Chapter 1

Introduction

1.1 General introduction

Proteins perform vital functions in all living organisms. After being synthesised in the cytosol, most of the proteins are transferred to other places in the cell or sent out to the extracellular space where they carry out their specialised tasks. One of the important questions modern biology has been trying to address is how new born proteins can find their ways in reaching their destinations. Throughout their journey they come across many obstacles, including different organelle or cell membranes, pores that have to be passed, and pathways that must be followed. Studies on the structure of the secretory pathway by George Palade were awarded with a Nobel prize in Physiology or Medicine in 1974. This and other pioneering studies yielded the theory that proteins carry intrinsic signals that govern their localisation, which sounds simple and natural to us now. This important discovery brought Günter Blobel the 1999 Nobel prize in Physiology or Medicine, less than a decade ago.

The identification of protein subcellular localisation has been the subject of

numerous experimental and computational studies. However, despite the advances made in understanding the underlying mechanisms, this complicated process has still not been fully explained. Most of the protein targeting mechanisms have been identified and well studied, certain localisation-related protein signals have been discovered, but it is still not possible to determine every proteins' localisation by inspecting only their amino acid sequences. In automatic protein localisation annotation, computational methods that rely on sequence and structure homology could be advantageous only if some other similar protein localisations have already been fully characterised in experiments. Furthermore, these methods, while performing well, cannot help much in explaining the underlying biological processes and interactions involved in protein targeting.

In this study, in accordance with the general notion that "signals govern protein targeting", my aim was to investigate whether an *ab initio*, signal-based computational prediction system can adequately help us to predict and classify subcellular localisation, without using any kind of sequence similarity, text-mining, or any kind of database searches to check for known localisation signal matches. Using known localisation signals, protein domain motifs etc., but no sequence similarity could still be anticipated as a valid *ab initio* methodology, however, in this work, in addition to predicting localisation, as a secondary goal I tried to directly discover potentially localisation-related amino acid sequence motifs as well, by extending a robust, *ab initio*, probabilistic DNA motif discovery tool program, NestedMICA (Down & Hubbard, 2005) to work on amino acid sequences.

As can be seen in the "thesis graph" (Figure 1.1), Chapter 2 is devoted to motif finding using NestedMICA, which was originally developed for transcription

factor binding site motif finding. This chapter is an extended version of our study, published under the title "NestedMICA as an *ab initio* protein motif discovery tool" (Doğruel *et al.*, 2008), where I added protein support to the program and fine tuned it for optimal protein motif discovery. A comparison of the protein-capable NestedMICA with another popular program, MEME (Bailey & Elkan, 1994), is also given in the same chapter.

In Chapter 3, I introduce Lokum, or localisation by using motifs, a novel eukaryotic protein subcellular localisation prediction program which mainly uses motifs discovered by the new NestedMICA. In addition to NestedMICA motifs found from datasets of proteins with experimentally determined localisations, I modified and used a hierarchical motif finder, Eponine (Down & Hubbard, 2002), for discovering and modeling multi-component localisation motifs such as the bipartite nuclear localisation signals. I changed Eponine, originally developed for finding transcription start site motif models, to work with amino acid sequence, too. Lokum incorporates both mono and bipartite motifs along with amino acid composition, and finally transmembrane topology statistics. In Lokum, predictions based on these features are made by a Support Vector Machine (SVM), a robust machine learning strategy.

The predicted eukaryotic localisation categories are:

1. Cytoplasmic
2. Nuclear
3. Plasma membrane

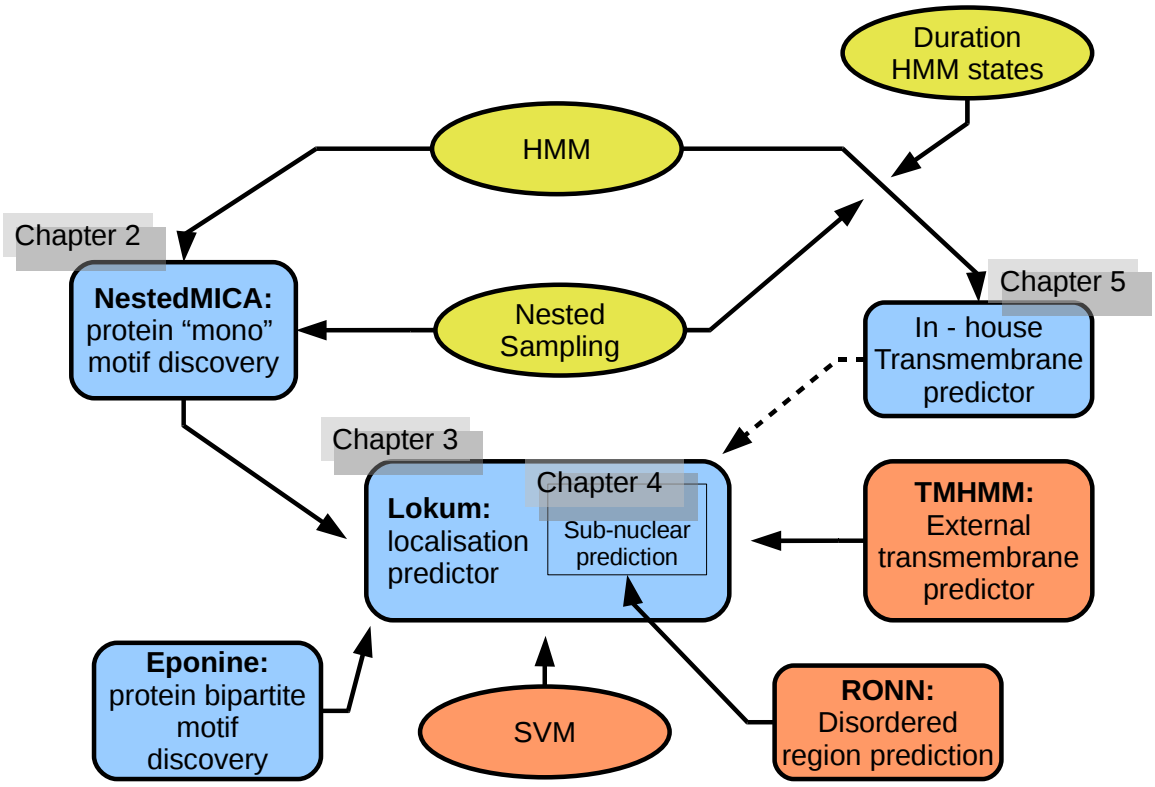

Figure 1.1: **The "thesis graph".** Main relations between the thesis chapters are shown. Rectangular shapes indicate programs that can produce "deliverables" such as a motif or a prediction, whereas elliptical shapes indicate the used intermediate computational methodologies or algorithms. Orange shapes represent external tools used, while the others are developed, implemented or modified programs.

4. Endoplasmic reticulum (ER)
5. Golgi
6. Extracellular / secretory
7. Mitochondrial
8. Peroxisomal
9. Lysosomal
10. Vacuolar
11. Chloroplast

The first nine localisations above represent the protein localisation categories for animals. Another nine categories predicted for fungi are the first eight classes with the addition of vacuolar proteins (instead of lysosomes). Finally, in addition to the categories in the list of fungal localisations, proteins targeted into the chloroplast are predicted, too, to have a total of ten categories for plants.

Chapter 4, which can be considered as an application of what is learned in Chapters 2 and 3, discusses whether it is possible to fine tune predictions by classifying some nuclear proteins in terms of their sub-localisation categories. I chose nuclear proteins as an example, because it was possible to find a significant number of protein sequences from databases, annotated as "nuclear" or "nucleolar". As an addition to the features used in Lokum, here I also evaluate the use of protein disordered regions as predicted by the RONN (Yang *et al.*, 2005) disorder predictor (Figure 1.1).

Transmembrane topologies are predicted by an external program, TMHMM (Krogh *et al.*, 2001). As an alternative to this, I developed a hidden Markov model (HMM)-based, prototype predictor (see Chapter 5) which can be plugged into this system. The underlying HMM of this predictor was optimised for its transition probabilities by using a novel procedure developed that relies on nested sampling. This new approach is introduced in the same chapter along with the prototype transmembrane topology predictor. For this HMM approach to work more efficiently I also implemented "duration capable HMMs" which are defined in this chapter.

The main chapters in the thesis have their own introduction sections that will be useful while reading a particular chapter. In this general introduction, I summarise the most popular subcellular localisation prediction programs, briefly describe three main computational techniques I used in the developed tools, namely support vector machines, HMMs, and lastly motif finding by using sampling strategies.

Finally, in the Conclusions (Chapter 6, page 165) I summarise the developed computational tools and discuss their applications in biology, together with their pros and cons. In the rest of this introductory chapter, I briefly mention previous work done on automatic protein subcellular localisation prediction and the main computational tools used.

1.1.1 Previous work on subcellular localisation prediction

Dozens of software applications are available that deal with particular aspects of subcellular localisation prediction. Some of the popular ones are listed in Table

1.1. I will first mention three widely used sets of prediction programs, before discussing others: those developed at the Danish Technical University (DTU), programs by the Rost group in Columbia University, and those developed by Kenta Nakai of the University of Tokyo and his colleagues.

Predictor	Architecture	Features	Original reference
TargetP	ANN	N-terminal sequence	Emanuelsson *et al.* (2000)
SignalP	HMM and ANN	N-terminal sequence	Nielsen *et al.* (1999) Bendtsen *et al.* (2004b)
ChloroP	ANN	Presence of cTP	Emanuelsson *et al.* (1999)
LipoP	HMM	N-terminal sequence	Juncker *et al.* (2003)
PredictNLS	Template based	NLS look-up	Cokol *et al.* (2000)
LOChom	Database	Sequence similarity	Nair & Rost (2002b)
LOCkey	Lexical analysis	Sequence similarity	Nair & Rost (2002a)
PSORT	"If-then" rules	PSORT features	Nakai & Kanehisa (1991)
PSORT II	kNN	PSORT features	Horton & Nakai (1997) Nakai & Horton (1999)
iPSORT	Rule based	N-terminal patterns	Bannai *et al.* (2002)
WolfPSORT	kNN	PSORT features, aa	Horton *et al.* (2007)
PLOC	SVM	aa	Park & Kanehisa (2003)
SubLoc	SVM	aa features	Hua & Sun (2001)
CELLO	SVM	aa of k-words	Yu *et al.* (2004)
ELSpred	SVM	aa, BLAST	Bhasin & Raghava (2004)
Proteom Analyst	Naive Bayes	SwissProt keywords	Lu *et al.* (2004)
pTarget	SVM	PFAM domains	Guda & Subramaniam (2005)
MultiLoc	SVM	aa, motif DBs	Höglund *et al.* (2006)
BaCelLo	SVM	aa, decision tree	Pierleoni *et al.* (2006)

Table 1.1: **A list of some popular eukaryotic localisation predictors.** For each prediction tool the main computational methodology and features used are listed, along with related bibliographic reference(s). ANN stands for Artificial Neural Networks, kNN represents the "k-Nearest Neighbours" algorithm, "aa" indicates amino acid composition, SVM indicates support vector machines , cTP is Chloroplast targeting peptide, and finally DB means database.

One of the most popular protein subcellular localisation predictors that use N-terminal sorting signals is TargetP (Emanuelsson *et al.*, 2000), developed at the DTU. This program is limited to only three classes (signal peptides (SP),

mitochondrial, and "other") for non-plants, and four classes (SP, mitochondrial, chloroplast, and "other") for plants. The "other" class represents proteins that do not have N-terminal signals, and consists of only nuclear and cytosolic proteins. Until now, most of the novel programs that predict the presence of N-terminal targeting signals still use TargetP datasets as a benchmark set and compare their prediction performance with that of TargetP. Another popular tool, SignalP (Bendtsen *et al.*, 2004b; Nielsen *et al.*, 1997b) from the same group, predicts the presence and location of signal peptide cleavage sites, and can accept eukaryotic, Gram-positive and Gram-negative bacteria input. SignalP has two different architectures: one is based on artificial neural networks (ANN), while the other is an HMM predictor. ChloroP (Emanuelsson *et al.*, 1999) predicts chloroplast transit peptides (cTP) and the possible cleavage site position. Similar to TargetP, it is based on ANNs. LipoP (Juncker *et al.*, 2003) predicts lipoprotein signal peptides for Gram-negative bacteria, achieving a reported correct prediction rate of 96.8%. LipoP is an HMM based prediction system. Programs developed in this group are mainly specialised tools, and based on predicting certain localisation related features in proteins. Last year, the group published a Nature Protocols article (Emanuelsson *et al.*, 2007) describing the use of several localisation predictors that aim to detect N-terminal sorting signals, including TargetP, SignalP, and ChloroP which are all hosted at DTU's Centre for Biological Sequence Analysis.

Predictors developed in Rost's group can possibly be shortened by LOC*, with the exception of PredictNLS. PredictNLS (Cokol *et al.*, 2000) uses NLSdb (Nair *et al.*, 2003), a database of nuclear localisation signals (NLS) containing both experimentally verified and "extrapolated" NLSs, to predict nuclear pro-

teins. LOChom (Nair & Rost, 2002b) is a sequence similarity based classifier, and it is based on the findings of a large-scale analysis of the relation between sequence similarity and identity in subcellular localisation. Another "LOC" program, LOCkey, (Nair & Rost, 2002a) classifies proteins according to their localisations by a lexical analysis of SWISS-PROT keywords that assigns sub-cellular localisation. LOCtarget and LOCtree (Nair & Rost, 2004) are two programs that combine and use the other LOC* predictors, with the latter being based on SVM decision trees.

Predictors based on Prof. Nakai's "localisation knowledge base" (Nakai & Kanehisa, 1991, 1992) constitute the PSORT family of programs. This knowledge base is a set of "if-then" rules that are either determined from experimental observations or derived empirically. The first PSORT predictor was announced together with the knowledge base publication by Nakai & Kanehisa in 1991. This is an expert system which is based on detection of the compiled rules. An improved version, PSORT II (Horton & Nakai, 1997; Nakai & Horton, 1999) works by detecting the same PSORT features using a "k-nearest neighbours" classifier. Bannai *et al.* extended the PSORT family by a new predictor, iPSORT (Bannai *et al.*, 2002), which is the "TargetP counterpart" of this group. It basically has additional rules to check for some physiochemical patterns in signal peptide sequences. This program did not perform as well as the neural network based TargetP, nevertheless it directly used signals and signal properties for N-terminal sorting sequence prediction. After the development of PSORT-b (Gardy *et al.*, 2003, 2005) to predict Gram-negative bacterial localisation, the newest predictor of the PSORT family, WoLF PSORT (Horton *et al.*, 2007) was released (more

than one year before its publication). WoLF PSORT, a eukaryotic localisation predictor, is an extension of PSORT II. It uses the PSORT "if-then" rules, but additionally incorporates some of the iPSORT features. This program uses amino acid composition as well as some functional motifs such as DNA-binding motifs obtained from public protein databases. As in the previous version PSORT II, it is based on the k-nearest neighbour algorithm with feature selection.

In addition to the above, there are many other, mostly support vector machine -based protein classification programs. Examples of SVM-based methods using amino acid composition as their main feature to predict eukaryotic protein localisation categories include: PLOC (Park & Kanehisa, 2003), SubLoc (Hua & Sun, 2001), CELLO (Yu *et al.*, 2004), and ELSpred (Bhasin & Raghava, 2004) (also PLSpred (Bhasin *et al.*, 2005) from the same authors for bacteria), and so on. BaCelLo (Pierleoni *et al.*, 2006) is another SVM-based prediction system that can predict 4 localisation categories for non-plant and 5 for plant protein sequences. BaCelLo does not distinguish between secretory pathway proteins. It uses N- and C-terminal sequence features such as the composition rates of amino acid chunks of different lengths from both termini.

In spite of the numerous available methods to predict protein localisation, there are only a few programs that can predict all major eukaryotic localisation categories. Apart from the WoLF-PSORT program mentioned above, Proteom Analyst (Lu *et al.*, 2004), pTarget (Guda & Subramaniam, 2005) and MultiLoc (Höglund *et al.*, 2006) are the notable multi-class predictors.

Proteom Analyst predicts protein localisations for animal, plant, fungi, Gram-negative and Gram-positive bacteria with reported correct prediction rates of

around 81% for fungi, and at rates ranging from 92 to 94% for the other four categories. These high prediction accuracies are not surprising because this method, combined with some sequence features, looks up textual subcellular localisation annotations of other homologous sequences in annotated databases to report localisation.

pTarget is a subcellular localisation predictor that searches for the presence of over 2100 PFAM (Bateman *et al.*, 2004) domains in sequences, and also uses N- and C-terminal amino acid composition. It classifies mammalian proteins in nine localisation classes. Sequences used in pTarget's development and evaluation have been filtered to remove highly homologous sequences. However, only sequences having identity rates greater than 95% were eliminated in the localisation datasets used for training and testing of the program, which possesses the danger that sequences with too high identities will be 'recognised' by the program rather than 'predicted'.

MultiLoc (Höglund *et al.*, 2006) is a new, SVM-based eukaryotic localisation predictor which combines features such as N-terminal signals, amino acid composition, and protein motifs from databases including Prosite (Hulo *et al.*, 2006) and the nuclear localisation signals database NLSdb (Nair *et al.*, 2003). It predicts nine animal, nine fungal and ten plant subcellular localisation categories with an accuracy of around 74%. It uses a total of 5959 non-homologous sequences having a maximum identity rate of 80%.

Sprenger *et al.* (2006) compared five mammalian protein subcellular localisation programs including the multi-class predictors CELLO, MultiLoc, Proteom Analyst, pTarget, and WoLF PSORT, although these are not equivalent pro-

grams in terms of their methodology and training procedures in that some are not *ab initio*, and that they were originally trained from datasets having different sequence similarity rates. Nevertheless, all these prediction programs can predict the nine major subcellular localisation categories, and they are publicly available for download or use as a web service that can accept large number of input sequences. This comparative study showed that no individual method had a sufficient level of sensitivity for the datasets used in the evaluation that would enable reliable application to entirely new or different proteins. All methods showed lower performance than reported in the original publications. The benchmarking tests were performed with low-redundancy sequences from the LOCATE database (Fink *et al.*, 2006). However, the datasets were constructed such that two-thirds of them consist of only nuclear and extracellular proteins, while the remaining seven localisation categories make up the remaining portion.

Despite this, even when we judge from what these programs report as their accuracies, there is still a need for true *ab initio* automatic classifiers that can mimic the underlying biology and predict localisation with higher accuracies in the protein annotation field.

1.2 Sequence identity thresholds

Protein subcellular localisation predictors (see the previous section on page 6) use amino acid sequences from public protein databases such as SWISS-PROT (Bairoch & Apweiler, 1996, 2000) for program training and prediction accuracy assessment purposes. Highly homologous sequences present in datasets used in program training and testing phases could result in misleading reported prediction

accuracies, and therefore must be removed from sequence datasets prior to training and testing. Different programs allow different maximum mutual sequence identity thresholds to reduce sequence redundancy. Specialised programs that predict only a certain number of protein localisation categories tend to use non-homologous sequences as determined by some homology reduction algorithms (Hobohm *et al.*, 1992), or empirically determined sequence identity thresholds that could be as low as 30%. However, those covering the majority of protein localisation categories (generally 9-11 classes) tend to use higher thresholds, as demonstrated in Table 1.2.

Program	Max allowed sequence identity
MultiLoc (Höglund *et al.*, 2006)	80%
PLOC (Park & Kanehisa, 2003)	80%
pTarget (Guda & Subramaniam, 2005)	95%
PSORTb 2.0 (Gardy *et al.*, 2005)	100%
Proteome Analyst (Szafron *et al.*, 2004)	100%

Table 1.2: **Maximum mutual sequence identity rates allowed in the different predictors.** PSORTb datasets are not filtered to eliminate sequence redundancy. Sequence homology-based programs such as Proteome Analyst tend to use the entire protein sequence sets.

On the other hand, using very low sequence identity thresholds may dramatically reduce the number of available sequences in training and testing datasets. For example, the vacuolar sequence dataset used in MultiLoc (Höglund *et al.*, 2006) (see Chapter 3) normally contains 164 sequences with no redundancy reduction applied. In MultiLoc, the allowed maximum mutual sequence percent identity was taken as 80%, which reduces the number of vacuolar protein sequences to 103 (Table 1.3).

Chothia & Lesk (1986) demonstrated the relation between the divergence of

Max allowed sequence identity	Number of vacuolar proteins
100%	164
80%	103
40%	36
30%	26
25%	23

Table 1.3: **Several allowed maximum mutual sequence identity rates versus the number of vacuolar sequences.** The vacuolar sequences refer to the same dataset used in MultiLoc and in Chapter 3. Generally, as the percent identity decreases, sequence dataset size shrinks.

sequence and structure in proteins. Sander & Schneider (1991) later showed that sequence identity does not correlate linearly with sequence homology. Namely, to avoid homologous pairs in a protein sequence dataset, the maximum percent sequence identity for long amino acid sequences must be smaller than that of relatively shorter sequences. That is, even a pairwise alignment with only 30% sequence similarity over a length of 60 residues may imply homology, but it does not if the alignment length is around 40. Generally, 30% sequence identity is regarded as a good threshold. However, as the percent identity threshold is decreased, there is a danger that there won't be sufficient number of sequences required for healthy training and testing. Therefore, whenever possible, I used 30% (Chapter 4) and when the number of sequences was critically low, 40% sequence identities (for instance, for the training and testing of Lokum: see Chapter 3). Compared to the other sequence identity thresholds used by the other mentioned multi-class predictors, the 40% maximum threshold used in Lokum is significantly lower (Table 1.2).

I used the CD-HIT (Li & Godzik, 2006; Li *et al.*, 2001, 2002) clustering algorithm to eliminate the existince of homologous sequences in the various sequence

datasets that are employed in Chapters 2, 3 and 4. As explained in Li *et al.* (2002), CD-HIT, in principle, uses the same basic sequence clustering algorithm originally developed by Hobohm *et al.* (1992) that guarantees the elimination of homologous pairs, but also uses some alternative heuristic strategies instead of directly performing pairwise alignments that could normally be quite CPU intensive. In CD-HIT, the minimum number of identical short substrings, called 'words', such as dipeptides, tripeptides and so on, shared by two proteins is a function of their sequence similarity (Li & Godzik, 2006).

1.3 Computational methodologies

Below I summarise and describe briefly the main computational methods I used (See Figure 1.1). These are, primarily, hidden Markov Models (HMM), motif finding by Nested Sampling, and SVMs. Motif finding and inference by Nested Sampling is the topic of Chapter 2 and explained there in more detail in the context of NestedMICA. Another area where Nested Sampling is used is Chapter 5 which introduces a prototype HMM based transmembrane topology predictor. Nested Sampling in Chapter 5 is used for HMM parameter optimisation. SVMs form the crux of Lokum, combining all the features used.

1.3.1 HMMs

HMMs are useful for characterising sequentially changing behaviour, including signals such as speech or a string of amino acids, in a mathematically tractable way. An HMM is a stochastic finite automaton consisting of finite states. Each state in a model is associated with a probability distribution which usually has

multiple dimensions. An outcome or observable is said to be emitted from a state based on the emission probability distribution of that particular state. Transition probabilities reflect the transition frequencies between the states of a given model. They must be explicitly set in the model.

The three main problems HMMs can address are:

1. Computing likelihood (given a set of observables, find the corresponding probability of having that sequence). This problem is solved by the forward algorithm.

2. Viterbi decoding (given a model, find the most probable sequence of states which might have yielded a certain set of observables)

3. Model learning (inferring the model parameters, mainly that of the transition probabilities, that would best describe a set of observables) Parameter optimisation or learning can be achieved with the Baum-Welch algorithm which is an expectation maximisation (EM) procedure.

HMMs have been widely used in bioinformatics (Durbin *et al.*, 1999) particularly for computational gene prediction, secondary structure prediction, and modeling of protein families and domains. For example, gene prediction using HMMs involves the second and possibly the third tasks of the above HMM objectives. In the case of motif finding with NestedMICA, we use all three canonical objectives in Chapter 2, for tasks including sequence and background likelihood calculation, model learning and fitting.

Duration HMMs (Rabiner, 1989), which were originally developed for alleviating problems in speech recognition, have many benefits in most applications

of HMMs in bioinformatics. Probability distributions of state occupancy can be represented by continuous probability density functions. Duration HMMs may utilise functions like Gaussian or Gamma distribution functions, instead of a decaying exponential in the case of classic HMMs. Thus, in practical terms, it is possible to 'set' the minimum (and in certain circumstances the maximum) number of times a model has to emit from within a certain state, once it enters that state. Figure 1.2 shows an example state occupancy probability plot for a duration-enabled HMM state with a pre-determined minimum number of self-transitions.

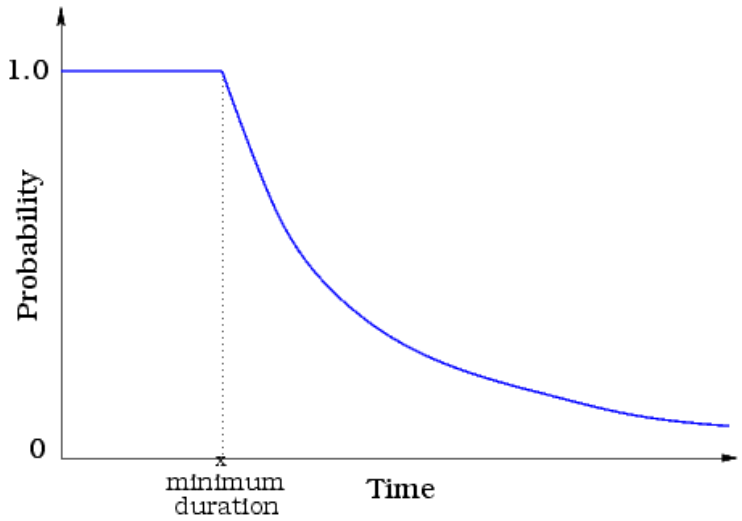

Figure 1.2: **Probability of staying in the same state in a minimum duration capable HMM state.** Normally the probability curve would be only a decaying function (of the form ae^{-x}) from a maximum probability towards zero. However, in duration-enabled states, it has to spend at least a certain number of emission times in the same state before it starts to decay (corresponding to the horizontal part in the curve).

A profile HMM (Gribskov *et al.*, 1987) consists of multiple states connected in series, none of which has a self-transition but usually a single transition to the

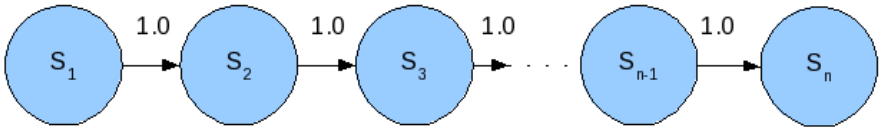

Figure 1.3: **Profile HMMs.** In this thesis, in what I call a "profile HMM" each state S_i has a single transition to the next state S_{i+1} with a duly set probability of 1.0, which makes them a way of representating position weight matrices (PWMs) in the context of HMMs.

next state. For instance, gapped multiple alignments can be represented as profile HMMs, in which case there is a need to add a "delete" and an "insert" state along with each "match" state (see Durbin *et al.* (1999) for use of HMMs in sequence alignment). However, throughout the thesis I will use the term "profile HMM" to indicate linearly constructed series of states, each of which has a transition probability of 1.0 to go to the next state, excluding the last state (Figure 1.3). In this regard, there is not much difference between such a construct and a sequence motif represented as a position weight matrix (PWM) where each fixed column has its own symbol distribution.

1.3.2 The general idea behind motif finding

Interesting motif regions and the remaining uninteresting parts of sequences can be represented as HMMs. These types of models can be referred to as sequence mixture models (SMM), as they contain states representing motifs as well as some prior models. Example of an SMM is the zero-or-one occurrences per sequence (ZOOPS) model which is the default strategy in most motif finders based on expectation maximisation (Dempster *et al.*, 1977), or Gibbs sampling (Smith, 1987), a typical example of which is the MEME (Bailey & Elkan, 1995) motif

discovery program.

NestedMICA differs from other ZOOPS models in that it does not perform a greedy search to discover the best single motif and then by masking it out focus on the next motif (if necessary). Instead, it considers different motifs at the same time and learns a model to best describe them based on independent component analysis (ICA) (Comon, 1994). In signal processing, ICA is a computational technique aiming to separate multivariate signals into independent subcomponents that constitute a given (generally noisy) signal. In linear, noiseless ICA:

$$x_i = a_{i,1}s_1 + \ldots + a_{i,k}s_k + \ldots + a_{i,n}s_n \tag{1.1}$$

where x represents the observed components vector, i.e:

$$x = (x_1, \ldots, x_m)^T \tag{1.2}$$

with the constituent components, each having a weight a_{ik}, being:

$$s = (s_1, \ldots, s_n)^T \tag{1.3}$$

The task is to be able to write s in terms of x : $s = Wx$, where W is some static transformation matrix. This situation is generally likened to the "cocktail party problem" which involves different people talking simultaneously in a room, and therefore one hears a constant random "noise". If individual components of the observed "noise" are independent, then using ICA one can try to map the individuals in the room to what each person has said. In the case of motif ICA (MICA), motifs correspond to the individual voices in this example.

1.3.3 Inference by Nested Sampling

Inferring optimal parameters for probabilistic models is a difficult task, particularly when the number of model parameters becomes large. NestedMICA performs inference using Nested Sampling (Skilling, 2004), a robust Bayesian sampling method for model selection and parameter optimisation. Nested Sampling is a Monte Carlo inference strategy which can find globally good solutions to high-dimensional problems. Classical Monte Carlo methods work by moving a single state (*i.e.* set of parameters) around the problem's parameter space, accepting or rejecting proposed moves depending on whether they increase or decrease the likelihood of the observed data. Nested Sampling is always applied to an ensemble of e different states, where the value of e is typically a few hundred. The process starts with an ensemble of states sampled uniformly from the prior.

Having sampled the states, they are then sorted in order of likelihood, and the least likely state is removed from the ensemble. To maintain the ensemble size, a new state is sampled, subject to the constraint that the new state must have a likelihood greater than that of the state it is replacing. Repeating this process many times means that nested samplers progressively move towards a small subset of the state space which contains high-likelihood states. This is somewhat analogous to simulated annealing methods where a temperature parameter is reduced to bring the model progressively closer to the posterior distribution, but nested sampling avoids the need to explicitly cool the model: progress towards high-likelihood states occurs automatically.

For each step of Nested Sampling, a certain fraction of state space is removed

from further consideration (since it contains states with likelihoods lower than the threshold). Over many steps, the fraction of prior mass that is removed from consideration at step t will tend towards [1]

$$W_t = \frac{1}{e}(\frac{e}{e+1})^t \tag{1.4}$$

where e is the ensemble size. Since all the states which have been removed from consideration will have a likelihood of approximately L_t, the likelihood of the state which was removed at step t, the Bayesian evidence for the model, Z, can be estimated as:

$$Z = \sum_{t=1}^{\infty} W_t L_t \tag{1.5}$$

Clearly, it is possibly to progressively accumulate an estimate of Z during the Nested Sampling process. The final estimate of Z can be used for model comparison purposes (for example, finding optimal parameters for the NestedMICA sequence model). NestedMICA also uses Z_t, the online Z estimate up to step t, to decide when to terminate the Nested Sampling process. Specifically, we terminate when:

$$\frac{1}{Z_t} L_t (\frac{e}{e+1})^t < 0.01 \tag{1.6}$$

i.e. the likely increase of Z in future iterations is small compared to the current value. Formally, this may lead to premature termination if L increases dramati-

[1] Derivation of this formula is explained in the 4-page Nested Sampling illustrations by David MacKay at http://www.inference.phy.cam.ac.uk/bayesys/box/nested.ps (URL last visited in 2008)

cally late in the training process, but in practice we find that this simple criterion is effective for motif discovery.

1.3.4 Support vector machines

SVMs are one of the most popular classification and regression algorithms, and are applied in a variety of disciplines for tasks including signal processing, pattern and image recognition, and biological sequence analysis. The support vector (SV) algorithm is a generalised, non-linear form of the "generalised portraits" concept developed by Vapnik in the 1960s (Vapnik & Lerner, 1963).

SVMs can be thought of as classifiers that try to maximise the geometric margin separating data points from different classes. Points are actually multidimensional feature or attribute vectors which are mapped by some selected function into some other mathematical space, where it would be more convenient to perform the required tasks such as classification or regression. This new space usually has a larger number of dimensions than the actual feature space, which in turn increases the separability of data.

This separability is determined by the VC (Vapnik-Chervonenkis) dimension of the model, which can be considered an upper theoretical limit for the set of points a classifier can "shatter" in that space. To "shatter" some given points belonging to different classes, SVMs "draw" hyperplanes near the support vectors of each class. SVs are those that are near the class boundaries, and contribute more than the other points in shaping the hyperplanes. Then, an optimal separating hyperplane is selected such that it maximises the geometric distance between any two drawn hyperplanes that define class zones. Although defining hyperplanes

associated with each class can in principle solve the problem of correctly assigning new, unseen data that is similar to the training data into one of the classes, for more "difficult" test points that lie between any two zone-determining hyperplanes, assignments can be done according to their distances to the maximum margin hyperplane.

A dot-product function can be used in simple problems for data mapping, but using kernel functions allows non-linear hyperplanes to be created. Apart from linear kernels, the most commonly used kernels are polynomial (Eq 1.7), radial-basis function (RBF)(Eq 1.8), and the sigmoid (Eq 1.9):

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d \tag{1.7}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \tag{1.8}$$

$$k(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \mathbf{x} \cdot \mathbf{x}' + c) \tag{1.9}$$

More information regarding SVMs can be found in the excellent tutorials of Burges (1998) and Smola & Scholkopf (1998), and also from SVM-dedicated web sites such as:

- `http://www.support-vector.net/`
- `http://www.support-vector-machines.org/`.

Examples of popular SVM implementations that are free for use in academic studies are *libsvm* (Chang & Lin, 2001), SVM^{light} (Joachims, 1999), and another

libsvm derivative, BSVM (Hsu & Lin, 2002). In Lokum we used both the C and Java implementations of *libsvm* (Chang & Lin, 2001), version 2.85 (see Sections 3.2.6 and 3.5).

Artificial Neural Networks (ANNs), similar to SVMs in terms of their goal and function, are widely used classifiers. ANNs differ substantially from SVMs in that their proposed solutions could correspond to some local maxima. As C. Burges put it in his SVM tutorial (Burges, 1998), "They (SVMs) differ radically from comparable approaches such as neural networks: SVM training always finds a global minimum, and their simple geometric interpretation provides fertile ground for further investigation".