University of Alberta

Library Release Form

Name of Author: Luca Pireddu

Title of Thesis: Pathway Analyst—Automating Biochemical Pathway Prediction

Degree: Master of Science

Year this Degree Granted: 2005

Luca Pireddu
1455 Marcel St.
Sudbury, Ontario, Canada
P3E 4G5

Date: _____

University of Alberta

Pathway Analyst—Automating Biochemical Pathway Prediction

by

Luca Pireddu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2005

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Pathway Analyst—Automating Biochemical Pathway Prediction** submitted by Luca Pireddu in partial fulfillment of the requirements for the degree of **Master of Science**.

_____

Duane Szafron

_____

Mike Deyholos

_____

Paul Lu

_____

Russ Greiner

Date: _____

Abstract

Pathways are crucial to our understanding of biology. The speed at which new organisms are being sequenced is outstripping our ability to experimentally determine their biochemical pathway information. In recent years several initiatives have been successful in automating the annotations of individual proteins in these organisms, either experimentally or by prediction. However, to leverage the success of pathways as an abstraction of complex biological processes we need to automate their identification in the rapidly growing list of sequenced organisms. This dissertation presents a prototype system for predicting the catalysts of important reactions and for organising the predicted catalysts and reactions into previously defined biochemical pathways. It compares a variety of predictors that incorporate sequence similarity (BLAST), hidden Markov models (HMM) and Support Vector Machines (SVM). It is shown that there is an advantage to using different predictors for different reactions. The prototype is validated on 10 metabolic pathways across 13 organisms and achieves a cross-validation precision of 71.5% and recall of 91.5% in predicting the catalyst proteins of all reactions.

Acknowledgements

A special thanks to my family for all their neverending love and support. I'm coming home now. I would like to thank my friends for keeping my spirits high during my challenging times.

I am grateful for the guidance that Dr. Duane Szafron and Brett Poulin provided over the course of my research, and for the discussions and suggestions provided by the entire Proteome Analyst team: Paul Lu, Russ Greiner, Roman Eisner, Brandon Pearcy, Kurt McMillan, Alona Fyshe, and Jordan Patterson.

Finally, I appreciate the effort and generosity of the open source software community. All the work pertaining to this dissertation was done and is being presented using freely available software tools.

This research was partially funded by graduate scholarships from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Informatics Circle of Research Excellence (iCORE), and the University of Alberta. Additional funding was provided by the Canadian Protein Engineering Network of Centres of Excellence (PENCE), and the Alberta Ingenuity Centre for Machine Learning (AICML).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the complex processes of organisms has been a long-standing challenge for biologists. Modern gene sequencing technologies have opened a new window into the biological world, which can help us understand biological systems using their very own language: DNA. However, the vast amounts of data represented by organisms' genomes prompts a need to put computational techniques to work to help with its interpretation. This dissertation presents a number of contributions that take another step in assisting biologists' investigative work with automated intelligent computational systems.

1.1 Mechanics of the cell

In the process of life, cells perform a myriad of functions. Like miniature factories furnished with assembly lines, cells are continuously assembling and dismantling molecules. Also like real-world assembly lines, the cell's processes accomplish their tasks in series of small steps: reactions.

A biochemical reaction is a transformation of one set of a molecules into another. The biochemical reactions that occur inside the cell tend to be different from the chemical reactions that occur outside of living organisms, because they are often transformations that chemistry is not eager to perform without some encouragement. Yet, nature has devised delicate ways to bring about the molecular transformations necessary for life, through the use of sophisticated *catalysts*.

A catalyst is a substance that facilitates a reaction, without being transformed or consumed by the process. In the biological world, catalysts normally take the form of *proteins*, or groups of proteins acting in unison as a *protein complex*. They are so essential to carrying out the cell's reactions that their presence alone can be used to discriminate between organisms where a particular reaction can exist, and ones where it cannot; this last point is important

for the techniques that are presented in the later chapters of this dissertation.

1.2 Enzyme classification

A particular class of catalyst proteins, known as *enzymes*, is responsible for catalysing transformations of molecules from one form to another through *enzymatic reactions* [27]. Enzyme proteins are labelled with the *enzyme classes* that specify the particular types of catalysis they are able to perform. There are wide variety of enzyme classes and they have been organised by the Biochemical Nomenclature Committee into the Enzyme Classification (EC) system [5]. The EC system is a nomenclature that organises enzyme classes in a hierarchy four levels high. Each class is identified by an EC number consisting of four numbers separated by periods, where each number identifies the particular classification at the corresponding level of the hierarchy. For example, the EC number 1.8.1.4 specifies an enzyme class that is an:

1. oxidoreductase,
   8. acting on a sulfur group of donors,
      1. with NAD+ or NADP+ as acceptor;
         4 specifically: dihydrolipoyl dehydrogenase.

Enzymatic reactions are also labelled with enzymatic classes. For reactions, the classes specify in which ways the reaction can be catalysed. There could be many reactions that fit the same enzyme class, and several enzyme classes can match one reaction. All metabolic reactions—i.e. the ones relating to the synthesis and degradation of molecules—are enzymatic, and thus can be annotated with at least one enzyme class. Enzyme classes provide a way to match enzymatic reactions with the proteins that are capable of catalysing them.

Number	**Class name**
EC 1	Oxidoreductases
EC 2	Transferases
EC 3	Hydrolases
EC 4	Lyases
EC 5	Isomerases
EC 6	Ligases

Table 1.1: The top-most classes of the enzyme classification hierarchy [5].

1.3 Biochemical pathways

The ensemble of an organism's biochemical processes consist of a map of chemical reactions, each catalysed by special purpose proteins. The complexity of these systems motivated the creation of an abstraction to provide a simpler view of their complex network: the pathway. Pathways are a way to segment the map of the biochemical processes—metabolic, signalling, etc.—into logical sections of related reactions. Each pathway contains two kinds of information: the *pathway structure* and the *pathway components*.

The pathway structure is the topology or map that defines the relationships between the reactions in the pathway, along with the compounds (small molecules) that are their reactants and products. For example, Figure 1.1 shows the relationship between seven reactions that are part of the Gluconeogenesis metabolic pathway [22]. Reactions are shown in the rectangular boxes, while passive compounds are in the ovals. For example, the reactant of reaction 1 is Lipoamide and the product is Dihydrolipoamide, while the catalyst protein (enzyme) is a Dihydrolipoyl dehydrogenase (labelled by the enzyme classification number of this enzyme class, EC 1.8.1.4). Some reactions have more than one reactant or product. For example, reaction 3 has two reactants (2-Hydroxy-ethyl-Thpp and Lipoamide) and two products (ThPP and 6-S-Acetyl-dihydrolipoamide). Other reactions have more than one class of catalysts. In such cases, catalysts from any of these classes can catalyse the reaction. For instance, reaction 4 has two potential catalyst classes: Pyruvate dehydrogenase (EC 1.2.4.1) and Pyruvate decarboxylase (EC 4.1.1.1).

The pathway components are the specific proteins that catalyse each of the pathway's reactions in a specific organism. For instance, reaction 1 is catalysed by protein hsa:1737(DLD) [1] in the organism *H. sapiens* (NCBI-GI 4557525, Uniprot P09622) and by protein cel:LLC1.3 in the organism *C. elegans*. Sometimes more than one protein from the same enzyme class can catalyse the same reaction in a single organism. For example, in the organism *A. thaliana*, reaction 1 is catalysed by three proteins in the class EC 1.8.1.4: ath:At1g48030, ath:At3g16950 and ath:At3g17240.

Although two species may have similar metabolic pathways, evolution generates some organism-specific variations. We refer to the variants of a specific metabolic pathway across different organisms as *pathway instances*. Pathway instances can differ in their pathway components, as illustrated by the organism-specific proteins in a single enzyme class. However, pathway instances can also differ in their pathway structure. For example, in Figure 1.1,

[1] The notation for protein names is the one used by the Kyoto Encyclopedia of Genes and Genomes (KEGG) PATHWAY database. Each name consists of a three character organism identifier followed by an abbreviated gene name from the original source—e.g., NCBI, Wormbase, TAIR, TIGR, MIPS, etc.

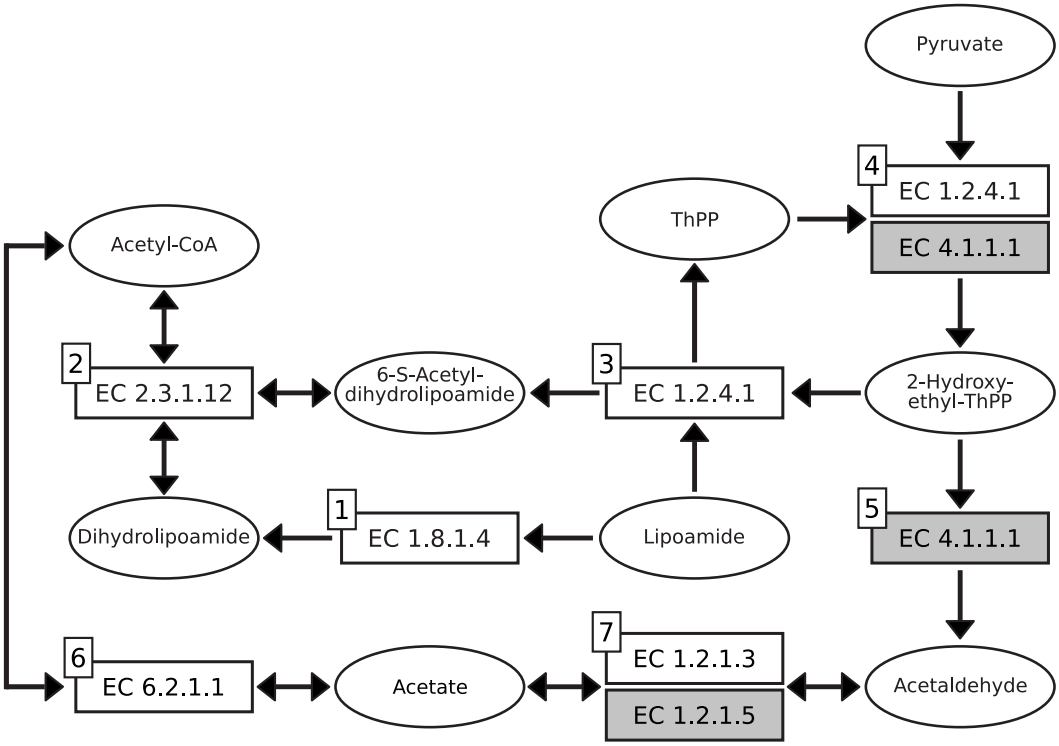


Figure 1.1: Seven reactions from the Gluconeogenesis pathway.

two of the enzyme classes denoted by gray boxes (EC 4.1.1.1) are present in the pathway instance for *A. thaliana*, but are not present in either *C. elegans* or *H. sapiens*. The other enzyme class in a gray box (EC 1.2.1.5) is present in *C. elegans* and *H. sapiens*, but is not present in *A. thaliana*. Since there is no catalyst for reaction 5 in *C. elegans* or *H. sapiens*, reaction 5 is not known to take place in either of these organisms so it is not part of the structure of their pathway instances.

There are different types of biochemical pathways. For instance, Figure 1.2 shows an excerpt from the Transforming Growth Factor–β (TGF–β) signalling pathway in *H. sapiens*. Signalling pathways are characterised by a chain of protein-protein interactions—phosphorylations, methylation, activation, binding, etc. In these pathways, in addition to being the components of reactions, proteins are usually also the reagents and products. The excerpt of the TGF–β pathway in Figure 1.2 shows reaction 9 where the Transforming Growth Factor beta (TGFβ) protein activates the Transforming Growth Factor beta Receptor 2 (TGFβ R2) (in the diagram we use "-A" to denote an activated state). In turn, TGFβ R2–A activates member A of the RAS homology family (RhoA) in reaction 10. That protein is then active in reaction 11, catalysing the ac-

tivation of a Rho-associated coiled-coil containing protein kinase 1 (ROCK1). Unlike in metabolic pathways, signalling pathways display molecules that are the products of some reactions, but are also the components of others.

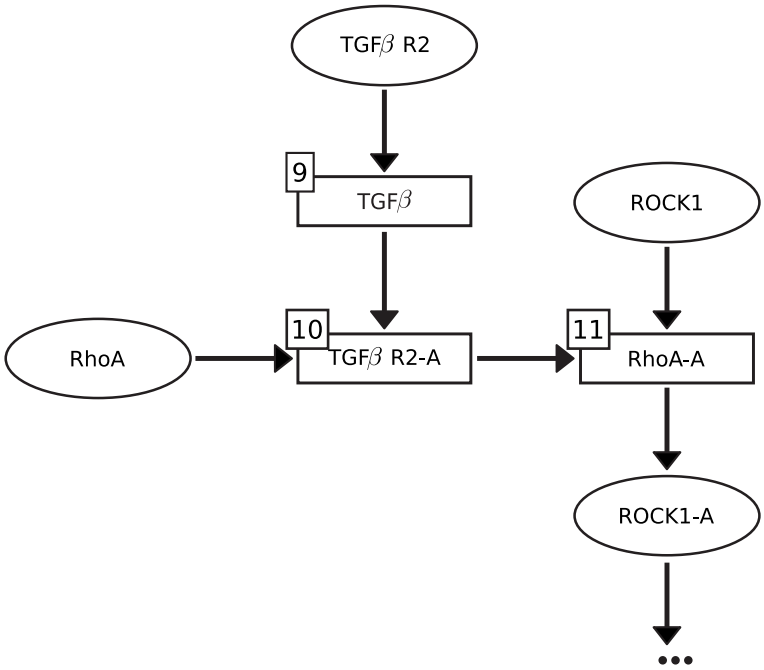


Figure 1.2: Excerpt of the TGF–β signalling pathway [17] in *H. sapiens*.

1.4 The problem

A major goal of biology is to understand the pathway instances of all known organisms. Biologists who study pathways tend to approach their work in one of two ways. The first approach is to study a particular pathway over many organisms. The second is to study one organism by trying to analyse all of its pathways. The current rate of knowledge acquisition prompts a broader systems approach. Genomic and proteomic sequence data is being generated so fast that analytical experimental methods to determine pathway structure and components cannot keep pace. To cope with this deluge of sequence data, an automated computational approach to the prediction of organism-specific biochemical pathways is proposed, as it could assist the study of many pathways in many organisms. The thesis of this dissertation is that the biological and sequence similarities between organisms can be exploited to predict the structure and the components of pathway instances. This dissertation makes four main research contributions:

1. It describes Pathway Analyst, a prototype high-throughput system that predicts pathway reactions and catalysts;

2. It demonstrates a simple but effective pathway prediction algorithm that incorporates machine learning techniques;

3. It provides empirical results that suggest the need for reaction-specific classifiers;

4. It validates the thesis that similarity is a powerful tool for predicting the structure and components of pathway instances.

1.5 Why predict pathways by similarity?

It is thought that the three domains of life stem from a common ancestor [24]. Because of their common origins, living organisms share some common traits. This observation leads to the hypothesis that it is possible to exploit the similarities between organisms, to project knowledge about the processes of life across species. More specifically, the similarities between organisms may encompass their biochemical pathways. Therefore, it should be possible to use well-understood pathway instances as models, to predict the structure and components of similar pathway instances in other organisms of interest.

1.5.1 Similarity of pathway structure across species

The structure of a pathway is relatively stable in evolution, and is often well-maintained even across species that have different phylogenies. It is a fairly large evolutionary step for an organism to develop a different structure in its metabolic system, to the point that the analysis of these mutations has been suggested as a means for investigating evolutionary trees [21].

1.5.2 Similarity of catalysts across species

Unlike the relative similarity of pathway structures that is seen across species, it is very common for different species to have proteins with different primary structure catalysing the same reaction. The differences between these proteins are normally spawned by mutations in the genome over the course of evolution. This phenomenon is a normal part of life, and is in fact a way in which the evolutionary process explores the genome space to find better solutions to nature's challenges.

As two species stemming from the same ancestor diverge in evolution, their genomes grow more dissimilar, and so do their proteins. However, because the

functional portions of a protein may be particularly important to the life of the organism, they are under greater selective pressure and thus are more likely than less important portions to be conserved over the course of evolution. The result is that the two diverging species may have *homologous* proteins, with identical functionality and similar—but not necessarily identical—amino acid sequences. These similarities in amino acid sequence can be detected by various techniques, such as BLAST [2] and hidden Markov models [8].

It should be noted that shared ancestry is not the only way several species can evolve to support the same biochemical reaction. Two species could independently evolve different proteins that catalyse the same reaction—i.e. independently arrive at different solutions to the same problem. This process is known as *convergent evolution*. Such proteins may be too dissimilar in primary structure for the techniques used in this dissertation to detect a common function.

Chapter 2

Sequence Classification Techniques

A binary classifier can be simply regarded as a separator in the problems input space. Elements that fall on one side of the classifier are in one class, while elements on the opposite side are in another class. There are many ways to formulate a separator. This chapter briefly describes the techniques that were used to classify proteins in the pathway prediction experiments explained later in this dissertation.

2.1 Basic Local Alignment Search Tool

Comparing protein sequences in a way that meaningfully reflects on their function is a tricky problem. One cannot simply compare each locus of two sequences and see if they are identical because the function of a protein can be resilient to many mutations such as insertions, deletions, and substitutions of elements. In addition, residues can sometimes be substituted without significant effect to the resulting protein. An approach to attain a meaningful comparison is to align the two proteins as well as possible. The alignment may require a number of mutations to the sequences—insertions, deletions, and substitutions. By assigning a cost to each of these actions, a total cost for aligning two sequences can be established. Then, the cost of alignment can be used as a similarity measure—the cheaper the alignment, the more similar the proteins.

When scoring an alignment one should consider the fact that not all amino acids are equally dissimilar, and therefore that not all substitutions in the alignment are equally undesirable. The varying compatibility between amino acids needs to be considered when scoring the matches and mismatches in the alignment. This can be achieved by using a *substitution matrix*. A substitution

matrix is a table that quantifies the effect of substituting one amino acid for another in a protein sequence. As an example, Figure 2.1 shows a section (amino acids A to G) of the BLOSUM-62 substitution matrix [14].

A	B	C	D	E	F	G	...	
4	-2	0	-2	-1	-2	0		A
	6	-3	6	2	-3	-1		B
		9	-3	-4	-2	-3		C
			6	2	-3	-1		D
				5	-3	-2		E
					6	-3		F
						6		G
								...

Figure 2.1: Section of the BLOSUM-62 substitution matrix [14]. Notice how the diagonal (a perfect match) always has a positive score. Most other substitutions incur a negative score on the alignment, but to a varying degree. On the other hand, some substitutions carry a positive score—for instance, aspartic acid (D) and glutamic acid (E).

The *Basic Local Alignment Search Tool* (BLAST) [2, 3] is a practical tool that calculates the local alignment of protein sequences (or DNA sequences) and summarizes the alignment quality as a score. A local alignment seeks relatively well-conserved regions of two proteins, even if small in comparison to the overall length of the protein sequences. In the context of this dissertaion, a local alignment is normally preferred to the global alignment—which seeks to align two entire sequences from beginning to end—since distantly related proteins may only be similar in a small section, perhaps near the active site. The standard algorithm for computing optimal local sequence alignments is the Smith-Waterman algorithm [32]. It uses a dynamic programming approach to compute the alignment, and although it works well, it has significant computational requirements which tend to limit its usefulness as the number of proteins to be compared—i.e. the size of protein database—grows. BLAST directly approximates the result of the Smith-Waterman algorithm, but significantly reduces computational requirements.

The idea behind BLAST's operation is the following. Given a pair of proteins, the BLAST algorithm searches them for *high-scoring segment pairs* (HSPs). A segment pair is a section of the alignment, while a *high-scoring* segment pair is a locally optimal segment pair whose score cannot be improved by extension or trimming. The authors of BLAST made the observation that a statistically significant HSP likely contains a short high-scoring aligned *word*

of about 3 residues. Therefore, they devised BLAST as the the following three-step algorithm.

The BLAST algorithm begins by segmenting the query protein into a list of query words, each of size w. By default, the size of the words is $w = 3$. It then uses the substitution matrix (usually the BLOSUM-62 matrix—see Figure 2.1) to construct a list of words that score above some threshold T when aligned with the query words.

The second step of the algorithm amounts to searching the target protein(s) for any of the short query words. This step can be accomplished relatively quickly. Any matches are stored as *hits*.

The final step of the process consists of extending the alignments of some of the hits found in the last step. The extension consists of continuing the hit's alignment on both sides. Since the alignment is extended with a dynamic programming algorithm, it is quite computationally expensive. It is therefore desirable to only perform the extension under favourable conditions. To reduce the number of futile hit extensions, Altschul et al. noted that a high-scoring segment pair of interest is much longer than a single word, and may consequently entail multiple hits within a short, ungapped window of the alignment. Therefore, a hit is extended only if one or more additional non-overlapping hits are found within a distance A. By carefully choosing the thresholds A and T BLAST finds most significant alignments (HSPs) while requiring a fraction of the time of a full-blown dynamic programming algorithm.

```
Query: 7   ILLKPESTRTQIDQIIDEAKAYKSVNPTHVKYAAERLASEVLVCTVIPLG 66
           ILL PE+T   + +++ +AK Y SV+P+ +      L  +V +  V P G
Sbjct: 15  ILLTPEATHNDVAKLVADAKKYWSVSPSMLPL---NLDGDVHLAVVCPSG 71
```

Figure 2.2: Section of BLAST's output from an alignment between Deoxyribose-phosphate aldolase in S. aureus (query, Uniprot ID P61108) and P. acnes (subject, Uniprot ID Q6A655). The center line outlines the alignment's scoring. The residue's letter is echoed where an exact match exists, while a '+' indicates that a substitution was made but its score is positive. On the other hand a blank indicates either a non-positive substitution (score $\leq$ 0) or a gap.

2.1.1 Scoring

The result of an alignment computed by BLAST is a raw score S that is calculated by summing the score in the substitution matrix for every aligned residue, plus any gap penalties incurred. The raw score is normalised to yield

the *bit score* S' according to the equation

$$S' = \frac{\lambda S - lnK}{ln2} \tag{2.1}$$

where λ and K are parameters calculated from the scoring system. Moreover, the number of HSPs with bit score of at least S' expected to be found when comparing two random protein sequences of sufficient lengths m and n is approximated by the equation

$$E = \frac{mn}{2^{S'}}. \tag{2.2}$$

When a protein is compared with a whole database rather than a single sequence the value of n becomes the size of the entire database in residues. The value of E is commonly referred to as the *e-value*, and can be used to quantify the similarity of two proteins; smaller e-values indicate more similarity.

2.2 Hidden Markov models

The hidden Markov model (HMM) is a statistical model of strings of elements from a known alphabet. It is well-suited to the problem of modelling proteins and DNA, which are sequences from the alphabets of the 20 amino acids and the four bases respectively. This section gives a brief explanation of how the HMM works using an sample problem of identifying the CpG islands in DNA taken from Durbin et al. [8], supplemented with information from Alpaydin's work [1]. For a more complete explanation of the properties and the algorithms for HMMs consult any of the numerous sources on Markov models. We will begin by explaining Markov chains, and then extend the explanation to describe a hidden Markov model.

2.2.1 Markov chains

A Markov chain is a stochastic system that at any given time $t = 1, 2, 3, \ldots$ is in one of a set of states $S = \{S_1, S_2, S_3, \ldots, S_N\}$. The state of the system at time t is denoted by q_t. At each new time $t + 1$, the system moves to its next state, which is chosen probabilistically depending on only the value of the current state q_t, and irrespective of the values of any previous states. Therefore, the probability of the next state q_{t+1} having a value S_a is:

$$\begin{aligned} &P(q_{t+1} = S_a | q_t = S_b, q_{t-1} = S_c, q_{t-2} = S_d, \ldots) \\ &= P(q_{t+1} = S_a | q_t = S_b) \text{ for } t \geq 0 \end{aligned} \tag{2.3}$$

In addition, the probabilities are independent of time, meaning that whenever the Markov chain is in state S_i a transition to state S_j is always equally likely. We can now define *transition probabilities* for the Markov chain as:

$$a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i) \tag{2.4}$$

subject to

$$a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1.$$

The fact that when choosing the next state of the system the past is not considered (only the current state matters) is a distinguishing feature of the Markov chain. Such a model is also known as a *first-order Markov model*.

Furthermore, as the Markov chain enters a state it emits a label corresponding to that state. As the model moves from state to state its emissions result in a sequence of observations O, which is composed of elements of the model's alphabet Σ. Because the Markov chain generates a sequence of labels it is said to be a generative model. There is a one-to-one correspondence between labels and states, so that by examining the emitted sequence of labels we know the Markov chain's state q_t for any time t along the sequence. Consequently, the Markov chain is said to be an observable Markov model.

The Markov chain can also be viewed as a *stochastic automaton* [1], which is a state machine with a complete set of probabilistic transitions. As an example, a stochastic automaton for Deoxyribonucleic Acid (DNA)is shown in Figure 2.3. The four states labelled A, C, G, and T correspond to the four different bases—adenine, cytosine, guanine, and thymine—that make up DNA. The states α models the start of a generated sequence, so the transitions starting from that state are assigned *initial probabilities* π_i that reflect the likelihood of a sequence starting at each labelled state i. On the other hand, the state ω models the end of a sequence, so there are no transitions leaving it.

Markov chains have applications in the realm of biological sequences analysis as discriminants, or classifiers, of DNA or proteins. In this sort of application, the Markov chain is constructed such that the set of states S is labelled with the alphabet of the sequences being analysed—either the four nucleic acids, or the 20 amino acids. Durbin et al. [8] give an example describing how a Markov chain can be used to classify sections of DNA as CpG islands. That example is summarised in the following paragraphs.

CpG islands are sections of DNA with a higher than average concentration of cytosine-guanine dinucleotides. They are biologically important because they often indicate the start region for a gene. To determine whether or not a slice of DNA is a CpG island we can use a Markov chain like the one in Figure 2.3. To use the Markov chain as a discriminant it is necessary to build

a chain that models the positive class (the CpG islands) and one that models the negative class (the rest of the DNA). The positive model is built such that its transition probabilities reflect those seen in sections of DNA known to be CpG islands. On the other hand, the negative model is built such that its transition probabilities estimate the average probabilities everywhere else in the DNA sequence. The probabilities can be estimated from a set of training sequences T using a maximum likelihood estimator:

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{i \in S} c{+}_{si}}. \tag{2.5}$$

The equation defines the transition probabilities a_{st}^{+} from a state s to a state t for the positive CpG class, where $c{+}_{st}$ is the number of times that state s is followed by state t in the positively labelled sequences from T. The transition probabilities a_{st}^{-} for negative model are set analogously, but counting the transitions in the negative training samples. Similarly, the maximum likelihood estimation of the initial probabilities for the positive class π_i^{+} can be written as:

$$\pi_i^{+} \equiv \sum_{x \in T^{+}} \frac{1(x_1 = q_i)}{|T^{+}|} \tag{2.6}$$

where $1(z)$ is 1 if z is *true* and 0 otherwise, and T^{+} is the positive subset of the training set. This equation simply computes the fraction of training sequences that start with state i's label. Of course, the initial probabilities for the negative model π_i^{-} are computed analogously.

Once the positive and negative Markov chains are built, we can use them to classify sequences as part of the positive or negative CpG island classes. Given a base sequence x to be tested, we classify it by calculating the log-odds ratio for x as:

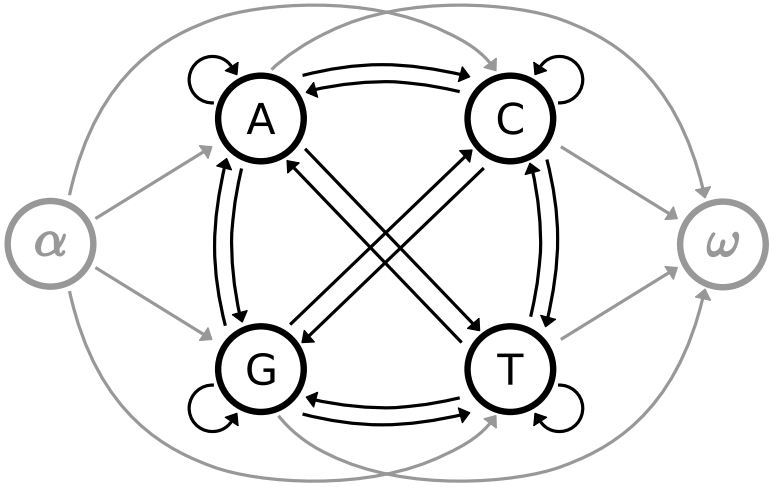
$$\begin{aligned} S(x) &= log \frac{P(x|model+)}{P(x|model-)} \\ &= log \frac{\pi_{x_1}^{+}}{\pi_{x_1}^{-}} + \sum_{i=2}^{|x|} log \frac{a^{+}(x_{i-1}, x_i)}{a^{-}(x_{i-1}, x_i)} \end{aligned} \tag{2.7}$$

where $a^{+}(x, y)$ is the transition probability function for the positive model, and $a^{-}(x, y)$ is the transition probability function for the negative model. Finally, if $S(x) > 0$ then the sequence x is deemed more likely to come from the positive model, and thus be a CpG island. On the other hand of $S(x) < 0$ then x is deemed more likely to be generated by the negative model and thus is probably not a CpG island. For example, consider classifying the DNA sequence $x = AGGCGCT$ with the model shown in Figure 2.3 and initial

probabilities $\pi_i^+ = \pi_i^- = 0.25$. Then,

$$\begin{aligned}
S(x) &= S(AGGCGCT) \\
&= log\frac{P(AGGCGCT|model+)}{P(AGGCGCT|model-)} \\
&= log\frac{\pi_A^+}{\pi_A^-} + \left(log\frac{a_{AG}^+}{a_{AG}^-} + log\frac{a_{GG}^+}{a_{GG}^-} + log\frac{a_{GC}^+}{a_{GC}^-} + log\frac{a_{CG}^+}{a_{CG}^-} + log\frac{a_{GC}^+}{a_{GC}^-} + log\frac{a_{CT}^+}{a_{CT}^-} \right) \\
&= log\frac{0.25}{0.25} + \left(log\frac{0.426}{0.285} + log\frac{0.375}{0.298} + log\frac{0.339}{0.246} \right. \\
&\quad \left. + log\frac{0.274}{0.078} + log\frac{0.339}{0.246} + log\frac{0.188}{0.302} \right) \\
&= 2.06,
\end{aligned}$$

meaning that the sequence is predicted to be a CpG island.



	CpG					**Non-CpG**			
a^+	A	C	G	T	a^-	A	C	G	T
A	0.180	0.274	0.426	0.120	A	0.300	0.205	0.285	0.210
C	0.171	0.368	0.274	0.188	C	0.322	0.298	0.078	0.302
G	0.161	0.339	0.375	0.125	G	0.248	0.246	0.298	0.208
T	0.079	0.355	0.384	0.182	T	0.177	0.239	0.292	0.292

Figure 2.3: A Markov chain for DNA with transition probabilities for the CpG and non-CpG models from Durbin et al. [8].

2.2.2 The hidden Markov model

In a Markov chain there is a one-to-one correspondence between the states in the model and the alphabet of its emitted label sequence. This is a limiting factor when tackling some problems. For instance, what if we want to identify all the CpG islands in a long string of DNA—as opposed to the previous example where the task is to determine whether a given section is a CpG island—the simple Markov chain will not suffice. What is required is a model capable of emitting the same labels while in more than one state—for example, cytosine in a CpG island in contrast to cytosine outside of a CpG island. That model is the *hidden Markov model*. Indeed, the key difference between a Markov chain and a hidden Markov model is that the hidden model relinquishes the one-to-one correspondence between the set of states and the set of state labels—i.e. elements in the alphabet. As a consequence of this change, it is no longer possible to positively determine the state of the model given a label observation. Hence, the model is no longer observable, and is therefore known as *hidden*.

An application of HMMs can be illustrated using the CpG island example. Conceptually, we can think of the hidden Markov model as merging multiple individual Markov chains that model disjoint states of a sequence into the same statistical model. For the purposes of identifying CpG islands, a DNA sequence can be thought to have two states: in a CpG island, and not in a CpG island. These states correspond to the positive class and negative class Markov chains described in the previous subsection. Thus, a suitable HMM to identify CpG islands could merge those two Markov chains, which become the HMM's hidden states. When the Markov chains are merged, they are connected by the addition of transitions that give the model a small but finite chance of switching from one chain to the other [8]. These transitions between hidden states typically connect all pairs of observable states, though inadmissible transitions can be given a probability of 0.

We now make an assumption regarding the transitions: the probability of transitioning into a particular observable state is only dependent on the current hidden state—not the current observable state. We can then replace the sub-models in each hidden state by the emission probability distribution of the labels, defined conditionally on the hidden state. The resulting hidden Markov model is shown in Figure 2.5.

More formally, each time a hidden Markov model λ visits a state it emits an observable label from its alphabet Σ. The process generates a sequence of observations O, while its true state sequence Q is hidden. The HMM is composed of the following elements:

- A set of (hidden) states $S = \{S_1, S_2, S_3, \ldots, S_N\}$

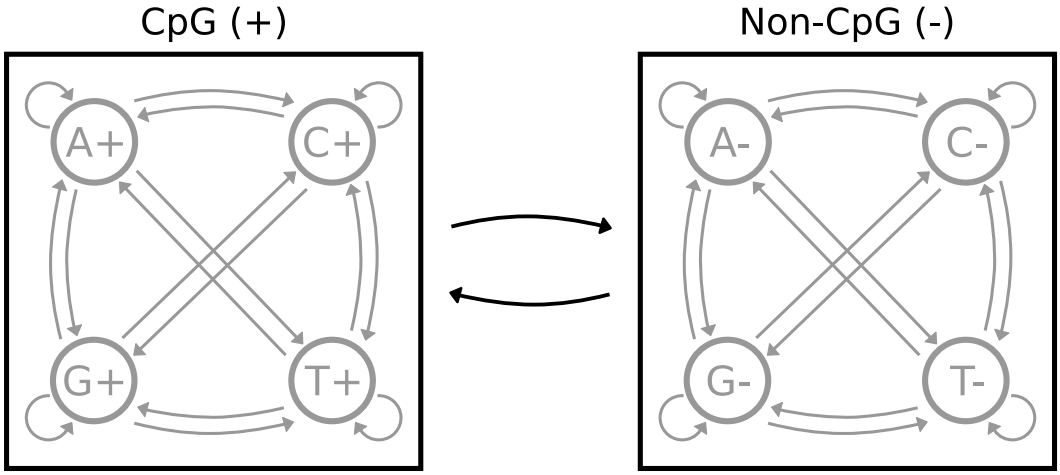


Figure 2.4: Conceptual model of an HMM of DNA with two hidden states outlined by the boxes. The CpG hidden state models the CpG islands in the DNA strand, while the Non-CpG state models the rest of the DNA. Each hidden state operates analogously to the individual Markov chains described in the previous section. In addition, there are transitions connecting the two hidden states, so that each pair of observable states is connected (though the connections are not shown for brevity).

- An alphabet of labels $\Sigma = \{v_1, v_2, v_3, \ldots v_M\}$
- State transition probabilities $a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i)$
- Emissive probabilities $e_j(m) \equiv P(O_t = m | q_t = S_j)$, $m \in \Sigma$
- Initial state probabilities $\pi_i = P(q_1 = S_i)$

Hidden Markov models have many applications. One of them lies in labelling each element of a sequence of observations with the hidden state that generated it. In keeping with the CpG island example, suppose we have a long sequence of bases, and we would like to identify any CpG islands within it. We assume that the DNA was generated by a Markov process that is modelled by the HMM in Figure 2.5. Our goal is to determine for each base in the DNA sequence which hidden state generated it. Although the HMM's state sequence Q is not observable, the DNA sequence O is its probabilistic function. Therefore, O should allow us to infer Q. There usually are a number of paths through the HMM that generate the same observation sequence, but each with a different probability. Therefore, we would like to find the path through the HMM that with the highest likelihood generates the given DNA sequence. This task is performed using an algorithm such as the Viterbi algorithm or

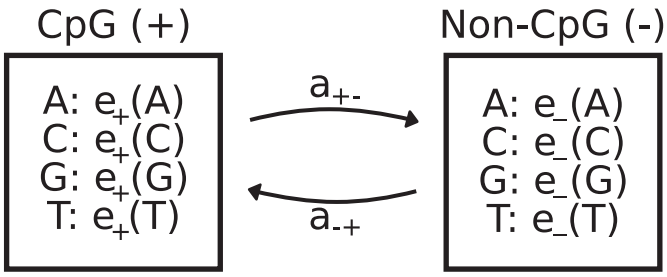


Figure 2.5: An HMM model for CpG islands in DNA. "CpG" and "Non-CpG" denote the hidden states, while $e_+(l)$ and $e_-(l)$ are the emissions probabilities of the labels given the positive and negative hidden states respectively.

the Forward-Backward algorithm [1, 8]. The deduced path Q consists of one state per element in the given DNA sequence O. Therefore, we can label each element O_i with its corresponding state Q_i: CpG, or Non-CpG.

Figure 2.6: An example state sequence Q corresponding to an observed DNA sequence O. The bases generated by the HMM's CpG island state are marked by a "+".

Also, like the Markov chains, hidden Markov models can be used as classifiers. For instance, an HMM can be built to represent a particular family of proteins (the following subsection will expand on how this can be done). While deducing the HMM's most probable state path to generate a given protein's amino acid sequence, the probability that the sequence was generated by the HMM is also computed. This probability can be used to classify the protein as a member or a non-member of the modelled protein family.

Whatever the application, before being used HMMs have to be trained—namely, we need to set its transition and emission probabilities. For this a training set T of sequences of the process we intend to model is required. If the sequences in T are labelled with their generating state sequences, then we can count the number of times particular transitions and emissions occur in the training data. Let A_{ij} be the number of number of times the transition from state i to state j occurred in the training data, and let $E_i(m)$ be the number of times state i emitted the label m. We can now use maximum likelihood

estimators for a_{ij} and $e_i(m)$ [8]:

$$a_{ij} = \frac{A_{ij}}{\sum_{s \in S} A_{is}} \tag{2.8}$$

and

$$e_i m = \frac{E_i(m)}{\sum_{l \in \Sigma} E_i(l)} \tag{2.9}$$

The state transition probabilities are estimated in the same manner as for Markov chains. The emission probabilities are estimated by dividing the number of times the state i emitted the label m by the total number of emissions by state i found in the training data.

In the case that the training sequences are not labelled with their generating states there is no closed-form equation to estimate the HMM's parameters. Instead, we are left to use iterative techniques. Although any standard algorithm for continuous function optimisation can be used, the standard is to use the Baum-Welch algorithm [8]—a special case of the expectation maximisation (EM) algorithm.

2.2.3 Profile hidden Markov models

Profile hidden Markov models are structured to take into consideration the positional information represented in sequences. Positional information can be important in biological applications. Evolution places different selective pressure on different protein residues, leading to the conservation of important sections or patterns of a protein throughout generations. These conserved protein sections are sometimes referred to as *domains* or *motifs*. Proteins can usually be classified into groups, or *families*, that exhibit the same motif. These groupings are important because as a consequence of sharing a functional domain the proteins are able to perform a similar function.

Motifs can be extracted from a protein family by performing a multiple alignment of the proteins' amino acid sequences with a tool such as ClustalW [33]. The resulting consensus sequence highlights positions within the alignment possessing the same or functionally compatible residues. Profile HMMs are hidden Markov models designed to capture the information provided by a multiple sequence alignment into a position-specific scoring model that can be used to recognise the presence of motifs in a given protein.

HMMER [11] is an implementation of a profile HMM. It is of particular interest in this dissertation because it was used as a classifier in a number of the experiments presented in the following chapters. The structure of HMMER's hidden Markov model is shown in Figure 2.8. For each column of the multiple alignment, the HMMER architecture includes three states: a match state (M),

```
1  SWDFAYAGKASRRPILASVGSYGAYLADGSEYSGIYGDAGASLFGGCCRTTPSAASK
2  SS--MDDKILKKRPILASVGSYGAYLADGSEYSGIYGDAGVSLLGGCCRTTRSL---
3  SG-------HSYNRALASIGSYGAYLADGSEYSGHYGELGAKLIGGCCRTT------
4  GG------------LSASVGPYGAALADGSEYRGCY--AGARIVGGCCRVRPPRG--
5  GT------------LLGSVGPYGAYLADGSEYRGDY--AGARLIGGCCRTT------
                 .*:*.*** ******* * *   *. :.*****.
```

Figure 2.7: A section of a multiple alignment of five proteins generated with ClustalW. The bottom line shows the quality of the alignment at each column:
<blank>: no match
"." weak match;
":" strong match;
"*": exact match.
The length of matches along the alignment is probably a conserved, homologous section of the proteins.

an insert state (I), and a delete state (D). The match state models the distribution of residues seen in the column. The insert state allows for the insertion of any number of residues between the matching column and the following match. Finally, the delete state allows the model to skip a particular residue of the pattern and advance to the following column. The I and D states allow the profile HMM to model patterns of variable length [29]. In addition, the architecture forms a probabilistic model of both local and global alignments, and also accommodates repeated protein motifs within the same sequence.

HMMER provides a suite a programs to build profile HMMs from multiple sequence alignments, and use those models to search a protein database for matching proteins. When scanning a protein database HMMER uses the Viterbi algorithm to evaluate the likelihood of a protein being generated by its profile HMM [10]. The proteins it deems as possible matches are returned with an accompanying e-value, which is the estimated number of false positives to arise from the scanned data set with the same probability of being generated by the model as the match in question. In short, this number is a measure of the confidence that the tested sequence possesses the domain modelled by the HMM, with smaller numbers indicating more confidence.

The Pfam database

The Pfam database [4] is collection of protein families (7973 in August 2005 [28]). Each family is complete with a multiple alignment of all its members, and a profile HMM that models it. The Pfam HMMs can be used to determine

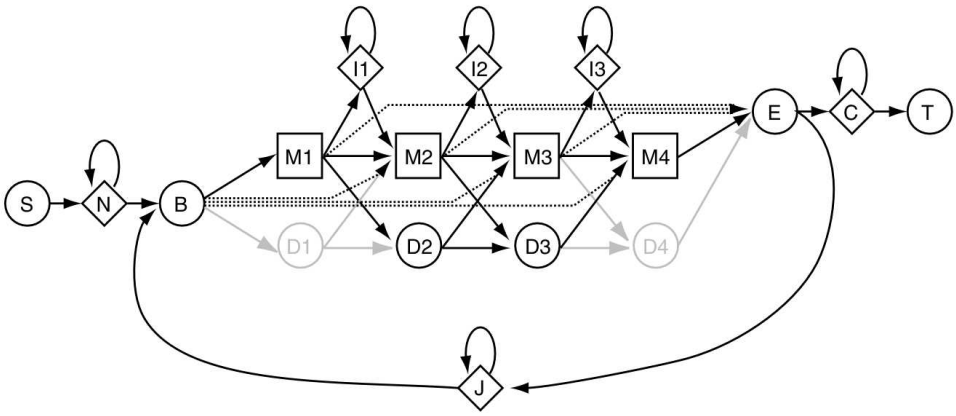


Figure 2.8: Architecture of HMMER's HMM. Squares indicate match states, diamonds indicate insert states, and circles indicate silent delete states. States S and T delimit the start and end of the analysed sequence, while B and E delimit the start and end of the aligning section. The transition $E \rightarrow J \rightarrow B$ allows for the repetition of the motif along the sequence. Finally, the insertion states N and C make it possible to build a model that places little importance on sections of the sequence preceding and following the motif, much like a local alignment. This diagram is taken from the HMMER manual [10].

whether a given protein contains the domain that defines the family. This is an important feature extraction tool. Chapter 4 of this dissertation discusses how one can use knowledge about the presence of specific domains in a protein—generated using the Pfam databases profile HMMs—to predict biochemical pathways.

2.3 Support vector machines

Like some other classification techniques, support vector machines (SVM) form a linear separator in the problem's feature space. However, SVMs distinguish themselves in the ingenious, yet intuitive way the separating hyperplane is calculated. The approach taken by the SVM is to find a separator such that the distance between itself and the nearest training data points is maximised. The remainder of this section summarises the derivation and application of SVMs as presented in the works of Alpaydin [1], Cristianini [7], and Hastie et al. [13].

The support vector machine's training data consists of N tuples $(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)$, where $\vec{x}_i \in \Re^n$ is a feature vector and $y_i \in \{+1, -1\}$

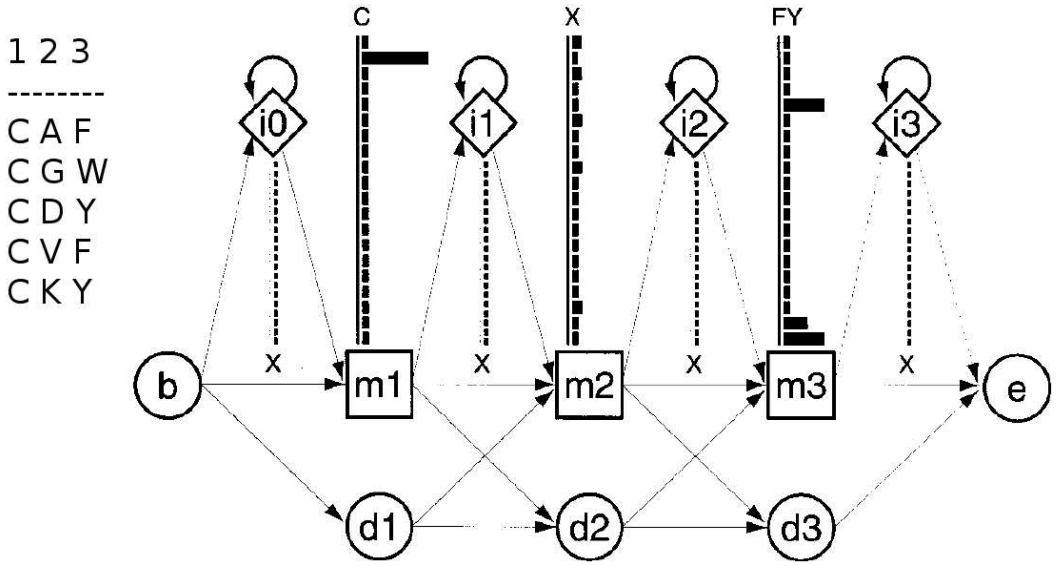


Figure 2.9: A small profile HMM modelling a multiple sequence alignment of five protein sections that are three columns long (example taken from Eddy, [9]). Each match state is assigned an emission probability distribution over the HMM's alphabet (the 20 amino acids) based on the values seen in the corresponding column of the multiple alignment. The insert states are also assigned appropriate emission distributions (not shown). On the other hand, the delete states are "mute" and do not emit a letter when entered. Therefore, they do not have any emission probabilities.

is the corresponding class label. We define the data's separating hyperplane with a normal $\vec{w}$ and a scalar offset b as

$$\{\vec{x} : \vec{w} \cdot \vec{x} + b = 0\}.$$

If such a separator exists for the training data, we can find it by solving the optimisation problem

$$\begin{aligned} &\min_{\vec{w},b} ||\vec{w}|| \\ &\text{subject to } \; y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \; i = 1, \ldots, N. \end{aligned} \tag{2.10}$$

The training points closest to the separator will just satisfy their constraints, such that $y_i(\vec{w} \cdot \vec{x}_i + b) = 1$. These points delimit the *margin* of the classifier and are known as the SVM's *support vectors*. In addition, the constraints imply that the distance from the support vectors to the separator will be

$$C = \frac{1}{||\vec{w}||}.$$

Moreover, the constraints on problem (2.10) require that each training vector be on the correct side of the separator. However, in the case where the data set's classes overlap, the above problem has no solution that can satisfy its constraints (no linear separator exists). To deal with this situation it is necessary to allow some of the data points to fall on the wrong side of the hyperplane, so we introduce "slack" variables ξ_i into the constraints yielding

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad \forall i \; \xi_i \geq 0. \tag{2.11}$$

As the value of ξ_i grows, the data point $\vec{x}_i$ is allowed to be closer to the separating plane than the value of C or, when $\xi_i > 1$, even on the wrong side of the plane resulting in a misclassification. To bound the total number of misclassifications we add an additional constraint to the problem:

$$\sum_{i=1}^{N} \xi_i \leq K.$$

The resulting definition of the support vector classifier is:

$$\min ||\vec{w}|| \quad \text{subject to} \quad \begin{cases} y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \;\; \forall i, \\ \xi_i \geq 0, \; \sum \xi_i \leq K. \end{cases} \tag{2.12}$$

Finally, since the separator is a simple hyperplane, it follows that a vector $\vec{x}$ could be classified using the rule

$$sign(\vec{w} \cdot \vec{x} + b). \tag{2.13}$$

2.3.1 Lagrangian (Wolfe) dual formulation

Using Lagrange multipliers it is possible to re-state the convex optimization problem (2.11) as the following quadratic program, known as the SVM's *Lagrangian (Wolfe) dual formulation*:

$$\max_{\vec{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{2.14}$$

subject to

$$0 \leq \alpha_i \leq \gamma, \tag{2.15}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \tag{2.16}$$

$$\alpha_i[y_i(\vec{w} \cdot \vec{x}_i + b) - (1 - \xi_i)] = 0, \tag{2.17}$$

$$\mu_i \xi_i = 0, \tag{2.18}$$

$$y_i(\vec{w} \cdot \vec{x}_i + b) - (1 - \xi_i) \geq 0, \tag{2.19}$$

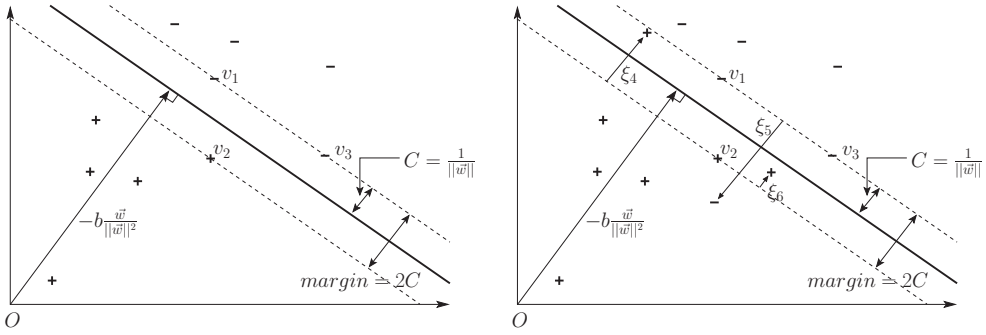


Figure 2.10: Geometric representation of SVMs for 2-dimensional, 2-class, separable (on the left) and inseparable (on the right) data sets. The decision boundary (shown as the the solid line) is placed to maximise the smallest distance C between a data point and the separator. The broken lines delimit the margin of the classifier, and the support vectors—lying on the edge of the margin—are labelled $\vec{v}_1, \vec{v}_2, \vec{v}_3$. On the right, the slack variables ξ_i are added to allow training points on the wrong side of the separator.

where

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i, \tag{2.20}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{2.21}$$

$$\alpha_i = \gamma - \mu_i, \ \forall i. \tag{2.22}$$

For the derivation of this formulation consult Hastie et al. [13]. Notice from (2.20) that the vector normal to the separator $\vec{w}$ is determined only from those training instances i for which $\alpha_i > 0$. Those training instances are the support vectors.

The dual formulation of the support vector machine is of particular importance in large part because of the following development. From equation (2.20) we can rewrite the decision function (2.13) as

$$\begin{aligned} G(v) &= sign(\vec{w} \cdot \vec{v} + b) \\ &= sign((\sum_{i=1}^{N} \alpha_i y_i \vec{x}_i) \cdot \vec{v} + b) \\ &= sign(\sum_{i=1}^{N} \alpha_i y_i (\vec{x}_i \cdot \vec{v}) \ + b). \end{aligned} \tag{2.23}$$

With the revised decision function, the data vectors only appear in the dot products—in equations (2.14) and (2.23)—which allows us to easily replace the dot product with any operation $K(x, v)$, referred to as a *kernel*, that is symmetric and positive semi-definite (Mercer's conditions). The equations (2.14) and (2.23) revised to use a kernel function are:

$$max_{\vec{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \tag{2.24}$$

$$G(v) \quad = \quad sign(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{v}) \ + b). \tag{2.25}$$

The kernel formulation is of particular importance for a number of reasons, including the decoupling of the SVM learning machine from the nature of the input data, and the ability to map data vectors into a space where they are linearly separable [7]. Some examples of kernel functions are:

- linear: $K(\vec{x}, \vec{v}) = (\vec{x} \cdot \vec{v})$;
- d^{th} degree polynomial: $K(\vec{x}, \vec{v}) = (1 + \vec{x} \cdot \vec{v})^d$;
- radial basis: $K(\vec{x}, \vec{v}) = e^{-||\vec{x}-\vec{v}||^2/c}$.

Chapter 3

Predicting pathways

This chapter presents an algorithm to predict biochemical pathways. For simplicity the algorithm is explained using metabolic pathways. This chapter begins by describing a data model to represent biochemical pathways in a way that makes computational prediction feasible. It then describes a general pathway prediction algorithm, and discusses the experiments that are done to evaluate it. Finally, section 3.4 explains the straightforward application of the algorithm to other kinds of biochemical pathways—e.g. signalling pathways.

3.1 Representing metabolic pathways

From the pathway example in Figure 1.1, we can see that the general biochemical pathway is in essence a network of reactions. Therefore, a pathway is represented as a directed graph. This graph has two types of nodes: *reaction nodes* and *compound nodes*. In the diagrams these are distinguished by rectangular boxes and ovals respectively. Reaction nodes represent the reactions that make up the pathways. They are annotated with the identifier of the reaction they represent. More importantly, they are annotated with the one or more proteins that are able to catalyse their reaction. This annotation states that any of catalysts specified is able to catalyse the reaction. On the other hand, compound nodes represent a chemical compound that is either required or produced by a reaction node. Finally, the arcs of the graph follow the flow of chemicals through the process, or more generally, represent a dependency.

Two extensions are made to the pathway data model to enhance its expressiveness. First, any of the annotating proteins in a reaction node is sufficient for the node's reaction to take place. However, some reactions require a protein complex (a set of proteins) instead of a single protein to occur. To correctly express these reactions a mechanism is needed to differentiate between a required protein complex and a set of proteins for which any protein in the set

can catalyse the reaction. The pathway model can be easily extended to support protein complexes by generalizing the protein list in a reaction node to include either individual protein or protein complexes. Secondly, reaction 2 in Figure 1.1 highlights that this graph structure can fail to distinguish the two sides of a reversible metabolic reaction. This issue could be addressed by extending the reaction node to be have a small subgraph consisting of two nodes labelled as *left* and *right*. The reaction's compounds are then connected to the appropriate side thus explicitly forming the two groups of compounds.

Neither of these extensions are used in this work since the original pathway data model suffices.

3.1.1 A useful simplification of the general pathway model: *reaction graphs.*

The general graph model for biochemical pathways just presented is very expressive at the cost of simplicity. The fact that the graph is comprised of two different classes of nodes makes its analysis more complex. However, the pathway prediction strategies presented in this thesis are solely concerned with the proteins involved in the biological process—i.e. the reaction nodes. These analyses do not require the information provided by the compound nodes. Therefore, a useful simplification of the pathway model is to represent it as a network of reactions only. We will refer to this data model as a *reaction graph*. In a reaction graph, the reactions that are connected via an intermediate compound in the general pathway graph are now connected directly. Connections that do not go through a compound remain unchanged. Figure 3.1 presents the reaction graph model of a section of the instance of the Gluconeogenesis pathway for *C. elegans* that spans reactions 1, 2 and 3 from Figure 1.1. Each node contains the names of the proteins in *C. elegans* that catalyse the reaction. The arcs from reactions 1 and 3 to node 2 indicate that some chemical products of the reactions 1 and 3 are the reactants of reaction 2.

The reaction graph representation reduces the data model's complexity by only using one type of node. While this model may not be expressive enough for all conceivable pathway prediction strategies, it is sufficiently powerful to organize the data used by the techniques studied herein. Thus, the reaction graph will henceforth be the pathway model of choice in this dissertation.

3.2 The pathway prediction algorithm

Having a data model to represent pathways in a structured way lays the foundations for an algorithm that can use the pathway model to make predictions. To exploit the similarity between organisms, well-studied versions of a path-

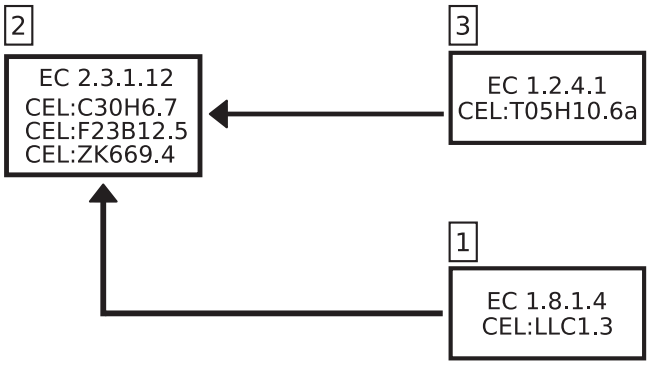


Figure 3.1: The reaction graph for a section of *C. elegans*'s instance of Gluconeogensis spanning reactions 1, 2 and 3 from Figure 1.1.

way (such as in model organisms) can be used to predict what the pathway may look like in an organism of interest, where the pathway is not as well-characterised. Some terminology is necessary. The *target organism* is the organism whose pathway instance is being predicted. A *training pathway* is a pathway instance that is given as input to the prediction algorithm. A *training reaction* is a reaction in a training pathway. A *training protein* is a protein that labels a training reaction. The prediction algorithm takes as input a single training pathway [1] and the proteome of a target organism. The goal of the algorithm is to predict:

1. whether the pathway exists in the target organism and, if it exists, then
2. the structure of the pathway,
3. for every reaction in the pathway, its set of potential catalyst proteins.

To achieve the goal, each of the training pathway's reactions is analysed one at a time. For each of these reactions, the algorithm must decide whether or not it exists in the target organism. The algorithm assumes that the reaction needs to have at least one protein catalyst to occur. Therefore, the algorithm can determine whether the reaction exists in the target organism by determining whether the target organism has one or more proteins capable of performing the same function as the training reaction's proteins. If such candidate proteins are found, the reaction is added to the predicted pathway. In addition, the predicted reaction is annotated with the candidate proteins found in the target organism's proteome. On the other hand, if no such proteins are found then the algorithm predicts that the training reaction does not exist in the target

[1] In the next section this approach is generalised to utilize multiple training pathways.

organism. Finally, if none of the training reactions are predicted to exist in the target organism then the algorithm decides that the entire pathway does not exist in the organism. The pathway prediction algorithm appears as pseudo-code in Algorithm 1.

Algorithm 1 Pathway prediction algorithm.

```
Require: training_pathway
Require: proteome
Ensure: prediction
  prediction ← Pathway.new
  for all reaction in training_pathway do
    predicted_proteins ← reaction.find_able_proteins_in(proteome)
    if not predicted_proteins.empty? then
      new_reaction ← prediction.add_node(reaction)
      new_reaction.add_proteins(predicted_proteins)
    end if
  end for
  if prediction.empty? then
    prediction ← nil
  end if
  return prediction
```

The prediction algorithm is quite intuitive. However, it abstracts a critical step—how to decide whether a protein from the target organism is capable of performing the same function as the training protein. The particular task is handled by a classifier and is hidden in the algorithm as the `find_able_proteins_in(proteome)` function call. For the purposes of this algorithm, a classifier is a black box with the inputs and outputs specified in Table 3.1. This design allows the prediction algorithm to work for any type of pathway that can be represented by the general reaction graph model, as long as a suitable classifier can be created.

3.2.1 The classifier

The classifier is a critical component of the pathway prediction algorithm. It is a computational device that predicts which proteins from the target organism are capable of catalysing a training reaction. The classifier is a "black box" whose inputs consist of the target proteome and a training reaction. However, both these parameters are composite items, making available to the classifier the various bits of information summarized in Table 3.1. The classifier filters the target organism's proteome, returning only those proteins that are functionally compatible with the training reaction's catalysts. Therefore, the

classifier in the pathway prediction problem must make a very specific function prediction based on a small number of positive training samples. In our experimental data set the number of catalysts in a single reaction of a single pathway instance varies from 1 to 17, with mean 1.8. In the next section we illustrate how the number of positive training samples can be increased to the range 1 to 50 with a mean of 11.5. . However, even with this increase, the problem is different from many other protein function prediction problems, such as high level Gene Ontology [12] classification, because a much more specialized function must be predicted for the target protein and the number of positive training samples is still very low.

Reaction

Parameter	Description
Reaction identifier	Identifies the reaction represented by the node
Catalysts	Proteins known to catalyse the reaction
Connections	Other reactions in the pathway that are connected to this one (consume its products or produce its reagents)

Target proteome

Parameter	Description
Protein list	The amino acid sequences of all the target organism's proteins
Organism	The prediction's target organism

Table 3.1: Inputs to the protein classifier

In chapter 4 we describe the different classifiers we have used in our pathway prediction architecture. In Chapter 5 we compare the prediction accuracies of these classifiers.

3.3 The model pathway

As presented above, the pathway prediction algorithm only learns about the pathway of interest from a single pathway instance. There are two major problems in trying to predict the structure and components of a pathway based on a single pathway instance. First, we have shown that the structure of a pathway varies between organisms. Using only a single training pathway will increases the chance that the training pathway has a different structure than the target pathway. In this case, predicting the true structure of the target pathway becomes impossible since no predictions would be attempted on any reaction not found in the training instance. For example, if the training

pathway is the Gluconeogenesis pathway in *C. elegans* and the target is the same pathway in *A. thaliana* then there is no chance of finding the reaction denoted 5 in Figure 1.1, since this reaction does not appear in the training pathway. Second, as indicated in the last section, if a classifier uses only a few positive training instances to predict the pathway components, then the classifier will have poor accuracy.

Our approach is to make our structure and component predictions using all available pathway instances. This enables the algorithm to use all of the diverse instances of the training pathway to match the structure of the target pathway and to predict components more accurately. To add this capability to the algorithm we introduce the notion of a *model pathway*. A model pathway is an abstract version of a pathway that combines multiple pathway instances. To create a model pathway we effectively take the "union" of a number of training pathways. At the structural level, we define the union of two pathways A and B as $U = A \cup B$, where U is a new pathway whose structure includes all the reactions occurring in either A or B. For each reaction in pathway U, if that reaction existed in both A and B, then the reaction's protein catalyst set in U is the union of the catalyst sets from the same reaction in A and B. Therefore, the reaction in U is considered to be catalysed by any of the reaction's catalysts from the two original instances.

Figure 3.2 illustrates the union of part of the *C. elegans* instance of the Gluconeogenesis pathway (left subfigure) and part of the *S. pombe* instance of the same pathway (middle subfigure). *C. elegans* has reaction 6 and 7, from Figure 1.1, but not reaction 5. *S. pombe* has reaction 5 and 7, but not reaction 6. The union pathway has all three reactions. The catalyst set for reaction 7 contains all of the proteins that catalysed this reaction in either organism, regardless of the EC family where the protein originated. For brevity, reactions 5 and 6 are shown with only one of their protein catalysts.

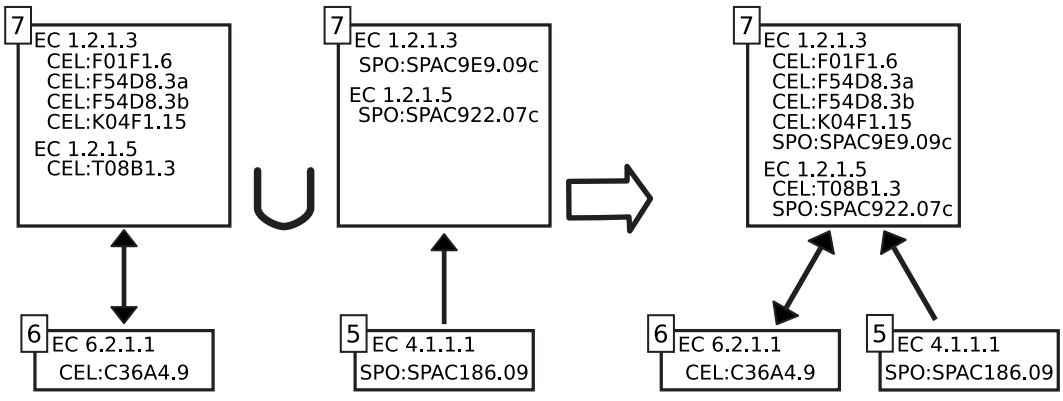


Figure 3.2: The union of two partial pathway instances into a model pathway.

To use the model pathway algorithm 1 is not modified, except that the model pathway is used as the training pathway input instead of using a single instance pathway. By using model pathways, the pathway prediction algorithm can even predict instances of a pathway with variations in structure that were never observed in the training pathway set—and perhaps never found in any physical laboratory. Such *emergent structures* can be computationally predicted before being observed.

3.4 Predicting other types of pathways

The pathway prediction algorithm was explained using only metabolic pathways. However, there are other types of pathways at work in living organisms—for instance, signal transduction pathways. It is important to be able to apply the prediction algorithm to these types of pathways as well. Since there have been no particular assumptions made in the algorithm regarding any traits specific to metabolic pathways, the application to other types of pathways is automatic. The only requirement is that the pathway be representable as a reaction graph. This requirement is not very restrictive since the type of reaction is not confined.

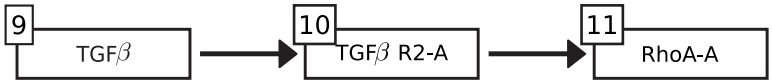


Figure 3.3: Reaction graph representation of the TGF–β signalling pathway excerpt from Figure 1.2. In this example, the reactions are generic protein activations.

3.5 Experimental methodology

To evaluate the effectiveness of our automated pathway prediction technique, we performed a cross-validation of pathway predictions to simulate the situation where the pathway instances of the target organism are completely unknown. We then combine the statistics from each prediction into a single set of statistics to evaluate the effectiveness of the algorithm.

3.5.1 Cross-validation

Cross-validation is a method commonly used to test predictors that are "learned" from training data. It works by simulating the situation where part of the

training data is unknown. The training data set is partitioned into n groups, or *folds*, numbering them $i = 1 \ldots n$. Then, for each evaluation i we remove fold i from the training set, train a predictor with the remaining $n-1$ folds, and use it to predict fold i. The accuracy of the predictor can then be measured for each fold by comparing the predicted results to the known values.

In applying cross-validation, we began with a data set consisting of n instances of the same pathway, where each instance was from a different organism. We used $n - 1$ organisms to build a model pathway and then used this model pathway to build a classifier. We used this classifier to predict the remaining n^{th} pathway instance and compared the predictions to the known structure and components of this n^{th} pathway instance. We repeated this cycle n times, each time predicting the pathway instance of a different one of the organisms. Since our data set had 125 different pathway instances, we repeated this cross-validation process 125 times, once for each instance. Finally, we aggregated all of the results to compute overall statistics.

3.5.2 Measurements

The algorithm is evaluated by comparing the predicted pathway instance to the known pathway information—structure (reactions) and components (catalysts). We computed statistics for each protein that catalyses each reaction in a pathway and called these statistics the *component* or *catalyst scores*. We also separately computed statistics for the existence of each reaction in a pathway, where a reaction is predicted to exist in a pathway if at least one protein is predicted to catalyse that reaction. We call these statistics *structure* or *reaction scores*. The goal of the predictor is to find all the reactions with all their proteins for all the known pathways in the target organism. Also, the predictor should not produce any noise—false positive predictions. With these criteria in mind three basic elements to be counted are identified:

1. **True positive (TP):** a positive prediction that matches the known information,
2. **False positive (FP):** a positive prediction that does not match,
3. **False negative (FN):** a negative prediction that does not match.

Although these counts give some insight into the effectiveness of the pathway prediction algorithm, they leave a desire for simpler, more intuitive measurements. For this reason two standard measurements based on TP, FP, and FN are adopted:

1. **Precision:**

$$\frac{TP}{TP + FP}$$

2. **Recall:**

$$\frac{TP}{TP + FN}.$$

Recall (R) measures what fraction of the known pathway information was discovered by the predictor. On the other hand, precision (P) measures the fraction of the predictions that are true, thus measuring the amount of noise in the predictions.

Finally, recall and precision are combined into *f-measure* [34] to yield a single value that can be used for easy comparison between different results. The experiments presented in this thesis use an f-measure that places equal weight on precision and recall. It is defined as

$$\frac{2pr}{p+r} \quad \text{where} \quad \begin{array}{rcl} p & = & \text{precision} \\ r & = & \text{recall} \end{array}$$

Note that f-measure is an important statistic because it combines recall and precision. It is easy to inflate either precision or recall separately at the expense of the other. For example, a classifier that always predicts "yes" has perfect recall since FN = 0. A classifier that always predict "no" has perfect precision since FP = 0. Note also that we do not compute the specificity, which gives credit for true negatives, since in the context of this problem the negative set is extremely large compared to the positive set so the specificity would be high even for a poor classifier that always predicts negative.

3.5.3 False positives versus discoveries

The standard definition of true positive (TP) is the only choice in perfect information scenarios. However, in imperfect information situations, there is an alternative. In the case of pathway data (and other biological data), we cannot assume that our testing data is complete. Usually publications report only positive results. Therefore, the absence of a protein from a reaction's list of catalysts could indicate that an experiment has not been performed to determine whether or not that protein catalyses that reaction. Given that the data is incomplete, it may be desirable not to penalize the algorithm for predicting the existence of a previously unknown catalyst for a reaction known to exist in the target organism. Such a prediction may not really be a false positive—it may be a *discovery*. On the other hand, every false positive could be treated as a discovery, so that a predictor that always says "yes" would have perfect precision. This is not desirable either.

We propose a compromise. To predict pathways, an algorithm makes catalyst predictions and reaction predictions. The probability of discovering another protein that catalyses a reaction that is known to occur in an organism

is higher than the probability of discovering that a reaction occurs in an organism, when it was previously not known to occur. In other words, small discoveries are more probable than large discoveries. When evaluating reaction scores, false positives should be considered normally, since discoveries are improbable. For example, there is no reaction 5 for *C. elegans* in the Gluconeogenesis pathway. If a predictor predicted such a reaction then it would be considered a false positive reaction. The predictor knows about reaction 5, since the model pathway contains this reaction (from *S. pombe*). Therefore, it would be possible for the classifier to predict a catalyst for it. In fact, to predict such a reaction for *C. elegans*, the classifier would only need to predict that a single protein in *C. elegans* has the same function as a protein in S. pombe that catalyses reaction 5. However, this would be the first protein ever discovered in *C. elegans* that catalyses this reaction, making it improbable. On the other hand, if a reaction is known to exist in an organism and a false positive occurs when predicting a catalyst for this reaction, it could be a discovery of another protein that catalyses the same reaction (many reactions have more than one catalyst). This is a small discovery so it is more probable.

For a molecular biologist, a false positive on a catalyst in an organism that is known to have a reaction should be considered a "lead" for an experiment that could make a "small" discovery of a new catalyst for a known reaction in the metabolic pathway of this organism. A false positive on a catalyst in an organism that is not known to have this reaction, is a "lead" for a riskier experiment that could lead to a "large" discovery of a new reaction in the metabolic pathway for this organism.

In this dissertation, all of the statistics presented use the traditional (more conservative) definition of false positive, even though some false positives are probably discoveries. Therefore the actual accuracies of our classifiers are probably higher than reported. However, some specially identified statistics are reported using the relaxed "false positive is a discovery" evaluation to illustrate the differences that arise.

3.6 Experimental data set

To perform the cross-validation experiment requires a data set with certain characteristics. Namely, because of the way the prediction algorithm works it requires an instance of a pathway to find its variant in the target organism. The data set has to have at least two instances of any pathway that is to be used in cross validation. In this manner, while one instance can be in the withheld fold, at least one other can be in the training set. However, that number is only the minimum requirement for operation. To fairly and reliably test the utility of the pathway prediction algorithm several instances of each

pathway must be available.

Another requirement is that the data structure and components of the experimental data set be experimentally verified, or at least manually curated—not be automatically generated by computational means. It would be undesirable to use another pathway predictor's results to evaluate the quality of the predictor presented in this dissertation. In such a case, this predictor would be striving to model another predictor, rather than learning from biology. It would score well for repeating the other predictor's successes, but also for repeating the other predictor's mistakes! Moreover, if the predictors use similar techniques—sequence similarity for example—the comparison would be reduced to a self-confirmation.

In addition, the data must be available in a format that can be easily used for computation. It should also not be burdened by restrictive licenses that would limit the use and publication of any results derived from it. Finally, the most obvious requirement is that the amino acid sequences of the proteins involved in the pathways must be part of the data set, since the prediction algorithm requires them to operate.

Even with all these restrictions, one might suspect that given the elevated status of biochemical pathways in biology, finding a suitable data set would not be too difficult. Consider that pathways exist in all living cells. Further, the concept of a pathway has been understood by biologists for a relatively long time—for example, Hans Krebs won the Nobel Prize in medecine in 1953 for his discovery of the Krebs Cycle [26]. Because of these reasons much work has been done by biologists in this general area. Unfortunately, much of this work still lies in paper manuscripts, written in free text. Such information remains impractical to access for the time being, and thus remain out of the reach of most work in bioinformatics. This restriction in particular quickly limited the number of candidate data sources that could furnish a data set useful for this project.

A search of the Internet reveals a number of potential sources of biochemical pathway data—for instance, KEGG [17], MetaCyc [23], Reactome [15], aMAZE [25], TAIR [30]. Regrettably, most of these have shortcomings with respect to the listed requirements of this project, making their data unsuitable for these experiments. Consequently, the search for an ideal testing data set was rather lengthy. Along the way various options were investigated. Some of the better ones are briefly described below, along with the reasons for their abandonment.

Amalgamating several organism-centric data sources

There are several online pathway databases that concentrate on a small number of organisms. This approach allows the database maintainers to focus their

efforts, producing detailed, high quality information for their model organisms. However, given our experimental design, a data set that only spans one or two organisms is not suitable.

To address this issue, the option of integrating the data from several of these focused databases was considered. Such a move should result in one larger data set spanning several organisms, and thus suitable for the experiments. Unfortunately, the integration problem proved to be a complicated one. The reason is that there is a lack of standardization and structure among the different data sources, making it very difficult to recognize equivalent entities and avoiding redundancy in any integrated data set. Redundancy in the pathway data set would falsify the results of our cross-validation to some degree, as the our algorithm could be rewarded or penalized more than once for the same prediction. In the end this idea was abandoned by the author of this work, but it is still the subject of research elsewhere [31].

Using EC numbers to annotate organism-independent template pathways

Proteins are annotated with EC numbers when they are known to catalyse the corresponding class of reactions. Also, the KEGG and MetaCyc resources make available a number of template organism-independent metabolic pathways where all the component reactions are labelled with EC numbers. To create an organism-specific pathway one could take a template pathway and annotate its reactions with proteins from the organism that have matching EC numbers, and then remove all reactions that do not have any catalysts. This technique was used by earlier versions of KEGG to create their organism-specific pathways [16].

The feasibility of creating a data set by combining proteomes including EC number annotations with template pathways was investigated. Unfortunately it had to be abandoned for two reasons. First, proteomes with EC annotations are not common. Further, the ones that can be found do not usually explain how the annotations are derived, leading to suspicion that they may be at least in part generated by automated sequence similarity techniques such as BLAST [2]. Such data would violate one of our requirements. Secondly, the EC hierarchy classes are not disjoint, meaning that a single reaction can have multiple EC numbers. In fact, some EC classes are proper subsets of others. This point is important because it introduces two possibilities for annotation errors:

1. False annotation: a protein that only catalyses a subset class can be mistakenly annotated as catalysing the superset class;

2. Missed annotation: a protein that catalyses an entire superset is not

labelled as a catalyst for each of the individual subset classes.

The combination of these problems was sufficient to convince the author to abandon this path.

The MetaCyc database

At first glance the MetaCyc database appears to be the ideal source for the required experimental data. It documents 445 pathway instances spanning 158 organisms (as of 2002 [19]). Further, the information it provides is experimentally verified. Finally, its data is available for download, although its use is subject to a license agreement which restricts the distribution of any derived work. But a closer look exposes a major problem. Most of MetaCyc's gene annotations (33% as of April 2005 [18]) do not include the corresponding DNA or amino acid sequence, nor do they provide sufficient information to form a link to an external sequence database. Since the algorithm presented in this thesis works from proteins' amino acid sequences this data is unusable in the evaluation.

3.6.1 The final data source selection

After scrutinizing the other sources and evaluating the alternatives, it was decided that the Kyoto Encyclopaedia of Genes and Genomes PATHWAY [17] database is the source of data best-suited for this project. The KEGG resource is an ensemble of databases that strives to integrate biochemical, genomic, and pathway data [17]. Their integration efforts make it an idea source for pathway structure information, since it comes complete with useful information about the reactions and the compounds that comprise it. More importantly, the data set includes the catalysts of a significant number of metabolic pathways, and includes the complete proteomes of 251 organisms (as of July 2005).

As previously mentioned, the manner in which the pathway data is derived is important to the validity of these experiments. In particular, the KEGG data would not be suitable if it came directly from an automated predictor, without any manual curation. Fortunately the KEGG pathway data is not derived in this manner. Instead, a number of organism-independent reference pathways are manually constructed. The reactions in these pathways are labelled with a special function identifier. A team of curators then analyzes organism genomes and assigns a corresponding function identifier to each gene based on various bits of available information. These include features such as motifs, phylogenetic profiles, information from literature, etc. [20]. The final step in creating organism-specific pathways consists of associating the reaction in the reference pathways with the proteins possessing the equivalent function identifier.

Finally, the data from KEGG are made available in a format that is suitable for computation, and are not encumbered by any restrictions on the use by academic users.

With all its positive traits, the KEGG data source does have a shortcoming. Although the database does contain some data regarding non-metabolic pathways, that part of the collection is somewhat limited in breadth. Since the amount of data available is not suitable to provided representative results it was decided to only use metabolic pathways for the experiments presented in this dissertation.

Selected pathways and organisms

From the KEGG PATHWAY database a selection of 10 metabolic pathways spanning 13 organisms—a total of 125 pathway instances—was extracted to create the experimental data set. The complete listing of pathways and organisms are in Tables 3.2 and 3.3 respectively. Since two organisms do not have instances of all 10 pathways, only 11- or 12-fold cross-validation was done for those pathways. The missing instances are listed in Table 3.4. Consequently, cross-validation resulted in 125 invocations of the prediction algorithm, totalling 3523 reaction nodes classifications.

Category	**Pathway**
Carbohydrate metabolism	Aminosugars metabolism Citrate cycle (TCA cycle) Galactose metabolism Glycolysis / Gluconeogenesis Propanoate metabolism
Amino acid metabolism	Alanine and aspartate metabolism Cysteine metabolism Glutamate metabolism Methionine metabolism Urea cycle and metabolism of amino groups

Table 3.2: The 10 pathways selected for the experimental data set

Species	**Strain**	**Proteome size**
Agrobacterium tumefaciens	C58 (Cereon)	5290
Arabidopsis thaliana		28014
Bacillus subtilis		4106
Caenorhabditis elegans		21177
Chlamydia trachomatis		895
Drosophila melanogaster		16098
Escherichia coli	K-12 MG1655	4208
Helicobacter pylori	J99	1488
Homo sapiens		23148
Mycobacterium tuberculosis	CDC1551	4178
Mycoplasma pneumoniae		689
Saccharomyces cerevisiae		5794
Schizosaccharomyces pombe		4969

Table 3.3: The 13 species selected for the experimental data set.

Species	**Pathways**
C. trachomatis	Galactose m., Urea cycle
M. pneumoniae	Aminosugars m.,TCA cycle, Urea cycle

Table 3.4: Species missing test pathways

Chapter 4

Classifiers

Section 3.2.1 describes the requirements placed by our pathway prediction algorithm on the classifier used to select proteins from the target organism with the function required by the training reaction. Several different classifiers were implemented, tested, and evaluated in the Pathway Analyst prototype pathway prediction system. The classifiers were based on three different technologies: BLAST, hidden Markov models (HMM) and Support Vector Machines (SVM). Other classification technology could be used, but these technologies were sufficient to establish the utility of this approach to high-throughput pathway prediction. In this chapter we explore the different classifiers that were implemented and evaluated. In addition, we illustrate the operation of the classifiers with the the prediction of reaction 1 from Gluconeogenesis (see Figure 1.1) in *C. elegans*. For this reaction there are 3 catalysts from *C. elegans* in the KEGG database: cel:C30H6.7 cel:F23B12.5 cel:ZK669.4. The model reaction is catalysed by 19 proteins from 11 organisms (listed in Table 4.4).

4.1 BLAST-based classification

One approach to the classification problem is to compare the primary sequence of the training proteins—which are known to catalyse a specific reaction node—to the target organism's proteins, and select the most similar ones. BLAST [2] is a tool that performs this type of comparison. Two classifiers based solely on BLAST were implemented.

4.1.1 BLAST nearest-neighbour classifier

The BLAST nearest-neighbour (NN) classifier selects the protein from the target organism's proteome that is most similar to any of the training reaction's proteins (as determined by BLAST). In other words, for a given reaction and

target proteome, the BLAST NN classifier compares all the training proteins to all the sequences of the target proteome, and then returns the single target protein with the smallest e-value (most similarity). Therefore, to predict reaction 1 in *C. elegans* the NN classifier applies BLAST to search the *C. elegans* proteome with each of the 19 proteins from the model reaction (see Table 4.4) as a query. Then the classifier merges the results of each search and sorts them by e-value. The result of this step is shown in Table 4.1. Since the BLAST NN classifier chooses BLAST's closest match, its prediction is that reaction 1 exists and protein cel:F23B12.5, which has the lowest e-value, catalyses it. Therefore, the classifier did find one of the correct catalysts, but it missed 2 other ones (target proteins 2 and 3 in the table).

	Model	Target	e-value
1.	hsa:1737	cel:F23B12.5	6e-128
2.	bsu:BG10209	cel:ZK669.4	6e-54
3.	hsa:8050	cel:C30H6.7	2e-48
4.	bsu:BG12560	cel:W02F12.5	2e-46
	continues...		

Table 4.1: Result from searching *C. elegans*' proteome for proteins similar to reaction 1's model catalysts (see Table 4.4). The BLAST NN classifier predicts protein 1 as a catalyst.

This simplistic classifier provides a baseline for comparison with the other classifiers. In particular, the classifier's limitation of only selecting a single protein from the target proteome makes it impossible for the classifier to attain good recall scores, since it is common for several proteins in an organism to catalyse the same reaction. In addition, the fact that the BLAST NN classifier always makes a prediction—regardless of the dissimilarity of the best-matching protein—undoubtedly harms its precision, since in some organisms certain reactions do not occur as the necessary protein catalysts are not present.

4.1.2 BLAST Threshold classifier

Similarly to the BLAST NN classifier, the BLAST Threshold classifier uses BLAST as a metric to compare each training protein to each protein of the target proteome. However, this classifier eliminates some of the BLAST NN classifier's obvious limitations, by predicting all those proteins from the proteome whose comparison with any of the training proteins resulted in an e-value no greater than a significance threshold ϵ. To illustrate consider predicting reaction 1 in our running example with a threshold $\epsilon = 10^{-50}$. In this case, the BLAST Threshold classifier selects the top two proteins in Table 4.2—

i.e. cel:F23B12.5 and cel:ZK669.4. However, the classifier misses the known catalyst cel:C30H6.7.

	Model	Target	e-value
1.	hsa:1737	cel:F23B12.5	6e-128
2.	bsu:BG10209	cel:ZK669.4	6e-54
3.	hsa:8050	cel:C30H6.7	2e-48
4.	bsu:BG12560	cel:W02F12.5	2e-46
	continues...		

Table 4.2: Result of searching with BLAST the proteome of *C. elegans* for proteins similar to reaction 1's model catalysts (see Table 4.4). The BLAST Threshold classifier predicts proteins 1 and 2 as catalysts since their e-values fall below the threshold $\epsilon = 10^{-50}$.

4.2 Profile-HMM-based classification

In cases where the functionality of a protein depends mainly on a small conserved portion of its amino acid sequence—perhaps due to a motif—profile HMMs may be more sensitive than alignment methods such as BLAST for identifying candidate catalysts from the target proteome. Profile hidden Markov models can weight similarity to these conserved regions more heavily than similarity to the rest of the sequence. They are constructed to recognize recurring protein segments, or motifs, that are common to most of the model catalysts.

To use profile HMMs for pathway prediction, we build a profile HMM model for each reaction in the training pathway. This process involves computing a multiple alignment of all the training reaction's catalyst proteins with ClustalW [33], and then using HMMER [11] to build a profile hidden Markov model. Finally, the model is calibrated with the `hmmcalibrate` tool [11] to empirically estimate the its score significance.

4.2.1 Profile HMM threshold classifier

Our Profile HMM classifier uses HMMER to iterate over the target organism's proteome and calculate for each protein the likelihood of being emitted by the reaction's hidden Markov model. Like BLAST, HMMER returns an e-value for each protein (although the e-values for the two tools do not have exactly the same meaning), so our predictor uses a threshold to filter out proteins that do not match well enough.

To demonstrate this classifier's operation, consider again predicting reaction 1 in *C. elegans*, but this time using the profile HMM classifier. The first

	Target	e-value
1.	cel:F23B12.5	2.9e-221
2.	cel:ZK669.4	1.5e-84
3.	cel:W02F12.5	3.7e-59
4.	cel:C30H6.7	5e-26
5.	cel:F26F4.7	1.1

Table 4.3: Result of searching *C. elegans*' proteome for proteins similar to reaction 1's model catalysts (see Table 4.4). The Profile HMM classifier predicts proteins 1, 2, and 3 as catalysts, since their e-values fall below the threshold $\epsilon = 10^{-50}$.

step in the process consists of building and calibrating the HMM model for the model proteins listed in Table 4.4. Then the model is used to scan *C. elegans*' proteome, resulting in Table 4.3. If we require an e-value of 10^{-50} to consider a proteins a catalyst, the classifier returns proteins cel:F23B12.5, cel:ZK669.4, and cel:W02F12.5 (rows 1, 2, and 3 in the table). In this case the classifier scores two true positives, 1 false negative and one false positive.

4.2.2 Mixing BLAST and HMMs

The evaluation of the BLAST Threshold and the HMM classifiers show a significant trade-off between precision and recall. In an attempt to take advantage of the strengths of each, we implemented a classifier that combines the BLAST and HMM threshold classifiers—the BLAST-HMM classifier. The BLAST-HMM classifier's prediction consists of the intersection of the prediction of its two component classifiers (BLAST and HMM). This classification rule implies a tougher standard to be met by proteins before being classified as catalysts, since both BLAST and HMM classifiers need to agree. We used higher e-value thresholds to allow more true positive catalysts to be predicted by each classifier (increasing recall), while the requirement for agreement filtered the extra false positives generated by the individual classifiers because of their higher thresholds (increasing precision).

Continuing the example predicting reaction 1 in *C. elegans*, we can use the results of the BLAST and HMM searches shown in tables 4.2 and 4.3 to calculate a prediction for the BLAST-HMM classifier while keeping the same BLAST and HMM thresholds. The BLAST Threshold classifier predicted cel:F23B12.5 and cel:ZK669.4 as catalysts, while the Profile HMM classifier decided that cel:F23B12.5, cel:ZK669.4, and cel:W02F12.5 are likely catalysts. The intersection of these two predictions results in cel:F23B12.5, cel:ZK669.4. Since both of these proteins are true positives we have raised the precision of the prediction, although the level of recall did not change.

4.3 SVM-Pfam-based classification

A Support Vector Machine is a statistical classification technique to compute a separator for a two-class data set. Indeed, our classifier's task is to separate the proteins that catalyse a reaction from the ones that do not. Unlike BLAST and HMMER, the SVM itself is not a sequence analysis technique. Therefore, it cannot work with raw amino acid sequences. Instead, it is necessary to produce a representative feature vector for each protein and use it for training and prediction. Given that motifs often determine enzymatic activity, the Pfam families [4] identify motifs that may be used as features for our problem. The feature vectors used by the SVM classifier consisted of 7673 boolean values, each stating the presence of absence in the protein of the corresponding Pfam motif.

We computed the motifs for each of the 120,054 proteins in our 13 test organisms using the `hmmpfam` tool. The training set for the classifier consisted of feature vectors for all the proteins of all the training organisms. The proteins that catalyse the reaction of interest were positive examples for the training process while the rest were negative examples. The training data set was used to compute a support vector machine using LibSVM software [6]. The SVM was then used to predict which of the target organism's proteins are catalysts for the reaction, given their Pfam motifs.

The unbalanced nature of the training set—there are far more negative samples than positive samples—can be problematic in the application of SVM to this particular problem. In particular, when no perfect separator can be found between the set of catalysts and non-catalysts, the SVM training algorithm may find it better—according to its optimization function—to take a small penalty for leaving the few known catalysts on the wrong side of the separator rather than leaving a larger number non-catalysts on the wrong side. To combat this symptom the weight associated with the positive training samples is raised. This change raised the penalty for placing a positive training sample on the wrong side of the partition, making the training algorithm behave as if there were more positive samples in the training set.

To illustrate the operation of the Motif SVM classifier, we can again refer to the prediction of reaction 1 in *C. elegans*. Table 4.5 lists the model proteins with their Pfam motif annotations. Each row of the table is used to form a feature vector for reaction 1's positive training set. For each protein, the listed motifs are labelled as "present" in the feature vector, while all the other known motifs are labelled as "absent." The compiled positive training set is then added to an analogous negative training, and is then used to train the Motif SVM classifier. The trained SVM correctly predicts all three known catalysts for this reaction: cel:C30H6.7, cel:F23B12.5, and cel:ZK669.4. The correct predictions may be explained by observing the common motifs between the

model proteins (Table 4.5) and the predicted catalysts (Table 4.6). Notice how the 2-oxoacid_dh motif is in all the model proteins, and the Biotin_lipoyl, and E3_binding motifs appear also very often. This may indicate that these motifs are important to catalysing reaction 1, so the classifier weighs their presence heavily. Since the known catalysts also contain most of those important motifs, the classifier readily identifies them.

Organism	**strain**	**Protein**
Agrobacterium tumefaciens	C58 (Cereon)	atc:AGR_C_2641 atc:AGR_L_2719
Arabidopsis thaliana		ath:At1g34430 ath:At1g54220 ath:At3g13930 ath:At3g25860 ath:At3g52200
Bacillus subtilis		bsu:BG10209 bsu:BG12560
Chlamydia trachomatis		ctr:CT247
Drosophila melanogaster		dme:CG5261-PA dme:CG5261-PB
Escherichia coli	K-12 MG1655	eco:b0115
Homo sapiens		hsa:1737 hsa:8050
Mycobacterium tuberculosis	CDC1551	mpn:MPN391
Mycoplasma pneumoniae		mtc:MT2570
Saccharomyces cerevisiae		sce:YNL071W
Schizosaccharomyces pombe		spo:SPCC794.07

Table 4.4: Model proteins for predicting of reaction 1 in *C. elegans* (therefore, proteins from *C. elegans* are withheld).

Protein	Motifs
atc:AGR_C_2641	2-oxoacid_dh, Biotin_lipoyl, DUF475, E3_binding, HlyD, S-antigen
atc:AGR_L_2719	2-oxoacid_dh, Biotin_lipoyl, E3_binding
ath:At1g34430	2-oxoacid_dh, AIRC, Biotin_lipoyl, E3_binding
ath:At1g54220	2-oxoacid_dh, Biotin_lipoyl, E3_binding, GCV_H
ath:At3g13930	2-oxoacid_dh, Agenet, Biotin_lipoyl, E3_binding, GCV_H
ath:At3g25860	2-oxoacid_dh, Biotin_lipoyl, E3_binding
ath:At3g52200	2-oxoacid_dh, BASP1, Biotin_lipoyl, Caudal_act, E3_binding, GCV_H
bsu:BG10209	2-oxoacid_dh, Biotin_lipoyl, E3_binding, GCV_H
bsu:BG12560	2-oxoacid_dh, Biotin_lipoyl, E3_binding, PYNP_C, STAT_alpha, SURF6
ctr:CT247	2-oxoacid_dh, Biotin_lipoyl, E3_binding
dme:CG5261-PA	2-oxoacid_dh, Biotin_lipoyl, E3_binding
dme:CG5261-PB	2-oxoacid_dh, ARS2, Biotin_lipoyl, E3_binding
eco:b0115	2-oxoacid_dh, Biotin_lipoyl, DUF601, E3_binding, GCV_H, HlyD
hsa:1737	2-oxoacid_dh, Biotin_lipoyl, DNA_PPF, E3_binding, Extensin_2
hsa:8050	2-oxoacid_dh, Biotin_lipoyl, E3_binding
mpn:MPN391	2-oxoacid_dh, Biotin_lipoyl, DEC-1_N, GCV_H, HMA, PT
mtc:MT2570	2-oxoacid_dh, Biotin_lipoyl, E3_binding, GCV_H
sce:YNL071W	2-oxoacid_dh, Biotin_lipoyl, E3_binding, GRASP55_65
spo:SPCC794.07	2-oxoacid_dh, Biotin_lipoyl, E3_binding, PYNP_C, TolA

Table 4.5: Motifs (Pfams) of the model proteins for the prediction of reaction 1 in *C. elegans*.

Protein	Protein families
cel:C30H6.7	2-oxoacid_dh, E3_binding
cel:F23B12.5	2-oxoacid_dh, Biotin_lipoyl, E3_binding, GCV_H
cel:ZK669.4	2-oxoacid_dh, Biotin_lipoyl, E3_binding

Table 4.6: Motifs (Pfams) of the proteins that catalyse reaction 1 in *C. elegans*.

Chapter 5

Experimental results

The effectiveness of the pathway prediction algorithm using each of the classification techniques described in chapter 4 was evaluated via the experimental methodology described in section 3.5. The results of those experiments are presented in this section. We focus on catalyst prediction scores over structure prediction scores because this test is certainly the more stringent one. Good catalyst predictions will imply good reaction predictions, while the converse is not necessarily true. In fact, comparing the two measurements showed that for all of the tested classifiers the reaction prediction score was always higher than the corresponding catalyst prediction score. Most of the classifiers have parameters that affect their performance. Over the course of these experiments we varied some of them in an effort to obtain the best possible performance from each classifier. Results specific to each classifier type are presented in the following subsections, while all the classifiers' best component and structure scores are summarized together in Table 5.2 and Table 5.3 for easy comparison.

5.1 BLAST NN and BLAST Threshold

When testing the BLAST NN classifier, the default parameters were used. With the BLAST Threshold classifier, only the e-value was changed. Figure 5.1 shows the precision, recall, and f-measure statistics for the pathway predictor using these two classifiers. The graph shows how the effectiveness of the classifier is significantly affected by its threshold. The threshold classifier is better than the baseline NN classifier over most of the range of reasonable e-values (10^{-14} to 10^{-128}), when comparing the results by f-measure. The threshold classifier's curves show that a reasonable threshold value needs to be chosen to guarantee good performance, so that enough similarity is required to try to match the functional parts of the protein, but enough variation is

permitted to allow for the divergence between species.

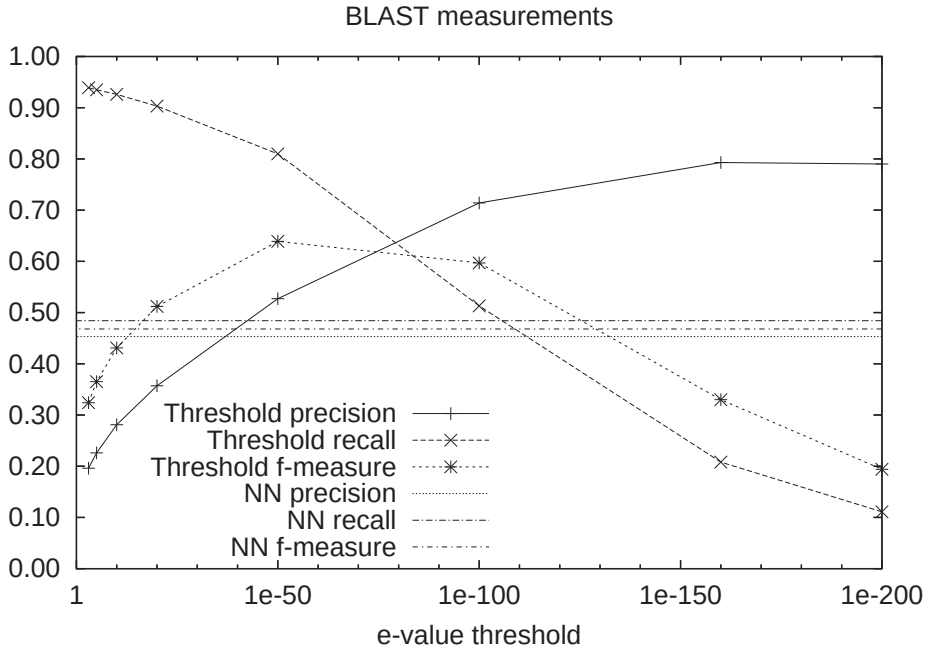


Figure 5.1: Statistics for catalyst prediction using BLAST classifiers.

5.2 Profile HMM

A variety of threshold e-values were used for the profile HMM classifier. The results of the experiments are plotted in Figure 5.2. This classifier is more precise than its BLAST-based counterpart, which also makes it slightly better than the BLAST classifier when compared by f-measure. We see that both classifiers exhibit similar behavior when varying their e-value thresholds, peaking their overall performance at an optimal point. Therefore, choosing a reasonable threshold is also important for the HMM classifier.

5.3 BLAST-HMM

By intersecting the predictions of the BLAST Threshold and profile HMM classifiers, and lowering the threshold of the HMM classifier, this classifier reached an f-measure almost 2% above the HMM classifier. In fact, the combined BLAST-HMM classifier represents an improvement in precision over both the individual classifiers (13% over BLAST, 3% over HMM). The results also show

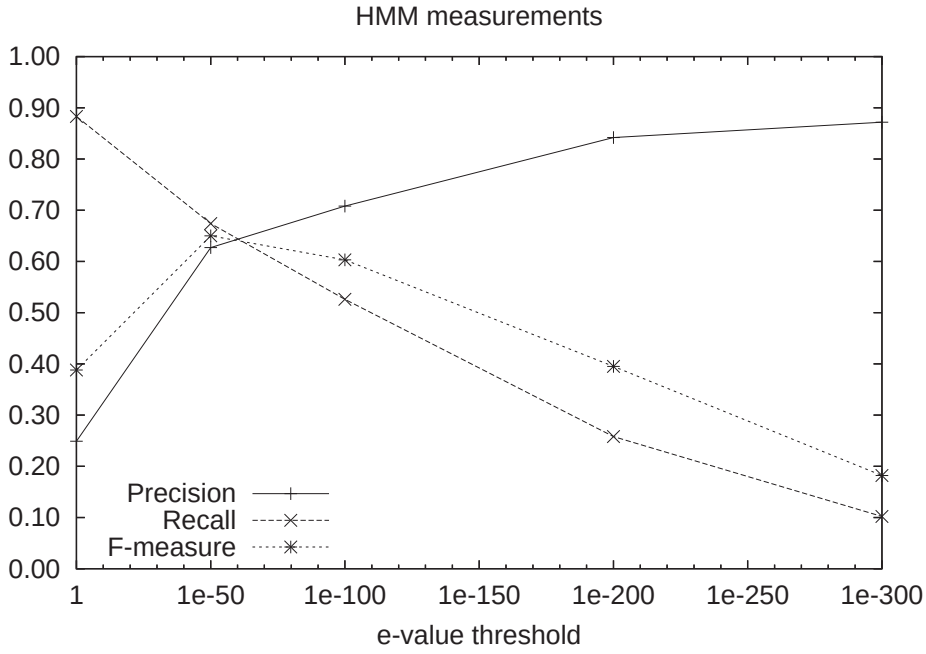


Figure 5.2: Statistics for catalyst prediction using HMM classifiers.

that there is no drop in recall when compared to the HMM classifiers; actually, there is a small increase. The relatively constant recall indicates that there probably is a large overlap between the true positive predictions of the two individual techniques, to the degree that HMM classifier's TP set is a subset of the TP BLAST predictions. This claim is further substantiated by the results shown in Table 5.1, where we can see that at an HMM e-value of 10^{-50}, the TP, FP, and FN catalyst counts for the BLAST-HMM classifiers are identical to the corresponding counts for the Profile HMM classifier. The statistics for the BLAST-HMM classifier are shown in the results summary Tables near the end of this chapter.

Classifier	**HMM e-value**	**TP**	**FP**	**TN**
BLAST-HMM	10^{-50}	2122	1261	1028
HMM	10^{-50}	2122	1261	1028

Table 5.1: Catalyst counts for the BLAST-HMM and HMM classifiers, both using an HMM threshold e-value of 10^{-50}.

Interestingly, the speed of this classifier is comparable to running BLAST only—which is significantly faster than HMMER. The reason for this performance is that this classifier runs BLAST in the work pipeline before evaluating

the proteins with the HMM model, thereby using BLAST as a filter and reducing the number of proteins that the HMMER has to evaluate to only a few per prediction. Consequently, the computation time is dominated by the BLAST searches, and the BLAST-HMM is significantly faster than the HMM classifier, while also improving on the quality of the HMM classifier's predictions.

5.4 Motif SVM classifier

The Motif SVM classifier was tested with a linear kernel and varying weights for the positive class instances. The results show that a slight increase in the weight of the positive training samples is necessary to obtain good prediction scores. This classifier did not outperform any of the BLAST or HMM threshold classifiers. In particular, the classifier had poor precision. The results of the Motif SVM classifier's evaluation are presented in Figure 5.3.

It should be noted that some experiments were also run using a radial basis function kernel, but it did not improve the performance of the classifier.

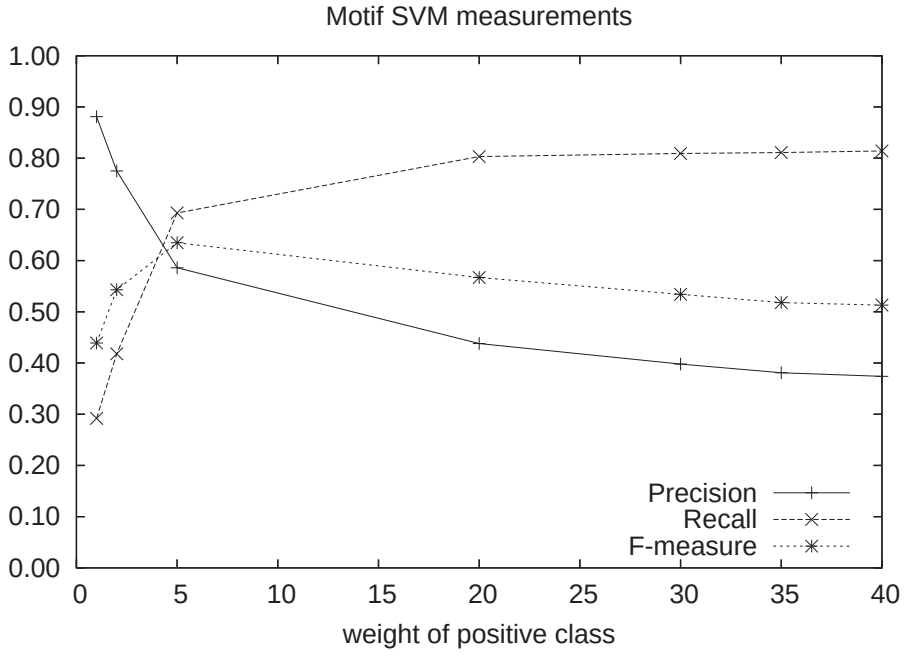


Figure 5.3: Statistics for the Pfam motif SVM classifier using a linear kernel.

Classifier	F-measure	Precision	Recall
BLAST-HMM	0.667	0.657	0.677
HMM	0.650	0.627	0.674
BLAST Thresh	0.639	0.527	0.810
Motif SVM	0.635	0.586	0.693
BLAST NN	0.468	0.453	0.484

Table 5.2: Best catalyst prediction scores (selected by F-measure) for each classifier type.

Classifier	F-measure	Precision	Recall
BLAST Thresh	0.860	0.840	0.880
HMM	0.831	0.880	0.787
BLAST-HMM	0.827	0.886	0.775
Motif SVM	0.817	0.899	0.750
BLAST NN	0.665	0.506	0.970

Table 5.3: Pathway structure prediction scores corresponding to the catalyst scores in Table 5.2.

Classifier	F-measure	Precision	Recall
BLAST Thresh	0.706	0.625	0.810
BLAST-HMM	0.704	0.733	0.677
HMM	0.689	0.706	0.674
Motif SVM	0.660	0.629	0.693
BLAST NN	0.648	0.979	0.484

Table 5.4: Best catalyst prediction scores (selected by F-measure) while not counting discovered catalysts as FP. These statistics ignore the proteins predicted to catalyse reactions that are not known to exist in the target organism.

5.5 Discussion

Table 5.2 summarizes the quality of the catalyst predictions of the presented algorithm with its various classifiers, according to the experimental procedure described in chapter 3. The results show that the classifiers making use of an HMM perform better according to the f-measure criterion, in particular due to the higher precision. However, the BLAST Threshold classifier achieves a recall significantly higher than the other classifiers. If discoveries are not counted as false positives, there is a 10% increase in the BLAST Threshold classifier's precision, which is sufficient to place it at the top of Table 5.4. This classifier's statistics are the most affected by that table's change in evaluation criteria, indicating that, out of all the tested classifiers, the BLAST Threshold

classifier is placing the largest fraction of its positive predictions in reactions that are not known to exist in the target organisms. This conclusion is also supported by the structure prediction statistics in Table 5.3, where we see that the BLAST Threshold classifier produces the lowest precision out of all classifiers except the baseline.

Table 5.3 summarizes the quality of the pathway structure predictions made by the various classifiers. Predicting the pathway structure is an easier problem than predicting the individual catalysts. This fact is reflected in this table's statistics, as the precision and recall of every classifier is higher when measuring structure predictions than when measuring catalyst predictions.

Finally, Table 5.4 shows the statistics for the catalyst predictions while ignoring discovered reactions. In other words, any positive catalyst prediction made for reactions that are not known to exist in the target organism is excluded from the false positive count. We can see the fraction of the predicted reactions that are being ignored by glancing at the precision scores in Table 5.3 (the fraction is equal to $1 - precision$). It is interesting to note that the Motif SVM classifier is placing the smallest fraction of its predicted catalysts in reactions that are not known to exist in the target organism—this is indicated by the small 4% increase in precision between the regular catalyst counting statistics and the ones that exclude the false positives in discovered reactions. This observation partially explains the high precision of its structural predictions (90%). It also implies that most of this classifier's false positives reported in Table 5.2 are proteins predicted to catalyse reactions that are already thought to exist. These may be proteins that have been missed by the analyses performed to create our experimental data set, and are potential leads for biologists to follow in the search for previously unknown catalysts. The HMM, BLAST-HMM, and BLAST Threshold classifiers also make a significant portion of their false positive predictions in reactions known to exist—8%, 8%, and 10% respectively—but to a slightly lower extent than the Motif SVM classifier.

Chapter 6

Reaction-specific classifiers

The experimental results for both the BLAST threshold and HMM threshold classifiers show that different overall e-value thresholds vary the effectiveness of the predictor. However, when analyzing these overall scores, any sense of the success of the classifier at the individual reaction nodes is lost. A more detailed analysis shows that at different reaction nodes, a different e-value threshold maximizes the f-measure. Table IV shows the f-measure scores at different e-values when predicting reaction 1 (EC 1.8.1.4 - Dihydrolipoyl dehydrogenase) and reaction 2 (EC 2.3.1.12 - Dihydrolipoyllysine-residue acetyltransferase) for the *C. elegans* instance of the Gluconeogenesis pathway. This example shows that choosing a single e-value threshold for both nodes results in sub-optimal performance for the two classifiers.

We analyzed the results of our experiments with the BLAST and HMM threshold classifiers and calculated the overall scores that could be achieved by using the best threshold at each reaction node. The results of this analysis are presented in Table 6.3 and Table 6.4, with the other predictors. They are

e-value	**f-measure 1.8.1.4**	**f-measure 2.3.1.12**
1e-03	0.250	0.750
1e-05	0.333	0.857
1e-10	0.333	0.857
1e-20	0.400	0.857
1e-50	0.667	0.800
1e-100	1.000	0.500
1e-160	1.000	0.000
1e-200	1.000	0.000

Table 6.1: F-measure metric at different e-value thresholds for the BLAST threshold classifier predicting reaction in *C. elegans*' Gluconeogenesis pathway.

e-value	f-measure 6.2.1.5	f-measure 4.1.1.9
1	0.286	1.0
1e-50	0.333	1.0
1e-100	0.4	0
1e-200	1.0	0
1e-250	0	0
1e-300	0	0

Table 6.2: F-measure metric at different e-value thresholds for the HMM threshold classifier predicting reactions in *H. sapiens*'s Propanoate metabolism pathway.

Classifier	F-measure	Precision	Recall
Opt BLAST	0.803	0.715	0.915
Opt HMM	0.767	0.706	0.881
BLAST-HMM	0.653	0.670	0.638
HMM	0.650	0.627	0.674
BLAST Thresh	0.639	0.527	0.810
Motif SVM	0.635	0.586	0.693
BLAST NN	0.468	0.453	0.484

Table 6.3: Best catalyst prediction scores (selected by F-measure) for each classifier type.

named Opt BLAST and Opt HMM. The results show that the BLAST predictor gains 19% in precision and 10% in recall, while the HMM predictor gains 8% in precision and 16.5% in recall over their constant-threshold counterparts.

Another interesting observation arises from the histogram of the best thresholds for the BLAST classifier (Figure 6.1). The histogram shows two significant peaks, indicating that there are two categories of metabolic reactions. The first one (near 1e-100) is very sensitive to the variations in the catalyst's amino acid sequence. The second (near 1e-3) is rather forgiving of variations, perhaps only being functionally affected by a small section of the protein.

This evidence suggests that for accurate pathway prediction, the decision as to whether a particular protein from the target organism is suitable to catalyse a reaction node should be made by a reaction-specific classifier. The classifier should adopt prediction techniques and parameters that are specialized for recognizing proteins that meet its particular requirements. The parameter searching involved in constructing a large number of specialized classifiers may seem like a daunting task—choosing which of the presented classifiers to use, in addition to its particular parameters. However, we believe that this burden

Classifier	F-measure	Precision	Recall
Opt BLAST	0.889	0.857	0.924
Opt HMM	0.864	0.847	0.881
BLAST Thresh	0.860	0.840	0.880
HMM	0.831	0.880	0.787
BLAST-HMM	0.821	0.908	0.749
Motif SVM	0.817	0.899	0.750
BLAST NN	0.665	0.506	0.970

Table 6.4: Pathway structure prediction scores corresponding to the catalyst scores in Table 6.3.

can be eased by utilizing machine learning techniques to select and tune the classifiers. We have started this process, but the results are beyond the scope of this dissertation.

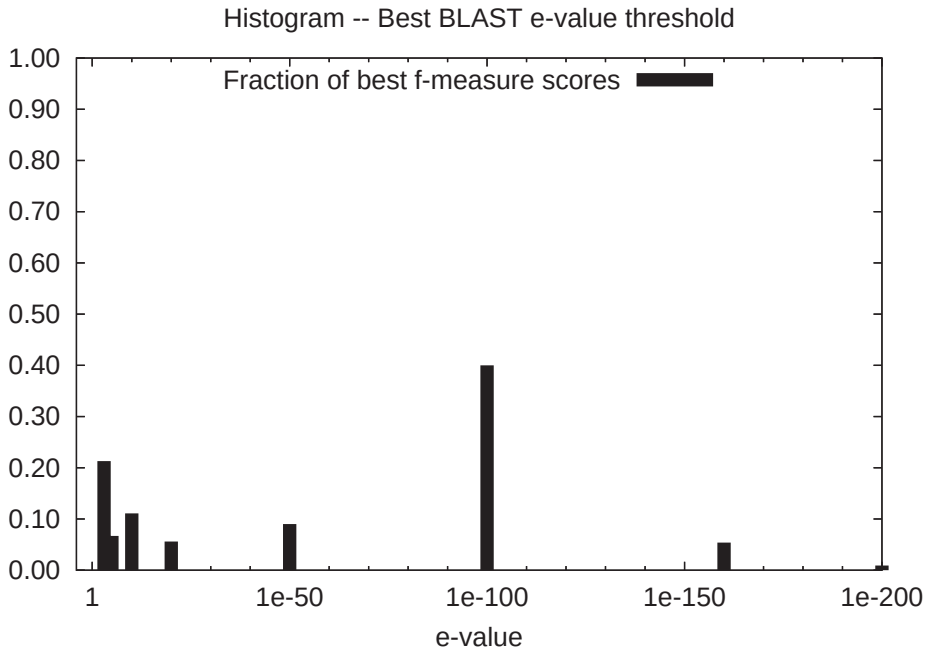


Figure 6.1: Best e-values for BLAST threshold classifier.

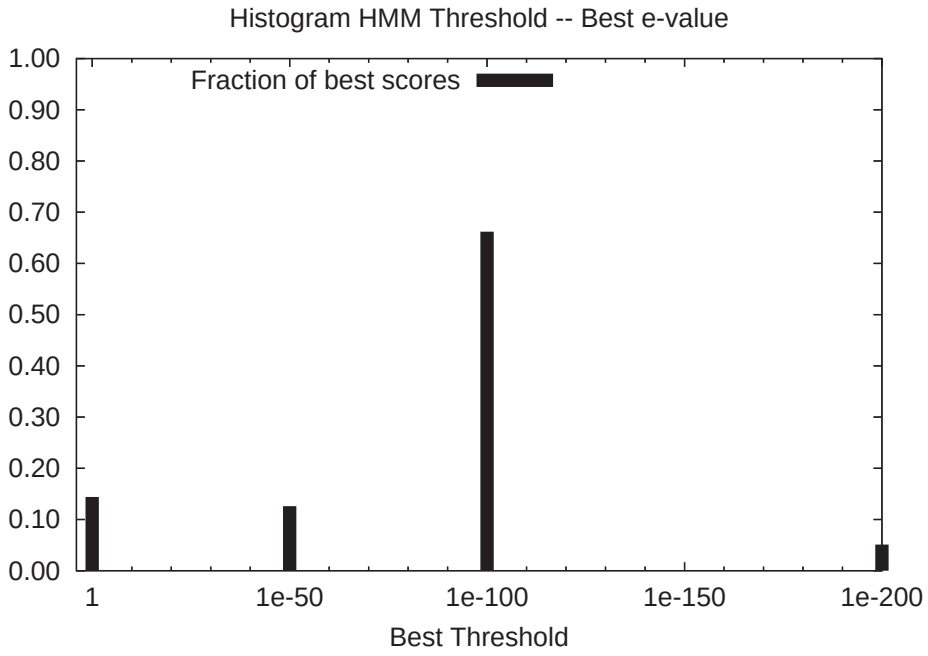


Figure 6.2: Best e-values for HMM threshold classifier.

Chapter 7

Conclusion

7.1 Future work

This section outlines some possible directions in which the work presented in this dissertation could be extended.

7.1.1 A learned reaction-specific classifier

The results presented in Chapter 6 are evidence that making good pathway predictions benefits greatly from the use of specialised classifiers for each reaction of the model pathway, as well as combinations of classifier types (BLAST-HMM). Unfortunately, the strategy of selecting a single best type and parameter combination for each model reaction by a rule, the one with greatest f-measure scored in cross-validation, presents itself as an inflexible scheme.

An alternative strategy that deserves investigation is to compute predictions with a number of classifier type and parameter combinations for each model reaction. The type and parameter combinations could be strategically chosen, perhaps based on the distribution of the best scores of the classifiers shown in Figures 6.1 and 6.2. All the individual predictions could then be collected into a single feature vector representing the "opinions" of each of the parameter variations. By collecting one such feature vector for each step of the usual cross-validation procedure, we could build a training data set that could be used to *learn* a reaction-specific rule that chooses a specialised combination of classifier types and parameters for each model reaction. The rule might consist of a support vector machine, or some other type of binary classifier. In addition, adopting this approach opens the door to using other types of information to help improve the quality of the prediction. For instance, the prediction rule could integrate the sub-cellular localization of the model proteins and the target protein into its decision.

7.1.2 Considering operons in protein classification

In bacteria, proteins that are involved in the same pathway are normally transcribed from a single section of the bacteria's DNA whose expression is under the control of a single operator gene. Such segments of DNA are known as *operons*. This knowledge could be used as a verification technique when predicting bacterial pathways, since components of the pathway coming from an operon differing from the rest could be flagged as potential false positives.

Furthermore, by a process of elimination it may be possible to use information on the proteins' source operon to predict which protein is responsible for catalysing a reaction that appears as a "gap" in a predicted pathway. In other words, suppose the classifiers from chapter 4 have resulted in a predicted bacterial pathway that contains a reaction without any assigned catalyst. If all the assigned proteins come from the same operon, and that operon has one more unassigned protein, there is a high probability that the unassigned protein is responsible for catalysing the gap reaction.

7.1.3 Test other types of reaction-specific classifiers

Due to time constraints, only the BLAST threshold and HMM threshold classifiers were tested with reaction-specific parameters. It would be interesting to see whether finding reaction-specific parameters brings the same degree of improvement to the effectiveness of the other types of classifiers.

7.1.4 Performance with other types of pathways

Because of the unavailability of data, it was not possible to evaluate the prediction performance of the presented pathway prediction techniques with non-metabolic pathways. In the event that such pathway data becomes available in a suitable form, it would be desirable to complete these tests.

7.1.5 A web-based tool for pathway prediction

The Pathway Analyst prototype discussed in this dissertation is currently being transformed into a web-based tool that will be available to the public via the Internet. This will allow users to browse and download pre-calculated pathway predictions, as well as submitting their own proteomes for a custom analysis.

7.2 Summary

In this dissertation, a computational technique has been presented for predicting biochemical pathways' reactions and catalysts. The technique is based on the premise that similarity between organisms can be exploited to use the knowledge of well-understood pathway instances to predict the structure and components of other pathway instances in organisms of interest. The algorithm and classifiers have been implemented in the Pathway Analyst prototype system, and tested by cross-validation showing that they can make accurate predictions of metabolic pathways. In particular, the Opt BLAST classifier is found to be the most effective technique out of the ones tested in maximising the f-measure of the catalyst predictions. This finding evidences the necessity to tailor the choice of classifier and its parameters to each particular reaction being predicted.

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning.* The MIT Press, 2004.

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

[3] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403—410, 1990.

[4] Alex Bateman, Ewan Birney, Lorenzo Cerruti, Richard Durbin, Laurence Etwiller, Sean R. Eddy, Sam Griffiths-Jones, Kevin L. Howe, Mhairi Marshall, and Erik L. L. Sonnhammer. The Pfam protein families database. *Nucl. Acids Res.*, 30(1):276–280, 2002.

[5] Biochemical nomenclature committees. `http://www.chem.qmw.ac.uk/iupac/jcbn/`, July 2005.

[6] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] Nello Cristianini. Support vector and kernel methods for learning. Tutorial in the Eighteenth International Conference on Machine Learning (ICML 2001), June 2001.

[8] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge University Press, 1998.

[9] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

[10] S. R. Eddy. *HMMER User's Guide*. Howard Hughes Medical Institute and Department of Genetics, 2.3.2 edition, October 2003.

[11] S. R. Eddy. HMMER: Profile hidden Markov models for biological sequence analysis. http://hmmer.wustl.edu, 2005.

[12] Michael Ashburner et al. Gene Ontology: tool for the unification of biology. *Nat Genet*, 25(1):25–29, 2000.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning; Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[14] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89(22):10915–10919, 1992.

[15] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G.R. Gopinath, G.R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucl. Acids Res.*, 33(suppl. 1):D428–432, 2005.

[16] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto encycolpedia of genes and genomes. *Nucl. Acids Res.*, 28(1):27–30, 2000.

[17] Minoru Kanehisa, Susumu Goto, Shuichi Kawashima, Yasushi Okuno, and Masahiro Hattori. The KEGG resource for deciphering the genome. *Nucl. Acids Res.*, 32(90001):D277–280, 2004.

[18] Peter D. Karp. Personal Communication, April 2005.

[19] Peter D. Karp, Monica Riley, Suzanne M. Paley, and Alida Pellegrini-Toole. The MetaCyc database. *Nucl. Acids Res.*, 30(1):59–61, 2002.

[20] Toshiaki Katayama. Personal Communication, 2005. Member of the Bioinformatics Center Institute for Chemical Research at Kyoto University (the maintainers of KEGG).

[21] Stuart Kauffman. Personal Communication, 2005. Director, Institute for Biocomplexity and Informatics.

[22] Kegg pathway database. `http://www.genome.ad.jp/kegg/pathway.html`.

[23] Cynthia J. Krieger, Peifen Zhang, Lukas A. Mueller, Alfred Wang, Suzanne Paley, Martha Arnaud, John Pick, Seung Y. Rhee, and Peter D. Karp. MetaCyc: a multiorganism database of metabolic pathways and enzymes. *Nucl. Acids Res.*, 32(90001):D438–442, 2004.

[24] Nikos Kyrpides, Ross Overbeek, and Christos Ouzounis. Universal protein families and the functional content of the last universal common ancestor. *Journal of Molecular Evolution*, 49(4):413—423, October 1999.

[25] Christian Lemer, Erick Antezana, Fabian Couche, Frederic Fays, Xavier Santolaria, Rekin's Janky, Yves Deville, Jean Richelle, and Shoshana J. Wodak. The aMAZE lightbench: a web interface to a relational database of cellular processes. *Nucl. Acids Res.*, 32(90001):D443–448, 2004.

[26] Nobelprize.org. `http://nobelprize.org/`. Official online source for Nobel Prize winner information. Developed by the Nobel Foundation web group in Stockholm.

[27] Peter Paolella. *Introduction to Molecular Biology.* McGraw-Hill Companies, Inc., 2001.

[28] Pfam: Pfam home page. `http://www.sanger.ac.uk/Software/Pfam/`. Home page of the protein families database of alignments and HMMs.

[29] Brett Poulin. Sequence-based protein function prediction. Master's thesis, University of Alberta, 2004.

[30] Seung Yon Rhee, William Beavis, Tanya Z. Berardini, Guanghong Chen, David Dixon, Aisling Doyle, Margarita Garcia-Hernandez, Eva Huala, Gabriel Lander, Mary Montoya, Neil Miller, Lukas A. Mueller, Suparna Mundodi, Leonore Reiser, Julie Tacklind, Dan C. Weems, Yihe Wu, Iris Xu, Daniel Yoo, Jungwon Yoon, and Peifen Zhang. The Arabidopsis Information Resource (TAIR): a model organism database providing a centralized, curated gateway to Arabidopsis biology, research materials and community. *Nucl. Acids Res.*, 31(1):224–228, 2003.

[31] Sorhab P. Shaw, Yong Huang, Tao Xu, Macaire M. S. Yuen, John Ling, and B. F. Francis Ouellette. Atlas—a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6(34), February 2005.

[32] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195—197, 1981.

[33] J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, November 1994.

[34] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.