the number of distance comparisons is $O(npts)$, where s is the number of samples in the knn training set.

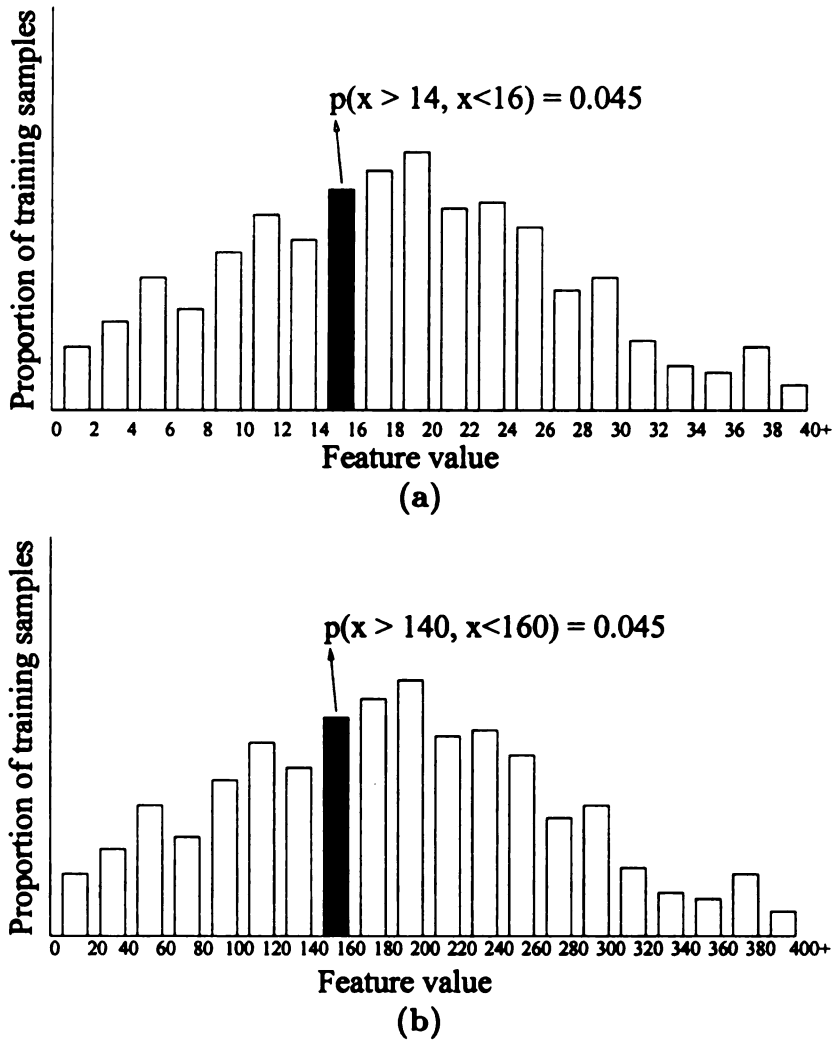


Figure 4.1: The Bayes decision rule is invariant to linear transformations of the feature space. For the feature shown here, the raw feature values (**a**) have been multiplied by 10 in (**b**). Using a nonparametric Bayes classifier, we find that the original feature value falls in the bin 14–16 (black rectangle) in the original histogram. The scaled feature falls in the equivalent bin of histogram **b**, and the histogram values (marginal probabilities) of the two bins are identical, so the scaling has no bearing on the classification results.

all the features being considered, and allow an EC-based optimizer to search for the covariance matrix which best describes the true multivariate distribution of the training data. Unfortunately, the search space involved in finding this covariance matrix grows as 2^{d^2}, even if the elements of the covariance matrix are binary-valued. For real valued matrix elements, the search space quickly becomes intractable, even for small problems.

We can simplify the problem somewhat by looking at the Bayesian classifier as a discriminant function. As outlined in Section 2.1.2, the Bayes decision rule can be written as follows:

$$\text{given } \vec{x}, \text{ decide } \omega_i \text{ if}$$

$$P(\omega_i|\vec{x}) \quad > P(\omega_j|\vec{x}) \quad \forall j \tag{4.1}$$

This rule can be rewritten as a *discriminant function*—a function g of the feature vector $\vec{x}$. Consider, by way of example, a two-class decision problem. The Bayes discriminant function can be written as:

$$g(\vec{x}) = P(\omega_1|\vec{x}) - P(\omega_2|\vec{x}) \tag{4.2}$$

Here, we would decide class 1 if $g(\vec{x}) > 0$, and class 2 if $g(\vec{x}) < 0$. The classification when $g(\vec{x}) = 0$ is arbitrary. The discriminant function, then, is uniquely associated with a particular classifier, mapping an input feature vector to a value associated with a particular class. According to Duda and Hart:

> We can always multiply the discriminant functions by a positive constant or bias them by an additive constant without influencing the decision. More generally, if we replace every $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$, where f is a monotonically increasing function, the resulting classification is unchanged. (Duda and Hart, 1973, pp. 17–18).

Thus, we can design a parameterized classifier based on the concept of the discriminant function. We begin with a discriminant function based on the Bayes decision rule. Using this function as a model, we can design similar functions which classify well, but are more easily parameterizable for hybridization with EC optimization methods. After designing such a discriminant function and identifying the tunable parameters, we can use an EC to optimize these parameters with regard to a particular set of training and tuning data.

Two distinct techniques were designed and tested using this method. The first is a simple nonlinear weighting of the Bayes discriminant function, while the second is a novel discriminant function based on the summation of the class-conditional marginal probabilities.

4.1.2 Nonlinear Weighting of the Bayes Discriminant Function

Consider the Bayes discriminant function,

$$\begin{aligned} g(\vec{x}) &= P(\omega_1|\vec{x}) - P(\omega_2|\vec{x}) \\ &= \frac{P(\vec{x}|\omega_1) \times P(\omega_1) - P(\vec{x}|\omega_2) \times P(\omega_2)}{\sum_{i=1}^{2} P(\vec{x}|\omega_i) \times P(\omega_i)} \end{aligned} \tag{4.3}$$

The denominator can be eliminated, since it does not affect the sign of $g(\vec{x})$, and thus does not affect the resulting classification. Since $a > b \Rightarrow \log(a) > \log(b)$, we can apply the log function to the *a posteriori* probabilities without changing the resulting classification. Thus, the following discriminant function is equivalent to the naïve Bayes discriminant:

$$g(\vec{x}) = \log\left(P(\vec{x}|\omega_1) \times P(\omega_1)\right) - \log\left(P(\vec{x}|\omega_2) \times P(\omega_2)\right) \tag{4.4}$$

$$\begin{aligned} &= \left(\log\left(P(\vec{x}|\omega_1)\right) + \log\left(P(\omega_1)\right)\right) \\ &\quad - \left(\log\left(P(\vec{x}|\omega_2)\right) + \log\left(P(\omega_2)\right)\right) \end{aligned} \tag{4.5}$$

where

$$\log\left(P(\vec{x}|\omega_i)\right) = \log\left(P(x_1|\omega_i)\right) + \log\left(P(x_2|\omega_i)\right) + \ldots + \log\left(P(x_d|\omega_i)\right) \tag{4.6}$$

Finally, we can parameterize this discriminant function, while maintaining a similar level of classification accuracy, by adding coefficients to each of the marginal proba-

bilities.

$$\begin{aligned} P^*(\vec{x}|\omega_i) &= C_1 \log\left(P(x_1|\omega_i)\right) + C_2 \log\left(P(x_2|\omega_i)\right) + ... \\ &+ C_d \log\left(P(x_d|\omega_i)\right) + \log\left(P(\omega_i)\right) \end{aligned} \tag{4.7}$$

The values for the coefficients, $C_{1...d}$, are supplied by an EC optimizer. The effect of these coefficients is to apply a nonlinear weighting to each of the marginal probabilities, which are then combined to produce a confidence value, P^*, for each class. While $P^*(\vec{x}|\omega_i)$ is no longer a joint probability distribution, the discriminant function is equivalent to the naïve Bayes discriminant function when $C_1 = C_2 = ... = C_d = 1$. Furthermore, the new function has several desirable features for hybridization with an EC optimizer. When a particular coefficient, C_j, is reduced to zero, the associated feature value, x_j, is effectively eliminated from consideration by the classifier. This allows us to perform feature selection in conjunction with classifier tuning. Furthermore, when the value of a coefficient, C_j is increased, the marginal probability value for the associated feature, x_j, has an increased influence on the value of the confidence value, $P^*(\vec{x}|\omega_i)$, for each class.

GA Optimization of the Nonlinear Discriminant Coefficients

The implementation for this discriminant function was based on the previously described nonparametric naïve Bayes classifier. The marginal probability distributions for each feature were approximated using histograms with 20 bins each. Gaussian smoothing was used to mitigate sampling anomalies, as described earlier, with

$\sigma = 2.0$. The GA cost function was identical to that described in section 3.1.2, with the exception that there is no voting involved in classification using the discriminant function classifier, so there was no corresponding *incorrect_votes* term in the fitness function. The coefficient values, except for C_{votes}, were set identically to those shown in Table 3.1.

The EC chromosome was also similar to that designed for the GA/knn hybrid classifier, although there is no k value included. Each coefficient is represented as a 32-bit integer scaled by division to produce a real value over the range [0.0, 100.0]. Masking was used to aid feature selection, with 5 mask bits for each coefficient. An example of the EC chromosome for optimization of the nonlinear discriminant coefficients is shown in Figure 4.2. GA and EP run parameters were set identically to those used in the corresponding hybrid knn classifiers, as shown in Tables 3.2 and 3.3.

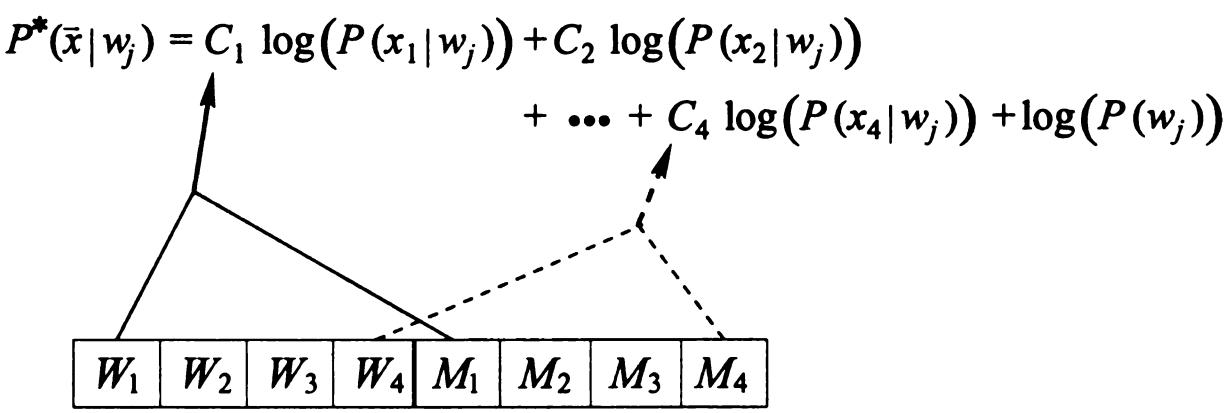


Figure 4.2: An example of the EC chromosome for optimization of the nonlinear discriminant coefficients. A four-dimensional problem is shown. Each coefficient, C_i, in the discriminant function is determined by the chromosome weight, W_i, and the masking field, M_i.

4.1.3 A New Discriminant Function Based on Summation of the Class-Conditional Marginal Probabilities

In the process of developing the nonlinear discriminant function, several alternatives, also derived from the Bayes discriminant function, were formulated. Of these alternatives, a discriminant function based on a linear combination of the marginal probability values showed initial promise in combination with EC coefficient optimization. As with the nonlinear discriminant, this sum-based discriminant function assigns a confidence value, $P^*(\vec{x}|\omega_i)$, to each class, ω_i. In this case, however, the value of P^* is simply a weighted sum of the marginal probabilities for each feature value. That is,

$$P^*(\vec{x}|\omega_i) = C_1 P(x_1|\omega_i) + C_2 P(x_2|\omega_i) + ... + C_d P(x_d|\omega_i) \tag{4.8}$$

Again, the marginal probabilities, $P(x_1|\omega_i)...P(x_d|\omega_i)$, were estimated using a histogram consisting of 20 bins. Gaussian smoothing was also employed with $\sigma = 2.0$. Additionally, the representation of the coefficients on the chromosome, EC cost function, and EC parameter settings were identical to those detailed above for the Bayes-derived nonlinear discriminant function.

4.2 Results

A GA was employed to optimize the parameters of each of the discriminant function based classifiers. The two resulting hybrid classifiers were then tested on the same

artificial and real-world data sets as the EC/knn hybrid classifiers, under identical training and testing conditions. Based on the similarity of the results for the GA and the EP optimizers when hybridized with a weighted knn classifier, only the GA optimizer was tested in conjunction with the discriminant function classifiers.

4.2.1 Artificial Data Sets

Results of the two parameterized discriminant function classifiers on data sets consisting of independent, Gaussian-distributed features are summarized in Table 4.1. Performance on these simple data sets is similar to that of the EC/knn hybrid classifiers, but the computation time is significantly less since all-pairs neighbor searching is no longer required. The nonlinear discriminant function seems to show slightly better feature selection performance than the other classifiers, as it is the only classifier to reduce the feature set for $G_{10,1}$ down to 3 features, and $G_{10,2}$ down to 9 features. There is, however, a performance penalty associated with this reduction in features—the EC/knn hybrid classifiers obtained better accuracy than the nonlinear discriminant on both of these data sets. When the number of features used is equal, as in the G_5 data set, the performance of the nonlinear discriminant is similar to that of the EC/knn hybrid classifiers.

Table 4.2 summarizes the results for data sets consisting of feature values drawn from a mixture of uniform random and Gaussian distributions. These experiments were conducted identically to the previous set, and the results were similar. The nonlinear discriminant function outperformed all other classifiers for the data set

Table 4.1: Performance of the nonlinear discriminant function (**Nonlinear**) and the summation-based discriminant function (**Sum**) on artificially-generated data sets consisting of feature values drawn from independent Gaussian distributions. For both discriminant functions, five GA experiments were conducted using the same data sets but differing initial random starting conditions, and the best result is reported. The accuracy on reclassification of the training data (**Train**), the accuracy when reclassifying the GA tuning data (**Tune**), and the accuracy when classifying an independent test set (**Test**) are provided along with the number of non-zero feature weights (**Features**). The results for the naïve Bayes classifier (**Bayes**), and the two EC/knn hybrid classifiers are reproduced here from Table 3.4 for comparison.

$G_{10,1}$	**Train**	**Tune**	**Test**	**Features**
Bayes	100	—	100	10
Nonlinear	99.0	99.6	99.1	3
Sum	99.1	99.6	98.9	4
GA/knn	98.9	100	99.4	4
EP/knn	100	100	100	4

$G_{10,2}$	**Train**	**Tune**	**Test**	**Features**
Bayes	92.9	—	91.5	10
Nonlinear	91.3	92.5	89.9	9
Sum	88.6	90.2	87.7	10
GA/knn	94.2	94.8	93.1	10
EP/knn	94.6	93.8	93.1	10

G_5	**Train**	**Tune**	**Test**	**Features**
Bayes	96.0	—	95.6	5
Nonlinear	94.8	96.2	95.1	3
Sum	91.5	94.7	93.3	3
GA/knn	94.7	97.2	95.7	3
EP/knn	95.8	97.2	96.2	3

N_5, but was outperformed by the EC/knn hybrids for M_6. The summation-based discriminant function found a unique solution for the N_5 data set. Although the accuracy for this solution was considerably lower than that of the other classifiers, the data set was classified using only a single feature.

Table 4.2: Performance of the nonlinear discriminant function (**Nonlinear**) and the summation-based discriminant function (**Sum**) on artificially-generated data sets consisting of features with a mixture of random distributions. For both discriminant functions, five GA experiments were conducted using the same data sets but differing initial random starting conditions, and the best result is reported. The accuracy on reclassification of the training data (**Train**), the accuracy when reclassifying the GA tuning data (**Tune**), and the accuracy when classifying an independent test set (**Test**) are provided along with the number of non-zero feature weights (**Features**). The results for the naïve Bayes classifier (**Bayes**), and the two EC/knn hybrid classifiers are reproduced here from Table 3.5 for comparison.

M_6	**Train**	**Tune**	**Test**	**Features**
Bayes	89.5	—	87.7	6
Nonlinear	89.2	89.0	88.1	5
Sum	88.2	88.1	87.6	6
GA/knn	92.4	93.6	90.8	5
EP/knn	90.8	92.2	89.0	4

N_5	**Train**	**Tune**	**Test**	**Features**
Bayes	94.4	—	81.6	5
Nonlinear	81.4	83.7	81.5	4
Sum	69.1	71.0	66.5	1
GA/knn	85.1	86.2	80.9	4
EP/knn	85.6	85.8	79.3	4

Once again, the balance in predictive accuracy among classes is an important factor to consider when comparing the results of the various EC-hybrid classifiers with that of the Bayes classifier. Since the EC cost function is biased to favor solutions with good balance in accuracy between classes, it will prefer such solutions to those

with disparity among the classes, possibly at the cost of overall prediction accuracy. Table 4.3 summarizes the predictive balance obtained by the various classifiers for the M_6 and N_5 data sets. All the EC-hybrid methods except the EP/knn classifier outperform the naïve Bayes classifier in terms of class balance.

Table 4.3: Balance in predictive accuracy among classes for the various EC/hybrid classifiers on the data sets M_6 and N_5. Predictive accuracy on the independent test set is shown for each class. **Balance** is defined as the difference in predictive accuracy between class 1 and class 2, as described in Section 3.1.2.

M_6	Class 1	Class 2	Overall	Balance
Bayes	91.80	84.40	88.10	7.4
Nonlinear	89.40	85.80	87.60	3.6
Sum	93.20	88.40	90.80	4.8
GA/knn	92.00	86.00	89.00	6.0
EP/knn	93.00	82.40	87.70	10.6

N_5	Class 1	Class 2	Overall	Balance
Bayes	76.60	86.40	81.50	9.8
Nonlinear	62.20	70.80	66.50	8.6
Sum	79.20	82.40	80.80	3.2
GA/knn	78.60	80.00	79.30	1.4
EP/knn	72.20	91.00	81.60	18.8

4.2.2 Classification of the UCI Data Sets

Classification of real-world data from the UCI machine learning data set repository not only allows a more realistic assessment of the performance of the various classifiers on data with erroneous and missing feature values, but also allows the classifiers to be tested on data sets with larger numbers of features than the artificially constructed data sets described above. Table 4.4 compares the classification and feature

selection performance of the two discriminant-function-based classifiers with that of the EC/knn hybrid classifiers and the naïve Bayes classifier.

The most evident aspect of the results on these four data sets is the feature selection capability demonstrated by the nonlinear discriminant function. For three of the four data sets, the minimum number of features used in classification was found by the nonlinear discriminant function in conjunction with the GA. Additionally, for the hepatitis data, the test accuracy obtained by the two discriminant function classifiers surpassed the other classifiers tested. For the other three data sets the accuracies obtained by the discriminant methods were similar to those obtained by other methods tested. The notable difference between **Tune** and **Test** results for the hepatitis and ionosphere data sets suggest that the discriminant classifiers may be more prone to overfitting of the training and tuning data than the other classifiers.

Examination of the run times for the UCI data sets illustrates the advantage held by the discriminant-function-based classifiers over the EC/knn hybrid classifiers in terms of computational efficiency. Table 4.5 compares the execution times for 200 generations of GA optimization in conjunction with the nonlinear discriminant function and the knn classifier. In all cases the nonlinear discriminant classifier is significantly faster than the GA/knn—in the case of the Pima Indian diabetes data set the difference is nearly tenfold.

Table 4.4: Results of the nonlinear-weighted discriminant function (**Nonlinear**) and the discriminant function based on summation of the marginal probabilities (**Sum**) on various data sets from the UCI Machine Learning data set repository, averaged over 50 runs. **Train/Tune** refers to the accuracy obtained when reclassifying the data used by the EC in tuning (optimizing) feature subsets and weights. **Test** refers to the accuracy obtained on an independent test set for each experiment, disjoint from the training and tuning sets. The number of features is the mean number of features used in classification over all 50 runs. Performance for the EC/knn classifiers is repeated here from Table 3.6 for comparison.

Hepatitis	**Train/Tune**	**Test**	**Features**
Bayes	85.3	65.7	19
Nonlinear	95.4	79.4	6.5
Sum	91.4	79.4	8.2
GA/knn	86.0	69.6	8.1
EP/knn	87.2	73.1	8.9

Wine	**Train/Tune**	**Test**	**Features**
Bayes	98.8	94.7	13
Nonlinear	97.8	91.3	4.5
Sum	99.0	92.3	7.6
GA/knn	99.7	94.8	6.0
EP/knn	99.5	93.2	6.2

Ionosphere	**Train/Tune**	**Test**	**Features**
Bayes	93.0	90.1	34
Nonlinear	97.6	87.5	8.5
Sum	93.8	84.2	11.2
GA/knn	95.0	91.9	8.5
EP/knn	93.2	92.3	13.5

Pima	**Train/Tune**	**Test**	**Features**
Bayes	76.1	64.6	8
Nonlinear	76.2	70.4	3.9
Sum	74.0	70.8	3.4
GA/knn	80.0	72.1	3.1
EP/knn	79.1	72.9	3.9

Table 4.5: Execution times (wall clock time) for 200 generations of GA optimization of the knn and nonlinear discriminant function classifiers. For each data set, the number of features (d), the number of classes (C), the combined training and tuning set size (n), and the mean execution time (hours:minutes:seconds) over 50 runs are shown. Each run was executed on a single 250MHz UltraSPARC-II cpu of a six-cpu Sun Ultra-Enterprise system with 768 MB of system RAM. Runs were executed in sets of 5 with no other user processes present on the system.

Data set	d	C	n	knn	nonlinear
Pima	8	2	400	1:40:13	0:10:52
Hepatitis	19	2	240	1:05:48	0:24:42
Ionosphere	34	2	400	2:02:25	0:43:37
Wine	13	3	240	0:23:59	0:14:39

4.2.3 Classification of Medical Data

For the thyroid and appendicitis data, the two discriminant function-based classifiers were trained and tested in the same manner as the EC-hybrid classifiers (Section 3.2.3). For each data set, five experiments were conducted for each classifier. The appendicitis data set was re-partitioned into disjoint training/tuning and testing sets for each experiment. The much larger thyroid data set was pre-partitioned into training and testing sets in the UCI database (Quinlan *et al.*, 1986). For this data, only the initial random GA population was changed for each experiment. The results of these experiments are summarized in Table 4.6.

Both the discriminant function based classifiers performed well on the hypothyroid diagnosis data. The nonlinear classifier utilized a smaller feature set than the GA/knn on this data, at a slight cost in bootstrap test accuracy. Both discriminant function classifiers seem to exhibit overfitting of the tuning data for the appendicitis data set.

Table 4.6: Accuracy of various classifiers on the hypothyroid and appendicitis diagnosis data sets. Results for the discriminant function classifiers are averaged over five GA experiments. Results for the GA/knn classifier represent the best of five experiments. **Train/Tune** refers to the accuracy obtained in reclassifying the GA tuning set; **Test** refers to bootstrap accuracy over 100 bootstrap sets.

Thyroid	**Train/Tune**	**Test**	**Features**
GA/knn	98.5	98.4	3
Nonlinear	97.7	97.2	2.7
Sum	97.8	97.4	4.2
Appendicitis	**Train/Tune**	**Test**	**Features**
GA/knn	90.4	90.6	2
Nonlinear	80.4	67.0	2.6
Sum	83.0	74.2	2.2

This behavior, along with the previous results for the hepatitis and ionosphere data sets, suggests that the discriminant function classifiers may be prone to overfitting when presented with small (< 50 samples of each class) data sets for training, tuning, and testing.

4.3 Discussion

A key advantage of the discriminant function classifiers over the nearest neighbor methods is the gain in computational efficiency obtained by estimating the class-conditional feature value distributions based on the training data, rather than storing every training sample and performing an all-pairs search for near neighbors for each test sample. While the experiments here were all executed for a fixed number of EC generations, it would be worthwhile to run several experiments constrained

instead by wall-clock time. In this way, the efficiency advantage of the discriminant function-based classifiers might be translated into further gains in classification accuracy relative to the near-neighbor methods.

The nonlinear discriminant function classifier, in conjunction with the GA feature extraction method, seems to exhibit the best feature selection capability of all the classifiers evaluated. In several cases, however, the additional reduction in the number of features, as compared to the GA/knn classifier, incurred a slight cost in terms of classification accuracy.

The promise exhibited by the discriminant function-based classifiers suggests several avenues for further investigation. One possible improvement would be to include the prior probabilities for each class on the EC chromosome for optimization. Intuitively, this might allow the hybrid classifier more ability to maintain more control over the balance in predictive accuracy among classes, even when there is disparity in the number of training and tuning samples available for each class. Initial experimentation in this area, however, suggested that inclusion of the prior probabilities on the chromosome can exacerbate the problem of overfitting the training and tuning data, resulting in poor performance on independent test data.

Since the capabilities of the Bayes classifier have been well explored in the literature, an alternate approach would be to avoid direct modification of the Bayes discriminant function. Instead, the independence assumption of the naïve Bayes classifier might be abandoned, and the relationship among the various distributions of feature values encoded on the EC chromosome as a covariance matrix, or some sub-

set thereof. Essentially, this approach would allow the EC to estimate the covariance matrix for each class based on both the training and tuning data provided. In conjunction with the previously proposed technique of including the *a priori* probabilities for each class on the chromosome, all the parameters of a traditional Bayes classifier might be optimized by the EC for a particular data set and error cost function.

Chapter 5

Identifying the Determinants of Solvent Binding in Proteins

5.1 Introduction

The most extensive application and testing of the various EC-based feature selection and extraction methods described here has been performed in the field of biochemistry. The specific problem area investigated was that of evaluating and understanding protein-water interactions. Understanding the nature of protein solvation (water binding) is a significant outstanding problem in structural biochemistry, and has been the subject of numerous studies, yielding a substantial body of literature (Palma *et al.*, 1993; Westhof, 1993; Ladbury, 1995; Karplus and Faerman, 1994; Levitt and Park, 1993). Elucidation of the nature of protein-water interactions would be a significant contribution to a broad range of biochemical endeavors, including *de novo* drug design, database screening for drug leads, ligand docking, understanding of protein structure

and function, and even the grand challenge "protein folding" problem.

5.1.1 An Overview of the Protein-Water Problem

Proteins are macromolecules consisting of linear (non-branching) chains of amino acids. The twenty common amino acids are joined covalently during protein synthesis to form the protein "main chain" or "backbone". A particular protein can be completely specified by the order of the different amino acids that form its main chain. After synthesis, a protein will fold into a particular three-dimensional structure (Figure 5.1). Since this structure is generally a compact globule, folded, soluble proteins are occasionally referred to as "globular" proteins. In the context of a living organism, numerous water molecules will be bound, via hydrogen bonds, steric, and other interactions, to the surface of the protein in its globular form.

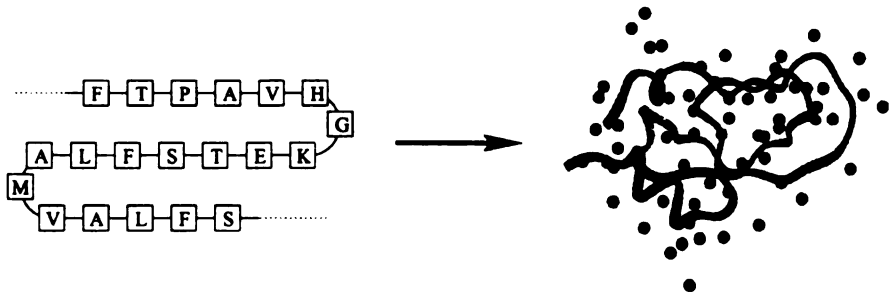


Figure 5.1: A particular sequence of amino acids will consistently fold into the same **three**-dimensional, "globular" structure. Each of the 20 common amino acids is **identified** by a one-letter code. The linear protein chain (left) folds into its globular **structure** (right) after synthesis. Water molecules, shown here as spheres, bind to the **surface** of the protein via hydrogen bonds.

While proteins have many functions in the context of a living organism, most

proteins bind to a specific molecule (or set of molecules) and perform some action upon that molecule, such as transportation, cleavage, or catalysis of other chemical reactions. The particular molecule that a protein binds in this manner is called the *ligand*, and the area where the ligand binds is referred to as the protein's *active site*. Like the rest of the protein surface, the residues in the active site will typically bind water molecules. When a protein binds to a particular ligand, some of the water molecules in the active site will be displaced by the ligand, while some will remain in the area of the interface between the protein and the ligand. Those water molecules that remain in the interface will often form hydrogen bonds to both the protein and the ligand, thus helping the protein to bind to the ligand by forming a water-mediated "bridge" between the two molecules (Raymer *et al.*, 1997; Poornima and Dean, 1995c). One study of 20 protein/ligand complexes found that the average protein/ligand interface includes 10 water molecules and 17 water-mediated bridges between the protein and ligand (A. Cayemberg & L. A. Kuhn, unpublished results).

In the process of designing a ligand to bind to a particular active site, such as a pharmaceutical drug, water molecules that are not displaced from the active site upon ligand binding must be considered, as they form an essential part of the protein surface (Poornima and Dean, 1995a). Unfortunately, during ligand design it is generally unknown which water molecules will remain bound upon ligand binding, and which will be displaced. If the sites of interfacial water binding were known, this information could be used to improve the chemical and shape complementarity of the ligand in the design process.

The primary goal in analyzing protein-water interactions is to isolate the determinants of water binding—the physical and chemical features of a local protein environment that determine whether or not the site is conducive to water binding. In terms of classification, it is desirable to categorize areas near a protein surface into one of three classes: sites that are unlikely to bind water molecules, "weak" solvation sites that bind water molecules but are not likely to be conserved during ligand binding, and "strong" solvation sites that are likely to be conserved between the ligand-bound and unbound forms of a protein (Ringe, 1995).

5.1.2 State of the Art in Predicting Protein Solvation

X-ray Crystallographic Data Collection

The input for most algorithms that predict protein solvation consists of the three-dimensional structure of the protein, usually represented as the Cartesian coordinates of each heavy (non-hydrogen) atom in the protein. This structure is generally obtained via X-ray crystallography. In this process, a protein is first purified and dissolved in an aqueous solvent. From this solution, microscopic crystals (with edges on the order of tenths of millimeters) are grown. These crystals consist of roughly equal amounts of the protein, arranged in a lattice, and interstitial water molecules. The crystals are then suspended in small capillaries, and exposed to a beam of X-ray radiation. The electrons in the protein crystal diffract the X-rays, and the diffraction pattern is captured in two dimensions by exposing X-ray sensitive film or an electronic detector to the diffracted X-rays. The diffraction pattern on the exposed

film can then be digitized and subjected to a Fourier transformation, resulting in a three-dimensional map of the electron density in the crystal. This electron density map is equivalent to a time-averaged and space-averaged[1] snapshot of the locations of the electrons in the protein crystal. From this information, the three-dimensional structure of the protein can be deduced.

The result of an X-ray crystallographic experiment—often referred to as a crystallographic structure—contains several other pieces of information along with the coordinates of the atoms. Each atom is assigned a crystallographic temperature factor, or B-value, which provides a relative measure of the thermal mobility of the atom. A related measurement, the occupancy of the atom, provides an estimate of the percentage of time that a site is occupied by a given atom type over the experiment time and throughout the various copies of the protein in the crystal. Generally, the information used for training and testing in empirical solvent-site prediction algorithms is calculated from the atomic coordinates, thermal mobility, and occupancy data obtained via crystallography. Occasionally, raw electron density data is also considered. From the crystallographic coordinates, other physical and chemical features of each protein atom can be calculated. Further details and examples are provided in Section 5.2.3.

[1]The electron density map is space-averaged in the sense that the average electron density over all the copies of the protein in the crystal is measured.

Current Approaches to Solvent Site Prediction

Current algorithms for predicting the locations of bound water molecules can be divided into two classes. Theoretical approaches, such as GRID (Wade *et al.*, 1993), use a potential energy function to evaluate the favorability of a probe site. For GRID, the potential energy function includes terms for Lennard-Jones interactions, electrostatic interactions, and a detailed evaluation of potential hydrogen bonds. For all theoretical approaches, the relative contribution of the terms in the potential energy function must be determined before solvation sites can be predicted. In contrast, empirical methods determine the favorability of a solvation site by analogy with known sites. A site is evaluated and compared to a database of known solvation sites and non-solvated sites, and predicted as being more similar to one than the other. A set of features to observe and compare between solvation sites and non-solvated sites must be selected prior to classification. Several empirical methods for prediction of protein-bound water molecule locations have been developed. AQUARIUS2 (Pitt *et al.*, 1993) uses a knowledge base of the distributions of water molecules around polar atoms at the protein surface. A "likelihood" value is assigned to a putative water molecule location based on its geometric relationship to nearby polar protein atoms. If the site lies in a region highly occupied by water molecules in the knowledge base and has significant electron density, as determined by X-ray crystallography, it receives a higher score. The highest scoring positions in a 3-dimensional matrix surrounding the protein are selected as likely water molecule locations. AUTO-SOL (Vedani and Huhta, 1991) predicts water sites based on the directionality of hydrogen bonds. A database of

small-molecule crystal structures was analyzed to find the distribution of hydrogen-bond lengths and directions, and possible solvent sites are evaluated by AUTO-SOL according to how well their hydrogen-bond geometry matches this database. Current methods can reproduce ~70% of crystallographically observed solvent molecules within 1.5 Å of the experimental locations (Vedani and Huhta, 1991). There remains, however, a tendency for many current methods to produce false positives, predicting solvation sites where none are observed in the crystal structure.

Both empirical and theoretical methods require the evaluation of a set of features describing the environment of the site. Commonly-used features include hydrogen bond geometry, electrostatic interactions, local thermal mobility, protein surface shape, solvent accessibility, and the packing geometry, atom type, and residue type of neighboring protein atoms. Several studies have evaluated the correlation between individual features and water binding. For instance, in a study of 56 high-resolution crystallographic structures, a strong relationship was found between local surface shape and water binding (Kuhn *et al.*, 1992). Deep grooves in the solvent-accessible surface bound three times as many water molecules as non-groove surface areas. Results obtained later also showed that deep surface grooves were the preferred sites for tightly-bound water molecules, which were unlikely to be displaced by ligand binding (Poornima and Dean, 1995b). Furthermore, conserved polar contacts between water molecules and the protein were found to be associated with solvation-site conservation (Poornima and Dean, 1995c).

As noted in previously, X-ray crystallographic assignment is an imperfect source of

information for known water sites, resulting in both false-positive and false-negative sites in any database based on crystallographic data. One method for identifying reliable water molecule positions is to utilize solvation sites conserved in multiple crystallographic structures. In a study of eighteen crystallographically independent structures representing ten unique crystal forms of T4 lysozyme, it was found that ~79% of the 20 most frequently occupied sites were conserved in all structures when allowing for potential steric interference induced by the crystal lattice (Zhang and Matthews, 1994). A recent study of conserved solvation sites in thrombin, trypsin and bovine pancreatic trypsin inhibitor (BPTI) found that 19% of unique water sites in 10 superimposed thrombin structures were conserved in at least half of the structures, while 77% of the unique sites in three superimposed trypsin structures were conserved in at least two of the three structures (Sanschagrin and Kuhn, 1998). In perhaps the most extreme change in crystallization conditions, the structure of subtilisin Carlsberg was solved in an organic solvent, anhydrous acetonitrile, and compared with the structure solved in an aqueous environment (Fitzpatrick *et al.*, 1993). Of the 119 bound water sites observed in the aqueous structure, 99 were conserved in the acetonitrile structure. Together, these studies show that one-half or more of bound water sites are typically conserved in independent structures of a protein solved under diverse conditions.

A slightly different approach was taken by Gschwend and Karplus (1991) in their study of features correlating with increasing electron density of water sites in glutathione reductase. The water molecules in a 1.54 Å resolution structure were ranked

in order of electron density in the final $2F_O - F_C$ map. Eight parameters were measured for each water molecule, weighted, and correlated with the electron density rankings; they included the energy from XPLOR refinement, solvent accessibility, strong hydrogen bonds to protein atoms and to other water molecules (donor to acceptor bond length between 2.5 and 3.2 Å), weak hydrogen bonds (bond length less than 2.5 Å or from 3.2 to 3.5 Å) to protein atoms and water molecules, the average temperature factor of hydrogen-bonding partners of the water molecule, and the average occupancies of hydrogen-bonding partners. The parameters that correlated best with increasing electron density included decreasing average temperature factor of hydrogen-bonding partners of the water molecule, decreasing solvent accessibility of the water, and increasing the number of strong hydrogen bonds to protein atoms. The linear correlation coefficient between this set of features and the density ranking was 0.8.

An experimental approach was taken by Ringe (1995) to investigate the characteristics of ligand-binding sites on the protein surface. Crystals of elastase were grown in aqueous mother liquor, then transferred to an organic solvent. A shell of bound water molecules remained, even after the organic solvent was exchanged several times, though some organic solvent molecules also bound. The features associated with binding sites for the polar organic solvent molecules included the local surface shape, exposure of hydrophobic groups, and the presence of water molecules bound to polar groups near the site.

The approach presented here is a hybrid between prediction of bound water lo-

cation and characterization of features favorable for water binding. Three classes of protein sites were studied: non-solvated sites, sites that were solvated in a single crystallographic structure of a protein, and solvation sites that were conserved in two crystallographically independent structures. The hybrid genetic algorithm/k-nearest-neighbors (GAknn) classifier was provided with a set of features representing each site, and trained to distinguish between the classes of sites.

Motivation and Experimental Design

Unfortunately, one of the steps in the crystallographic determination of protein structures, the assignment of water molecule locations based on the electron density data from an X-ray crystallographic experiment, is an imperfect process. The resulting 3-dimensional structure may include false sites via noise in the electron density, and omit many sites due to limited resolution. Additionally, the structure of the crystal lattice may produce favorable environments for bound water molecules where none were present in the soluble form of the protein. By applying the feature selection and extraction methods described here to protein crystallography data sets, we hope to improve water assignment by identifying the physical and chemical features associated with water-binding sites, and by producing a classifier that is capable of distinguishing solvation sites from non-solvated sites near the protein surface.

In order to train such a classifier, however, the current, imperfect information about protein-water binding available from X-ray crystallographic structures must be used. To mitigate the effect of noisy water coordinate data, we can superimpose mul-

tiple crystallographic structures of the same protein and examine the water-binding sites that are conserved between them. In this way we can identify the physical and chemical features associated with the most reliable water binding sites available—those that are conserved among multiple, independently-solved structures of the same protein. Finally, by using this information during the assignment of water molecules in future structures, we can improve the reliability of crystallographic water assignment by considering not only the electron density information obtained from the X-ray experiment, but also the physical and chemical features of the local protein environment in determining whether or not to assign a water molecule to a particular location near the protein surface. While the electron density is the experimental observable and the most important criterion, evaluating the likelihood of a given protein environment for water binding gives us insight into the protein-water recognition process and also can help identify missing or extraneous water sites in crystallographic structures.

In pursuit of this goal of understanding protein-water interactions more completely, the classification and feature selection and extraction methods described in earlier chapters were applied to two distinct classification problems:

Prediction of solvation sites: Given a location near the surface of a protein, determine whether the site is likely to be a water-binding site or a non-solvated site.

Prediction of solvent-site conservation: Given a known solvent site from a non-ligand-bound protein, determine whether the water molecule will be conserved

(remain bound) in the protein-ligand complex.

For each of these problems, two subproblems related to understanding protein-water binding were investigated. The first was to develop and train a classifier capable of distinguishing between the two classes of sites: solvated vs. non-solvated, and conserved vs. non-conserved, respectively. The second objective was to utilize the feature selection and data mining capabilities of the EC-hybrid classifiers to identify the features of protein bound water molecules that are associated with solvent binding, and with conservation of solvent sites between structures.

5.2 Methods

5.2.1 Compilation of a Protein Solvation Database

The task of understanding protein-water interactions was undertaken in two stages. The objective of the first stage was to design a hybrid EC/knn classifier capable of distinguishing water molecules from non-solvated sites on the surface of a protein, and to use the feature selection capabilities of this classifier to identify the physical and chemical features associated with solvation sites. The second goal was to further classify the water sites into those likely to be conserved between a ligand-bound and an unbound structure of a protein, and those likely to be non-conserved (displaced). Again, once this classification was made, the data mining and feature selection capabilities of the classifier were utilized to identify the determinants of solvent conservation between ligand-bound and unbound structures.

Prerequisite to any of these experiments it was necessary to establish a database of conserved and non-conserved water molecules as well as non-solvated sites near protein surfaces to serve as training and tuning data for the GA/knn hybrid classifier. To this end, 60 protein structures were selected in pairs from the Brookhaven Protein Databank (Abola *et al.*, 1987; Bernstein *et al.*, 1977). Each pair consisted of the crystallographic structure of a protein bound to a ligand and the corresponding structure of the unbound protein. Protein pairs were selected based on several criteria. First, all proteins included in the database were non-homologous[2] (Hobohm *et al.*, 1992). In addition, high resolution ($\leq$ 2.0 Å) structures were preferred, and structures exhibiting significant conformational changes upon ligand binding were excluded, since the concept of conserved water is no longer well-defined in this case. Pairs of structures with a root-mean-squared positional deviation (RMSD) of $\leq$ 2.0 Å upon superposition of main-chain atoms were also preferred. The 30 pairs of proteins selected for the database are detailed in Table 5.1.

[2]Homology, in this case, refers to proteins composed of similar sequences of amino acids. Inclusion of homologous proteins in the database can introduce redundant information, which can lead to unnecessary bias in the training and tuning data. No two proteins included in the database described here had > 25% amino acid sequence identity when the linear sequence of amino acids comprising each protein were aligned.

Table 5.1: 30 protein pairs included in the database.

PDB Code[1]	Protein / Ligand Complex	Res(Å)[2]	Waters	RMSD[3]
1ahc/1ahb	α-momorcharin / formycin 5′-monophosphate	2.0/2.2	163/80	0.23
1apm/1atp	cAMP-dependent protein kinase / MnATP	2.0/2.2	207/103	0.35
1bia/1bib	bira bifunctional protein / biotinylated lysine	2.3/2.8	43/19	0.48
1bsa/1brn	barnase / D(CGAC)	2.0/1.76	258/229	0.55
1ca2/1bcd	carbonic anhydrase II / trifluoromethane sulphonamide	2.0/1.9	167/215	0.20
1cgf/1hfc	fibroblast collagenase / HAP[4]	2.1/1.56	181/88	0.39
1cgt/1cgu	cyclodextrin glycosyltransferase / glucose	2.0/2.5	588/478	0.34
1chp/1xtc	cholera toxin β pentamer / polypeptide chain	2.0/2.4	248/138	0.93
1dr2/1dr3	dihydrofolate reductase / biopterin	2.3/2.3	73/110	0.12
1gta/1gtb	glutathione S-transferase / praziquantel	2.4/2.6	118/84	0.20
1hel/1mlc	hen egg-white lysozyme / monoclonal antibody Fab D44.1	1.7/2.1	185/210	0.49
1lib/1lic	adipocyte lipid-binding protein / hexadecanesulfonic acid	1.7/1.6	89/68	0.32
1nsb/1nsc	neuraminidase / N-acetyl neuraminic acid	2.2/1.7	446/506	0.12
1poa/1pob	phospholipase A2 / transition-state analogue	1.5/2.0	151/242	0.72
1syc/1syd	staphylococcal nuclease / 2′-deoxy-3′-5′-diphosphothymidine	1.8/1.7	69/83	0.41
1thm/2tec	thermitase / eglin-C	1.37/1.98	193/224	0.24

Table 5.1 (cont.)

PDB Code[1]	Protein / Ligand Complex	Res(Å)[2]	Waters	RMSD[3]
1udg/1udh	uracil-DNA glycosylase / uracil	1.75/1.75	121/94	0.19
2act/1aec	actinidin / E64[5]	1.7/1.86	272/268	0.11
2apr/3apr	acid proteinase / reduced peptide inhibitor	1.8/1.8	373/344	0.13
2cla/3cla	chloramphenicol acetyltransferase / chloramphenicol	2.35/1.75	104/204	0.41
2ctv/5cna	concanavalin A / α-methyl-D-mannopyranoside	1.95/2.0	146/533	0.42
2sga/5sga	proteinase A / tetrapeptide Ace-Pro-Ala-Pro-Tyr	1.5/1.8	220/185	0.08
2wrp/1tro	Trp repressor / synthetic operator	1.65/1.9	170/572	2.18
3cox/1coy	cholesterol oxidase / 3-β-hydroxy-5-androsten-17-one	1.8/1.8	453/424	0.24
3dni/2dnj	deoxyribonuclease I / DNA	2.0/2.0	375/252	0.37
3enl/5enl	enolase / 2-phospho-D-glyceric acid	2.25/2.2	353/355	0.21
3grs/1gra	glutathione reductase / glutathione disulfide	1.54/2.0	523/530	0.12
3tln/3tmn	thermolysin / Val-Trp	1.6/1.7	173/173	0.10
5cpa/6cpa	carboxypeptidase A / phosphonate	1.54/2.0	315/148	0.36
See note 6	RTEM-1 β-lactamase / penicillin G	1.7/1.7	182/189	0.22

[1]Ligand-free/ligand-bound
[2]Resolution of the crystallographic structures in Ångstroms.
[3]Main-chain root-mean-squared positional deviation from superposition of the ligand-bound and free structures.
[4]HAP is (N-(2-hydroxamatemethylene-4-methyl-pentoyl)phenylalanyl) methyl amine.
[5]E64 is [N-(l-3-trans-carboxyoxirane-2-carbonyl)-l-leucyl]-amido(4-guanido)butane.
[6]Provided by Drs. Natalie Strynadka and Michael James, University of Alberta, Edmonton.

The water molecules from the unbound structures were used to construct the training and tuning database. The ligand-bound structures were used only to label

the water molecules from the unbound structures as conserved or displaced. To obtain this labeling, the main-chain atoms of the ligand-bound and unbound structures were superimposed using the rigid-body superimposition algorithm of the *InsightII* molecular graphics software (Molecular Simulations, Inc., San Diego, CA). Each water molecule in the unbound structure was then compared with the superimposed ligand-bound structure—if a corresponding water molecule was found in the ligand-bound structure within 1.2 Å (Zhang and Matthews, 1994) of the water from the unbound structure, the water was labeled as conserved. Water molecules from the unbound structure with no corresponding water molecule in the superimposed ligand-bound structure were labeled as non-conserved. Only first-shell water molecules from the unbound structures were included in the database, where the first shell is defined as the set of water molecules within hydrogen bonding distance (3.6 Å) of a protein surface atom.

5.2.2 Generation of Non-Solvated Probe Sites

For distinguishing solvation sites from non-solvated sites near the protein surface, it was necessary to generate a collection of non-solvated probe sites near the surface of each of the 30 non-ligand-bound proteins selected for the database. To generate these probe sites, the solvent accessible molecular surface for each of the non-ligand-bound structures in Table 5.1 was generated using the Molecular Surface (MS) software from Connolly (1983). A probe radius of 1.2 Å was used, with a surface density of 1 dot/Å^2 and unit normals generated at each surface dot. For each protein, the minimum

distance of any crystallographically observed water molecule from the protein surface, d_{min}, and the maximum distance, d_{max} were determined. For each surface point, a probe-site was generated and placed at a distance along the surface normal, selected at random over the range of distances from d_{min} to d_{max}. Any non-sites overlapping crystallographic water sites or other non-sites (with positions within 1.2 Å) were removed. Finally, the same number of non-sites were selected for each protein as there were crystallographic water sites. This selection was done so that the distribution of distances of non-sites from the protein surface matched the distribution of distances for observed water molecules for that protein.

5.2.3 Feature Identification and Measurement

Once a database of non-solvated sites, non-conserved water-binding sites, and conserved solvation sites had been established, the next step was to identify and measure a collection of features to represent each water binding site or non-solvated site for purposes of classification. For each site, six physical and chemical features of the local protein environment were computed from the crystallographic coordinates. For conserved and non-conserved solvation sites, two additional features describing the thermal mobility of the water molecule in the crystal structure were included. The eight features included in the waters database are included in Table 5.2.

Table 5.2: The physical and chemical features used to represent protein-bound water molecules and protein surface sites. BVAL and MOB were only measured for conserved and non-conserved molecules, as the concept of thermal mobility is not defined for non-solvated probe sites.

Tag	Feature	Description
ADN	Atomic density	The number of protein atom neighbors within 3.6 Å of the water molecule. This feature correlates with the local protein topography. Water molecules bound in deep grooves will have high ADN values, while those bound to protrusions will have low ADN values (Kuhn *et al.*, 1992).
AHP	Atomic hydrophilicity	The hydrophilicity of the neighborhood of the water molecule is based on the frequency of hydration for each atom type in 56 high-resolution protein structures (Kuhn *et al.*, 1995). Each water molecule is assigned an AHP value equal to the sum of the atomic hydrophilicity values of all atom neighbors within 3.6 Å of the water molecule.
BVAL	B-value	The crystallographic temperature factor from PDB file of the crystal structure. This feature measures the thermal mobility of the water molecule.
HBDP	Hydrogen bonds to protein	The number of hydrogen bonds between the water molecule and neighboring protein atoms. Each donor or acceptor atom within 3.5 Å is considered a potential hydrogen bond.
HBDW	Hydrogen bonds to water	The number of hydrogen bonds between the water molecule and other water molecules in the ligand-free protein structure, based on $\leq$ 3.5 Å distance between oxygen atoms in the two water molecules.

Table 5.2 (cont.)

Tag	Feature	Description
MOB	Mobility	A normalized measure of thermal mobility, defined in terms of the B-value and occupancy, two measures of the thermal mobility of the water molecule provided in the crystallographic structure of the protein. MOB = $\left(B_w/\overline{B}\right) / \left(Occ/\overline{Occ}\right)$. Where B and $\overline{B}$ are the B-value of the water molecule and the average B-value of all the water molecules in the protein respectively. Similarly, Occ and $\overline{Occ}$ are the occupancy of the water molecule (from the PDB file) and the average occupancy of all water molecules in the protein structure (Craig *et al.*, 1998).
ABVAL	Average B-value of protein atom neighbors	The average (mean) temperature factor of all protein atoms within 3.6 Å of the water molecule.
NBVAL	Net B-value of protein atom neighbors	The sum of the B-values of all protein atoms within 3.6 Å of the water molecule.

The eight features described in Table 5.2 comprise most of the factors currently believed to be related to protein/water binding, as discussed in Section 5.1.2. These features do not, unfortunately, establish a well-separated or easily classified data space. Figure 5.2 shows 800 waters selected randomly from the conserved and non-conserved protein-bound water molecule data set, plotted according to their first two principal components. This plot demonstrates the high degree of overlap between the two classes. This overlap is further evidenced by the poor classification results obtained by linear discriminant analysis (Fisher, 1936), which obtains a classification accuracy of only ~50% (equivalent to random prediction) in distinguishing between conserved and non-conserved solvation sites based upon this data set.

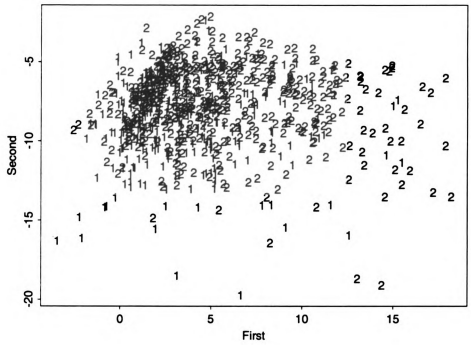


Figure 5.2: Conserved and non-conserved water molecules plotted according to the first two principal components of the eight-feature water binding data set. Eight hundred randomly-selected water molecules from the data set are plotted. Non-conserved water molecules are plotted as 1's, conserved water molecules are plotted as 2's. The first principal component is composed primarily of the features HBDP, ADN, and AHP (with coefficients 0.63, 0.52, and 0.38, respectively). The second principal component is composed primarily of the three features based on temperature factor: ABVAL, BVAL, and NBVAL. (The respective coefficients are -0.54, -0.53, and -0.44.) There is no clear separation between the two classes in this feature space.

5.2.4 Distinguishing Solvation Sites from Non-Solvated Sites Near the Protein Surface

The training and tuning data for distinguishing solvation sites from non-solvated regions of the protein surface consisted of all the first-shell water molecules from the non-ligand-bound structures in Table 5.1, a total of 5542 water molecules, and an equal number of non-solvated probe sites, for a total of 11,084 samples. The non-solvated sites were generated from the same 30 non-ligand-bound structures, as described in Section 5.2.2. The six features from Table 5.2 not related to thermal mobility were used to represent each sample, and the patterns were labeled "class 1" for non-solvated probe sites and "class 2" for solvation sites.

For each GA/knn experiment, 1000 patterns of each class were selected at random from the database to serve as the training set. Likewise, the tuning data consisted of 1000 patterns of each class. After the GA/knn experiment, bootstrap testing was performed as described in Section 3.1.6 to test the predictive accuracy of the best weight set and k value found by the GA. The bootstrap testing pool consisted of the 3542 samples from each class not included in the training or tuning data. Individual test sets of size 500 were drawn from this pool during the bootstrap testing.

The GA run parameters and the cost function coefficients for these experiments were identical to those detailed in Tables 3.1 and 3.2.

5.2.5 Distinguishing Conserved Solvation Sites from Sites not Conserved Between Ligand-Bound and Unbound Protein Structures

The training and tuning data for classifying conserved solvation sites between ligand-bound and unbound protein structures consisted of the 5324 first-shell water molecules from the 30 non-ligand-bound structures in Table 5.1. Of these water molecules, the 2137 non-conserved water molecules were labeled "class 1", and the 3405 conserved water molecules were labeled "class 2". All eight features described in Table 5.2 were used to represent each sample.

Training and testing data were compiled in a similar manner to the previous set of experiments, but due to the smaller size of this data set, the training and testing data consisted of 400 patterns of each class. The remaining 2605 conserved and 1337 non-conserved water molecules composed the pool of samples for bootstrap testing. Again, the bootstrap test set size was 500 samples.

As with the previous set of experiments, the GA run parameters and the cost function coefficients were set as described in Tables 3.1 and 3.2.

5.3 Results

5.3.1 Differentiation of Solvation Sites from Non-Solvated Sites

The protein solvation site and non-solvated site experiments were conducted with several objectives. In addition to optimizing classification accuracy, the feature weights produced by the GA were analyzed to determine if a subset of the available features was consistently selected for inclusion in the highest performance weight sets. Fifteen GA/knn experiments were conducted with differing initial random populations. 100 bootstrap test experiments were then conducted on the best 6 resulting weight sets. Descriptive statistics were calculated for each set of bootstrap tests, and the results are summarized in Table 5.3, sorted according to mean bootstrap accuracy.

In spite of randomized initial conditions for each GA experiment, the top performing runs for distinguishing protein solvation sites from non-sites (refer to Table 5.3) exhibited notable consistency in feature subset selection across runs. Although we cannot assume that this consistency is indicative of finding a globally optimal set of feature weights for classifying this data, we can conclude that the features that are consistently included in the optimized weight sets are a sufficient set to predict, with ~68% accuracy, water binding sites at the protein surface. In this respect, then, these features can be concluded to be associated with water binding.

For distinguishing observed solvation sites from pseudo-water sites, the number of hydrogen bonds to protein atoms (HBDP) was the most common highly-weighted

Table 5.3: Bootstrap test results, k-values, and weight sets from the top six GA experiments for distinguishing crystallographically-observed solvation sites from non-sites. Bootstrap testing accuracy is given as a percentage for observed solvation sites (**Site**), non-solvated sites (**Non**), and both classes together (**Tot**). Balance (**Bal**) is the average difference between solvated-site and non-solvated-site prediction accuracy (%) over all bootstrap runs. Features with no weights were masked by the GA; i.e. their feature weights were zero, and they did not participate in classification. The feature weights for all six features are normalized to sum to 1.00 for each experiment.

Bootstrap %					Feature Weights					
Site	**Non**	**Tot**	**Bal**	**K**	ADN	AHP	HBDP	HBDW	ABVAL	NBVAL
68.60	67.75	68.18	3.50	23	0.284		0.524			0.193
66.87	68.74	67.81	3.50	75	0.545		0.235			0.219
65.93	65.93	65.93	3.20	53	0.530		0.197			0.274
65.21	70.44	67.83	5.49	45	0.194		0.721		0.086	
66.19	69.40	67.79	4.39	35	0.332		0.643		0.024	
67.32	67.74	67.53	3.52	61	0.651		0.289		0.060	

feature, followed closely by atomic density (ADN). Some measure of the temperature factor of neighboring protein atoms was also included in each of the top weight sets. For half of the runs the average temperature factor (ABVAL) was included, while the sum of the temperature factors (NBVAL) was included in the other half. The local atomic hydrophilicity (AHP) and the number of hydrogen bonds to other water molecules (HBDW) were not included in any of the highest-performance weight sets. The average weights for each feature across the top fifteen weight sets are shown in Figure 5.3.

These results correspond well with several other studies of conserved solvation sites in specific protein structures. In a study of conserved solvation sites in thrombin, trypsin, and bovine pancreatic trypsin inhibitor (BPTI), various features of the local protein environment were correlated with the degree of conservation of solva-

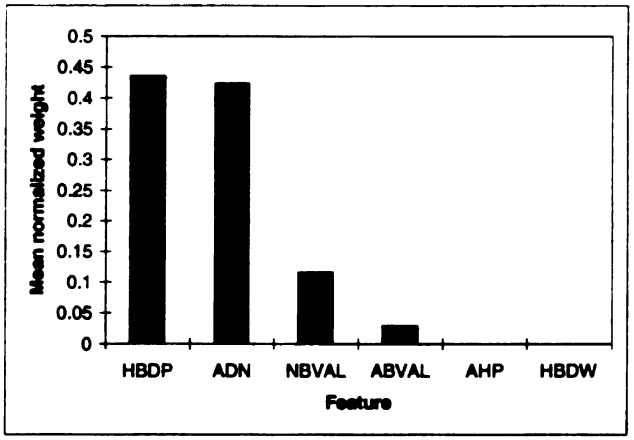


Figure 5.3: Average normalized weight of each feature in the top fifteen weight sets for distinguishing crystallographically- observed solvation sites from non-solvation sites.

tion sites across multiple superimposed structures for each protein (Sanschagrin and Kuhn, 1998). The three features which demonstrated the strongest correlation with water site conservation were hydrogen bonds to neighboring protein atoms, atomic hydrophilicity, and atomic density. Two of these features, hydrogen bonds to protein atoms and atomic density, were highly weighted by the GA in all of the weight sets reported in Table 5.3. A possible explanation for the GA's failure to include atomic hydrophilicity in any of these weight sets is the strong correlation between atomic density and atomic hydrophilicity. Since both of these features are strongly dependent on the number of protein atoms neighboring a given water molecule, the two values are correlated ($r = 0.642$).

Even more closely related to features found by the GA are those found to be related to water electron density rankings by Gschwend and Karplus (1991)—average temperature factor of hydrogen bonding partners, solvent accessibility, and the number

of strong hydrogen bonds to protein atoms. The hydrogen bonding and temperature factor features were selected by both methods, while the atomic density feature used by the GA is closely related to the solvent accessibility term used in Gschwend and Karplus's density correlations. Water molecules bound in deep grooves will have low solvent accessibility and higher atomic density values, while the reverse holds for waters bound to protein surface protrusions.

Studies conducted by Kuhn *et al.* (1992) and Poornima and Dean (1995b) both demonstrated a relationship between water binding and local surface shape. The GA results identify the same relationship, as atomic density is included in all of the top performing weight sets; high density of protein atoms surrounding a water site correlates with groove-like topography.

5.3.2 Distinguishing Between Conserved and Non-Conserved Solvation Sites

The problem of identifying solvation sites likely to be conserved between ligand-bound and unbound structures of a protein was one of the first applications of the GA/knn hybrid classifier. As such, significantly more experiments have been conducted for this data set than for any of the other data sets presented here. The top 21 weight sets from all such experiments were selected for bootstrap testing. Again, 100 bootstrap tests were conducted for each weight set. Accuracy and balance results for each weight set and k value tested are summarized in Table 5.4.

Table 5.4: Bootstrap results of the best 21 weight sets for identifying conserved solvation sites. Mean bootstrap testing accuracy is given as a percentage (**Acc**). Mean balance (**Bal**) is the average difference between conserved and non-conserved prediction accuracy (%) over all bootstrap runs. Features with no weights were masked by the GA; i.e. their feature weights were zero, and they did not participate in classification. The feature weights for all eight features are normalized to sum to 1.00 for each experiment.

			Feature Weights							
Acc	**Bal**	**K**	ADN	AHP	BVAL	HBDP	HBDW	MOB	ABVAL	NBVAL
64.20	3.88	65			0.413	0.135	0.137	0.315		
63.62	3.80	29			0.667			0.333		
63.16	5.35	25			0.463			0.323		0.214
63.11	4.15	37			0.891			0.109		
63.00	3.52	77			0.308	0.163	0.225	0.304		
62.87	5.36	17			0.841			0.159		
62.47	7.74	97	0.459	0.291	0.250					
62.36	3.27	27		0.371	0.629					
62.16	3.79	23			0.372	0.240		0.203		0.184
62.15	3.50	7			0.571	0.156		0.273		
62.02	4.26	87		0.118	0.558	0.323				
61.72	4.14	17		0.252	0.352	0.397				
61.70	4.20	67			0.421			0.579		
61.58	3.58	13	0.018	0.388	0.441					0.153
61.46	3.84	63			0.227		0.773			
61.42	3.18	19		0.051	0.417	0.058		0.474		
61.36	3.39	15			0.392	0.293		0.207		0.108
61.16	7.14	19			1.000					
60.62	4.58	57			0.881					0.119
60.30	2.78	75		0.317						0.683
60.25	3.40	69				0.336				0.664

In applying the feature selection and extraction capabilities of the EC-hybrid classifiers for data mining and analysis, consistency in feature weighting across multiple runs is a desirable result. While it is never possible to guarantee that the EC has found an optimal or near-optimal set of feature weights, features that are consistently included in weight sets that provide high classification accuracy are likely to be associated with the natural classes in the data. For the water conservation data, the two features related to thermal mobility—B-value (BVAL) and mobility (MOB)—are included in all of the top 6 performing weight sets. Features such as atomic density (ADN), number of hydrogen bonds to other water molecules (HBDW) and average B-value of neighboring protein atoms (ABVAL), which are included in very few of the top 21 weight sets, are likely to be omitted either because they are not as closely associated with water binding, or because they are closely related to another included feature, and thus providing redundant information. Because the EC cost function penalizes for each feature included in the classification, the EC search is directed towards a minimal feature set that provides good classification accuracy. It is interesting to note that atomic hydrophilicity (AHP) and mobility are nearly mutually exclusive, suggesting that some information needed for classification may be provided by both of these features. The linear correlation coefficient for AHP and MOB is $r = -0.374$.

It is not surprising that the features that indicate conserved solvation across multiple structures differ somewhat from those that identify likely solvation sites in a single structure. Since both conserved and non-conserved water molecules are present in at least one crystallographic structure, they might be expected to have similar values for

the features associated with bound water molecules in a single crystallographic structure. Although the task appears to be more difficult than identifying water-binding sites, the GA/knn classifier obtained a set of features which further distinguishes those sites that are likely to be conserved across multiple structures. The features thus identified include B-value (BVAL), which appeared in 19 of the top 21 weight sets, and was the highest-weighted feature in 15 of these, and mobility (MOB), appearing in 11 weight sets, always as the first or second most strongly-weighted feature. Also selected in 9 weight sets, including the set which obtained the best overall accuracy in bootstrap testing, was the number of hydrogen bonds to protein atoms. The best performing weight set included these three features as well as the number of hydrogen bonds to other water molecules.

The best bootstrap accuracy obtained by the GA/knn was 64.2%, with a mean bootstrap balance (difference in predictive accuracy between conserved and non-conserved solvation sites) of 3.88%. This accuracy is a notable improvement over the near random classification obtained by linear discriminant analysis. Further improvement in accuracy can be obtained by using the k-nearest-neighbors vote tally as a measure of classification confidence. Intuitively, one would expect that a site for which the knn vote was definitive (say, 38 votes for conserved and 1 vote for non-conserved) was more likely to be correctly classified than one for which the vote was more evenly split (for example, 20 votes for conserved and 19 for non-conserved), and this intuition was supported by knn voting results. A weighted knn classifier was used with the top weight set from Table 5.4 to classify as conserved or non-conserved

all water-binding sites from the data set that were not selected as training or tuning sites for this GA run. A total of 3942 solvation sites were tested. Figure 5.4 shows the relationship between number of votes in the majority category, m, and the predictive accuracy for all test sites where the number of votes cast in the majority was $\geq m$. The correlation coefficient between m and cumulative predictive accuracy was $r = 0.905$. As demonstrated by the figure, the knn classifier exhibits a strong tendency toward more accurate classification of sites for which the voting consensus is more definitive. Greater predictive accuracy can thus be obtained by allowing "don't know" classifications when the number of majority votes does not exceed a specified cutoff value, and the prediction is thus less likely to be correct. Also shown in the figure is the number of test sites for which the number of majority votes meets or exceeds each value of m. For example, by classifying only sites where 43 or more votes were cast for the same class we can classify 1344 of the 3942 test sites with a predictive accuracy of 70.46%. While some sites are eliminated from classification, those that remain are classified with greater accuracy.

5.4 Discussion

In distinguishing solvation sites from non-solvated sites near the protein surface, the classification accuracy of the GA/knn hybrid classifier is comparable to other contemporary methods for solvent-site prediction, but the GA/knn does not have a tendency to overpredict solvation—a feature common to other techniques. For the problem of distinguishing conserved from non-conserved water-binding sites, the accuracy of the

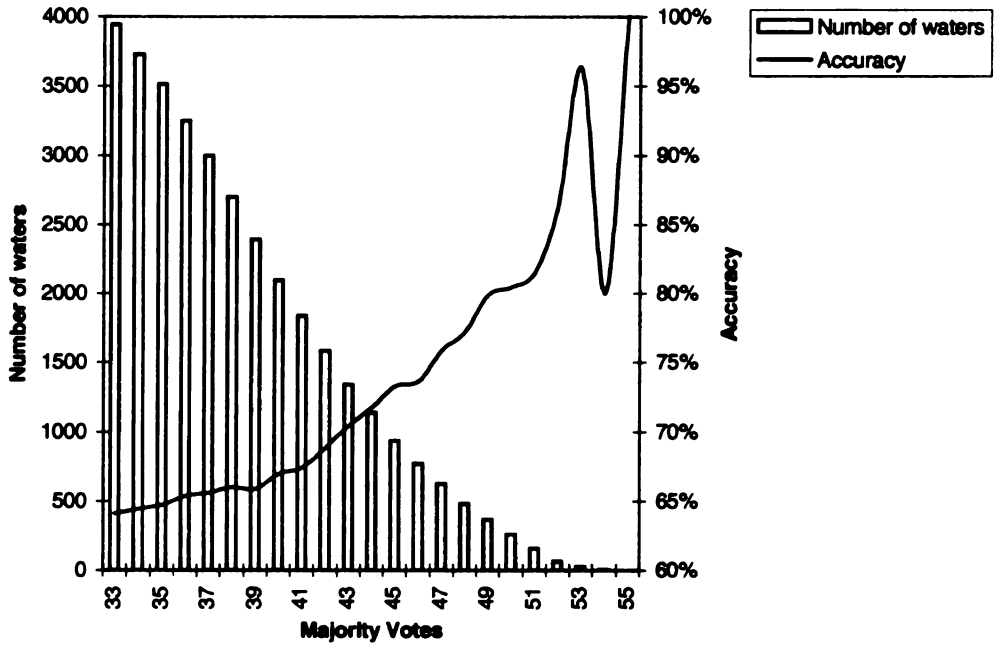


Figure 5.4: Relationship between majority votes and accuracy. For each possible number of majority votes, m, (x-axis) the solid line shows the cumulative predictive accuracy for all test sites for which m or more votes were cast in the majority. This accuracy value corresponds to the scale on the right y-axis. The outlined rectangles show the actual number of sites for which m or more votes were cast in the majority, corresponding to the scale on the left y-axis. Since the k-value for the weight set tested was 65, the number of majority votes ranges from 33 (the smallest possible majority) to 65.

GA/knn surpassed that of other classical classification techniques tested. In addition, the consistency in feature weighting exhibited for both problems provides some insight into the determinants of water binding and water conservation in crystallographic protein structures. In the case of predicting solvation sites, the determinants identified by the GA/knn experiments correspond well with other computational and experimental results.

Taken together, the results of both sets of experiments paint a general picture of the physical and chemical features related to water binding and conservation. From these results, we can envision a continuum of water binding favorability based on various features. Local atomic hydrophilicity, atomic density, and the hydrogen bonding potential of local protein atoms are good indicators of likely solvation sites. Of those sites that do bind water molecules, a low crystallographic temperature factor, high occupancy, and a large number of hydrogen bonds to neighboring protein atoms indicate that the water molecule is likely to be conserved between independently determined structures.

Bibliography

Bibliography

Abola, E. E., Bernstein, F. C., Bryant, S. H., Koetzle, T. F., and Weng, J. (1987). *Protein Data Bank*, pages 107–132. Data Commission of the International Union of Crystallography, Bonn/Cambridge/Chester.

Aeberhard, S., Coomans, D., and de Vel, O. (1992a). The classification performance of RDA. Technical Report 92-01, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

Aeberhard, S., Coomans, D., and de Vel, O. (1992b). Comparison of classifiers in high dimensional settings. Technical Report 92-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In Palaniswami, M., Attikiouzel, Y., Marks, R., Fogel, D., and Fukuda, T., editors, *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, Piscataway, NJ.

Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1:1–23.

Bäck, T. and Schwefel, H.-P. (1996). Evolutionary computation: An overview. In *Proceedings of the Third IEEE Conference on Evolutionary Computation*, pages 20–29, Piscataway NJ. IEEE Press.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. Roy. Soc.*, 53.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, Jr., E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., and Tasumi, M. (1977). The Protein Data Bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542.

Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth, Pacific Grove.

Bremermann, H., Rogson, M., and Salaff, S. (1966). Global properties of evolution processes. In Pattee, H., Edlsack, E., Fein, L., and Callahan, A., editors, *Natural Automata and Useful Simulations*, pages 3–41. Spartan Books, Washington, D.C.

Cestnik, G., Konenenko, I., and Bratko, I. (1987). Assistant-86: A knowledge-elicitation tool for sophisticated users. In Bratko, I. and Lavrac, N., editors, *Progress in Machine Learning*, pages 31–45. Sigma Press.

Connolly, M. L. (1983). Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221:709–713.

Cover, T. M. and Campenhout, J. M. V. (1977). On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:657–661.

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

Craig, L., Sanschagrin, P. C., Rozek, A., Lackie, S., Kuhn, L. A., and Scott, J. K. (1998). The role of structure in antibody cross-reactivity between peptides and folded proteins. *J. Mol. Biol.*, 281(1):183–201.

Crosby, J. (1967). Computers in the study of evolution. *Sci. Prog. Oxf.*, 55:279–292.

De Jong, K. A. (1992). Genetic algorithms are NOT function optimizers. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, pages 5–18, Vail, Colorado.

Diaconis, P. and Efron, B. (1983). Computer-intensive methods in statistics. *Scientific American*, 248.

Domingos, P. and Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier. In Saitta, L., editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, San Francisco, CA. Morgan Kaufmann.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7:1–26.

Efron, B. (1982). The jackknife, the bootstrap, and other resampling plans. In *CBMS-NSF Regional Conf. Series in Applied Mathematics, no. 38. SIAM. 91.*

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7:179–188.

Fitzpatrick, P. A., Steinmetz, A. C. U., Ringe, D., and Klibanov, A. M. (1993). Enzyme crystal structure in a neat organic solvent. *Proc. Natl. Acad. Sci. USA*, 90:8653–8657.

Fogel, D., editor (1998). *Evolutionary Computation The Fossil Record.* IEEE Press, New York, NY.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution.* John Wiley, NY.

Fraser, A. (1957). Simulation of genetic systems by automatic digital computers. I. Introduction. *Austrailian J. Biological Sciences*, 10:484–491.

Fukunaga, K. and Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, pages 750–753.

Gates, G. W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-18:431–433.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, San Mateo, CA.

Goodman, E. D. (1996). An introduction to GALOPPs, the genetic algorithm optimized for portability and parallelism. Technical Report 96-07-01, Michigan State University.

Gschwend, D. A. and Karplus, P. A. (1991). Bound water visibility in protein structure determination by x-ray crystallography. Unpublished senior thesis.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:515–516.

Hobohm, U., Scharf, M., Schneider, R., and Sander, C. (1992). Selection of representative protein data sets. *Protein Sci.*, 1:409–417.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI.

Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(3):260–270.

Jain, A. K. and Chandrasekaran, B. (1982). Dimensionality and sample size considerations in pattern recognition in practice. In Krishnaiah, P. R. and Kanal, L. N., editors, *Handbook of Statistics*, volume 2, pages 835–855. North-Holland.

Jain, A. K., Dubes, R. C., and Chen, C. C. (1987). Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):628–633.

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

Jaynes, E. T. (1968). Prior probabilities. In *IEEE Transactions on Systems Science and Cybernetics*, volume SSC-4, pages 227–241.

Karplus, P. A. and Faerman, C. H. (1994). Ordered water in macromolecular structure. *Curr. Opin. Struct. Biol.*, 4:770–776.

Kelly, J. D. and Davis, L. (1991). Hybridizing the genetic algorithm and the k nearest neighbors classification algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, pages 377–383.

Koza, J. R. (1992). *Genetic Programming: On the Programming of a Computer by Means of Natural Selection.* MIT Press.

Kuhn, L. A., Siani, M. A., Pique, M. E., Fisher, C. L., Getzoff, E. D., and Tainer, J. A. (1992). The interdependence of protein surface topography and bound water molecules revealed by surface accessibility and fractal density measures. *J. Mol. Biol.*, 228:13–22.

Kuhn, L. A., Swanson, C. A., Pique, M. E., Tainer, J. A., and Getzoff, E. D. (1995). Atomic and residue hydrophilicity in the context of folded protein structures. *Proteins: Str. Funct. Genet.*, 23:536–547.

Ladbury, J. E. (1995). Counting the calories to stay in the groove. *Structure*, 3:635–639.

Levitt, M. and Park, B. H. (1993). Water: now you see it, now you don't. *Structure*, 1(4):223–226.

Mao, J., Mohiuddin, K., and Jain, A. K. (1994). Parsimonious network design and feature selection through node pruning. In *Proc. of the Intl. Conf. on Pattern Recognition*, pages 622–624, Jerusalem.

Marchand, A., Lente, F. V., and Galen, R. (1983). The assessment of laboratory tests in the diagnosis of acute appendicitis. *American Journal of Clinical Pathology*, 80(3):369–374.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta.*, 405:442–451.

McRee, D. E. (1992). A visual protein crystallographic software system for X11/XView. *J. Molecular Graphics*, 10:44–46.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT Press.

Narendra, P. M. and Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26:917–922.

Nozaki, K., Ishibuchi, H., and Tanaka, H. (1996). Adaptive fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 4(3):238–250.

Palma, M. U., Palma-Vittorelli, M. B., and Parak, F., editors (1993). *Water-Biomolecule Interactions*, volume 43 of *Italian Physical Society Conference Proceedings*. Editrice Compositori, Bologna, Italy.

Parzen, E. (1962). On the estimation of a probability density function and the mode. *Ann. Math Statistics*, 33:1065–1076.

Pitt, W. R., Murray-Rust, J., and Goodfellow, J. M. (1993). AQUARIUS2: Knowledge-based modeling of solvent sites around proteins. *J. Comp. Chem.*, 14(9):1007–1018.

Poornima, C. S. and Dean, P. M. (1995a). Hydration in drug design. 1. Multiple hydrogen-bonding features of water molecules in mediating protein-ligand interactions. *Journal of Computer-Aided Molecular Design*, 9:500–512.

Poornima, C. S. and Dean, P. M. (1995b). Hydration in drug design. 2. Influence of local site surface shape on water binding. *Journal of Computer-Aided Molecular Design*, 9:513–520.

Poornima, C. S. and Dean, P. M. (1995c). Hydration in drug design. 3. Conserved water molecules at the ligand-binding sites of homologous proteins. *Journal of Computer-Aided Molecular Design*, 9:521–531.

Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1,119–1,125.

Punch, W. F., Goodman, E. D., Pei, M., Chia-Shun, L., Hovland, P., and Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. In *Proc. International Conference on Genetic Algorithms 93*, pages 557–564.

Quinlan, J. R. (1986a). The effect of noise on concept learning. In Michalski, R., Carbonnell, J., and Mitchell, T., editors, *Machine Learning: an Artificial Intelligence Approach*, pages 149–166. Morgan Kaufmann.

Quinlan, J. R. (1986b). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, pages 221–234.

Quinlan, J. R., Compton, P. J., Horn, K. A., and Lazurus, L. (1986). Inductive knowledge acquisition: A case study. In *Proceedings of the Second Australian Conference on Applications of Expert Systems*, Sydney, Australia.

Raymer, M. L., Sanschagrin, P. C., Punch, W. F., Venkataraman, S., Goodman, E. D., and Kuhn, L. A. (1997). Predicting conserved water-mediated and polar ligand interactions in proteins using a k-nearest-neighbors genetic algorithm. *J. Mol. Biol.*, 265:445–464.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart.

Reed, J., Toombs, R., and Barricelli, N. (1967). Simulation of biological evolution and machine learning: I. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *J. Theoret. Biol.*, 17:319–342.

Ringe, D. (1995). What makes a binding site a binding site? *Curr. Opin. Struct. Biol.*, 5:825–829.

Sanschagrin, P. C. and Kuhn, L. A. (1998). Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Sci.*, 7(9).

Schnecke, V. and Kuhn, L. A. (2000). Virtual screening with solvation and ligand-induced complementarity. *Perspectives in Drug Discovery and Design*, 20:1–22.

Schraudolph, N. N. and Grefenstette, J. J. (1992). A user's guide to GAucsd. Technical Report CS92-249, Computer Science Department, University of California, San Diego, CA.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie.* Birkhäuser, Basel.

Siedlecki, W. and Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10:335–347.

Sigillito, V. G., Wing, S. P., Hutton, L. V., and Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*, pages 261–265. IEEE Computer Society Press.

Smith, R. E., Goldberg, D. E., and Earickson, J. A. (1991). SGA-C: A C-language implementation of a simple genetic algorithm. Technical Report 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Department of Engineering Mechanics, Tuscaloosa, AL 35487.

Trunk, G. V. (1979). A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:306–307.

Vafaie, H. and De Jong, K. (1998). Evolutionary feature space transformation. In Liu, H. and Motoda, H., editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, chapter 19, pages 307–323. Kluwer Academic Publishers, Norwell, MA.

Vedani, A. and Huhta, D. W. (1991). An algorithm for the systematic solvation of proteins based on the directionality of hydrogen bonds. *J. Am. Chem. Soc.*, 113:5860–5862.

Wade, R. C., Clark, K. J., and Goodford, P. J. (1993). Further developments of hydrogen bond functions for use in determining energetically favorable binding sites on molecules of known structure. *J. Med. Chem.*, 36:140–147.

Weiss, S. and Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 781–787, Detroit, MI. Morgan Kaufmann.

Weiss, S. and Kapouleas, I. (1990). *An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods.* Morgan Kaufmann.

Westhof, E., editor (1993). *Water and Biological Macromolecules.* Topics in Molecular and Structural Biology. CRC Press Inc., Boca Raton, FL.

Whitney, A. (1971). A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 20:1100–1103.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2:408–421.

Yang, J. and Honavar, V. (1998). Feature subset selection using a genetic algorithm. In Liu, H. and Motoda, H., editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, chapter 8, pages 117–136. Kluwer Academic Publishers, Norwell, MA.

Yang, J., Parekh, R., and Honavar, V. (1998). DistAI: an inter-pattern distance-based constructive learning algorithm. In *Proceedings of the International Joint Conference on Neural Networks*, Anchorage, Alaska.

Zhang, X.-J. and Matthews, B. W. (1994). Conservation of solvent-binding sites in 10 crystal forms of T4 lysozyme. *Protein Sci.*, 3:1031–1039.