Feature Learning and Graphical Models for Protein Sequences

Subhodeep Moitra

CMU-LTI-15-003

May 6, 2015

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Large Scale Feature Selection, Protein Families, Markov Random Fields, Boltzmann Machines, G Protein Coupled Receptors

To my parents. For always being there for me. For being my harshest critics. For their unconditional love. For raising me well. For being my parents.

Abstract

Evolutionarily related proteins often share similar sequences and structures and are grouped together into entities called *protein families*. The sequences in a protein family can have complex amino acid distributions encoding evolutionary relationships, physical constraints and functional attributes. Additionally, protein families can contain large numbers of sequences (deep) as well as large number of positions (wide). Existing models of protein sequence families make strong assumptions, require prior knowledge or severely limit the representational power of the models. In this thesis, we study computational methods for the task of learning rich predictive and generative models of protein families.

First, we consider the problem of large scale feature selection for predictive models. We address this in the context of a target application of designing drug cocktails against HIV-1 infection. We work with a large dataset consisting of around 70,000 HIV-1 protease and reverse transcriptase sequences. The core challenge in this setting is scaling up and selecting discriminatory features. We successfully accomplish this and provide strategies for designing cocktails of drugs robust to mutations by examining the fitness landscape learned by our predictive models.

Next, we present a framework for modelling protein families as a series of increasingly complex models using *Markov Random Fields* (MRFs). We hypothesise that by adding edges and latent variables in the MRF, we can progressively relax model assumptions and increase representational power. We note that latent variable models with cycles fail to learn effective models due to poor approximate inference thus defeating their purpose. This motivates the need for special architectures which allow efficient inference even in the company of latent variables.

Next, we extend the utility of the learned models beyond generative metrics. We introspect and interpret the learned features for biological significance by studying allostery in *G Protein Coupled Receptors* (GPCRs). We identify networks of co-evolving residues, a minimal binding pocket and long range interactions all by learning the structure of a MRF trained on the GPCR protein family.

Finally, we develop the first *Restricted Boltzmann Machines* (RBMs) and *Deep Boltzmann Machines* (DBMs) for protein sequence families. We demonstrate that these models significantly outperform their MRF counterparts in terms of imputation error. Additionally, we also consider Boltzmann Machines with sparse topologies and provide a strategy for learning their sparse structures. We note that the sparse Boltzmann Machines perform similar to MRFs thus reinforcing our hypothesis that non-sparse Boltzmann Machines are required for modelling the complex relationships inherent in protein families.

Acknowledgments

First and foremost, I would like to thank my advisor, Christopher Langmead. He has always given me unrestricted freedom to pursue my research ideas. Never on any occasion has he been unavailable for anything I wanted to discuss. I am grateful to him both for his patience and his guidance to keep me on the right track when I inevitably went astray. Next, I would like to thank Judith Klein Seetharaman. I was jointly advised by her in my initial years at CMU. I am deeply indebted to her for teaching me the value of slicing and dicing data and analyzing it for hidden treasures. It's a lesson I will carry forward for the rest of my life.

I would like to thank my thesis committee members, Jaime Carbonell and Bhiksha Raj. Their insights and critical feedback during the proposal helped me put my work on a solid foundation. I am extremely grateful to Hetunandan Kamisetty, for being on my thesis committee. A lot of the ideas in my thesis were built on his earlier work. He has been a constant presence during my time at grad school, and was always able to offer a solution whenever I was stuck on a difficult problem. I would like to thank Kalyan Tirupula and Arvind Ramanathan, who were students in Judith's and Chris's labs, respectively. It was a joy working and learning from them.

I would also like to thank the DNA team at Google, especially my mentor Sam Gross. I had a very positive experience during my internship at Google and I learned a lot from both an engineering and a scientific perspective. A major chunk of my thesis work was done in my final year at CMU, and I owe it to the "get it done" attitude that I picked up there.

My colleagues and friends at CMU have been a constant source of inspiration and support. A special mention for Prasanna Muthukumar, Ashwati Krishnan and Shriphani Palakodety. Several lunches, dinners and tea sessions have facilitated great discussions and calmed nerves. Warm thanks to Saurabh Basu, my dear friend and roommate from undergrad and grad school. Pittsburgh has been very kind to me and my friends in the Explorers Club of Pittsburgh have greatly enriched my experience here. Thanks to Michael Jehn for sharing his tastefully decorated house with me. I am very lucky to have a set of loyal friends from my high school days. Thanks to Krithika Rao, Marimuthu Ram, Rohit Ramachandran and Rose Antony for picking up my calls and keeping in touch.

Last but not the least, I am grateful to my parents. They have always had high expectations of me, but more importantly they always believed in me. I would never have made it this far without the reassurance that they are always there for me.

Funding Acknowledgement: This work was supported by NSF Grant *IIS-0905193* and NIH Grant *US NIH P41 GM103712*. Also, computational resources were provided through grants from Amazon Web Services and NVIDIA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Proteins are tiny molecular machines and are essential for life amongst all organisms. They are involved in signal transduction, structural components, transport, catalysis, immune response, etc. The function of a protein is governed by its three dimensional structure which in turn is governed by its protein sequence. A protein sequence consists of a chain of amino acids ranging anywhere between a handful to several hundred amino acids in length. Each amino acid belongs to an alphabet of 20 characters and this leads to an exponential number of possible protein sequences that can exist in nature. It is much more expensive to obtain the three dimensional crystal structure of a protein than its protein sequence. As a result the number of protein sequences in public databases far outnumbers the protein structures.

Evolutionarily related proteins known as a *protein family*, generally have similar sequences, structures, and functions. The statistical patterns within the sequences comprising a protein family can provide insights into the constraints that determine structure and function. These sequences are typically arranged in a data structure known as a *multiple sequence alignment* (MSA). A multiple sequence alignment will be the standard form of the data that we will deal with in this thesis.

Generative models of protein families are often learnt from multiple sequence alignments

[4, 42, 49]). The popularity of generative models is due in part to the fact that they can be used to perform important tasks such as structure and function classification (e.g., [42]) and to design new protein sequences (e.g., [4, 84]) such as antibodies for vaccines [91].

Additionally, *predictive models* of protein families can also be learnt from multiple sequence alignments. Such predictive models can be used to map sequence to vectors of real or categorical values with applications ranging from predicting fitness values, predicting binding partners to determining if a protein will fold or remain disordered. Predictive models directly learn a function $f : \mathcal{X} \rightarrow y$ where $\mathcal{X}$ are the predictors and y are the values to be predicted. Predictive models differ from generative models in that they do not attempt to learn a joint distribution of the data $P(\mathcal{X}, y)$ or do density estimation of the input data $P(\mathcal{X})$.

Unfortunately, despite decades of research, such models still have limitations in terms of predictive accuracies, possibly due to the hand-crafted features used in their construction. Existing models of protein sequence families make strong assumptions, require prior knowledge or severely limit the representational power of the models. In this thesis, we will learn and evaluate effective predictive and generative models of protein sequence families. We will do so by (1) large-scale feature selection for predictive models and (2) learning rich feature representations via undirected graphical models for generative models.

Figure 1.1 summarizes the generative models in the thesis at a quick glance. We learn increasingly complex models of protein families starting with *Visible Markov Random Fields* continuing on to *Hidden Markov Random Fields* and finally culminating in *Boltzmann Machines*. These models will be described in detail in subsequent chapters.

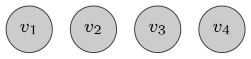
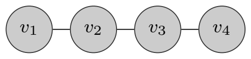
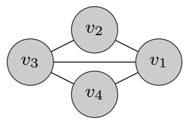
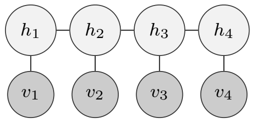
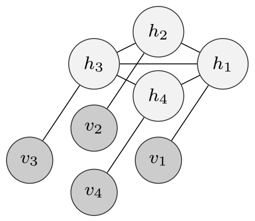
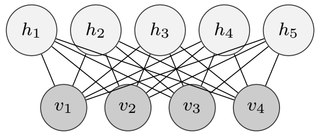
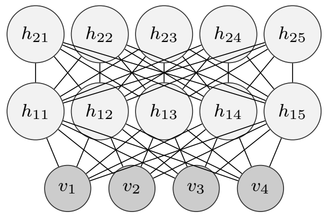
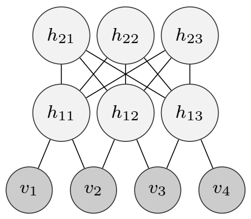
Visible Markov Random Fields	
Independent Visible MRF	
Linear Visible MRF	
General Visible MRF	
Hidden Markov Random Fields	
Linear Hidden MRF	
General Hidden MRF	
Boltzmann Machines	
Restricted Boltzmann Machine	
Deep Boltzmann Machine	 h_{21} h_{22} h_{23} h_{24} h_{25} h_{11} h_{12} h_{13} h_{14} h_{15} v_1 v_2 v_3 v_4
Locally Connected	 h_{21} h_{22} h_{23} h_{11} h_{12} h_{13} v_1 v_2 v_3 v_4

Table 1.1: Evolution of Generative Models from Visible MRFs to Boltzmann Machines

1.1 Summary of Thesis Work

In this thesis we learn and evaluate effective predictive and generative models of protein sequence families. The following are the principal contributions:

- *Large scale feature selection* for predictive models. We address this in the context of designing drug cocktails against HIV-1 infection. We work with a large dataset consisting of around 70,000 HIV-1 protease and reverse transcriptase sequences. We successfully scale our regression model and provide strategies for designing cocktails of drugs robust to mutations.
- *Visible and Hidden Markov Random Fields* of protein sequence families. We present a framework for modelling protein families as a series of increasingly complex models using Markov Random Fields (MRFs). We note that latent variable models with cycles fail to learn effective models thus motivating the need for special architectures which allow efficient inference even in the presence of latent variables.
- *Feature interpretation* of GPCRs. We extend the utility of the MRFs beyond generative metrics by introspecting and interpreting the learned features for biological significance. We study allostery in G Protein Coupled Receptors (GPCRs) and identify networks of co-evolving residues, a minimal binding pocket and long range interactions.
- *Restricted and Deep Boltzmann Machines* of protein sequence families. We develop the first Restricted Boltzmann Machines (RBMs) and Deep Boltzmann Machines (DBMs) for protein sequence families. We demonstrate that these models significantly outperform their Markov Random Field counterparts in terms of imputation error.
- *Sparse Boltzmann Machines* of protein sequence families. We introduce Boltzmann Machines with sparse topologies and provide a strategy for learning their sparse structures. We note that the sparse Boltzmann Machines perform similar to MRFs thus reinforcing our hypothesis that non-sparse Boltzmann Machines are critical for modelling the com-

plex relationships inherent in protein families.

1.2 Thesis Outline

Chapter 2 introduces relevant biology background for the thesis. We describe protein sequences and their three-dimensional structures. We also define protein families, their structural and evolutionary relationships. Finally we illustrate how multiple sequence alignments can be used to store sequences belonging to a protein family.

In chapter 3, we look at a feature selection for predictive models by addressing feature selection in the context of large scale data. We address the problem of drug cocktail design against HIV-1 infection. We work with a large dataset consisting of around 70,000 HIV-1 protease and reverse transcriptase sequences annotated with fitness values after administering 15 anti-retroviral drugs. We learn a regression model that models the sequence-fitness landscape. The regression model is then used to design a personalized cocktail of drugs that is robust to mutations arising in the viral sequence. They key challenge in this approach was scaling up our regression models to this large dataset and selecting discriminatory features.

In chapter 4, we formally introduce unsupervised feature learning for generative models of protein sequence families. We describe undirected graphical models also known as Markov Random Fields (MRFs). We introduce the notation, models, learning algorithms and evaluations techniques for MRFs. We further make a distinction between Visible Markov Random Fields (VMRFs) and Hidden Markov Random Fields (HMRFs) depending on whether they contain latent variables. We show that despite the greater representational power of HMRFs, they are unable to learn effective models due to poor inference properties. This sets up the motivation for densely connected yet learneable generative models described in the next chapter.

In chapter 5, we study the problem of feature interpretation in an unsupervised setting. We explore signal transduction in G protein coupled receptors (GPCRs) ; which relay signals across cell membranes. We identify networks of co-evolving residues from multiple sequence alignments by learning the topology of a Markov Random Field trained on GPCR sequences. We find that pairwise interactions containing residues in the ligand binding pocket are enriched. An analysis of these interactions reveals a minimal GPCR binding pocket containing four residues (T118$^{3.33}$, M207$^{5.42}$, Y268$^{6.51}$ and A292$^{7.39}$). Additionally, the ten residues predicted to have the most long-range interactions, are also part of the ligand binding pocket. This suggests that the activation in rhodopsin (a canonical GPCR) involves these long-range interactions between extracellular and intracellular domain residues mediated by the retinal domain.

In chapter 6, we approach the problem of learning rich feature representations using unsupervised generative models. A good representation can be understood as one that models the posterior distribution of the latent factors for an observed input. We focus on architectures comprising of single or multiple layers of hidden layers which operate via non-linear transformations of the data. We hypothesise that these architectures are good candidates for modelling protein families given the complex factors underlying the evolutionary processes in proteins. We employ energy based graphical models known as Boltzmann Machines and characterize them both theoretically and experimentally.

Specifically, we employ *Restricted Boltzmann Machines* (RBMs), *Deep Boltzmann Machines* (DBMs) and find that they outperform MRFs using a variety of generative metrics. We also introduce sparse versions of Boltzmann Machines *viz. Sparse Restricted Boltzmann Machines* (SRMBs) and *Locally Connected Deep Boltzmann Machines* (LC-DBMs) as a tradeoff between model complexity and representational power. We introduce a sparse MRF to sparse RBM mapping in order to create the sparse Boltzmann Machines.

Chapter 2

Background

In this chapter, we introduce the relevant background material. Throughout this thesis we make references to proteins, protein sequences, protein families, multiple sequence alignments and protein structures. This section attempts to briefly introduce these concepts, such that the reader can follow most of the biological terms used. Finally, we also provide a summary of the databases referred to in this thesis.

2.1 Proteins

Proteins are organic macromolecules that function as molecular machines. They perform a variety of different functions in living beings such as catalysis, transport, signal transduction, structural support, immune response, etc. The function of a protein is determined by its structure which in turn is determined by its sequence.

Proteins consist of long chains of amino acids. Amino acids are organic compounds containing an amino group ($-NH_2$) and a carboxyl ($-COOH$) functional group along with a unique side chain specific to the amino acid. There are 20 principal types of amino acids denoted by a single letter. These are Alanine (A), Arginine (R), Asparagine (N), Aspartic Acid (D), Cysteine (C), Glutamic Acid (E), Glutamine (Q), Glycine (G), Histidine (H), IsoLeucine (I), Leucine (L), Lysine (K), Methionine (M), Phenyalanine (F), Proline (P), Serine (S), Threonine (T), Tryptophan (W), Tyrosine (Y) and Valine (V). Amino acids in the protein are also referred to as *residues* and we will make this reference throughout the thesis.

The chain of amino acids is known as the *primary sequence* of the protein. Note that a protein sequence consisting of N amino acids can have 20^N possible combinations. The chain folds in three dimensional space and adopts both local and global shapes. The local shapes are known as the *secondary structure* of the protein consisting of α-helices, β-sheets and loops. The global shape of the protein is known as the *tertiary structure* of the protein. Note that when we refer to *protein structure* in this thesis we will be referring to the *tertiary structure* of a protein.

See Figure 2.3 for examples of two proteins. Figure 2.1 shows the structure of HIV-1 Protease, a protein which functions as molecular scissors by cutting other proteins. Figure 2.1 shows the structure of HIV-1 Reverse Transcriptase, a protein which functions converts RNA to DNA and is critical for the HIV-1 infection process in humans.

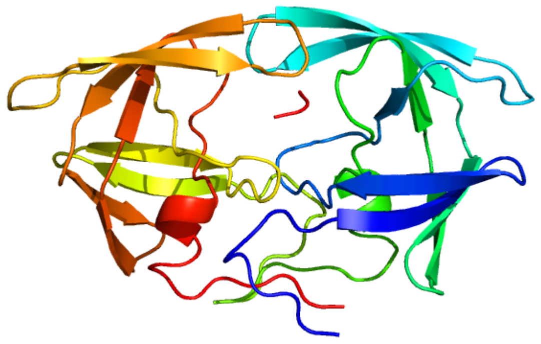


Figure 2.1: HIV-1 Protease

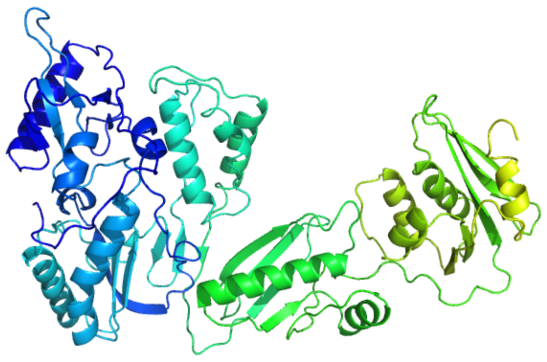


Figure 2.2: HIV-1 Reverse Transcriptase

Figure 2.3: The structures of HIV-1 Protease (pdb:1A30) and HIV-1 Reverse Transcriptase(pdb:1DLO)

2.2 Protein Families and Multiple Sequence Alignments

2.2.1 Protein Families

Evolutionarily related proteins tend to have similar sequences, structures and functions. Similar groups of sequences can be organized in entities known as *protein families*. The sequences belonging to a protein family typically tend to have similar patterns also known as its *sequence profile*. Learning predictive and generative models of protein families will be the major thrust of study in this thesis.

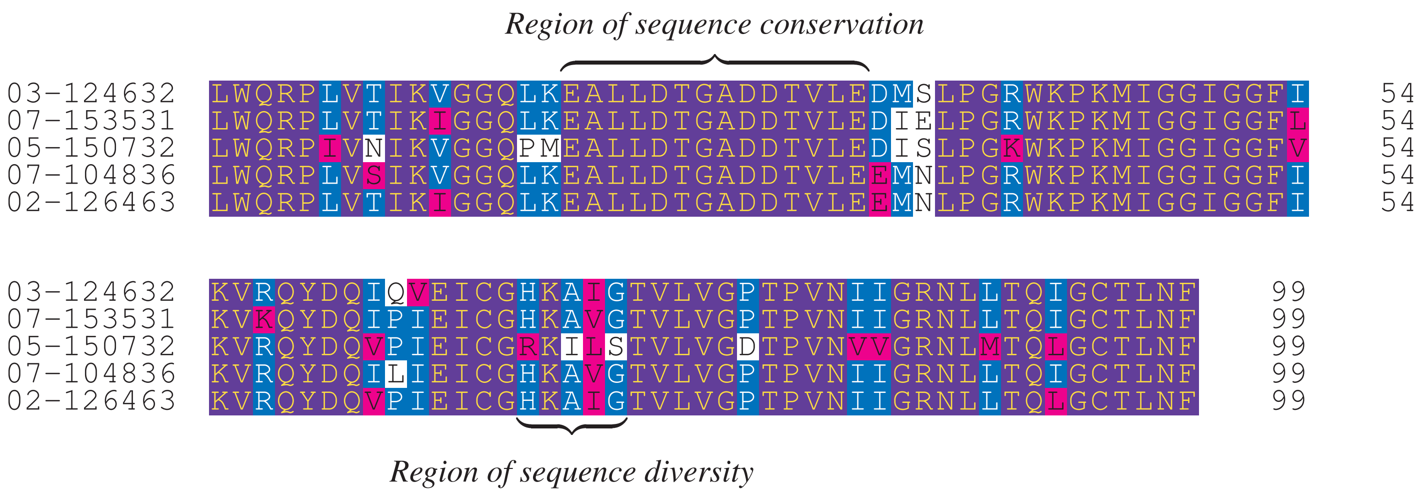
2.2.2 Multiple Sequence Alignments

Groups of protein sequences, especially those belonging to a protein family are typically arranged in a data structure known as a *Multiple Sequence Alignment (MSA)*. A MSA resembles a matrix where the rows correspond to the individual protein sequences whereas the columns correspond to the amino-acid positions. See figure 2.4 for an example of a multiple sequence alignment. Positions in the MSA which do not have any amino acids in them are represented by dashes and are known as *gaps*.

In this thesis, we will download and use pre-created MSAs of protein families. Nevertheless, it is useful to know how MSAs are created. MSAs are usually created by grouping together similar sequences and then aligning them by a dynamic programming algorithm with a custom insertion/deletion and substitution matrix. BLAST and PSI-BLAST [2] are common tools for fetching sequences from protein sequence databases such as Uniprot [15].

Figure 2.4 shows a sample multiple sequence alignment (MSA) of HIV-1 Protease containing 10 sequences. The rows corresponds to the sequence and the columns correspond to the amino acid positions in the protein sequence. This is a rather small alignment compared to the size of the

MSAs we will deal with in this thesis which contain thousands of sequences.



```
03-124632  LWQRPLVTIKVGGQLKEALLDTGADDTVLEDMSLPGRWKPKMIGGIGGFI  54
07-153531  LWQRPLVTIKIGGQLKEALLDTGADDTVLEDIELPGRWKPKMIGGIGGFL  54
05-150732  LWQRPIVNIKVGGQPMEALLDTGADDTVLEDISLPGKWKPKMIGGIGGFV  54
07-104836  LWQRPLVSIKVGGQLKEALLDTGADDTVLEEMNLPGRWKPKMIGGIGGFI  54
02-126463  LWQRPLVTIKIGGQLKEALLDTGADDTVLEEMNLPGRWKPKMIGGIGGFI  54

03-124632  KVRQYDQIQVEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF  99
07-153531  KVKQYDQIPIEICGHKAVGTVLVGPTPVNIIGRNLLTQIGCTLNF  99
05-150732  KVRQYDQVPIEICGRKILSTVLVGDTPVNVVGRNLMTQLGCTLNF  99
07-104836  KVRQYDQILIEICGHKAVGTVLVGPTPVNIIGRNLLTQIGCTLNF  99
02-126463  KVRQYDQVPIEICGHKAIGTVLVGPTPVNIIGRNLLTQLGCTLNF  99
```

Figure 2.4: A Multiple Sequence Alignment (MSA) of HIV-1 Protease containing 10 sequences. The rows corresponds to the sequence and the columns correspond to the amino acid positions in the protein sequence.

2.3 GPCRs

We briefly introduce G Protein Coupled Receptors (GPCRs) since they are discussed further in Chapter 5. In particular, we are interested in *Rhodopsin* a light signal transmitter. GPCRs are *transmembrane* proteins i.e. they are embedded in the cell membrane. They act as signal transducers by transmitting signals from outside the cell (extracellular region) to inside the cell (intracellular region). See figure 2.5 for an illustration of rhodopsin embedded in the cell membrane.

Rhodopsin has a region near its extracellular region known as the *binding pocket* where small molecules called *ligands* bind in order to facilitate the signal transduction. The ligand that binds in the ligand binding pocket is called *retinal*.

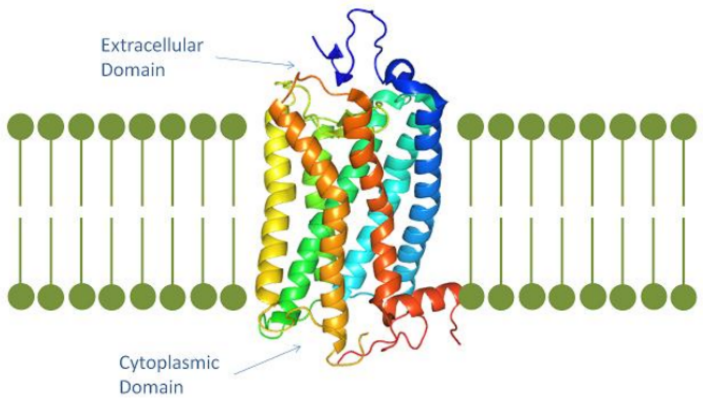


Figure 2.5: Cartoon image of Rhodopsin, a GPCR. It consists of seven transmembrane helices embedded in the cell membrane. The external face of the GPCR is known as the extracellular domain whereas the face pointing inside the cell is known as the cytoplasmic domain

2.4 Relevant Databases

In this section we briefly describe the databases referred to in this thesis and the content stored in them.

Gremlin Webserver (`http://gremlin.bakerlab.org/`) is a webserver maintained by Baker lab at the University of Washington. It provides a curated list of alignments and predictions of pairs of co-evolving residues identified by running the GREMLIN [4] algorithm.

PDB - Protein Data Bank (`http://www.rcsb.org/`) is the default repository of three dimensional structures of proteins and peptides. The structures are referred to by a *PDBid* for e.g. the PDBid of a HIV-1 reverse transriptase structure is 1DLO.

GPCRDB (GPCR database) (`http://www.gpcr.org/7tm/`) is the primary resource for information for G Protein Coupled Receptors. It includes sequence, structure, ligands, taxonomy and other related information.

Uniprot (Universal Protein Resource) (`http://www.uniprot.org/`) is an extremely exhaustive resource of proteins. It contains protein sequences, taxonomies, structures, etc. It also lists a number of cross-references of proteins to other useful databases.

Chapter 3

Feature Learning for Predictive Models of Protein Families

Predictive models directly learn a function $f : X \rightarrow y$ where X are the predictors and y are the values to be predicted. Predictive models differ from generative models in that they do not attempt to learn a joint distribution of the data $P(X, y)$ or do density estimation of the input data $P(X)$. Depending on the task, a number of different predictive models exist such as Linear Regression, Logistic Regression, Support Vector Machines, Artificial Neural Networks, etc. Regardless of the chosen predictive model, the form of the feature space is important. In particular, the data pre-processing steps of feature transformation and feature selection are critical to successful application of the predictive model for the machine learning application.

We will illustrate some of these feature transformation and feature selection strategies in the context of learning predictive models of protein sequence families. We will use the problem of drug cocktail design against HIV-1 infection both as an illustrative example and a novel application. We focus the application of our methods on a large dataset consisting of around 70,000 HIV-1 protease and reverse transcriptase sequences annotated with fitness values after administering each of the 15 anti-retroviral drugs. Specifically, we learn a regression model that models the sequence-fitness landscape. The regression model is then used to design a personalized cocktail of drugs that is robust to mutations arising in the viral sequence. They key challenge in this approach was scaling up our regression models to this large dataset and selecting discriminatory features. We accomplish this by employing several feature reduction strategies.

3.1 Background

HIV-1 (Human Immunodeficiency virus) is a retrovirus that causes AIDS in humans. HIV-1 infects humans by attacking CD4+ T cells and then releasing the contents of its viral capsid into the cytoplasm. These comprise a RNA payload and several helper viral proteins such as protease and reverse transcriptase. Figure 3.1 and figure 3.2 show the structures of protease and reverse transcriptase, respectively. Protease and reverse transcriptase are often targets of anti-retroviral drugs. These drugs often belong to three major classes - (i) Protease Inhibitors (PI) (ii) Nucleoside Reverse Transcriptase Inhibitors (NRTI) and (iii) Non-Nucleoside Reverse Transcriptase Inhibitors (NNRTI). However, The HIV virus has low fidelity and mutates rapidly upon selective pressures such as the presence of anti-retroviral drugs. A common strategy is to administer a cocktail of drugs to account for all mutation scenarios. However, this all-out strategy is suboptimal and has several drawbacks such as early immunity, increased dosage and added expense.

In order to prescribe a regimen of drugs to combat HIV-1, the key resource needed is a drug cocktail design scheme which models the viral sequence-fitness landscape. We work with a HIV-1 sequence dataset [67] containing around 70,000 HIV-1 protease and reverse transcriptase sequences from HIV-1 subtype B infected individuals undergoing routine drug resistance testing. Each of these sequences was annotated with a *replication coefficient value (RC)* that corresponds to the fitness of a viral sequence under the administration of 15 different drugs. We then proceeded to learn a multivariate regression model that maps protein sequences to the RC values. This subsequently allows us to create a drug cocktail design method robust to resistance mutations.

Previously, Hinkley et al. [29] studied the same dataset with goal of learning a regression model as well. They used a Generalized Kernel Regression model (GKRR) and used single amino acid as well as intra/intergenic pairwise mutations as their features. They mention that they would have preferred to use the sparsity inducing lasso regression model [86] had but were unable to scale because of the size of the dataset. For validation they report the percentage-deviance and not RMSE between predicted and actual values since their model does not optimize for RMSE. They also report the non-normality of RC values and transform it using the square-root transform before learning the GKRR model. They followed it up with [46, 47] where they explore the complexity of the sequence-fitness landscape using random walks.

We solve the lasso scaling problem using a variety of strategies including sparse matrix vectorization and feature reduction strategies with theoretical guarantees such as *strong rules* [87] (defined in section 3.2.3). Additionally, we employ the lasso regression model in a method for designing drug cocktails robust to HIV mutations. We define a quasi-species which uses the predicted RC values from a neighbourhood of allowed mutations to simulate a worst-case outcome. This provides helps insure the patient against mutations using the available budget of drugs. Previously, Kamisetty et al. [40] developed Gamut, a drug cocktail design method and posed the drug cocktail design problem as a graphical game. We note that a regression model is a necessary component of Gamut as well. We validate our predicted drug cocktails on a held out test data set using regret analysis commonly used in multi-armed bandit problems [50]. We find that our cocktail design method compares favorably against a variety of competing baseline strategies under a limited budget setting.

3.2 Methods

In this section we expand upon the feature transformation and feature reduction strategies used. We detail the RC value tranformation using (i) Gremlin (ii) lasso regression (iii) the various feature reduction schemes involving marginal regression (iv) El Ghaoui and strong rules (v) sparse

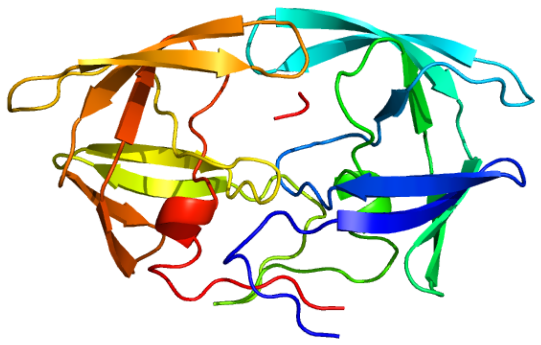


Figure 3.1: HIV-1 Protease

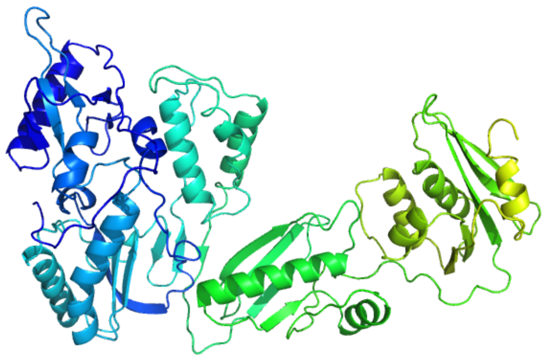


Figure 3.2: HIV-1 Reverse Transcriptase

Figure 3.3: The structures of HIV-1 Protease (pdb:1A30) and HIV-1 Reverse Transcriptase(pdb:1DLO)

matrices and (vi) the drug cocktail design strategy. The experiments and results are discussed in 3.3.

3.2.1 Gremlin Contact Map

The RC values are approximately proportional to a monontonic function of the concentration of viral particles in the patient [46]. However, the exact form of the monotonic function is not known. It is important to know the form of the monotonic function since it allows us to estimate the relative abundance of viral sequences in-vivo. Additionally, this monotonic function might define a kernel space in which deviations are Gaussian distributed, thus allowing application of $l2$ loss based regression methods [86].

Gremlin [41] is an accurate method to predict the pairwise amino acid contacts also known as the *contact map* of a protein, using just the sequence alignment of the protein. Given the inherent difficulty in predicting protein contact maps [18], we can leverage Gremlin to be used as an independent validation technique for selecting amongst candidate monotonic function transforms.

Additionally, we also examine the Gaussianity of the transformed data using Skew and kurtosis metrics. Based on evidence from the Gremlin contact map and the Gaussianity tests, we choose "square root" as the appropriate monotonic transform.

3.2.2 Lasso Regression

Lasso is a shrinkage regression and selection model [86]. We used a modified version of the lasso called an elastic net model which has an additional $L2$ loss term. The elastic net solves the following optimization problem.

$$\min_w L(w) = \frac{1}{2n_{train}} \|Xw - y\|_2^2 + \alpha\rho|w|_1 + \frac{\alpha(1-\rho)}{2}\|w\|_2^2$$

Here n_{train} is the number of training sequences, X is the sparse design matrix which is an indicator matrix consisting of single and pairwise amino acid features, w are the regression parameters, y is the transformed RC values, $\alpha \geq 0$ is the regularization penalty, $0 \leq \rho \leq 1$ is the tradeoff between the $L1$ and $L2$ loss. The regularization penalty α, is chosen by 5-fold cross validation on a held out validation set. ρ is set to 0.9 thus encouraging sparsity. We use the lasso implementation from scikit-learn [66] since it supports sparse design matrices.

3.2.3 Memory and Feature Reduction

In this section we discuss the strategies taken towards memory footprint reduction and pre-selecting features before the application of lasso.

Occurence Filtering

This is similar to the approach taken in the GKRR paper [29]. We restrict the possible single and pairwise amino acid features to the ones that appeared at least 10 times in the dataset. This shrinks the possible set of features from 35900046 to 517660.

Marginal Regression

Using single and pairwise amino acid features alone, a sequence $s_1 s_2 \cdots s_p$ can have $2^{p+\binom{p}{2}}$ possible feature functions. This is a prohibitively large space to enumerate. Marginal regression is a heuristic that enumerates only $\binom{p}{2}$ feature functions by defining the feature space to consist of all single amino acid features and one pairwise feature $\{i < j \in (1, \dots p) : s_1, s_2, \dots, s_p, s_{ij}\}$. This restricted feature space is regressed against the output y. Pairwise s_{ij} features with large magnitudes are then selected since they are likely to be predictive of y.

Strong Rules

El Ghaoui *et al.* proposed SAFE rules [26] as a technique for discarding predictors in lasso regression. It involves taking dot products between each predictor and the outcome and provably discarding predictors that don't pass a fixed threshold. Strong Rules [87] was proposed by Tibshirani et al. as a technique that improves upon SAFE rules and discards several more predictors. Consider the Lasso problem

$$\min_w \frac{1}{2}\|Xw - y\|_2^2 + \alpha\|w\|_1$$

where X is a $N \times p$ matrix of predictors with ith row x_i and jth column $\mathbf{x}_j$.

The SAFE rule would discard the jth predictor if

$$|\mathbf{x}_j^T y| < \alpha - \|x_j\|_2 \|y\|_2 \frac{\alpha_{max} - \alpha}{\alpha_{max}}$$

where $\alpha_{max} = \max_i |\mathbf{x_j}^T y|$ is the smallest penalty value at which all the coefficient are zero.

Strong Rules would discard a predictor if

$$|\mathbf{x}_j^T (y - X w_{\alpha_{k-1}})| < 2\alpha_k - \alpha_{k-1}$$

where $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$ is a grid of penalty values and $w_{\alpha_{k-1}}$ are the parameters of the regression model at penalty α_{k-1}.

We use both SAFE and strong rules for filtering out features. We use a slightly different version of strong rules. In the original strong rules paper, the design matrix is required to be mean-centered and normalized. However, centering would destroy the sparse matrix property (see 3.2.3). So we build the feature set by iteratively growing the feature set starting from the strictest penalty.

Sparse Matrices

We note that the majority of elements in the design matrix are zero. This allows us to represent the design matrix as a sparse matrix, thus keeping the entire matrix in memory. However, the gradient update calculations in lasso can no longer be vectorized and require an iterative update. We feel that this is a worthwhile space-time tradeoff.

3.2.4 Drug Cocktail design

In this section, we stipulate the strategies to choose drug cocktails given a new test sequence. We also define relevant metrics for evaluating the predicted cocktails.

Cocktail Design Strategies

- *Min-Strategy* - Pick drug cocktail based on predicted RC suppression from our regression models.
- *MinMax-Strategy* - Pick drug cocktail based on worst case predicted RC suppression from our regression models allowing sequence to mutate.
- *MinExp-Strategy* - Pick drug cocktail based on expected predicted RC suppression from our regression models allowing sequence to mutate.
- *LowestMean-Strategy* - Pick drug cocktails based on overall mean RC suppression statistics without using regression models.

- *Random-Strategy* - Pick a random drug cocktail.

Evaluation

- *Regret* - Measured as the average RC value difference between the best choice of drug cocktail in hindsight and what our strategy chose. (The lower the better)
- *Accuracy* - Retrieval of the RC value presented by the best drug cocktail. (The higher the better)

3.3 Experiments and Results

In this section, we describe the dataset and data preparation protocols. We also describe the experimental setup and results from contact map based feature transformation, feature reduction, lasso regression model and drug cocktail design studies.

3.3.1 Data description

The protein sequence dataset was obtained from Monogram Inc [67]. The dataset after cleaning contains $71,091$ HIV protease and reverse transcriptase sequences. The protease alignment had a width of 99 columns while the reverse transcriptase alignment had a width of 305 columns. Each of the sequences is annotated with 16 real non-negative Replication Coefficient (RC) values (15 drugs + 1 non-drug).

See Appendix A.1.1 for a detailed description of the data and the data preparation protocol.

3.3.2 Feature Transformation - Contact Map Analysis

The motivation and methods for the feature transformation scheme is described in section 3.2.1. Figure 3.5 shows the contact map precision after applying gremlin to a 10,000 protease alignment. The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI).

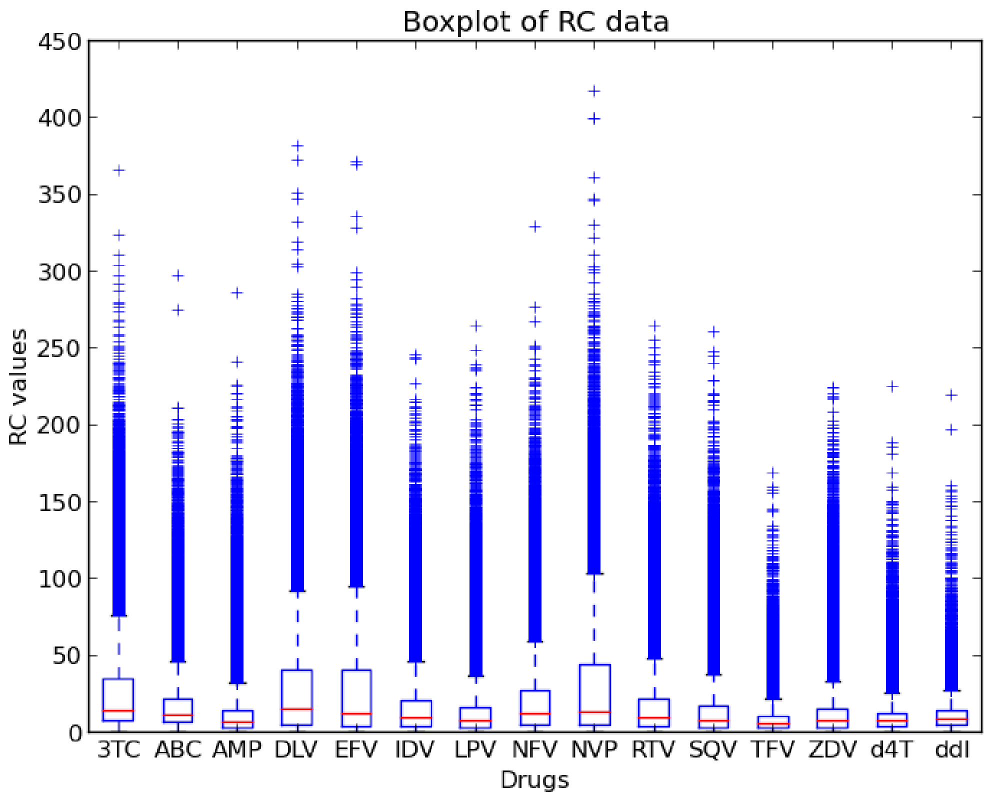


Figure 3.4: A boxplot of the RC data distribution

The X axis lists all the monotonic function transformations used (norc - no weighting, raw-untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \ldots 10\}$, $\log$ and $\log \log$). See Figure A.10 for the contact map precision plot for the reverse transcriptase alignment.

We make the following observations. For gremlin trained on protease : Protease Inhibitors (PIs) are enriched (redder) as compared to NRTI and NNRTI. For gremlin trained on reverse transcriptase: NNRTIs are enriched when compared to NRTI. A reasonable hypothesis for this result is that NNRTIs function by binding to RT and hence depend on the 3D structure of RT. NRTIs on the other hand do not bind to RT. They supply fake nucleotides which RT uses to convert RNA to DNA. Hence NRTIs do not need the 3D contact information of RT in order to be effective. This also might explain why the contact map of NNRTIs is much better predicted than NRTIs. The contact map analysis indicates for most drugs that square root transformation works fine. We note that the cube root is better in some cases but we decided to go with the square-root transformation to stay consistent with the GKRR study and also because it reduces the skew and kurtosis of the distribution to make it more Gaussian-like. Note, that the authors in the GKRR study [29] do not provide an independent validation of their choice of square-root as the monotonic RC transform.

All further analysis was done after taking the square root of the RC values.

3.3.3 Lasso Regression Model

We define `s20` as the strongest 20K features selected using strong rules. The `s20` pre-selected features are used to learn elastic net models from the entire 50K training+validation dataset. We test on the $21,091$ test sequences which have so far never been touched. Overall, 16 (15 drug + 1 drug free) regression models are learned. The train error, test error and standard deviation of the data is shown in Table 3.1. We note that the train and test error are consistent indicating that there is no overfitting. Also, the test error is lower than the standard deviation of the RC values. The standard deviation itself acts as a baseline method which predicts the mean every time. See figure 3.6 for the scatter plot of predicted vs actual RC values from the regression model.

3.3.4 KL divergence validation

The RC values can be used to compute a weighted histogram of the residue types at each position. This gives a distribution over amino acids based on the weighting scheme. We compute a KL divergence between pairs of distributions. See figure 3.7 for the different KL divergence comparisons. Pred-Real(i.e. Predicted RC and Actual RC) ; Mean-Real(i.e.Mean RC and actual RC) ; Nodrug-Real(i.e. Drug Free RC and Actual RC). A low KL divergence indicates that the distributions agree with each other. Indeed, Pred-Real has the lowest KL divergence when compared to the other cases. This further instils confidence that our models are extracting valuable

Drug	**Train Err**	**Test Err**	**Std. Dev**
N	1.4874	1.4869	1.9298
Q	1.6573	1.6614	2.0813
E	1.5513	1.5557	2.0390
G	1.8196	1.8162	2.2759
I	1.7078	1.7242	2.2021
L	1.5800	1.5802	2.0508
A	1.8593	1.8529	2.5843
R	1.4910	1.4841	1.8949
K	1.0006	0.9999	1.2267
M	1.4011	1.3993	1.8895
F	1.1123	1.0972	1.2946
P	1.0718	1.0595	1.1816
D	2.1877	2.1809	2.9369
C	2.2190	2.2093	3.0092
H	2.2931	2.2816	3.0946
NONE	2.4816	2.4581	2.9791

Table 3.1: RMSE Training and Testing Errors using Lasso

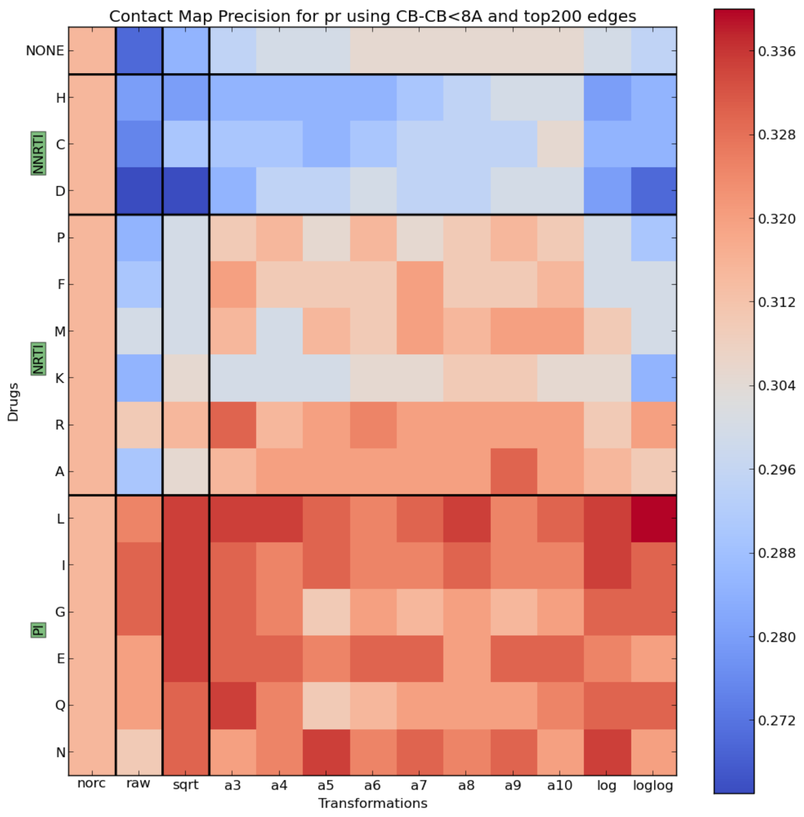


Figure 3.5: **Contact Map Analysis**: The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI). The X axis lists all the transformations used (norc - no weighting, raw- untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \dots 10\}$, $\log$ and $\log \log$) The colorbar corresponds to the precision for contact map recovery using the gremlin method. The top 200 and top 300 edges were chosen for visualization purposes. We observe that there are distinct bands according to drug type. For Protease, PIs are enriched (redder) as compared to NRTI and NNRTI. For RT, NNRTIs are enriched when compared to NRTI.

signal from the data and predicting the right outputs.

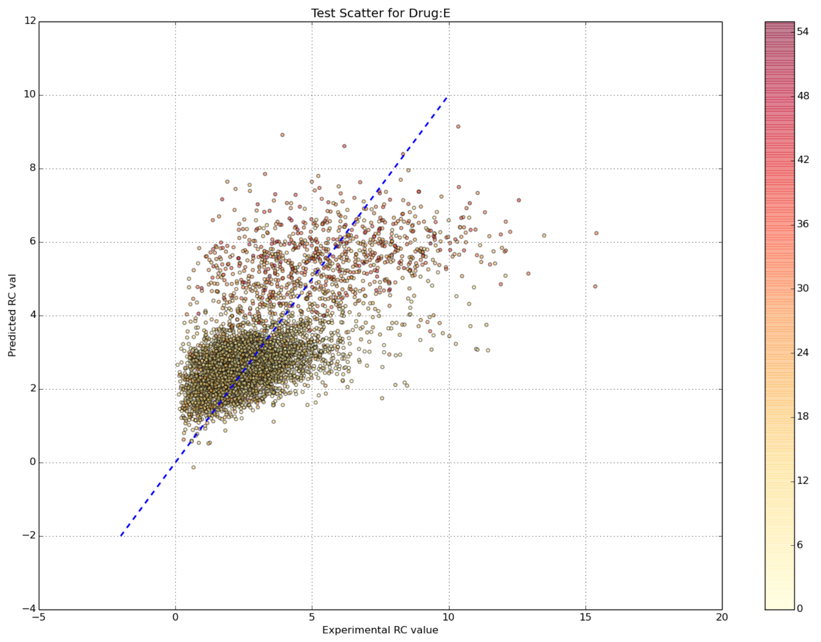


Figure 3.6: Scatter plots using test data for Elastic Net Model trained on 50K training sequences using S20 strong rules features. The color denotes edit distance from the root of a phylogenetic tree of the test sequences.

3.3.5 Cocktail Design

In section 3.2.4 we described the different drug cocktail design strategies. In this section, we evaluate those predictions. We consider the following drug cocktail budgets - *Each-3* refers to a choice of a single drug from each drug type (PI,NRTI,NNRTI). *Any-k* refers to a choice of any $k \in \{1, 2, 3, 4, 5\}$ drugs from among the 15 drugs used in the study. We also consider the following scenarios - *Radius-k* for $k \in \{1, 2, 5, 10\}$. These scenarios depict differing viral mutational proclivities ; *Radius-k* stipulates that the viral sequence is allowed to mutate upto k mutations away. The evaluation is done on held out test sequences and those within its k-radius edit distance neigbhorhood.

Figure 3.8 shows the comparison between the different cocktail design strategies using regret

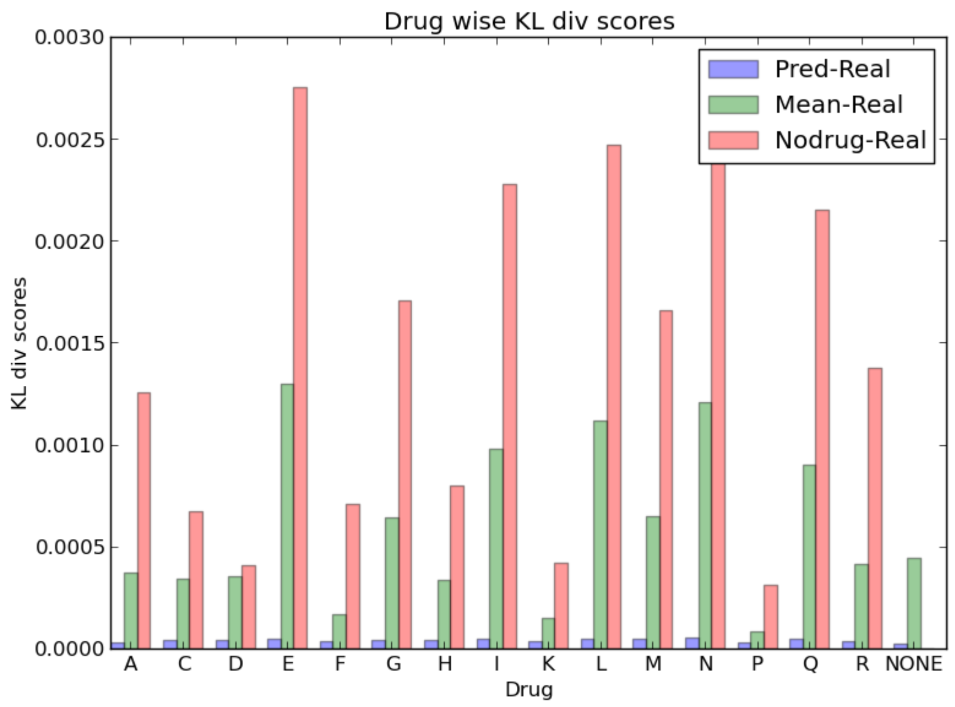


Figure 3.7: The RC values are used to compute a weighted histogram of the residue types at each position. This gives a distribution over amino acids based on the weighting scheme. We compute a KL divergence between pairs of distributions. Pred-Real:Predicted RC and Actual RC ; Mean-Real: Mean RCand actual RC ; Nodrug-Real: Drug Free RC and Actual RC. A low KL divergence indicates that the distributions agree with each other.

analysis. The X axis contains shows the different drug cocktail choices. The plot on the left (Radius 1) allows only point mutations while plot on the right (Radius 10) allows the test sequence to mutate up to 10 mutations. MinExp, MinMax and Min Strategies all use our lasso regression model. We observe that for limited budgets (upto 3 drugs) our strategies outperform the competing baseline. This suffers with increasing mutational load, but the trend generally holds. This allows us to conclude that our cocktail design method can be used for personalized medicine under a limited budget (upto 3 drugs) and is robust to viral mutations. When a greater budget of drugs is available it makes sense to use overall drug wise statistics as opposed to personalized sequence based suggestions.

See appendix figure A.15 for a comparison between the different cocktail design strategies using accuracy as a metric.

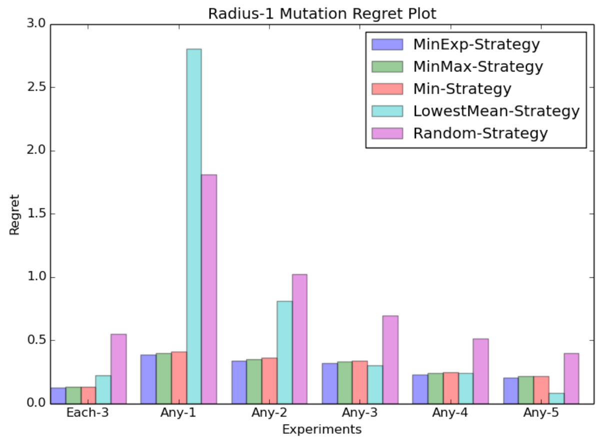


Figure 3.8: Radius 1 Regret

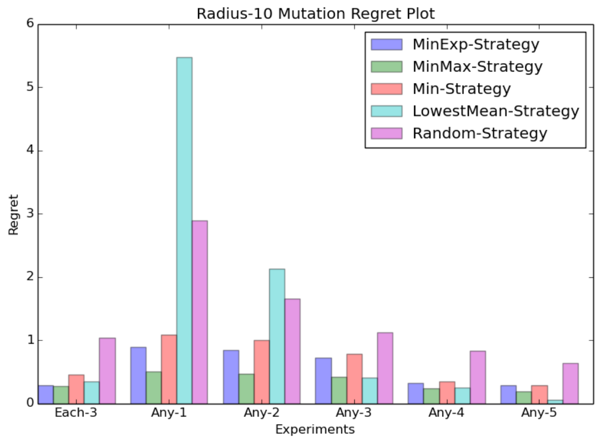


Figure 3.9: Radius 10 Regret

Figure 3.10: The X axis contains shows the different drug combination choices. Each-3 refers to a choice of a single from each drug type (PI,NRTI,NNRTI). Any-k refers to a choice of any k drugs from among the 15 drugs in the study. The Y axis shows the regret obtained in hindsight in choosing a drug combination when compared with the best possible outcome. The plot on the left (Radius 1) allows only point mutations while plot on the right (Radius 10) allows the test sequence to mutate up to 10 mutations. MinExp, MinMax and Min Strategies are all derived from our lasso regression model. We observe that for limited budgets (upto 3 drugs) our strategies outperform the competing baseline. This suffers with increasing mutational load however the trend generally holds.

3.4 Chapter Summary

In this chapter, we examined learning large scale predictive models of protein sequence families. We were able to scale the Lasso regression model to the entire HIV-1 dataset which the previous authors Hinkley et.al [29] failed to address. In particular, using drug cocktail design as an illustrative example we highlighted feature transformation and feature selection strategies. We work with a HIV-1 sequence dataset [67] containing around $70,000$ HIV-1 protease and reverse transcriptase sequences from HIV-1 subtype B infected individuals undergoing routine drug resistance testing. Each of these sequences was annotated with a replication coefficient value (RC) that corresponds to the fitness of a viral sequence. We then proceeded to learn a regression model that maps sequences to the real RC values. We solve the Lasso scaling problem using a variety of strategies including sparse matrix vectorization and feature reduction strategies such as strong rules [87]. We handle the non-normality of the RC values by transforming the RC values using contact-map prediction accuracies as an independent validation metric using Gremlin [41]. We are able to report the RMSE and this provides a concrete measurement of the predictive utility of our method. The previous authors do not report the RMSE, R^2 or the p-values of their model and this renders their results incomparable with standard regression models.

We further validate our model using a KL-divergence based metric to compare the distributions between actual and predicted RC values. Finally, we leverage the lasso regression model in a method for designing drug cocktails robust to HIV mutations. We validate our predicted drug cocktails on a held out test data set using a regret analysis. We find that our cocktail design method can be used for personalized medicine under a limited budget (upto 3 drugs) and is robust to viral mutations. We additionally suggest that for larger budgets it is beneficial to use overall drug statistics than personalized cocktail approaches.

Chapter 4

Markov Random Fields of Protein Families

In the previous chapter, we looked at large scale feature selection for predictive models. In this chapter, we will switch our focus to learning generative models of protein sequence families. Similar to predictive models, generative models too depend on high quality features for learning effective models of protein families.

We seek to learn generative graphical models of protein sequence families which can be used to model the probability density of the input data distribution [6]. A generative model can be useful in (1) describing the data distribution [41] (2) defining a prior for predictive tasks [33, 51] (3) transfer and multi-task knowledge sharing [14]. The popularity of generative models in computational biology is due in part to the fact that they can be used to perform important tasks such as structure and function classification (e.g., [42]) and to design new protein sequences (e.g., [4, 84]) such as antibodies for vaccines [91].

Unfortunately, despite decades of research, such models still have limitations in terms of predictive accuracies, possibly due to the hand-crafted features used in their construction. Existing models of protein sequence families make strong assumptions, require prior knowledge or severely limit the representational power of the models.

We will present a framework of increasingly complex models that progressively relax independence assumptions in order to define a richer feature space. Additionally, we will also intro-

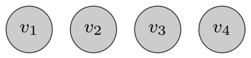
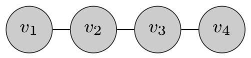
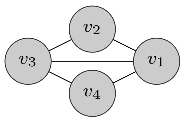
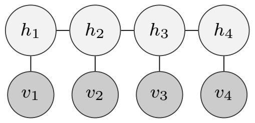
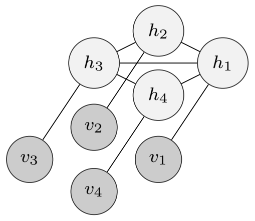
Visible Markov Random Fields	
Independent Visible MRF	
Linear Visible MRF	
General Visible MRF	
Hidden Markov Random Fields	
Linear Hidden MRF	
General Hidden MRF	

Table 4.1: Visible and Hidden Markov Random Fields

duce latent variables which attempt to model aspects of amino acid distributions not represented by visible variables alone. We will use Markov Random Fields (MRFs) as the default general class of generative models. We shall define, learn and evaluate visible and hidden Markov Random Fields. See figure 4.1 for a quick glance of the topology of models that shall be covered in this chapter.

4.1 Inputs and Notation

This section introduces the inputs and notation used for generative models of protein sequence families. This notation will be used throughout Chapter 4 and Chapter 6.

We represent a multiple sequence alignment (MSA) having N columns using a set of indices

$\mathcal{V} = \{1, \ldots, N\}$. Each of the positions is associated with a corresponding multinomial random variable v_i and together the MSA can be represented by the set $\mathbf{V} = \{v_i\,; i \in \mathcal{V}\}$. Each random variable, v_i can take one of 21 states corresponding to the twenty amino acids plus a gap character. A variable v_i taking state $k \in \{1, \ldots, 21\}$ is represented by v_i^k.

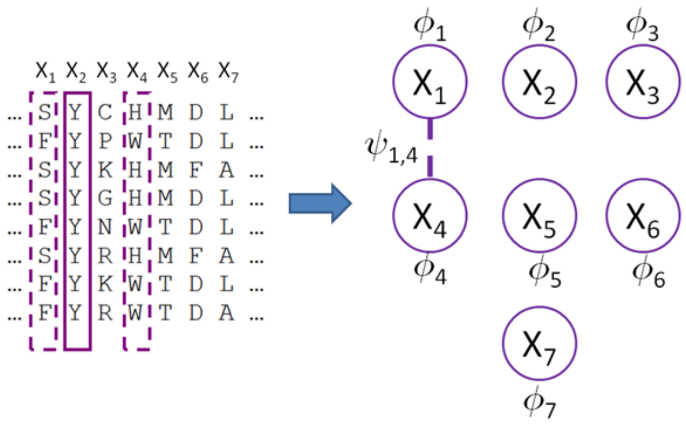


Figure 4.1: A sample MSA mapped to a MRF. Notice that column X_1 and X_4 covary and hence have a statistical dependency between them represented by an edge in the MRF

In the following part of this section, we will describe several MRF models used for generative modelling of protein families.

Independent Model The independent model (*ind*) is a simple baseline model which does not have any hidden variables and all the variables are independent of each other. See table 4.2 for an illustration. Each of the variables encodes a well defined multinomial distribution $\sum_{k=1}^{21} P(v_i^k) = \sum_{k=1}^{21} \theta_i^k = 1 \quad \forall i \in \mathcal{V}$.

The likelihood of a test sequence $\mathbf{V}$ is calculated as :

$$P(\mathbf{V} = \{v_i\ i \in \mathcal{V}\}; \Theta) = \prod_{i \in \mathcal{V}} P(v_i; \Theta) = \prod_{i \in \mathcal{V}} \theta_i^{v_i} \tag{4.1}$$

The *ind* model makes strong assumptions about the distributions of the amino acids in a protein family. These assumptions are too simplistic since adjacent and nearby amino acids in a protein sequence are connected to each other via chemical bonds and affect each other. Also, a protein molecule can fold in three dimensional space allowing distant amino acids to affect each other. These short range and long range dependencies imply corresponding correlations between the amino acids in the protein sequence which are observed as correlated mutations in a multiple sequence alignment [56].

4.2 Visible Markov Random Fields (VMRFs)

We can reduce the restrictions in the *ind* model by allowing amino acids to affect each other via statistical dependencies. These statistical dependencies give rise to what are referred to as structured models. *Markov random fields* (MRFs) are the prototypical structured models. They can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to the vertices and the edges in graph respectively . A pair of vertices (v_i, v_j) is said to be conditionally independent given their neighbours if $(i, j) \notin \mathcal{E}$. See figure 4.1 for an illustration. Notice that column X_1 and X_4 covary and hence have a statistical dependency between them represented by an edge in the MRF.

The parameters in a MRF are represented by $\Theta = (\Phi, \Psi)$ which specify the affinities of a single or a pair of amino acids to adopt a certain state. These are known as the node $\Phi = \{\phi_i\ ; i \in \mathcal{V}\}$ and the edge potentials $\Psi : \{\psi_{st}\ ; (s, t) \in \mathcal{E}\}$ respectively. We adopt a log linear

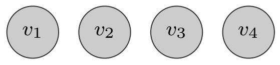
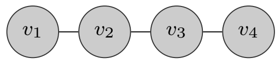
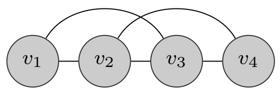
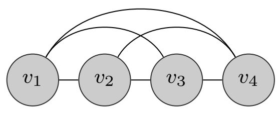
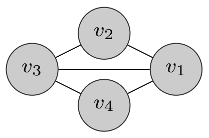
Model-name	**Architecture**	**Description**
ind		Independent Variables
linvis		Linear Chain
12vis		Every 2 connected
123vis		Every 2 and 3 connected
gremvis, 3dvis		gremlin edges, 3D edges

Table 4.2: A normal caption

parameterization of the node and the edge potentials.

$$\phi_i(v_i) = [e^{w_i^1}\ e^{w_i^2} \dots\ e^{w_i^{21}}] \tag{4.2}$$

$$\psi_{st}(v_s, v_t) = \begin{pmatrix} e^{w_{st}^{1,1}} & e^{w_{st}^{1,2}} & \cdots & e^{w_{st}^{1,21}} \\ e^{w_{st}^{2,1}} & e^{w_{st}^{2,2}} & \cdots & e^{w_{st}^{2,21}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{w_{st}^{21,1}} & e^{w_{st}^{21,2}} & \cdots & e^{w_{st}^{21,21}} \end{pmatrix} \tag{4.3}$$

4.2.1 Likelihood

The probability of a sequence $\mathbf{V}$ of amino acids is represented by:

$$P(\mathbf{V};\Theta) = \frac{1}{\mathcal{Z}(\Theta)} \prod_{i\in\mathcal{V}} \Phi_i(v_i) \prod_{(s,t)\in\mathcal{E}} \Psi_{st}(v_s, v_t) \tag{4.4}$$

where $\mathcal{Z}(\Theta)$ is an intractable sum known as the partition function and is defined as.

$$\mathcal{Z}(\Theta) = \sum_{v\in\mathbf{V}} \prod_{i\in\mathcal{V}} \Phi_i(v_i) \prod_{(s,t)\in\mathcal{E}} \Psi_{st}(v_s, v_t) \tag{4.5}$$

Finally, we define a *Visible Markov Random Field* (VMRF) as a MRF which does not contain any latent or hidden variables.

4.2.2 Parameter Learning

Parameter learning is carried out by maximizing the likelihood of the model given training sequences. Suppose that the training MSA contains M training sequences such that the m^{th} sequence is represented by a sequence of amino acids $\mathbf{V}^m$, then then MLE estimate of the parameters, Θ^* is:

$$\Theta^* = \arg\max_{\Theta=\{\Phi,\Psi\}} \prod_{m=1}^{M} P(\mathbf{V^m};\Theta) \tag{4.6}$$

$$= \arg\max_{\Theta=\{\Phi,\Psi\}} \log \prod_{m=1}^{M} P(\mathbf{V^m};\Theta) \tag{4.7}$$

$$= \arg\max_{\Theta=\{\Phi,\Psi\}} \sum_{m=1}^{M} \log P(\mathbf{V^m};\Theta) \tag{4.8}$$

The gradient with respect to the parameters for a training sequence $\mathbf{V}$ is calculated as:

$$\nabla_{\Phi_i} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[v_i] - E_{\Theta}[v_i] \tag{4.9}$$

$$\nabla_{\Psi_{st}} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[v_s v_t^T] - E_{\Theta}[v_s v_t^T] \tag{4.10}$$

where $E_{\mathbf{V}}[v_i]$ and $E_{\mathbf{V}}[v_s v_t^T]$ are the sufficient statistics of the data whereas $E_{\Theta}[v_i]$ and $E_{\Theta}[v_s v_t^T]$ are the model expectations given the current parameters of the model. This requires inferring the marginals of the variables given the parameters. Unfortunately, this is intractable for all MRFs except for tree structured graphical models. A commonly used approach is to approximate the partition function using a variational lower bound given by the loopy belief propagation algorithm [63].

4.2.3 Pseudo Log Likelihood Parameter Learning

For visible MRFs with cycles, using the *pseudo log likelihood* [9] as the objective function instead of the log likelihood can lead to better inference and thereby better learning [45]. The pseudo log likelihood is an efficient and a consistent estimator of the true log likelihood of the model.

$$\Theta^* = \underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \prod_{m=1}^{M} \prod_{i=1}^{N} P(\mathbf{v_i^m} | \mathbf{V_{-i}^m}; \Theta) \tag{4.11}$$

$$= \underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \sum_{m=1}^{M} \sum_{i=1}^{N} \log P(\mathbf{v_i^m} | \mathbf{V_{-i}^m}; \Theta) \tag{4.12}$$

The gradient with respect to the parameters for a training sequence $\mathbf{V}$ is calculated as:

$$\nabla_{\Phi_i} \log P(\mathbf{V}; \Theta) = E_{\mathbf{V}}[v_i] - E_{\mathbf{V_{-i}}}[v_i] \tag{4.13}$$

$$\nabla_{\Psi_{st}} \log P(\mathbf{V}; \Theta) = E_{\mathbf{V}}[v_s v_t^T] - E_{\mathbf{V_{-st}}}[v_s v_t^T] \tag{4.14}$$

where $E_{\mathbf{V}}[v_i]$ and $E_{\mathbf{V}}[v_s v_t^T]$ are the sufficient statistics of the data whereas $E_{\mathbf{V_{-i}}}[v_i]$ and $E_{\mathbf{V_{-st}}}[v_s v_t^T]$ are the model expectations given neighbouring variables. These expectations can be computed exactly and this makes the objective tractable even in the presence of cycles in the graph.

4.2.4 Evaluation

Our main metric for evaluating a MRF model is the imputation error. The *imputation error* is defined as the average error in predicting a particular position in the MSA given the others.

This is an unbiased test that demonstrates whether the model has learned something about the underlying probability distribution of the multiple sequence alignment. It is possible to calculate the imputation error exactly since in a VMRF the state of a random variable only depends on those of its neighbours.

$$P(v_i|v_{-i}) = P(v_i|\left\{v_j\,; \quad j \in \mathcal{N}(i)\right\}$$

where $\mathcal{N}(i)$ is the set of variables in the neighbourhood of v_i.

4.2.5 Models

In this section, we introduce particular instances of VMRFs in order of increasing number of edges.

Linear MRF

A linear VMRF (*linvis*) is a VMRF where a random variable is connected only to its adjacent neighbour in the MSA. See Figure 4.2 for an illustration. It allows us to inject the prior knowledge about the primary sequence ofa protein. Please note that this is not the same as a Hidden Markov Model since this does not have any hidden states.

Higher Order MRF

A Higher Order VMRF (*12vis, 123vis*) is a VMRF where a random variable is connected only to its neighbours in MSA within a fixed distance. See Figure 4.2 for an illustration. These edges try to capture secondary structure information e.g. α-helices can have a turn upto 4 amino acids away in sequence. Note that these models have cycles in the adjacency structure and are no longer exactly tractable.

General Graph MRF

A general graph VRMF can have an arbitrary topology. See table 4.2 for an illustration. We consider two cases of general graph VMRFs (1) Gremlin VMRF (*gremvis*) - topology learnt by running the GREMLIN algorithm [4] and (2) 3D VMRF (*3dvis*) : topology from the 3D contact map at a cutoff of $8\AA$. Note that like the Higher Order VMRF, these models have cycles in the adjacency structure and are no longer exactly tractable.

4.3 Hidden Markov Random Fields (HMRFs)

In this section we add latent variables to VMRFs and we refer to this class of models as *Hidden Markov Random Fields* (HMRFs).

The probability distribution can be affected by latent factors. Introducing hidden/latent variables in proteins can help learn rich representation arising out of evolutionary, physical and functional constraints. Latent variable models are commonly used in machine learning and computational biology [62]. Hidden Markov Models are an example of latent variable models used in several fields [43].

Hidden MRFs are undirected graphical models which contain hidden variables or unobserved states. The variable set in the graphical model is augmented with a set of hidden variables $\mathbf{H}$. We update the notation of indices of the variables to $\mathcal{V} = \{\mathcal{V}_V, \mathcal{V}_H\}$ corresponding to the indices of $\mathbf{V}$ and $\mathbf{H}$ respectively.

4.3.1 Likelihood

The likelihood of the visible variables is calculated by summing out hidden variables and is defined as:

$$P(\mathbf{V};\Theta) = \sum_{h\in H} P(\mathbf{V},h;\Theta) \tag{4.15}$$

$$= \frac{1}{\mathcal{Z}(\Theta)} \sum_{h\in H} \prod_{i\in\mathcal{V}_V} \Phi_i(v_i) \prod_{j\in\mathcal{V}_H} \Phi_j(h_j) \tag{4.16}$$

$$\prod_{(s,t)\in\mathcal{E}_V} \Psi_{st}(v_s,v_t) \prod_{(q,r)\in\mathcal{E}_H} \Psi_{qr}(h_q,h_r) \tag{4.17}$$

$$\prod_{(l,m)\in\mathcal{E}_{VH}} \Psi_{lm}(v_l,h_m) \tag{4.18}$$

$$= \frac{\mathcal{Z}_v(\Theta)}{\mathcal{Z}(\Theta)} \tag{4.19}$$

where $\mathcal{Z}(\Theta)$ is the model partition function and $\mathcal{Z}_v(\Theta)$ is the partition function after clamping down the visible units.

4.3.2 Learning

HMRF parameters are learned by maximizing the log likelihood of the training data.

$$\Theta^* = \arg\max_{\Theta=\{\Phi,\Psi\}} \sum_{m=1}^{M} \log P(V^m;\Theta) \tag{4.20}$$

$$= \arg\max_{\Theta=\{\Phi,\Psi\}} \sum_{m=1}^{M} \log \sum_{h\in H} P(V^m,H;\Theta) \tag{4.21}$$

The gradient calculation is the same for the visible variables as in VMRFs. The following changes are associated with the hidden variable parameters:

$$\nabla_{\Phi_j} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[h_j|\mathbf{V}] - E_{\Theta}[h_j] \tag{4.22}$$

$$\nabla_{\Psi_{qr}} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[h_q h_r^T|\mathbf{V}] - E_{\Theta}[h_q h_r^T] \tag{4.23}$$

where $E_{\mathbf{V}}[h_j|\mathbf{v}]$ and $E_{\mathbf{V}}[h_q h_r^T|\mathbf{v}]$ are the data dependent statistics; whereas $E_\Theta[h_j]$ and $E_\Theta[h_q h_r^T]$ are the model expectations. Similar to the VMRF, inference is intractable except for non-tree structured graphs and we resort to loopy belief propagation algorithm in those cases [63].

4.3.3 Evaluation

Evaluation is trickier for HMRFs when compared to a VMRFs. This is because of the need to perform inference in the hidden nodes. This calculation is tractable and exact only if the hidden nodes form a tree structured graph. In other cases we resort to approximate inference techniques as before. The imputation is $P(v_i|v_{-i}) = \sum_{h \in H} P(v_i, \mathbf{H}|v_{-i})$ approximated by performing inference using loopy BP. Whereas, the test log likelihood for a test sequence $\mathbf{V}^*$ is calculated as a ratio of partition functions $\mathcal{Z}_{\mathbf{V}^*}(\Theta)/\mathcal{Z}(\Theta)$.

4.3.4 Models

In this section, we introduce particular instances of VMRFs in order of increasing number of edges.

Linear HMRF

A Linear HMRF (*linhid*) is a HMRF where only adjacent hidden variables are connected to each other. The visible units have no direct connections to each other. See Table 4.3 for an illustration. This model is in principle the same as a traditional Hidden Markov Model with untied parameters and undirected edges. Inference and evaluation is performed efficiently using the sum-product algorithm [45] owing to the tree structure of a linear HMRF.

Grid HMRF

A grid HMRF (*gridhid*) is laid out in a grid like pattern with adjacent visible and hidden nodes connected to each other. See Table 4.3 for an illustration. Grid HMRFs are intractable due to the

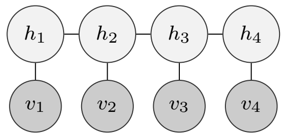
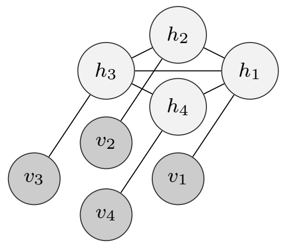
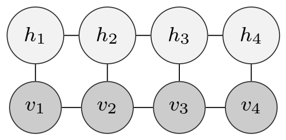
Model-name	**Architecture**	**Description**
linhid		Linear Hidden Variables
gremhid,3dhid		General Graph Hidden Layer
gridhid		Linear Chain Visible and Hidden

Table 4.3: A normal caption

introduction of cycles and we resort to loopy BP for inference in these models.

General Graph HMRF

A general graph HMRF (`gremhid, 3dhid`) allows for arbitrary connections in the hidden layer but not in the visible layer. See Table 4.3 for an illustration. Similar to the graph VMRF the hidden layer contains connections chosen with prior knowledge. Again, these models are intractable and we resort to loopy BP for inference and learning.

4.4 Experimental Results

In this section we describe the experimental results for VMRFs and HMRFs. First, we describe the datasets used. Next, we discuss algorithmic and implementation details and finally we discuss the results from the evaluations.

4.4.1 Data Description

We obtained multiple sequences alignments (MSA) for two protein families, *ubiquitin* (PF00240) and *PDZ* (PF00595), from the Baker-lab GREMLIN webserver `http://gremlin.bakerlab.org/`. These MSAs were create using Pfam HMM(s) as seeds.The alignments were further enriched with additional sequences using HHblits [69] and the uniprot database [15]. The alignments where filtered to have 75% coverage i.e. sites that had more than 75% gaps were removed. Also these alignments were processed to have at most 90% sequence identity.

The ubiquitin MSA contains 5538 sequences and 69 positions. The PDZ MSA contains 12887 sequences and 81 positions. The sequences in the alignment were randomized. The alignments were split into training, validation, and test sets using a 60%, 20% and 20% proportions, respectively. The validation sets were used to select the optimal hyper-parameters.

4.4.2 Algorithms and Implementation Details

As discussed in section 4.3.2, training VMRFs and HMRFs involves maximizing the log-likelihood of the training data. This requires calculating the gradients of the models with respect to the model parameters. The calculation of the gradients is complicated by the presence of the partition function $\mathcal{Z}(\Theta)$ which makes inference hard for certain kinds of graph structures. For tree-structured MRFs and HMRFs, the sum product algorithm can be used to perform exact inference.

For non-tree MRFs, exact inference is not possible. We use two different approximation techniques (1) loopy belief propagation (2) pseudo log likelihood. For non-tree structured HMRFs, we cannot use the pseudo log likelihood. This is due to the presence of hidden variables and the loopy belief propagation needs to be used. We use the UGM toolbox by Schmidt et.al [74] for performing inference.

Computationally heavy components were sped up using the `mex` functionality of Matlab. We used the LBFGS [94] algorithm for optimizing the parameters. The optimization routine was run for a maximum of 1000 iterations or till a convergence tolerance was reached. We used Wolfe line search [92] for adaptively picking the step length at each gradient step. Loopy belief propagation was run for a maximum of 1000 iterations or till convergence of the beliefs.

Hidden variables in HMRFs were binary-valued to make them directly comparable to the hidden states in Boltzmann Machines. We also used $l2$ regularization on the node and edge parameters with a penalty of 1 picked by cross-validation against other choices. For tree structured graphs i.e. (*linvis, linhid*) we used exact tree inference and loopy belief propagation in all other components. Also, the topology over the visible nodes in the (*gremvis*) model and in over the hidden variables in the (*gremhid*) models were computed using the GREMLIN algorithm and downloaded from the Baker-lab GREMLIN webserver (`http://gremlin.bakerlab.org/`).

4.4.3 Results and Discussion

Our primary evaluation metric was the imputation error with respect to held out test sequences. For VMRFs imputation error calculation is exact since the markov blanket of every node is fully specified. For HMRFs inference was carried out for every test instance. This is an expensive calculation for non-tree structured graphs and requires approximate inference via loopy bp. We distributed these computations over a commodity cluster.

We report the results for the ubiquitin family (PF00240) in this section. The results for PDZ family were largely similar and can be found in appendix B.1

Models trained with Log Likelihood and Loopy BP

Table 4.4 shows the train and test imputation error for the ubiquitin dataset for models trained with log likelihood and loopy belief propagation. All models do better than *ind* (completely

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
ind	0.7604	0.7572
3dhid	0.6443	0.6404
3dvis	0.6332	0.6371
12vis	0.6225	0.6300
gremvis	0.5879	0.6082
linhid	0.6069	0.6057
gridhid	0.0.5737	0.5857
linvis	0.5635	0.5753

Table 4.4: Imputation error ubiquitin : Models trained using log likelihood and loopy belief propagation

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
3dhid	0.6474	0.6437
gridhid	0.6026	0.6068
linhid	0.5971	0.5976

Table 4.5: Imputation error ubiquitin : Models trained using log likelihood and loopy belief propagation using 4 hidden states. Doubling the number of states in the hidden nodes, has no significant improvement in performance

independent). This shows that there is contextual information to be captured and most models are able to capture some aspect of it. The *linvis* and *linhid* models do fairly well. This can be attributed to exact inference in tree-structured graphs. They also have similar training and testing error, indicating that the models are not overfitting.

gremvis and *gridhid* are the next best. Surprisingly, *gremvis* which is closely related to the GREMLIN model did not have the best scores overall when compared to *linvis* (0.608 vs 0.575). We attribute this to the fact that GREMLIN internally uses pseudo log-likelihood to train the models. In the following section, we will show that pseudo log-likelihood indeed helps in better infer-

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
12vis-pseudo	0.5045	0.5374
gremvis-pseudo	0.4205	0.4952
3dvis-pseudo	0.4164	0.4856

Table 4.6: Imputation error ubiquitin : Models trained using pseudo log likelihood

ence properties for VMRFs. We chose to use log-likelihood as the objective function to keep our models comparable with latent variable models. HMRFs and Boltzmann Machines (discussed in chapter 6) contain latent variables and cannot use pseudo log-likelihood since conditioning on the other visible variables connects the graph connected via hidden nodes.

Extremely loopy graphs such as *12vis, 3dvis* and *3dhid* perform poorly. They also have a larger difference between train and test error indicating overfitting. We arrived at the conclusion, the 3D topology is not sufficient to provide features for a better generative model using log-likelihood as the objective function. The extra cycles it introduces stymies loopy belief propagation from finding correct beliefs.

We also report, the imputation errors for HMRFs using hidden nodes with 4 states in table 4.5. The models show no significant improvements when compared to the HMRFs having hidden nodes with 2 states. This provides evidence that the number of hidden states have less of an impact than the architecture of the models.

Models trained with Pseudo Log Likelihood

Table 4.6 shows the train and test imputation error for the ubiquitin dataset for models trained with pseudo log likelihood. The counterparts of the *12vis, gremvis* and *3dvis* are referred to as *12vis-pseudo*, *gremvis-pseudo* and *3dvis-pseudo*, respectively. We note that these models have much better imputation error scores. This shows that pseudo log likelihood helps in better inference and hence better learning of parameters.

Additionally, the *gremvis-pseudo* model has better scores than *12vis-pseudo* (0.495 vs 0.537). Similarly, the *3dvis-pseudo* model has better scores than *gremvis-pseudo* (0.485 vs 0.495). This seems to suggest that models with more features (denser edges) tend to have better imputation error scores. Note that the gap between the training and testing error is also widening from *linvis* to *3dvis-pseudo*, indicating that some amount of overfitting might be occuring.

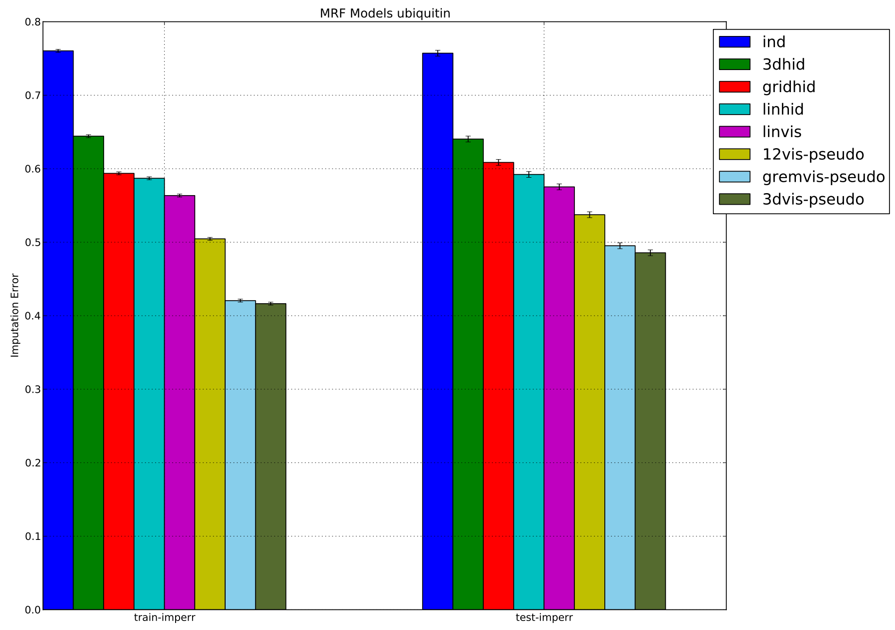
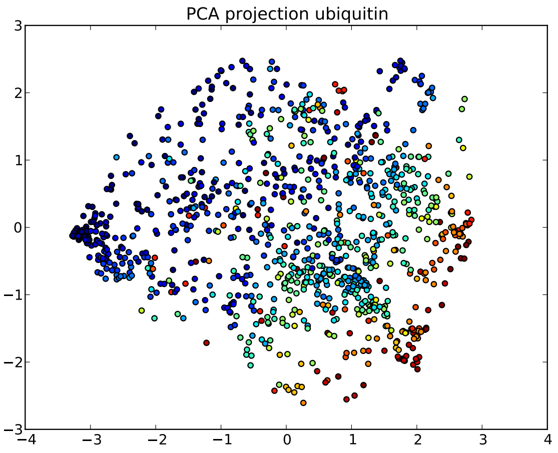


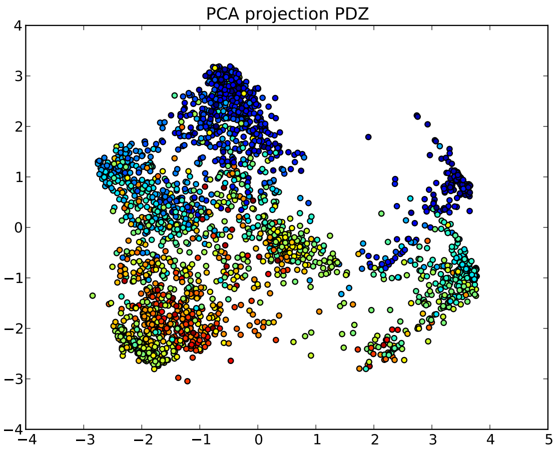
Figure 4.2: Imputation error for ubiquitin for MRF like models

We also calculated the test log-likelihood of the data. See appendix B.1 and figure B.1 for details.

Finally, we investigated the hidden layer representations to observe any patterns that emerge. We use Principal Components Analysis (PCA) on the hidden layer representation to visualize the trends in the models. Figure 4.3 shows the PCA projections of the hidden layer representations along the first two principal axes. These representations were obtained by inferring the hidden states of the *linhid* model given the input instances. The projected points were colored according to the gaps present in the sequence. A highly gapped sequence is likely to correlate with the



(a) PCA *linhid* ubiquitin



(b) PCA *linhid* PDZ

Figure 4.3: PCA projection of hidden representations from the *linhid* model. Projection colored by count of gaps in test sequences

evolutionary distance from the protein family. Notice the coloring pattern from bottom-right to upper-left in the case of ubiquitin ; and bottom-left to upper-right for PDZ. This suggests that the hidden representations might be capturing higher level attributes like evolutionary distance.

4.5 Chapter Summary

In this chapter, we discuss unsupervised generative models of protein families. We note that protein families encode complex distributions arising out of structural, functional and evolutionary constraints. Existing solutions for generative models of protein families such as PSSM matrices and Hidden Markov Models, make too many unrealistic assumptions.

We present a framework for modelling protein families as a series of increasingly complex models. These models belong to the class of undirected graphical models known as Markov Random Fields (MRFs). We show that by adding edges between variables in the MRF we can progressively relax the assumptions made about the model distribution. We present a series of models *linvis*, *12vis*, *gremvis* and *3dvis* which add pairwise, every-2, GREMLIN and 3D edges respectively.

Additionally, we make a distinction between MRF models that are completely visible and those with unobserved latent variables. The introduction of latent variables offers the flexibility of learning distributions that cannot be explained by the visible variables alone. Often, this comes at the cost of model complexity and computational tractability. We introduce the *linhid, gridhid* and *3dhid* models which add linear, grid-like and 3D edges to the hidden nodes respectively.

We learn and evaluate these models using generative metrics *viz.* imputation error. We find that all models beat a completely independent model and are hence able to model contextual information. Additionally, we find that VMRFs having many features and trained with pseudo likelihood, such as *gremvis-pseudo* and *3dvis-pseudo*, have the best performance. However, in general non-tree structured models trained using log likelihood as the objective function, do not learn well suffering from poor inference properties of loopy belief propagation. This problem is especially poignant for HMRFs which contain many cycles. They severely overfit and have poor generative performance.

We note that HMRF architectures offer the promise of modelling complex distributions, but are unable to do so in practice for all but tree-structured graphs. This motivates the need to explore other latent variable architectures that offer the possibility of better inference. We shall explore Boltzmann Machines in chapter 6, in order to tap into the potential of latent variables.

Chapter 5

Biological Analysis of Models

In chapter 4 and chapter 6 we discussed unsupervised generative models of protein families. In particular, we discussed Markov Random Fields and Boltzmann Machines. Apart from modelling the data density, these models can provide valuable insights into the data itself. This can help us learn useful biological facts about the protein family, such as important amino acid positions, pairwise correlation and even the contact map of the protein without providing any structural information.

In this chapter, we consider extending the utility of the generative models beyond just improving the test error. We do so by introspecting and interpreting the learned features beyond for biological significance. Similar to the first part of the thesis we approach this task through the lens of a target application. In particular, we examine signal transduction in *G protein coupled receptors* (GPCRs) ; which relay signals across cell membranes.

G protein coupled receptors (GPCRs) are seven helical transmembrane proteins that function as signal transducers. They bind ligands in their extracellular and transmembrane regions and activate cognate G proteins at their intracellular surface at the other side of the membrane. The relay of *allosteric* communication between the ligand binding site and the distant G protein binding site is poorly understood. In this study, we employ Gremlin [4] ,to identify networks of co-evolving residues from multiple sequence alignments, was used to identify those that may be involved in communicating the activation signal across the membrane. The Gremlin-predicted long-range interactions between amino acids were analyzed with respect to the seven GPCR structures that have been crystallized at the time this study was undertaken. See [60] for details regarding methods and results.

Gremlin significantly enriches the edges containing residues that are part of the ligand binding pocket, when compared to a control distribution of edges drawn from a random graph. An analysis of these edges reveals a minimal GPCR binding pocket containing four residues (T118$^{3.33}$, M207$^{5.42}$, Y268$^{6.51}$ and A292$^{7.39}$). Additionally, of the ten residues predicted to have the most long-range interactions (A117$^{3.32}$, A272$^{6.55}$, E113$^{3.28}$, H211$^{5.46}$, S186^{EC2}, A292$^{7.39}$, E122$^{3.37}$, G90$^{2.57}$, G114$^{3.29}$ and M207$^{5.42}$), nine are part of the ligand binding pocket.

We demonstrate the use of Gremlin to reveal a network of statistically correlated and functionally important residues in class A GPCRs. Gremlin identified that ligand binding pocket residues are extensively correlated with distal residues. An analysis of the Gremlin edges across multiple structures suggests that there may be a minimal binding pocket common to the seven known GPCRs. Further, the activation of *rhodopsin* involves these long-range interactions between extracellular and intracellular domain residues mediated by the retinal domain.

5.1 Background

G-protein coupled receptors (GPCRs) are an important class of proteins initiating major biochemical pathways sensing environmental stimuli. They are the largest protein superfamily with an estimated 1000 genes in the human genome alone [82]. An estimated 30% of known drug compounds target these receptors [64]. The GPCR family is divided into five distinct classes, class A - E [23]. The class A family is the largest class and includes rhodopsin, the prototypical GPCR, for which the first crystal structure of any GPCR was solved [65]. Its ligand is 11-cis retinal (RT), covalently attached to the protein. 11-cis RT isomerizes to all-trans RT upon light incidence, resulting in activation of the receptor. In GPCRs, the binding of a ligand in the EC or TM domain is the signal that is propagated to the IC domain wherein different effectors bind, in particular the G protein heterotrimer, GPCR receptor kinases (GRK) and -arrestin.

Receptor activation is an inherently allosteric process where the ligand binding signal is communicated to a distant site. The activation of rhodopsin and other class A GPCRs is thought to be conserved and involves rearrangements in structural microdomains [5] . Conformational changes of multiple "switches" in tandem activate the receptor [1] . These long-range interactions between distant residues are important for the function of the receptors and are also closely involved in their folding and structural stability [44, 68] . Identifying the residues involved in the propagation of signals within the protein is important in understanding the mechanism of activation. While much information can be directly extracted from crystal structures, allosteric interactions are dynamic and implicit in nature and thus are not directly observable in static crystal structures. Experimental methods for investigating dynamics, such as nuclear magnetic resonance, are presently incapable of resolving allosteric interactions in large membrane proteins, such as GPCRs.

Due to the limitations of experimental methods, statistical analysis of GPCR sequences is an alternative in identifying residues that may be involved in allosteric communication. Here, considerable effort has been directed towards identifying networks of co-evolving residues from multiple sequence alignments (MSA), i.e. residues that are statistically correlated in the MSA. Such correlations are thought to be necessary for function, and may provide insights into how signals are propagated between different domains. A number of computational methods have been developed to identify such couplings from MSAs, including Hidden Markov Models (HMMs) [22] , Statistical Coupling Analysis (SCA) [56, 81] , Explicit Likelihood of Subset Co-variation (ELSC) [17] , Graphical Models for Residue Coupling (GMRC) [83] , and Generative REgularized ModeLs of proteINs (Gremlin) [4]. Like the GMRC method, Gremlin learns an undirected probabilistic graphical model known as a Markov Random Field (MRF). Unlike HMMs, which are also graphical models, MRFs are well suited to modelling long-range couplings (i.e., between non-sequential residues). The SCA and ELSC methods return a set of residue couplings (which may include long-range couplings), but unlike MRFs, they do not distinguish between direct (conditionally dependent) and indirect (conditionally independent) correlations. This distinction is crucial in determining whether an observed correlation between two residues can be explained in terms of a network of correlations involving other residues. The key difference between the GMRC and Gremlin methods [4] is that Gremlin is statistically consistent and guaranteed to learn an optimal MRF, whereas the GMRC uses heuristics to learn the MRF. We have previously reported detailed comparisons of the GMRC and Gremlin methods and found that Gremlin achieved higher accuracy and superior scalability.

In accordance with the demonstrated advantages of Gremlin over other methods, we applied Gremlin to the same GPCR sequence alignment previously investigated by SCA [81] and GMRC [83] studies for comparability . Using Gremlin we identified statistically significant long-range couplings in class A GPCRs and analyzed the results with respect to all seven GPCRs that had been crystallized at the time of our study. Our findings indicate that the ligand binding residues are significantly enriched in these long-range couplings, mediating not only communication to the IC, but also to the EC side of the membrane. 9 out of the 10 residues with the largest number of long-range couplings belong to the ligand binding domain. There a total of 34 statistically significant long-range couplings involving these 10 residues, involving experimentally determined microdomains and activation switches in GPCRs. Our study describes a comprehensive view of the network of statistical couplings across the membrane in class A GPCRs. The details of this network are consistent with the hypothesis that the ligand-binding pocket mediates allosteric communication. The independent identification of a crucial role of the ligand binding pocket in mediating this communication provides the first sequence-based support for the early notion that all three domains in GPCRs are structurally coupled [37] . Finally, the extent of enrichment of edges in different GPCR structures allowed us to propose a novel minimal binding pocket predicted to represent the common core of ligand contact residues crucial for activation of all class A GPCRs. See [60] for details regarding methods and results.

5.2 Methods

5.2.1 Gremlin

We employed Gremlin [4] to learn a Markov Random Field (MRF) model Fig 5.1 from a MSA of class A GPCRs (see details below). MRFs are undirected probabilistic graphical models. We use the MRFs to model the conservation and coupling statistics observed in the MSA. In particular, each node in the MRF corresponds to a column in the MSA. An edge between two nodes indicates that they are coupled. Conversely, the absence of an edge between two nodes means that they are *conditionally* independent given other nodes. The conservation and coupling statistics in a MRF are encoded via node (ϕ) and edge potentials (ψ). Informally, these potentials can be thought of as un-normalized probabilities. Collectively, these potentials encode the joint probability distribution over protein sequences such that the probability of any given length p sequence $x = (x_1, x_2, \ldots, x_p)$ can be computed as:

$$P_M(x) = \frac{1}{Z} \prod_{s \in V} \phi_s(X_s) \prod_{(s,t) \in E} \phi_{st}(X_s, X_t)$$

Here, Z is the normalization constant, V and E are the nodes and edges in the MRF, respectively. We note MRFs are generative and can thus be used to sample new sequences (as in protein design). Fig 5.1 shows a toy example of the relationship between the input MSA and the MRF that Gremlin learns. Here, a 7-column MSA is shown. Column 2 is completely conserved, and is therefore statistically independent of the remaining columns. This independence is encoded in the MRF by the absence of an edge to the variable corresponding to the second column. On the other hand, columns 1 and 4 co-vary such that whenever there is an 'S' in column 1, there is a 'H'

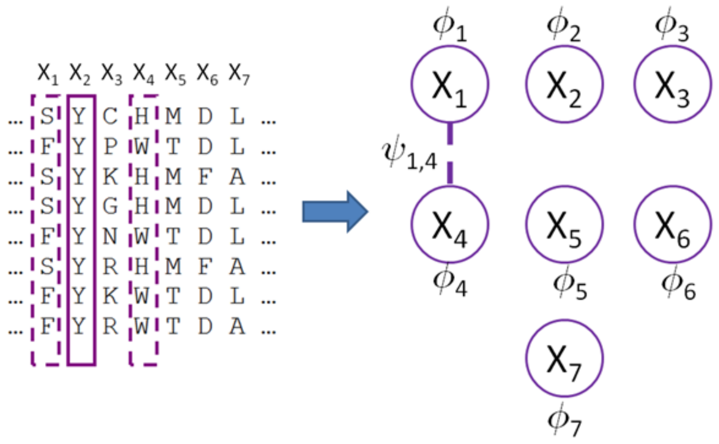


Figure 5.1: Shown in the figure is a cartoon figure of a multiple sequence alignment (MSA) and a corresponding Markov random field (MRF). There is one node in the MRF for each column in the MSA. The column-wise conservation statistics in the MSA are encoded by node potentials (ϕ_i). Similarly, the co-variation statistics in the MSA (e.g., between columns 1 and 4) are encoded by edge potentials ($\psi_{1,4}$) in the MRF. The lack of an edge between two nodes means that the corresponding columns are conditionally independent.

in column 4, and whenever there is an ‘F’ in column 1, there is a ‘W’ in column 4. This coupling is represented in the MRF by an edge between the variables corresponding to columns 1 and 4. In this paper, we examine the topology of the learned MRF to gain insights into the network of correlated mutations. Specifically, we are most interested in correlations that are observed between spatially distant residues from different domains of GPCRs.

5.2.2 Dataset Description and Preparation

See Appendix C.1 for details about multiple sequence alignment creation, model selection, structure files modeling, definition of binding pockets and definition of the control set.

Categories	**Control Set (Null)**		**Gremlin ($\lambda = 38$)**		**Gremlin <Null**	**Gremlin >Null**
	Total Edges	**% Edges**	**Total Edges**	**% Edges**	**p-value**	**p-value**
EC-EC	4095	6.78	169	23.80	0	1
EC-TM	14833	24.57	153	21.55	0.97	0.03
EC-IC	8554	14.17	56	7.89	1	0
TM-TM	13203	21.87	145	20.42	0.84	0.16
IC-TM	15322	25.38	81	11.41	1	0
IC-IC	4371	7.24	106	14.93	0	1
TOTAL	60378	100.00	710	100.00		
EC-RT	2125	3.52	114	16.06	0	1
RT-TM	3600	5.96	98	13.80	0	1
IC-RT	2350	3.89	67	9.44	0	1
RT-RT	300	0.50	51	7.18	0	1
SUB-TOTAL	8375	13.87	330	46.48		

Table 5.1: **Comparison of edge distribution from control set and Gremlin**

5.3 Results and Discussion

Gremlin was used to identify a network of correlated mutations in class A GPCRs. Our analysis has three main components. First, we used bovine rhodopsin (BR) as a template to map the edges (correlations) to the structure. Second, we identified the ligand binding pockets of all GPCRs with known structure to consider generality of our findings. Finally, we identified minimal binding pockets that capture the most general aspects of ligand binding across all GPCRs we examined. Additionally, See Appendix C.2.1 for a comparison of gremlin results with SCA and GMRC. See [60] for details regarding methods and results.

5.3.1 Bovine Rhodopsin Analysis

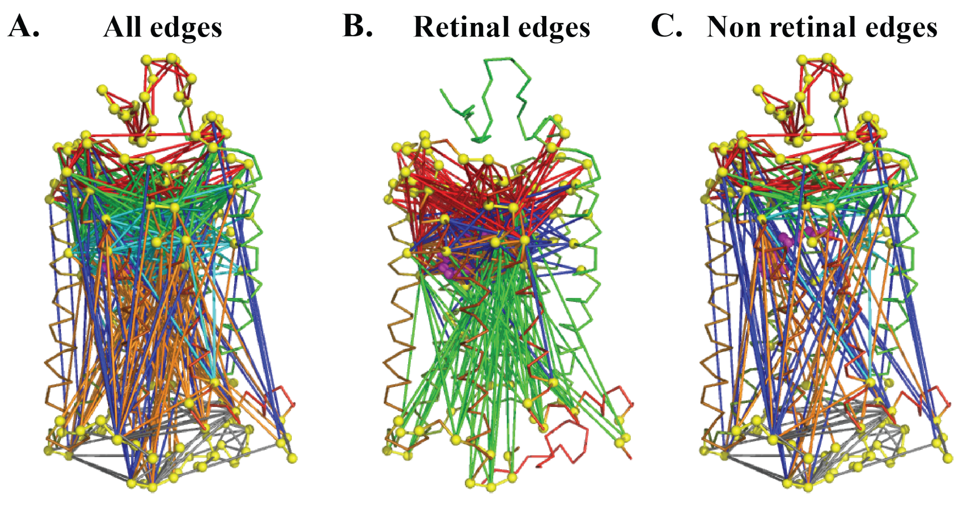
Our analysis revealed that most Gremlin edges involve residues in the RT ligand pocket; as compared to those between or within the residues belonging to EC, IC and TM domains outside of the RT pocket Fig 5.2. The RT pocket is located in the TM domain, at the interface with the EC domain. To quantify the observation that there were differences in the number of edges connecting EC, IC, TM domains and RT pocket, we enumerated the Gremlin edges and compared them to a control set, which included all possible edges (a total of 60,378 edges) involving all the 348 amino acids in rhodopsin. The results are summarized in Table 5.1. Assuming a significance level of $\alpha = 0.05$, we find that there is a significant enrichment of edges involving RT residues compared to the control set (46.48% for Gremlin vs. 13.87% for control; p-value of $\sim$ 0). Similar enrichment was observed in the relative distributions of EC-EC (23.8% for Gremlin vs. 6.78% for control; p-value of $\sim$ 0) and IC-IC (14.93% vs. 7.24%, p-value $\sim$ 0) edges. There was significant under-representation of edges in EC-IC (7.89% versus 14.17%, p-value $\sim$ 0), EC-TM (21.55% versus 24.57%, p-value $\sim$ 0.026) and IC-TM (11.41% versus 25.38%, p-value $\sim$ 0). There was no significant difference in TM-TM contacts (20.42% versus 21.87%, p-value $\sim$ 0.16).

The finding that there is significant enrichment in the EC-EC and IC-IC contacts and that there is an under-representation of EC-IC domain contacts is biologically meaningful, because EC-IC interactions would structurally be mediated via the TM domain. Interestingly, there is a lack of significant enrichment of edges within the TM domain and a slight under-representation of EC-TM and TM-IC edges. A lack of TM enrichment is in line with the general view of the TM helices as rigid bodies in the GPCR field . Furthermore, an important evolutionary pressure experienced by the amino acids in the TM region is to ensure that hydrophobic residues in the helices face the lipid bilayer. This pressure may override the importance of specific TM-TM contacts. However, it was puzzling that EC-TM and TM-IC contacts are under-represented since we would expect to find long-range couplings between EC and IC domains to be mediated via the intermediate TM domain. We therefore hypothesized that the EC-IC long-range contacts are more specifically mediated through a subset of TM and EC residues, namely those participating in binding RT. Indeed, 20 residues out of 27 in the RT pocket are in TM regions.

5.3.2 A minimal ligand binding pocket

We hypothesized that if there is a minimal binding pocket common to the seven known GPCRs, then Gremlin would significantly enrich the percentage edges for this pocket of residues compared to the null distribution set. To test this hypothesis we first defined ligand binding pockets B1, B2, B3, B4, B5, B6 and B7 representing residues common to at least one, two, three, four, five, six and seven receptor ligand binding pockets, respectively. We compared the percentage of edges formed by the residues in these pockets to that of the null distribution set and against each other. The percentage of edges for the null set decreased linearly from 32% to 2% with decreasing pocket size Fig 5.3. The percentage edges over the same range for Gremlin decreased 69% to 10% as expected because of the decreasing pocket size. However, the fold enrichment of edges for Gremlin over the null set increased from 2.2 to 5.2 for pockets B1 to B6. These results are statistically significant at a significance level of 0.05 with p-value ~ 0. The fold enrichment for B7 slightly decreased to 4.3 because the pocket is small with only 4 residues.

The four residues in B7 are T118$^{3.33}$, M207$^{5.42}$, Y268$^{6.51}$ and A292$^{7.39}$. These residues are uniquely positioned around the ligand (RT in rhodopsin; Fig 5.3) and make key interactions that stabilize RT .



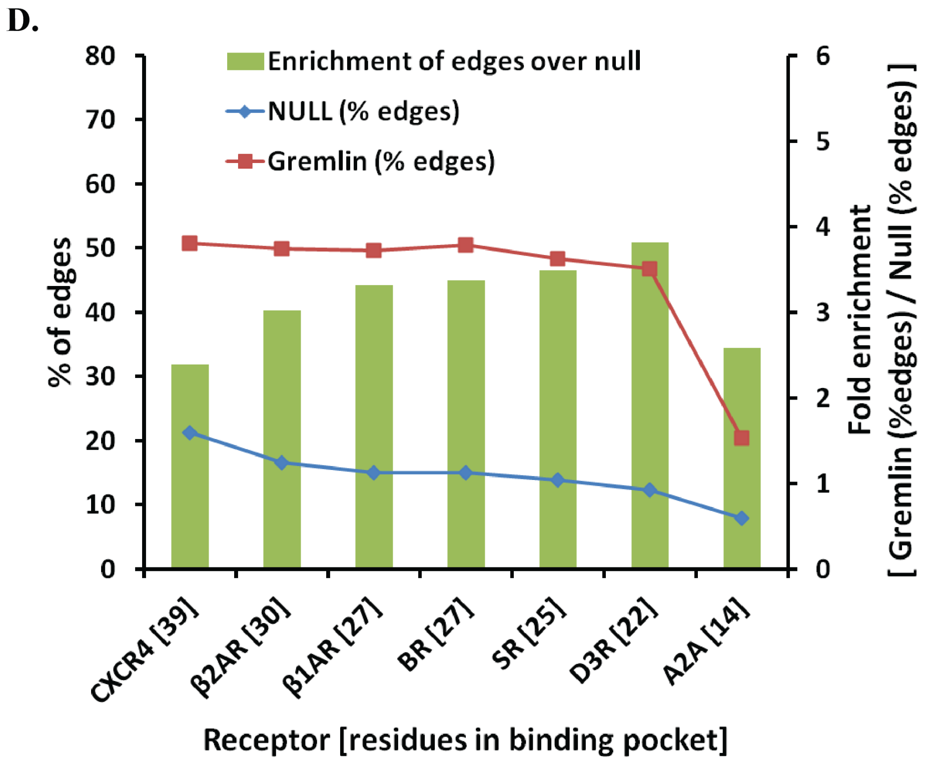
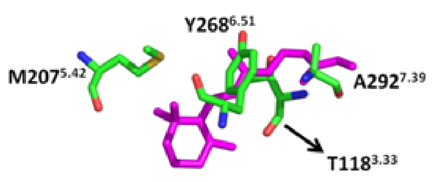
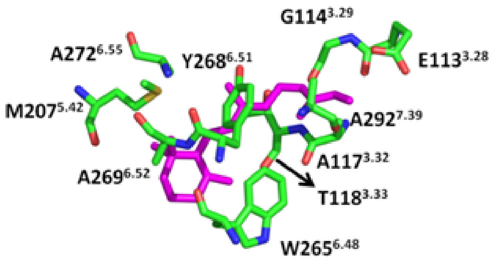


Figure 5.2: Mapping of (A) all (B) RT and (C) non RT edges identified by Gremlin (at =38) mapped onto the bovine rhodopsin structure (PDB ID: 1U19). The edges are EC-EC (red), EC-TM (green), EC-IC (blue), TM-TM (cyan), IC-TM (orange), IC-IC (grey40), EC-RT (red), RT-TM (blue), IC-RT (green) and RT-RT (orange) where EC, IC, TM and RT represent residues in extracellular, intracellular, transmembrane and RT (ligand binding) domains. In (D) The percentage of edges for Gremlin (squares) and null set (diamonds) are plotted against the common ligand binding pockets sorted by their size. The bars indicate fold enrichment (values on secondary y-axis) of edges in Gremlin over the null set.

A.



B.



C.

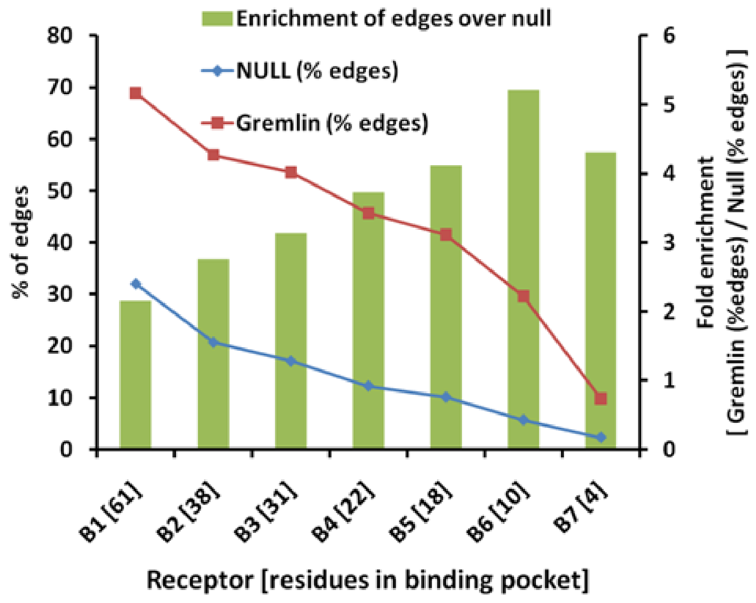


Figure 5.3: The spatial organization of residues in the minimal binding pocket (A) B7 and the larger pocket (B) B6 as present in the rhodopsin structure (PDB id 1U19). Rhodopsin numbering along with Ballesteros-Weinstein numbering (superscript) is given for comparison with other GPCRs. For clarity only the binding pocket residues are shown along with bound RT (in magenta). In (C), the percentage of edges for Gremlin (squares) and null set (diamonds) are plotted against the minimal ligand binding pockets sorted by their size. The bars indicate fold enrichment (values on secondary y-axis) of edges in Gremlin over the null set.

5.3.3 Residues involved in long-range interactions

The previous section showed that Gremlin is able to shed light on the biological and structural properties of the GPCR family. In this section we present a strategy for ranking Gremlin edges. This strategy can be used for exploratory purposes in order to discover novel couplings and residues that might play a key role in structure and function of the GPCR protein family.

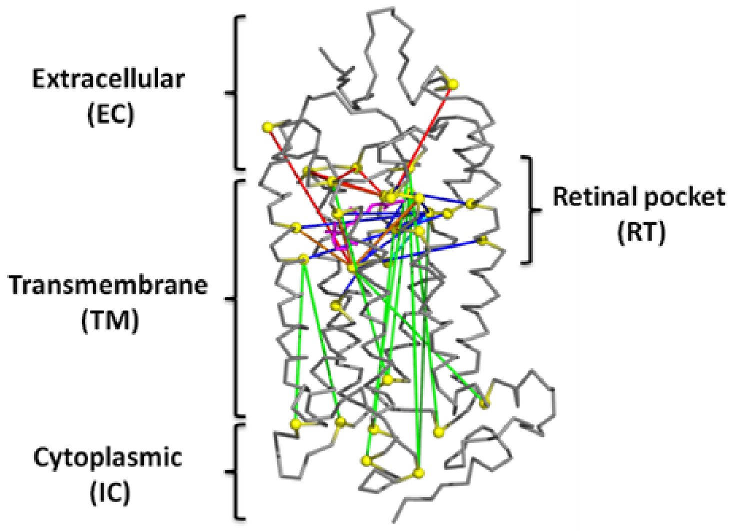


Figure 5.4: Persistent edges at penalty 140 for the top 10 residues are mapped onto the rhodopsin structure (PDB id 1U19). The residues forming the edges are represented as yellow spheres. The edges are TM-TM (cyan), IC-TM (dark green), EC-RT (red), RT-TM (blue), IC-RT (green) and RT-RT (orange), where EC, IC, TM and RT represent residues in extracellular, intracellular, transmembrane and RT (ligand binding) domains, respectively

The strategy is based on the following two key insights. The first insight is that the residues that have high degree in the graph of Gremlin couplings could be considered as hubs that lie on the communication pathways in GPCRs. This is motivated by the graphical model since a mutation/perturbation in the hub residue could affect a number of other residues. The second insight is based on the persistence of certain couplings even under stringent model complexity constraints. The larger the regularization parameter, λ, the sparser the Markov Random Field

(MRF), see Methods. Thus, each edge in the MRF can be assigned a persistence score equal to the maximum λuntil which the coupling was retained. The persistence score is an indicator of the importance of the couplings and the corresponding residues. See Fig 5.4.

We ranked the residues based on the number of edges at a penalty of $\lambda = 38$. The number of edges shown in the set of top 10 residues most frequently involved in an edge is shown in Table 5.2. Nine of these top ten residues (A117$^{3.32}$, A272$^{6.55}$, E113$^{3.28}$, H211$^{5.46}$, S186^{EC2}, A292$^{7.39}$, E122$^{3.37}$, G90$^{2.57}$, G114$^{3.29}$ and M207$^{5.42}$) are part of the RT pocket and are involved in packing and stabilizing of RT . Of these nine residues, eight are from the TM domain while S186^{EC2} is from the EC region. S186^{EC2} is involved in EC2 loop movement and its mutation to alanine alters the kinetics of activation . The remaining residue G90$^{2.57}$ that is not part of the RT pocket as defined by a $5\AA$ distance cut-off is nonetheless an important residue. The naturally occurring mutation G90$^{2.57}$D in the RT degeneration disease, *Retinitis pigmentosa*, results in the constitutive activity of the receptor .

5.4 Chapter Summary

In this chapter, we study the problem of feature interpretation in an unsupervised setting. We explore signal transduction in G protein coupled receptors (GPCRs) ; which relay signals across cell membranes. We identify networks of co-evolving residues from multiple sequence alignments by learning the topology of a Markov Random Field trained on GPCR sequences. We find that pairwise interactions containing residues in the ligand binding pocket are enriched. An analysis of these interactions reveals a minimal GPCR binding pocket containing four residues (T118$^{3.33}$, M207$^{5.42}$, Y268$^{6.51}$ and A292$^{7.39}$). Additionally, the ten residues predicted to have the most long-range interactions, are also part of the ligand binding pocket. This suggests that the activation in rhodopsin (a canonical GPCR) involves these long-range interactions between extracellular and intracellular domain residues mediated by the retinal domain.

Rank	**Position**	**Number of Edges** ($\lambda = 38$)	**Edges at** $\lambda = 140$
1	A117$^{3.32}$	41	G90$^{2.57}$, E247^{IC3}, F293$^{7.40}$, K296$^{7.43}$
2	A272$^{6.55}$	30	L72^{IC1}, G114$^{3.29}$, S176^{EC2}, Y178^{EC2}
3	E113$^{3.28}$	29	M44$^{1.39}$, L72^{IC1}, W126$^{3.41}$, Q237^{IC3}, F293$^{7.40}$
4	H211$^{5.46}$	29	F91$^{2.58}$, C140^{IC2}, F148^{IC2}
5	A292$^{7.39}$	28	Y29$^{EC(N-term)}$
6	S186^{EC2}	27	K67^{IC1}, Q244^{IC3}, P291$^{7.38}$
7	E122$^{3.37}$	26	I48$^{1.43}$, G90$^{2.57}$, E196^{EC3}, M207$^{5.42}$, A269$^{6.52}$, F293$^{7.40}$, C316$^{IC(C-term)}$
8	G90$^{2.57}$	23	A117$^{3.32}$, G120$^{3.35}$, E122$^{3.37}$, M207$^{5.42}$, Q237^{IC3}, A269$^{6.52}$, F293$^{7.40}$
9	G114$^{3.29}$	22	S176^{EC2}, A272$^{6.55}$, Y178^{EC2}
10	M207$^{5.42}$	22	G90$^{2.57}$, E122$^{3.37}$, C316$^{IC(C-term)}$

Table 5.2: **List of top ranked residues and the most persistent edges :** Residues in bold are part of the RT binding pocket extracted from the rhodopsin structure (PDB ID: 1U19). The Ballesteros-Weinstein numbering (superscript) is given for comparison with other GPCRs. Only long-range edges are reported i.e. the edges formed with neighboring residues (8 amino acids on either side) are filtered out

Chapter 6

Boltzmann Machines of Protein Families

In chapter 4, we introduced a framework of visible and hidden Markov Random Fields for generative modelling of protein families. We noted that, latent variable models with cycles performed poorly due to bad inference using loopy belief propagation. As a consequence, these models were not able to take advantage of the additional representational power afforded through the latent variables. In this chapter, we will correct the problems of the latent variable models by considering special architectures of Boltzmann Machines.

Protein sequence families have traditionally been modelled with PSSM matrices [80] and Hidden Markov Models (HMMs) [43]. These models make strong assumptions about the statistical independencies inherent in the data. For e.g. PSSM matrices require independence, while HMMs only allow linear dependence. Recently Balakrishnan *et.al* [4] introduced the GREMLIN VMRF model. This extended the list of allowed statistical dependencies by inserting edges between arbitrary pairs of variables.

While these models perform satisfactorily in practice, their representational power is limited. Complicated evolutionary and biological relationships necessitate more expressive models. However, greater representative power comes at a cost ; usually in the form of computational intractability and unlearneable models. In chapter 4 we introduced Hidden Markov Random Fields (HMRFs) which contain latent variables in order to increase the representational power of the MRF models. However, they suffered from badly learned models due to poor inference properties of loopy belief propagation in graphs with many cycles [38]. Additionally, the HMRF models need a prior specification of the statistical dependencies among the hidden layer units which might not be available.

In this chapter, we introduce *Boltzmann Machines* for learning generative models of protein sequence families. Boltzmann Machines are undirected graphical models with dense interconnections. Latent variables can also be added to Boltzmann Machines for greater modelling power. We show that by making very few restrictions to the structure of the Boltzmann Machines we can get models that are (1) expressive (2) learneable and (3) agnostic with respect to features.

In particular, we consider Boltzmann Machines with p-partite structures. These are known as *Restricted Boltzmann Machines* and *Deep Boltzmann Machines*. In addition, we also propose *Sparse Boltzmann Machines* and an algorithm for learning their sparsity structure.

See figure 6.1 for a quick glance at the architecture of models that shall be covered in this chapter.

6.1 General Boltzmann Machines

Boltzmann Machines were first proposed by Hinton and Sejnowski [32]. They are similar to Hopfield networks [35] which were contemporarily proposed. They are also known as energy based models [53] since the log-likelihood of model can be represented by an energy function which simplifies into a log-linear form [53].

Essentially, Boltzmann Machines are undirected graphical models and belong to the class of Markov Random Fields with pairwise potential functions that describe the interactions between the different variables. Boltzmann Machines can be fully observed in which case they belong to the class of Visible Markov Random Fields (VMRFs) as discussed in Section 4.2. They could also contain hidden units in which case they belong to the class of Hidden Markov Random Fields (HMRFs) as discussed in Section 4.3. See figure 6.1a and figure 6.1b for an illustration of

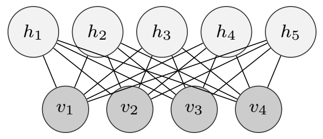
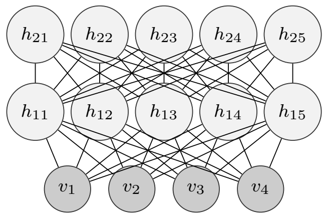
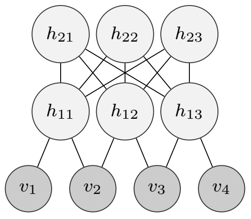
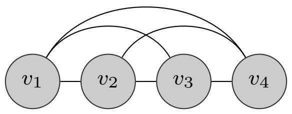
Boltzmann Machines	
Restricted Boltzmann Machine	
Deep Boltzmann Machine	 h_{21} h_{22} h_{23} h_{24} h_{25} h_{11} h_{12} h_{13} h_{14} h_{15} v_1 v_2 v_3 v_4
Locally Connected	 h_{21} h_{22} h_{23} h_{11} h_{12} h_{13} v_1 v_2 v_3 v_4

Table 6.1: Summary of Boltzmann Machine models

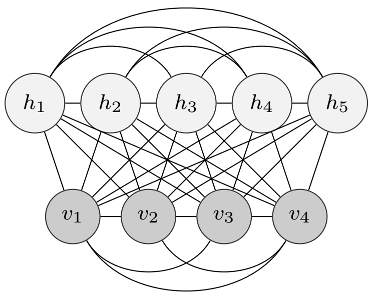
a visible and a hidden Boltzmann Machine respectively. Also, note that we will refer to *Visible General Boltzmann Machines* with the acronym VBMs.

The distinguishing factor of Boltzmann Machines when compared to general MRFs is that in general they tend to be much more densely connected. Hinton and Sejnowski, mention that their motivations for modelling with dense connections are rooted in Hebbian Learning and the connectionist interpretations of the brain [32]. The dense connections allow for the possibility of modelling much more complex relationships. Also, Boltzmann Machines tend to be agnostic with respect to the connections between the nodes, thus allowing for a larger hypothesis space for unsupervised learning. By allowing all visible variables to connect to all hidden nodes, the end-user remains agnostic about choosing any statistical edge over the other.

Despite the greater representational power these models tend to be largely computationally intractable [31]. The dense connections make exact inference impossible and approximate inference techniques inaccurate [6]. Typically, certain constraints are placed upon the topology of



(a) A completely observed Boltzmann Machine



(b) A Boltzmann Machine with hidden units

Figure 6.1: Boltzmann Machines with and without visible units

the Boltzmann Machines in order to make them more tractable. As we shall see in the following sections, special cases of Boltzmann Machines such as Restricted Boltzmann Machines (RBMs), Deep Boltzmann Machines (DBMs), Sparse Restricted Boltzmann Machines (SRBMs), Sparse Semi Restricted Boltzmann Machines (SSRBMs) and Locally Connected Deep Boltzmann Machines (LC-DBMs) can offer good tradeoffs between representational power and computational tractability.

6.1.1 Likelihood

As discussed above, Boltzmann Machines have a visible as well as a hidden formulation. In this thesis, we will mainly concern ourselves with the hidden formulation since hidden nodes let us go beyond pairwise potentials and model truly rich representations. Similar to section 4.3 on HMRFs, the probability of an input sequence $\mathbf{V}$ is represented by :

$$
\begin{aligned}
P(\mathbf{V};\Theta) &= \sum_{h\in H} P(\mathbf{V},h;\Theta) &(6.1)\\
&= \frac{1}{\mathcal{Z}(\Theta)} \sum_{h\in H} \prod_{i\in\mathcal{V}_V} \Phi_i(v_i) \prod_{j\in\mathcal{V}_H} \Phi_j(h_j) &(6.2)\\
&\prod_{(s,t)\in\mathcal{E}_V} \Psi_{st}(v_s,v_t) \prod_{(q,r)\in\mathcal{E}_H} \Psi_{qr}(h_q,h_r) &(6.3)\\
&\prod_{(l,m)\in\mathcal{E}_{V_H}} \Psi_{lm}(v_l,h_m) &(6.4)\\
&= \frac{\mathcal{Z}_{\mathbf{V}}(\Theta)}{\mathcal{Z}(\Theta)} &(6.5)
\end{aligned}
$$

where $\mathcal{Z}(\Theta)$ is the partition function of the full model, while $\mathcal{Z}_{\mathbf{V}}(\Theta)$ is the partition function of the model clamped with the input sequence $\mathbf{V}$.

6.1.2 Learning

Parameters of the model are learned by maximizing the log likelihood of the data.

$$
\begin{aligned}
\Theta^* &= \underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \sum_{m=1}^{M} \log P(V^m;\Theta) &(6.6)\\
&= \underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \sum_{m=1}^{M} \log \sum_{h\in H} P(V^m,H;\Theta) &(6.7)
\end{aligned}
$$

Gradient calculation is similar to HMRFs:

$$
\begin{aligned}
\nabla_{\Phi_i} \log P(\mathbf{V};\Theta) &= E_{\mathbf{V}}[v_i] - E_{\Theta}[v_i] &(6.8)\\
\nabla_{\Psi_{st}} \log P(\mathbf{V};\Theta) &= E_{\mathbf{V}}[v_s v_t^T] - E_{\Theta}[v_s v_t^T] &(6.9)\\
\nabla_{\Phi_j} \log P(\mathbf{V};\Theta) &= E_{\mathbf{V}}[h_j|\mathbf{V}] - E_{\Theta}[h_j] &(6.10)\\
\nabla_{\Psi_{qr}} \log P(\mathbf{V};\Theta) &= E_{\mathbf{V}}[h_q h_r^T|\mathbf{V}] - E_{\Theta}[h_q h_r^T] &(6.11)
\end{aligned}
$$

where $E_{\mathbf{V}}[v_i]$ and $E_{\mathbf{V}}[v_s v_t^T]$ are sufficient statistics of the visible variables; $E_{\mathbf{V}}[h_j|\mathbf{v}]$ and $E_{\mathbf{V}}[h_q h_r^T|\mathbf{v}]$ are the data dependent statistics; $E_{\Theta}[v_i]$, $E_{\Theta}[h_j]$, $E_{\Theta}[v_s v_t^T]$ and $E_{\Theta}[h_q h_r^T]$ are the model expectations. The calculation of the model expectations are intractable in general unless certain restrictions are imposed upon the structure of these models [70].

6.1.3 Evaluation

We will use generative metrics in order to evaluate the Boltzmann Machines. Generative metrics test the probability density estimate of held out data under the model. Our main evaluation metric is the imputation error / pseudo Log-likelihood [9]. Imputation error is relevant for modelling protein sequence families [4] as it is directly related to the task of protein design.

Another generative metric that will be used with Restricted Boltzmann Machines and Deep Boltzmann Machines is the reconstruction error, but only as a proxy for test log likelihood for model selection purposes. Reconstruction error is discussed in Section 6.2 on RBMs.

Pseudo Log Likelihood/Imputation Error

Imputation error is defined as the average error when predicting the amino acid type at each position in the sequence, when conditioning on the other amino acids in the sequence.

$$P(v_i|v_{-i}) = P(v_i|\left\{v_j\,;\quad j \in \mathcal{N}(i)\right\})$$

where $\mathcal{N}(i)$ is the set of random variables in the neighborhood of v_i.

6.2 Restricted Boltzmann Machines (RBMs)

Restricted Boltzmann Machines are a special type of Boltzmann Machines. The “restriction” in RBM comes from the fact that certain constraints are imposed upon the topology of the model. This allows tractability properties which are not otherwise enjoyed by other members of Boltzmann Machines. In particular, an RBM has a visible and a hidden layer arranged in a bipartite graph. See Figure 6.2 for an illustration. The shaded nodes v correspond to the visible layer. The lighter nodes h corresponds to the hidden layer. Unlike a VMRF/HMRF discussed in Section 4.2 4.3, the RBM does not contain any direct connections between the visible units. Instead all information is communicated via the hidden layer. The motivation behind this is that the potentials encoded by the hidden layers capture higher order interactions [24].

The bipartite nature of the graph allows the hidden nodes to factorize given the visible units and vice versa. This allows for efficient inference of the hidden node beliefs given the visible units. It also allows for efficient sequential parallel Gibbs sampling [12].

RBMs have been used extensively in the deep learning community as basic units in larger models such as Deep Neural Networks [33]. RBMs were first described by Smolensky et.al [76] as Harmoniums. They have been used in varied applications such as classification [51], collaborative filtering [73] and modelling documents [71]. They have even been used in computational biology applications such as predicting drug target interactions [89].

The hidden layer in RBMs can have an arbitrary number of nodes and each hidden variable has two states (logistic units) though it is not mandatory. Even though RBMs only have a single hidden layer, they can be very expressive and possess superior modelling power. Coates et.al [13] perform a detailed comparison of the hyperparameters involved in learning a RBM model. They find that the number of hidden nodes can play a critical role. When pushed to the limit i.e. very large number of hidden nodes ; the RBMs can have near state of the art performance.

6.2.1 Learning

Learning in the RBM is somewhat simplified owing to the bipartite nature of the connections between the visible and the hidden layer. Conditioned on the visible variables the hidden variables factorize. This leads to the equations 6.13 for conditional inference. Specifically calculation of $E_{\mathbf{V}}[h_j|\mathbf{v}]$ is exact and there is no need to calculate $E_{\mathbf{V}}[h_q h_r^T|\mathbf{v}]$ since there are no direct connec-

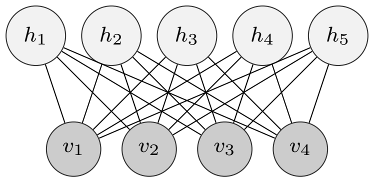


Figure 6.2: A Restricted Boltzmann Machine (RBM) is a Boltzmann Machine which consists of single hidden layer.The shaded nodes v correspond the visible layer. The lighter nodes h corresponds to the hidden layer.

tions within the hidden layer.

$$P(h_j = 1|\mathbf{V}) = \frac{\phi_j(h_j^1)\prod_{i\in\mathcal{V}_V}\psi_{ij}(v_i, h_j^1)}{1+\phi_j(h_j^1)\prod_{i\in\mathcal{V}_V}\psi_{ij}(v_i, h_j^1)} \tag{6.12}$$

$$P(v_i = a|\mathbf{H}) = \frac{\phi_i(v_i^a)\prod_{j\in\mathcal{V}_H}\psi_{ij}(v_i^a, h_j)}{\sum_{b=1}^{21}\phi_i(v_i^b)\prod_{j\in\mathcal{V}_H}\psi_{ij}(v_i^b, h_j)} \tag{6.13}$$

On the other hand, calculation of $E_\Theta[h_j]$ and $E_\Theta[v_i]$ is still intractable and requires approximate inference. Loopy belief propagation is not an effective choice in the case of RBMs and is known to perform very poorly in dense graph structures [38]. However, due to the bipartite graph structure, mixing of the markov chains can be fairly rapid when Gibbs sampling the layers consecutively. This is known as the contrastive divergence algorithm [30] and is a form of approximate gradient descent. Additionally, it has been shown that it is sufficient in practice to run just a few Gibbs steps to get an effective estimate of the model expectation [34] and is known as the CD-k algorithm.

6.2.2 Evaluation

Restricted Boltzmann Machines are evaluated using generative metrics for Boltzmann Machines as discussed in section 6.1.3. As in learning, the computation complexity of the evaluation metrics is simplified owing to the special structure of RBMs.

Cross entropy / Reconstruction error: is the easiest to calculate since it involves two Gibbs sampling passes ; once for each layer conditioned on the test input instance. In fact it is the default metric that is used to track the progress of learning since the other metrics can be computationally very expensive. Calculating the cross entropy is a two step process. First, the marginals of the hidden layer are inferred by conditioning on a sequence in the visible layer using equations 6.13. Second, the marginals of the visible layer are inferred by conditioning on the marginals of the hidden layer using equations 6.13. The cross entropy between the data distribution and the inferred distribution in the visible layer is defined as the reconstruction error.

Pseudo log-likelihood / Imputation error: is our primary evaluation metric. It is possible to calculate the pseudo log likelihood of an RBM exactly since the it is possible to calculate the conditional partition function efficiently.

$$P(v_i^a|v_{-i}) = \frac{\mathcal{Z}_a(\Theta)}{\sum_{k=1}^{21} \mathcal{Z}_k(\Theta)} \tag{6.14}$$

where $\mathcal{Z}_a(\Theta)$ is the partition function of the RBM conditioned on the input sequence $\mathbf{V} = \{v_i^a, v_{-i}\}$. This calculation is exact since the hidden layer factorizes conditioned on the visible layer.

6.3 Deep Boltzmann Machines (DBMs)

Deep Boltzmann Machines(DBMs) first introduced by Salakhutdinov et.al [71]. They generalize the RBM architecture to multiple layers. See Figure 6.3 for an illustration. The darker nodes V correspond to the visible layer while the lighter nodes H_1 and H_2 correspond to the hidden layers. Note the p-partite structure of the graph. RBMs can in theory represent arbitrary binary functions [24] and are universal approximators but they might need exponential number of units in order to do so [52]. DBMs have higher expressive power than RBMs due the connections between the hidden layer units. Bengio et.al [7] showed that deep networks need exponentially

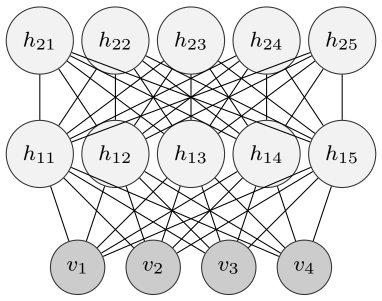


Figure 6.3: A Deep Boltzmann Machine (DBM) : Generalizes the architecture of a RBM to multiple hidden layers

less units than shallower networks when expressed as sum-product networks. DBMs have also shown to have better generative performance when compared to RBMs [71].

Unfortunately, this greater representative power comes at a cost. The tractability benefits present in the RBM are lost. With multiple hidden layers the factorization property present in the RBMs no longer holds and makes exact inference intractable. Despite the computational difficulties, state of the art models can be learned in practice, using approximate inference techniques such as Gibbs sampling and variational mean field inference [71, 72]. Deep Boltzmann Machines have been applied for modelling text [78], images [71] and even multiple modalities [77].

Note that a DBM is not the same as a Deep Belief Network (DBN). A DBN is a hybrid directed/undirected graphical model [33] and does not belong to the class of Boltzmann Machines. Conditioned on the visible nodes a DBN does not factorize. Also it cannot take both top down and bottom up influences during inference unlike a DBM.

6.3.1 Learning

Training Deep Boltzmann Machines is much more challenging than Restricted Boltzmann Machines due to the presence of a second hidden layer. As a consequence, the hidden layer units no longer factorize given the visible layer. The quantities $E_{\mathbf{V}}[h_j|\mathbf{v}]$, $E_{\mathbf{V}}[h_q h_r^T|\mathbf{v}]$, $E_{\Theta}[v_i]$, $E_{\Theta}[h_j]$ and $E_{\Theta}[h_q h_r^T]$ are not efficiently computable. Hence, directly applying stochastic gradient descent using Boltzmann Machine equations 6.11 is not possible.

Instead approximate inference techniques must be used. We used persistent contrastive divergence [88] to calculate the model expectations $E_\Theta[v_i]$, $E_\Theta[h_j]$ and $E_\Theta[h_q h_r^T]$; and variational mean field inference equations 6.16 to calculate training data expectations $E_\mathbf{V}[h_j|\mathbf{v}]$ and $E_\mathbf{V}[h_q h_r^T|\mathbf{v}]$. This is similar to the approach taken by Salakhutdinov et.al [71] for training their DBMs.

$$P(h_j = 1|\mathbf{V}, \mathbf{H_2}) = \frac{\phi_j(h_j^1) \prod_{i \in \mathcal{V}_V} \psi_{ij}(v_i, h_j^1) \prod_{k \in \mathcal{V}_{H_2}} \psi_{kj}(h_k, h_j^1)}{1 + \phi_j(h_j^1) \prod_{i \in \mathcal{V}_V} \psi_{ij}(v_i, h_j^1) \prod_{k \in \mathcal{V}_{H_2}} \psi_{kj}(h_k, h_j^1)} \tag{6.15}$$

$$P(h_k = 1|\mathbf{H_1}) = \frac{\phi_k(h_k^1) \prod_{j \in \mathcal{V}_{H_1}} \psi_{jk}(h_j, h_k^1)}{1 + \phi_k(h_k^1) \prod_{j \in \mathcal{V}_{H_1}} \psi_{jk}(h_j, h_k^1)} \tag{6.16}$$

Unfortunately, it has been empirically observed that even the approximate approach does not work very well in practice unless the initial set of parameters are already in a favorable location in parameter space [27, 71]. Greedy pretraining using RBMs is a common technique to warm start and initialize the parameters [8, 33]. Pretraining is a three step process. In the first step, an RBM model is trained on the input data. In the second step, the hidden layer representations of the RBM are extracted and are used to train a second RBM. In the third step the parameters are stitched together from the two RBMs to create a joint Deep Boltzmann Machine (DBM). This model is fine tuned using the variational inference update Equations 6.16, 6.17 and PCD-k [88] as discussed before.

$$P(v_i = a|\mathbf{H_1}) = \frac{\phi_i(v_i^a) \prod_{j \in \mathcal{V}_{H_1}} \psi_{ij}(v_i^a, h_j)}{\sum_{b=1}^{21} \phi_i(v_i^b) \prod_{j \in \mathcal{V}_{H_1}} \psi_{ij}(v_i^b, h_j)} \tag{6.17}$$

6.3.2 Evaluation

Similar to a RBM, evaluation on the DBM is done using two generative metrics. (1) Imputation Error and (2) Reconstruction Error. The evaluation metrics are largely similar to RBMs, however the additional computational difficulties necessitate certain modifications:

- **Imputation error:** calculation is no longer exact since the hidden layer is non-factorial. So we must resort to approximation inference. We have used variational mean field inference using Equation 6.17.
- **Reconstruction error:** calculation is no longer exact since exact inference is not possible on the hidden layer given the visible layer. We have used variational mean field inference using Equation 6.16 and equation 6.17.

6.4 Sparse Boltzmann Machines

In the previous section, we described the RBM and the DBM architectures. These models contain a very large number of parameters. We wondered if such a large number of parameters was actually necessary. Specifically, is it possible to (1) get similar performance with far lesser parameters (2) prevent overfitting and (3) inject prior knowledge into Boltzmann Machines by using fewer parameters. In this section, we attempt to address these question by introducing a new type of Boltzmann Machine.

We present a sparse variant of Boltzmann Machines. Sparsity in this context refers to sparsity in the parameters and topology of the Boltzmann Machine. Sparsity prevents overfitting by reducing the hypothesis space of the model. It also leads to data efficiency such that lesser data is needed to train the model by reducing the number of parameters. Sparse structure learning can also be useful for descriptive analysis in decoding the statistical relationships present in the data [4].

Sparse structure learning has been previously discussed in the context of MRFs [75] and Gaussian Graphical Models [25]. However, the problem is much harder for Boltzmann Machines such as RBMs and DBMs due to the presence of hidden variables. Sparse Boltzmann Machines have been conceptually proposed in the literature [61]. However, they have not been used in practice owing to the difficulty of sparse structure learning for hidden variable models. Previous approaches using sparse Boltzmann Machines assume that the connections are known apriori. The shape Boltzmann Machine [21] is an example of a sparse Boltzmann Machine which uses the spatial proximity of adjacent pixels to specify the connections allowed.

In this section, we present a method for learning the sparsity structure of Boltzmann Machines without using any prior information. This is important in the context in protein sequence families wherein long distance relationships can have important modelling consequences. We present three kinds of sparse Boltzmann Machines (1) *Sparse Restricted Boltzmann Machines* (SRBMs) (2) *Sparse Semi Restricted Boltzmann Machines* (SSRBMs) and (3) *Locally Connected Deep Boltzmann Machines* (LC-DBMs).

6.4.1 Models

In the following sections we describe the three sparse Boltzmann Machines:

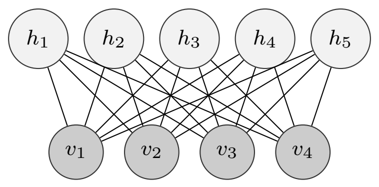
Sparse Restricted Boltzmann Machine (SRBMs)

These are RBMs with sparse edges. See figure 6.4 for an illustration. Figure 6.4a shows a non-sparse RBM which is a fully connected bipartite graph ; while figure 6.4b shows a SRBM with several edges and some nodes missing.

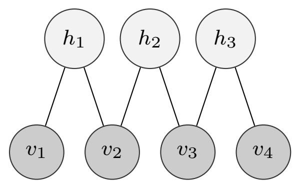
Learning and *Evaluation* for the SRBM is the same as a RBM (See Section 6.2.1) once the original topology and parameters have been learned (See Section 6.4.2).

Sparse Semi-Restricted Boltzmann Machines (SSRBM)

A *Sparse Semi-Restricted Boltzmann Machines* (SSRBM) is a SRBM with an added hidden layer attached to the visible layer. See Figure 6.5 for an illustration. An SRBM with its sparse edges is likely to have limited modelling power especially if it was created using a MRF to RBM map



(a) Restricted Boltzmann Machine (RBM)



(b) Sparse Restricted Boltzmann Machine (SRBM)

Figure 6.4: Comparison of Restricted Boltzmann Machines (RBMs) and Sparse Restricted Boltzmann Machines (SRBMs)

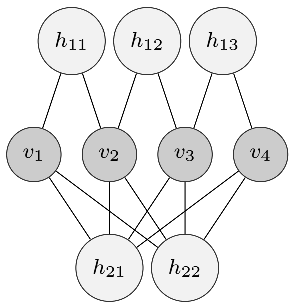


Figure 6.5: A Sparse Semi-Restricted Boltzmann Machine (SSRBM) is a Boltzmann Machine which consists of a sparse hidden layer (H_1) attached to the visible layer and a dense hidden layer (H_2) attached to the same visible layer. The dense layer (H_2) boosts the modelling capacity of the model without adding too many variables

(see section 6.4.2). The motivation behind the additional hidden layer is that it boosts modelling capacity without adding too many variables.

Learning and *Evaluation* for the SSRBM is the same as a RBM (See Section 6.2.1) once the original topology and parameters have been learned (See Section 6.4.2). The additional hidden layer does not make the RBM intractable since it still factorizes given the visible layer.

Locally Connected Deep Boltzmann Machines

A *Locally Connected Deep Boltzmann Machine* (LC-DBM) is similar to a DBM but has additional restrictions. The connections between the visible layer and the first hidden layer are required to be sparse. Compare figure 6.3, a Deep Boltzmann Machine with figure 6.7, a Locally Connected DBM. Notice that the connections to the first hidden layer are sparse. The motivation for sparsifying the lower layers is to inject prior knowledge into the lower layers while letting the higher layers deal with the modelling load of complex data distributions. This acts as a trade-off between allowing rich representations and statistical/computational efficiency. The shape Boltzmann Machine proposed by Eslami et.al [21] is a LC-DBM with the sparsity structure pre-defined. In section 6.4.2 we will show how to learn the sparsity structure automatically from the data.

Analogous to LC-DBMs are the concepts of convolution and pooling which take advantage of the topological structure of the input dimensions. Examples of topological structure are 2D layouts of pixels in images, 3D structures of videos and proteins, the 1D sequential structures of text or protein sequences. This is a commonly used approach in Convolutional Neural Networks [54] (CNN). The LC-DBM draws motivation from this idea in the context of DBMs. See figure 6.6 for an illustration of a CNN applied to a 2D image patch.

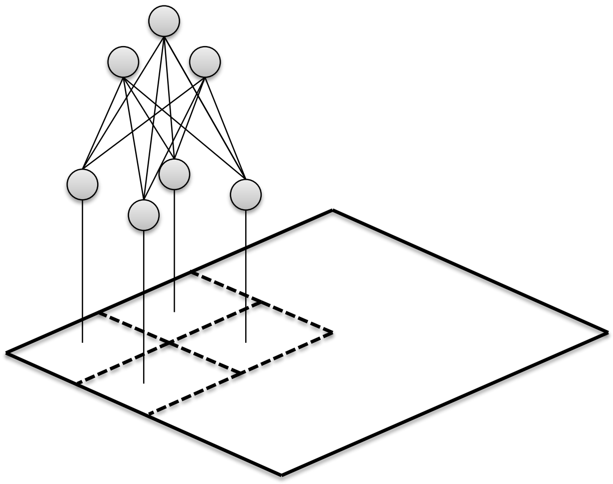


Figure 6.6: A Convolutional Neural Network used in computer vision has a local structure property at the lower layers and dense connections at higher layers. This approach allows incorporating spatial coherence into the statistical structure, an injection of prior knowledge.

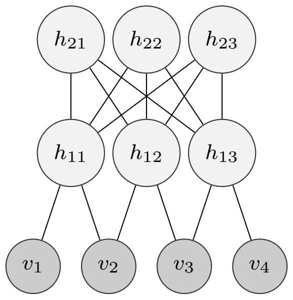


Figure 6.7: A Locally Connected Deep Boltzmann Machine (LC-DBM) : A DBM with sparse edges between the visible layer and the first hidden layer. Can be thought of as a SRBM with an additional RBM layer on top.

Learning and **Evaluation** for the LC-DBM is the same as a DBM (See Section 6.3.1) once the original topology and parameters have been learned (See Section 6.4.2). The additional hidden layer makes the SRBM intractable and only approximate inference techniques can be applied such as variational mean field inference.

6.4.2 Sparse Topology Learning using Cholesky Decomposition

Learning the sparsity structure is much harder for the case of latent variable models. In a sense the problem is ill-defined because the of non-identifiability of hidden nodes. For e.g. the sparsity structure of two hidden nodes h_i and h_j could be swapped without changing the objective function of the model. Previous sparsity inducing approaches do not enforce sparsity at the topology level [55] or enforce sparsity in the context of predictive networks such as auto-encoders [10, 57].

Our approach is based on three key insights (1) Sparse structure learning for MRFs (2) Mapping MRF parameters to a Gaussian RBM (3) Sparsity structure preservation in Cholesky decompositions. See Figure 6.10 for an illustration of the steps involved in creating the difference Sparse Boltzmann Machines. We describe these steps in greater detail below:

Sparse MRF structure learning: Balakrishnan et.al [4] introduce GREMLIN, a method for $l1$ regularized structure learning to learn a sparse visible MRF for a protein sequence alignment. Usually this method creates $O(N)$ edges, where N is the number of columns in the multiple sequence alignment. We use this method to learn a sparse MRF (figure 6.10b) from a fully connected MRF (figure 6.10a). Gremlin induces $l1$ group sparsity on the pseudo log-likelihood and learns a sparse set of edges.

$$\Theta^* = \left(\underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \sum_{m=1}^{M} \sum_{i=1}^{N} \log P(\mathbf{v_i^m} | \mathbf{V_{-i}^m}; \Theta) \right) - M\lambda_\Phi ||\Phi||_2^2 - M\lambda_\Psi ||\Psi||_1$$

MRF to Gaussian RBM Mapping: Martens *et.al* [58] introduced a method to convert a visible MRF to a Gaussian RBM using a Cholesky decomposition of the MRF parameter matrix. They use the fact that the energies of the Gaussian RBM and a visible MRF factorize to linear terms and can therefore be equated with each other.

Consider a RBM with gaussian hidden units such that the energy of the RBM is represented as

$$E(h, v) = \frac{1}{2} h^T h - h^T W v - f(v) \tag{6.18}$$

where h and v are the hidden and visible units respectively, W is the RBM parameter matrix and $f(v)$ is any arbitrary function of the visible variables alone.

We can rewrite the energy function $E(h, v)$ as:

$$E(h, v) = \frac{1}{2}(h - Wv)^T (h - Wv) - \frac{1}{2} v^T W^T W v - f(v) \tag{6.19}$$

We can calculate the marginal probability of the visible variables as

$$p(v;\Theta) = \exp\left(\frac{1}{2}v^T W^T W v + f(v)\right)\int \frac{1}{Z}\exp\left(-\frac{1}{2}(h - Wv)^T(h - Wv)\right) dh \tag{6.20}$$

$$= \frac{(2\pi)^{N/2}}{Z(\Theta)}\exp\left(\frac{1}{2}v^T W^T W v + f(v)\right) \tag{6.21}$$

$$\propto \exp\left(\frac{1}{2}v^T W^T W v + f(v)\right) \tag{6.22}$$

Now, suppose the parameters of a VMRF were represented via the matrix M. We can then calculate the probability of the VMRF as

$$p(v;\Theta) = \frac{1}{Z(\Theta)}\exp\left(\frac{1}{2}v^T M v\right) \tag{6.23}$$

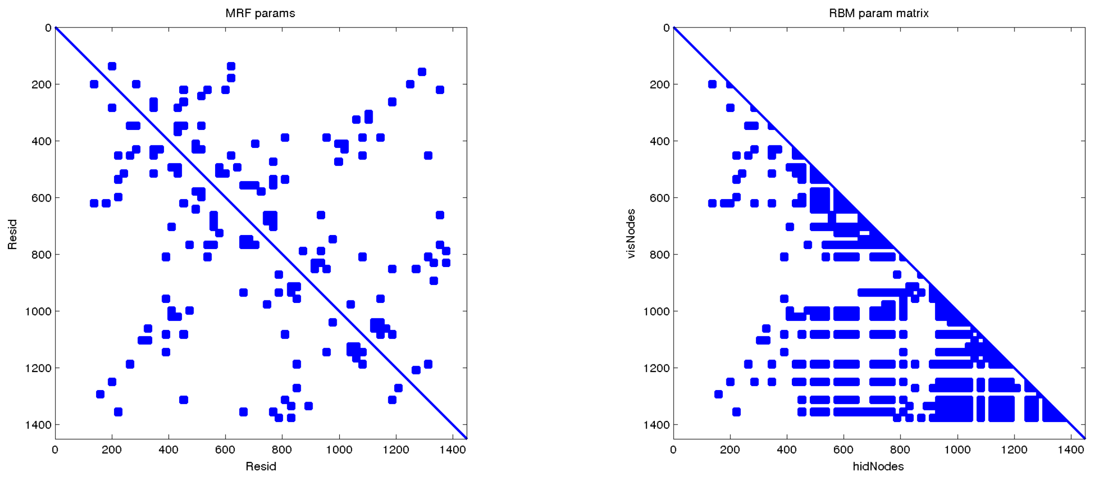
The Gaussian RBM can be made to represent the same distribution as the MRF by equating the energy terms of equation 6.20 and equation 6.23.

$$-\frac{1}{2}v^T M v = -\frac{1}{2}v^T W^T W v - f(v) \tag{6.24}$$

and this will be satisfied iff $M = W^T W + f(v)$. Importantly, W can be calculated as the Cholesky decomposition of M ; $W \leftarrow \text{chol}(M)$

Note that if M is positive definite, then $f(v)$ can be set to zero. If on the other hand, M is not positive definite we can consider a special form of $f(v) = 1/2v^T\text{diag}(d)v$. We can choose any d that makes $A - \text{diag}(d)$ positive definite. One such d is $d = \alpha\vec{1}$ where $\alpha = (1+\epsilon)\min(\lambda_{\min}, 0)$ and ϵ is a small positive value and $\lambda_{\min}$ is the smallest eigen value of M.

The Gaussian units can also be thought as linear units since their activations are a linear combination of the input units with some Gaussian noise. We empirically observed that performing



(a) The original MRF parameter matrix M for ubiquitin family (PF00240).

(b) Cholesky decomposition of the original MRF parameter matrix M

Figure 6.8: Converting MRF to RBM using Cholesky Decomposition

learning on the Gaussian hidden units tends to be unstable The instability can be countered by applying $l2$ regularization penalties, however the strong regularization prevents useful models from being learnt. As an alternative, we substitute the Gaussian units with regular bernoulli hidden units without much deterioration in performance. This helps in model stability and provides a warm parameter bootstrap despite losing the the 1-to-1 map between the MRF and the RBM.

Sparse Cholesky decomposition: The above MRF to RBM map uses a Cholesky decomposition. An interesting property of the Cholesky decomposition is that if the input matrix M is sparse then the factored matrix W tends to be sparse as well [16]. For most protein sequence families, Gremlin [41] produces a sparse set of edges that resemble the protein contact map (See `http://gremlin.bakerlab.org/`). So the input MRFs are likely to sparse and the sparsity is transferred to the RBM via the Cholesky decomposition. See Figure 6.8 for an example with the ubiquitin family (PF00240).

The Cholesky decomposition in the above step creates a SRBM (See Section 6.4.1). Hidden layers can be added to the SRBM to provide added representational power with minimal parameters. Adding a hidden layer to the visible layer creates a SSRBM (See Section 6.4.1) whereas adding a second hidden layer to the first hidden layer creates a LC-DBM (See Section 6.4.1)

Min fill permutations

The sparse Cholesky decomposition can be further sparsified by performing a min fill permutation on the input matrix [16]. We first show that any permutation matrix P, that permutes the rows of the positive definite MRF matrix M' can be used to get the same result as equation 6.24.

$$\begin{aligned}
\frac{1}{2}v^T M v - \frac{1}{2}v^T \text{diag}(d) v &= \frac{1}{2}v^T (M - \text{diag}(d)) v && (6.25)\\
&= \frac{1}{2}v^T M' v && (6.26)\\
&= \frac{1}{2}v^T P^T P M' P^T P v && (6.27)\\
&= \frac{1}{2}v^T P^T \left(X^T X\right) P v && (6.28)\\
&= \frac{1}{2}v^T (XP)^T (XP) v && (6.29)\\
&= \frac{1}{2}v^T Y^T Y v && (6.30)\\
& && (6.31)
\end{aligned}$$

where $Y = PX$ and $X \leftarrow \text{chol}(P(M - \text{diag}(d))P^T)$ and P is a permutation matrix.

Therefore we have the result that Y is the new RBM matrix.

$$-\frac{1}{2}v^T M v = -\frac{1}{2}v^T Y^T Y v - \frac{1}{2}v^T \text{diag}(d) v \qquad (6.32)$$

P can be chosen such that it causes minimum fill in the corresponding Cholesky factor X . The problem of finding the best ordering is in general, NP-complete. Instead heursitics must be used and we use the approximate minimum degree (AMD) heuristic due to Amestoy *et.al* [3]. This AMD algorithm has a tight upper bound of $O(nm)$ [28] where n and m are the number of vertices and edges in a graph respectively. See figure 6.9 for an example with the ubiquitin family (PF00240).

6.5 Regularization

Regularization describes a set of techniques used to prevent overfitting in machine learning models. This is especially important in models like Boltzmann Machines which are heavily overparameterized. A consequence of overfitting is low training error but high validation and test error. In this section, we discuss some commonly used regularization strategies that we tried.

$l2$ regularization

This technique modifies the objective function of the Boltzmann Machines by penalizing the parameter weights. The objective and the gradients are modified according to Equations 6.33 and 6.37. This discourages very strong activations in the hidden and the visible units. It also helps in keeping the learning algorithm stable and prevents numerical instability issues [34].

$$\Theta^* = \underset{\Theta=\{\Phi,\Psi\}}{\arg\max} \left(\sum_{m=1}^{M} \log P(V^m;\Theta) \right) - M\lambda_\Phi ||\Phi||_2^2 - M\lambda_\Psi ||\Psi||_2^2 \tag{6.33}$$

With the modification in the objective, the gradients become.

$$\nabla_{\Phi_i} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[v_i] - E_\Theta[v_i] - \lambda_\Phi \Phi_i \tag{6.34}$$

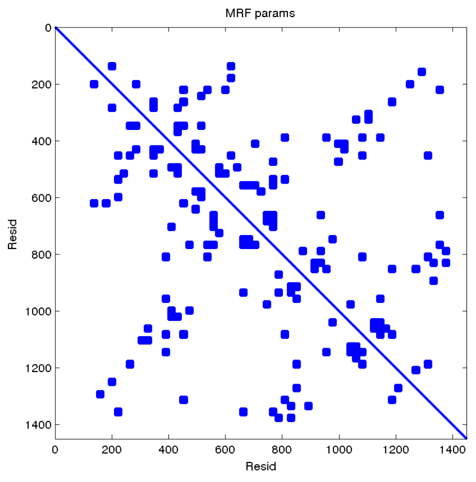
$$\nabla_{\Psi_{st}} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[v_s v_t^T] - E_\Theta[v_s v_t^T] - \lambda_\Psi \Psi_{st} \tag{6.35}$$

$$\nabla_{\Phi_j} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[h_j|\mathbf{V}] - E_\Theta[h_j] - \lambda_\Phi \Phi_j \tag{6.36}$$

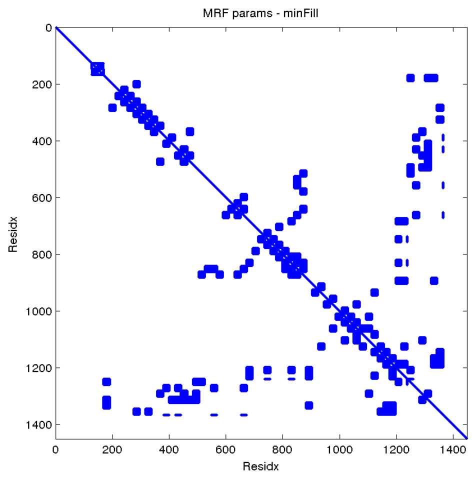
$$\nabla_{\Psi_{qr}} \log P(\mathbf{V};\Theta) = E_{\mathbf{V}}[h_q h_r^T|\mathbf{V}] - E_\Theta[h_q h_r^T] - \lambda_\Psi \Psi_{qr} \tag{6.37}$$

Dropout

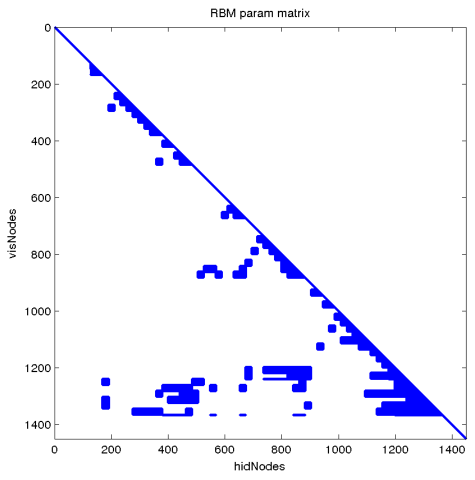
Dropout is a regularization technique introduced by Srivastava et.al [79] and has been previously used to regularize the parameters of a deep network. It involves randomly dropping some of



(a) The original MRF parameter matrix M for ubiquitin family (PF00240).

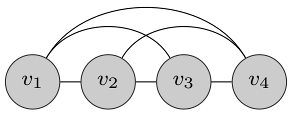


(b) A minfill ordering of the MRF param matrix M for ubiquitin family (PF00240)

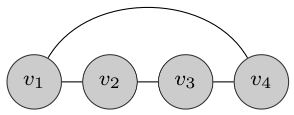


(c) Cholesky decomposition of minfill MRF 6.9b is sparser than 6.8b

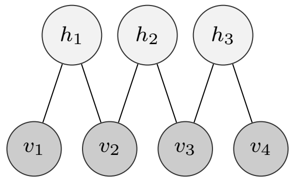
Figure 6.9: Converting MRF to RBM using min-fill Ordering



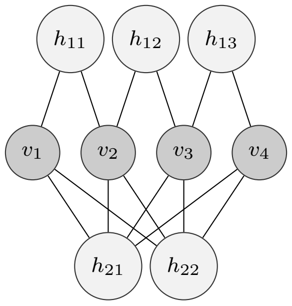
(a) Fully Connected MRF



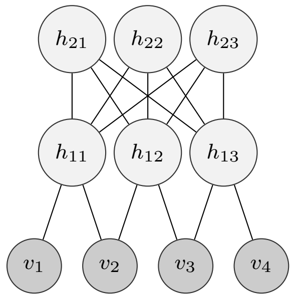
(b) Sparse MRF



(c) Sparse Restricted Boltzmann Machine (SRBM)



(d) Sparse Semi-Restricted Boltzmann Machine (SS-RBM)



(e) Locally Connected Deep Boltzmann Machine (LC-DBM)

Figure 6.10: **Sparse Boltzmann Machine Topology Learning:** The different steps involved in learning a Sparse Boltzmann Machine. **6.10a)** A fully connected MRF is used to model the protein sequence family. **6.10b)** A sparse MRF structure is learned by $l1$ regularized structure learning. **6.10c)** A Cholesky decomposition is performed to convert the sparse MRF to a sparse RBM using equation 6.24. **6.10d)** A hidden layer is added to the visible to convert the SRBM to a SSRBM. **6.10e)** A second hidden layer is added to the first hidden layer in the SRBM to convert it to a LC-DBM

hidden and visible nodes during an iteration of training. The central idea behind this is similar to the concept of bagging and boosting. It is a mechanism to sample several smaller submodels, train them and subsequently stitch them together at test time. While this may not optimize the log likelihood of the data it still forces the model to learn useful representations. We apply dropout to our models with great success.

6.6 Experiments and Results

The experiments will be divided into three main parts. In the first part, our goal was to ascertain if standard Boltzmann Machines such as RBMs and DBMs can learn superior generative models of protein sequence families. In the second part, our goal was to investigate the efficacy of sparse Boltzmann Machines. Finally, in the third part, we perform a large scale study with additional datasets using a diverse set of evaluation metrics.

6.6.1 Data Description

We obtained multiple sequences alignments for two protein families, ubiquitin (PF00240) and PDZ (PF00595), from the Baker-lab GREMLIN webserver `http://gremlin.bakerlab.org/`. These MSAs were create using HMMs from Pfam hmm(s) as seeds.The alignments were further enriched with additional sequences using HHblits [69] and the uniprot database [15]. The alignments where filtered to have 75% coverage i.e. sites that had more than 75% gaps were removed. Also these alignments were processed to have at most 90% sequence identity.

The ubiquitin MSA contains 5538 sequences and 69 positions. The PDZ MSA contains 12887 sequences and 81 positions. The sequences in the alignment were randomized. The alignments were split into training, validation, and test sets using a 60%, 20% and 20% proportion split, respectively. The validation sets were used to select the optimal hyper-parameters.

This is the same dataset as in chapter 4 to allow comparison with MRF models.

6.6.2 Algorithm and implementation details

In this section, we describe the experimental setup for Boltzmann Machine experiments. As with the MRFs 4.4, our task will be to test the generative performance of Boltzmann Machines and compare them against the MRF models. We undertake an extensive hyperparameter scan to figure out best practices for Boltzmann Machines when applied to protein families.

The “deepnet” package provided by Srivastava (`https://github.com/nitishsrivastava/deepnet`) was used for training the Boltzmann Machines. To speed up our calculations, we used Amazon EC2 cloud-computing and GPU-acceleration using the CUDAmat [59] library. Depending on the number of visible and hidden nodes and size of the training data, each model can take anywhere from 30 minutes to 12 hours to train.

RBMs and DBMs

This section corresponds to the first part of the experiments i.e. to ascertain if standard Boltzmann Machines such as RBMs and DBMs can learn superior generative models of protein sequence families

We describe the experimental setup for training and evaluating RBMs and DBMs. We also describe the hyperparameter scans associated with these models. These models use hidden layers that have binary or logistic units. The input visible layer has softmax units consisting of 21 states. DBM architectures were implemented using the template $I - H_1 - H_2$ corresponding to the input layer, the first hidden layer and the second hidden layer respectively. I contains 69 and 81 softmax units corresponding to ubiquitin and PDZ respectively. H_1 and $H2$ take 100, 500 and 1000 logistic units each. This gives rise to a total of 9 architectures for each protein family.

We employed the Persistent Contrastive Divergence (PCD) [88] algorithm for training our Boltzmann models. Optimization was accelerated using momentum acceleration. We used five Gibbs sampling steps to calculate model expectations, thereby leading to the PCD-5 algorithm. Additionally, we used five variational mean field inference steps for calculating the data dependent expectations.

Data was split into mini-batches of 200 sequences each. The weights were updated after each mini-batch was evaluated. We scanned over learning rates ϵ of 0.5, 0.1 and 0.01. The learning rate was decayed according to two schedules (1) linear $\epsilon/(2000+t)$ (2) exponential - $\epsilon \exp^{-t}$. We trained our RBMs for a total of 100,000 epochs and our DBM models for an additional 200,000 epochs after the RBM pretraining. Models were dumped every 1000 epochs.

We employed pre-training to initialize the parameters of the DBMs. Pre-training involves learning a single-layer RBM and then using the representations from the first RBM to train a second RBM. The parameters of these two RBMs are used to initialize a joint model which is then trained using variational mean field inference, as previously mentioned.

To prevent overfitting, we regularized our models. Our primary means of regularization was applying a $l2$ penalty to the parameters. We did not regularize the node biases, but applied it to edge parameters. Additionally, we investigated the use of "dropout" technique [79] for some models. We used a dropout probability of 0.5.

Visible General Boltzmann Machine (VBMs)

We train a visible general Boltzmann Machine (*vbm*) for comparison with RBMs and DBMs. This is also the same as a fully connected VMRF. We use pseudo log likelihood for training (see section 4.2.3). We used the LBFGS [94] algorithm for optimizing the parameters. The optimization routine was run for a maximum of 1000 iterations or till a convergence tolerance was reached. We used Wolfe line search [92] for adaptively picking the step length at each gradient step. We applied $l2$ regularization penalty of 1 on the nodes and edges similar to other VMRF models.

Training these models was much more computationally expensive than other visible MRFs.

Hyper parameters	**Options**	**Description**
ϵ	$(.5, .1, .01)$	Learning Rate
ϵ-decay	yes/no	Use ϵ decay with inverse time
$l2$-decay	$(0.01, 0.001)$	Apply $L2$ decay on parameter weights
dropout	yes/no	Use dropout
momentum	yes/no	Use momentum acceleration
hwidth	$(69/81, 100, 500, 1000)$	Width of the hidden layer
Gibbs-steps	5	Number of Gibbs iterations
mf-steps	5	Number of mean field iterations
epochs	(100K-RBM, 200K-DBM)	Number of epochs to run training
dropout probs	0.5	Drop node with probability

Table 6.2: Hyperparameters Boltzmann Machines

Hyperparameters and Model Selection: We follow similar practices adopted in the deep learning community for hyper-parameter selection [34]. We divide the hyperparameters into three categories (1) *Exhaustive* - scan over all variations (2) *Randomize* - randomly sample these (3) *Fixed* - best guesses based on prior experience (see Table 6.2). We exhaustively scan over the use of dropout and the number of hidden nodes. We randomly sample the learning rate, the decay schedule, the $l2$ penalty and the gradient momentum from a small set of candidate values. Additionally we fix other hyperparameters such as number of Gibbs sampling steps, variational inference steps, and dropout probabilities based on several quick experiments with these models.

Altogether, we ran a total of 300 different experiments for different values of hyperparameters.

We used cross-entropy on the validation set as the model selection metric. It is to be noted that the log likelihood and the imputation error are superior metrics. However, given the computational expense associated with computing them, we resort to validation cross entropy as an acceptable surrogate. In general, we find that low validation cross entropy is correlated with having low imputation error.

Sparse Boltzmann Machines

In this section we proceed to the second part of the experiments. We describe the setup for evaluating and comparing sparse Boltzmann Machines. In light of the computationally expensive nature of hyperparameter scanning, we fix the hyperparameters based on our prior experience. We then proceed to learn and evaluate different sparse Boltzmann Machines using these fixed hyperparameters. Additionally, we also train RBMs and DBMs using these same hyperparameters in order to allow for comparison with their sparse counterparts.

We obtain the sparse parameter matrix by performing a Cholesky decomposition on associated MRF parameters. The MRF parameters were obtained from the *gremvis* models for ubiquitin and PDZ which in turn obtained their sparse edges by running the GREMLIN algorithm. We consider two versions of these sparse edges (1) long edges (2) long plus short edges. The long edges correspond to GREMLIN connections between amino acids greater than 3 residue positions away. The short edges correspond to all pairs of edges within 3 residue positions away. To ensure that the MRF parameter matrix is positive definite we add the minimum eigenvalue of along the diagonal using a weight of 1.01.

The Sparse Restricted Boltzmann Machine (SRBM) (see section 6.4.1) model is an RBM with a sparse topology. We use logistic units for the hidden units instead of Gaussian unit as we find that it leads to better stability in learning. As a consequence of the Cholesky factorization, the hidden layer has the same number of nodes as the corresponding MRF. The ubiquitin SRBM has 1449 hidden nodes while the PDZ SRBM has 1701 hidden nodes.

The Sparse Semi Restricted Boltzmann Machine (SSRBM) (see section 6.4.1) model turbocharges the SRBM model described previously. It does so by adding additional logistic nodes to the first hidden layer. The model is still a RBM albeit with greater sparsity in its parameters.

We add 10, 50, 100 and 500 hidden nodes respectively to create four different SSRBM models for each protein family.

The Locally Connected Deep Boltzmann Machine (LC-DBM)(see section 6.4.1) also attempts to enhance the SRBM model. However it does so by adding an additional hidden layer above the first hidden layer. This makes the model more akin to a DBM with added computational difficulties. The hope is that the multilayer model allows for richer representation to be captured. We introduce 50, 100, 250, 500 and 1000 additional hidden nodes to create different LC-DBM models.

To compare the sparse Boltzmann Machines against RBMs and DBMs, we additionally trained vanilla RBMs and DBMs. The RBMs contained 50, 100, 500, 1000 and 1449/1701 hidden logistic units. The DBMs contained 1449 and 1701 units in the first hidden layer and 10, 50 and 100 in the second hidden layer.

In all the above models, we employed Contrastive Divergence (CD) [30] algorithm for training. Gradients were accelerated using momentum acceleration. We used five Gibbs sampling steps to calculate model expectations (CD-5). Additionally, we used five variational mean field inference steps for calculating data dependent expectations for the LC-DBM.

Data was split into mini-batches of 500 sequences each. The weights were updated after each mini-batch was evaluated. The learning rate ϵ was fixed at 0.1. The learning rate was decayed according to linear decay $\epsilon/(5000+t)$. We trained our RBMs, SRBMs and SSRBMs for a total of 100,000 epochs and our DBM and LC-DBM models for an additional 100,000 epochs. Models were dumped every 1000 epochs.

We employed pre-training to initialize the parameters of the DBMs. To prevent overfitting, we regularized our models using an $l2$ regularization penalty of 0.01. We did not regularize the node biases, but applied it to edge parameters. We did not use dropout on any of the models.

Evaluation was carried out by computing the imputation error. The calculations for RBMs, SRBMs and SSRBMs is exact whereas approximate inference is needed for DBMS and LC-

DBMs. Computation was sped up using GPUs.

6.6.3 Results and Discussion

In this section, we discuss the results from the two part experiments.

RBMs, DBMs and VBMs

This section contains the results from the experiments of the first part. We will present the results for the ubiquitin protein family (PF00240). The results for PDZ (PF00595) were largely similar and are discussed in appendix D.2.

We report the imputation errors calculated on the training and the test sets in Table 6.3. We also list the MRF results from section 4.4 for comparison. The results show that both Boltzmann Machines outperform the next-best models by around 13% points. This strongly suggests that these architectures are capable of extracting more information from the MSAs than the other models — including other latent variable models. In contrast, the difference in accuracies amongst the other structured models (excluding the straw-man model, `ind`) is at most 11%.

The General Boltzmann machine represented by *vbm*, has significantly better test error than *gremvis* (0.398 vs 0.495). However, it's training error is abysmally low (0.028) indicating that it has almost memorized the training data.

The other Boltzmann Machines also exhibit a large increase in test error, when compared to training error. This suggests that the models have over-fit their training data. However, our results demonstrate that the Boltzmann Machines still have far better generative performance than the other models.

Sparse Boltzmann Machines

In this section we discuss the results from the next set of experiments pertaining to the Sparse Boltzmann Machines. We discuss evaluation metrics for sparse Boltzmann machines boot-

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
ind	0.7604	0.7572
linhid	0.5870	0.5922
linvis	0.5635	0.5753
12vis-pseudo	0.5046	0.5374
gremvis-pseudo	0.4205	0.4952
3dvis-pseudo	0.4164	0.4856
vbm	0.0283	0.3979
rbm-hyp-scan	0.0501	0.3553
dbm-hyp-scan	0.1587	0.3510

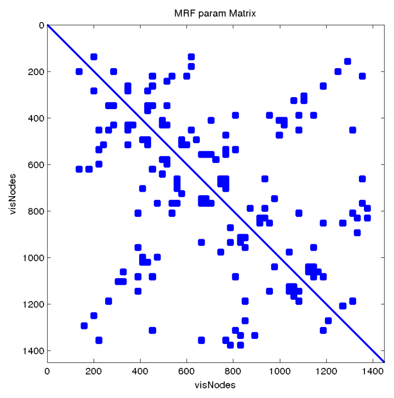
Table 6.3: Imputation error ubiquitin PF00240

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
rbm-50	0.4764	0.4947
rbm-100	0.3757	0.4335
rbm-500	0.2765	0.3946
rbm-1000	0.2770	0.3999
rbm-1449	0.2790	0.3984
dbm-1449-50	0.2344	0.3857
dbm-1449-100	0.2309	0.3819

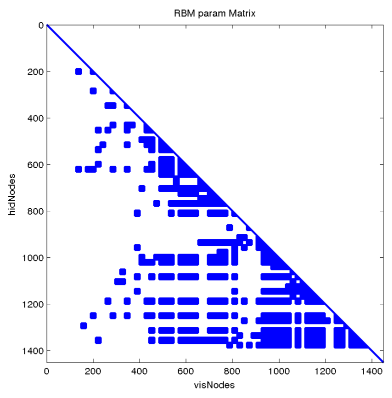
Table 6.4: Imputation error ubiquitin RBM and DBM

strapped from sparse MRFs. Figure 6.11 shows the edges for the ubuiquitin family and subsequent mapping to a RBM. Notice the sparsity preservation in the map from MRF (figure 6.11a) to RBM (figure 6.11b).

Table 6.4 shows the imputation errors calculated for baseline models of RBMs and DBMs. Note that even an RBM with 50 units (*rbm-50*) has a similar imputation error to the *gremvis-pseudo* MRF models (0.49) . As the number of hidden units is increased from 50 to 1449, the



(a) ubiquitin MRF param matrix with long range edges



(b) SRBM bootstrapped from ubiquitin MRF param matrix with long range edges

Figure 6.11: Cholesky decomposition of ubiquitin MRF param matrix with long range edges

imputation error drops sharply initially but levels off around 500 units. Further addition of units does not lead to more gains in modelling power. Note the difference in the training and testing error. The large gap suggests that the model is over-fitting to the training sequences but the generalization performance is still superior. Stricter $l2$ penalties could potentially help alleviate this while retaining the same test error. Additionally, the DBM model *dbm-1449-100* had the best test error.

Bootstrap from Long Range Interactions

This section discusses the results from long range interactions MRF bootstrap. Figure 6.11 shows the edges for the ubuiquitin family and subsequent mapping to a RBM. Note the absence of any edges within three residue positions.

Table 6.5 shows imputation errors for SRBM and SSRBM models. The number of additional hidden nodes in the SSRBM layer are varied. Our hypothesis is that SSRBMs turbocharge the SRBM models and improve the test error. Even the addition of 10 hidden units dramatically decreases the test error.

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
srbm-1449	0.5484	0.5671
10-ssrbm-1449	0.4917	0.5203
50-ssrbm-1449	0.4251	0.4656
100-ssrbm-1449	0.3830	0.4355
500-ssrbm-1449	0.3212	0.4052

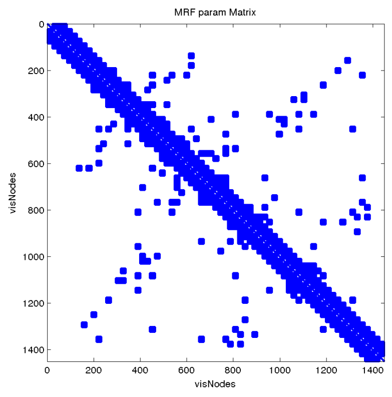
Table 6.5: Imputation error ubiquitin SRBM and SSRBM (long edges)

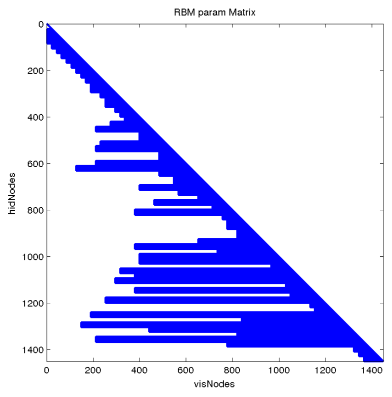
Model	**train-imperr**± 0.002	**test-imperr**± 0.004
srbm-1449	0.5484	0.5671
lcdbm-1449-50	0.5319	0.5657
lcdbm-1449-100	0.5095	0.5494
lcdbm-1449-250	0.5004	0.5399
lcdbm-1449-500	0.5032	0.5435
lcdbm-1449-1000	0.5052	0.5456

Table 6.6: Imputation error ubiquitin SRBM and LC-DBM (long edges)

This shows that the additional nodes help in modelling aspects which the SRBM and correspondingly, the MRF alone cannot model. However, note that the SSRBM models are about the same or slightly worse than the corresponding RBM models (see table 6.4). For e.g. *rbm-500* has a test error of 0.394 whereas *500-ssrbm-1449* has a test error of 0.405. This might suggest that RBMs by themselves are easily capable of learning the representations represented by MRFs and more. The sparse layer bootstrapped from the MRF biases the model towards distributions that the MRF represents and this interferes with other high level representations that a vanilla RBM is trying to learn.

Table 6.6 shows the imputation errors for the LC-DBM model when varying the number of hidden nodes in the second hidden layer. We expect to see the imputation error improve when adding additional hidden units in a second layer. However the test error improves for models





(a) ubiquitin MRF param matrix with short and long range edges

(b) SRBM bootstrapped from ubiquitin MRF param matrix with short and long range edges

Figure 6.12: Cholesky decomposition of ubiquitin MRF param matrix with short and long range edges

with additional nodes but these are only modest gains (only a drop of about 0.02 w.r.t SRBMs). *lcdbm-1449-50* has a similar test error of 0.565 while *lcdbm-1449-250* has a test error of 0.5399. Additional hidden units do not improver performance.

Bootstrap from Short Plus Long Range Interactions

This section discusses the results for sparse Boltzmann machines bootstrapped from short plus long range edges. Figure 6.12 shows the edges for the ubuiquitin family and subsequent mapping to a RBM. The RBM has many more parameters than the long range bootstrapped RBM (figure 6.11).

Table 6.7 shows imputation errors for SRBM and SSRBM models. The number of additional hidden nodes in the SSRBM layer are varied. Similar to the case with long range edges alone, the error drops with additional hidden nodes. However, the drop is less precipitous and it levels off around 500 additional hidden units. Also, note that short plus long range edge bootstrap SRBM has lower error than the long range SRBM alone (0.484 vs 0.567). This goes to show that the additional parameters help in reducing the error. The same holds for SSRBMs as well, wherein

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
srbm-1449	0.4392	0.4843
10-ssrbm-1449	0.4208	0.4703
50-ssrbm-1449	0.3864	0.4453
100-ssrbm-1449	0.3621	0.4270
500-ssrbm-1449	0.3471	0.4202

Table 6.7: Imputation error ubiquitin SRBM and SSRBM (short plus long edges)

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
srbm-1449	0.4392	0.4843
lcdbm-1449-50	0.4398	0.5081
lcdbm-1449-100	0.4223	0.4929
lcdbm-1449-250	0.4228	0.4925
lcdbm-1449-500	0.4226	0.4908
lcdbm-1449-1000	0.4207	0.4896

Table 6.8: Imputation error ubiquitin SRBM and LC-DBM (short plus long edges)

the addition of the short range edges reduces the error further. However, note that the short plus long *500-ssrbm-1449* model is worse than the long *500-ssrbm-1449* model alone ; indicating that the MRF bias is stronger for the short plus long model.

Table 6.8 shows the imputation errors for the LC-DBM model when varying the number of hidden nodes in the second hidden layer. These models do no better than the corresponding SRBMs and can in even hurt the performance by a tiny amount. We suspect that the addition of additional hidden nodes unsettles the already tuned SRBM without aiding recovery to vanilla RBM performance.

SSRBM and LC-DBM results summary

The LC-DBM results give rise to the following conclusions:

- SSRBMs turbocharge the SRBM models with much needed additional features and reduce the error significantly with even modest addition of hidden units.
- SSRBMs are biased depending on the MRF they were bootstrapped from, and cannot better the performance of vanilla RBMs with the same number of dense hidden units.
- LC-DBMs appear to have very little effect in reducing the imputation error. A second hidden layer can reduce the test error of a SRBM by (0.567 to 0.539) in the case of long-range edge bootstrap. Though these gains are fairly modest. For the case of short plus long range edge bootstrap the LC-DBM seems not to help at all.
- Adding additional hidden units in the second layer also does not lead to further improvement. This suggests that the features are bottlenecked at the first hidden layer.
- In conjunction with SSRBMs, we note that even a small addition of dense connections greatly reduces the test error. This suggests that protein families might have k-ary dependencies which a MRF and a SRBM fail to model.

6.6.4 Minfill trick

In this section we compare the performance of the minfill trick with the direct Cholesky decomposition. We also examine the effect of thresholding the decomposed Cholesky matrix, to check if dropping lower weight parameters can be an effective surrogate to the minfill trick.

We learn an SRBM model, *srbm-1449* by converting the *gremvis-pseudo* with short plus long range edges using equation 6.24. We also learn a SRBM using the minfill trick, *srbm-minfill-1449* from the same MRF using the minfill trick, equation 6.25. We report the imputation errors in table 6.9. Note that the test errors for SRBM models are largely similar to the original *gremvis-pseudo* model from which these sparse RBMs were bootstrapped. There is a jitter of $\pm$ 1 percentage points, which can be attributed to a Gaussian to Bernoulli approximation.

As an alternative to the minfill trick, we checked if dropping the lower weight parame-

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
gremvis-pseudo	0.4205	0.4952
srbm-1449	0.4392	0.4843
srbm-minfill-1449	0.4398	0.5081

Table 6.9: Imputation error ubiquitin: SRBM models with and without the minfill trick have similar imputation errors to *gremvis-pseudo* with a slight jitter due to the Gaussian to Bernoulli approximation

Model	**train-imperr**± 0.002	**test-imperr**± 0.004
srbm-1449-thresh-0.00	0.4392	0.4843
srbm-1449-thresh-0.25	0.4840	0.5298
srbm-1449-thresh-0.50	0.4785	0.5312
srbm-1449-thresh-0.75	0.4936	0.5427
srbm-1449-thresh-1.00	0.4942	0.5380

Table 6.10: Imputation error ubiquitin SRBM using direct Cholesky, minfill trick and thresholded Cholesky. We note that the thresholded Cholesky loses about 4 to 5 percentage points when compared to *srbm-1449*. It is also worse than the *srbm-minfill-1449*.

ters in the direct Cholesky decomposition could lead to similar performance. We interpolated between models containing the same number of paramters as the *srbm-minfill* and the *srbm-minfill-1449* models. The thresholded models are referred to as *srbm-1449-thresh-p* where $p \in \{0.25, 0.5, 0.75, 1.0\}$ corresponds to the percentage of parameters dropped. We report the results in table 6.10. The thresholded models lose about 4 percentage points just by dropping 25% of the parameters. Also, the *srbm-1449-thresh-1.00* model which has the same number of parameters as the *srbm-minfill-1449*, is about 4 percentage points worse. This goes to show that simply thresholding lower weight parameters is not an effective strategy to achieve greater sparsity.

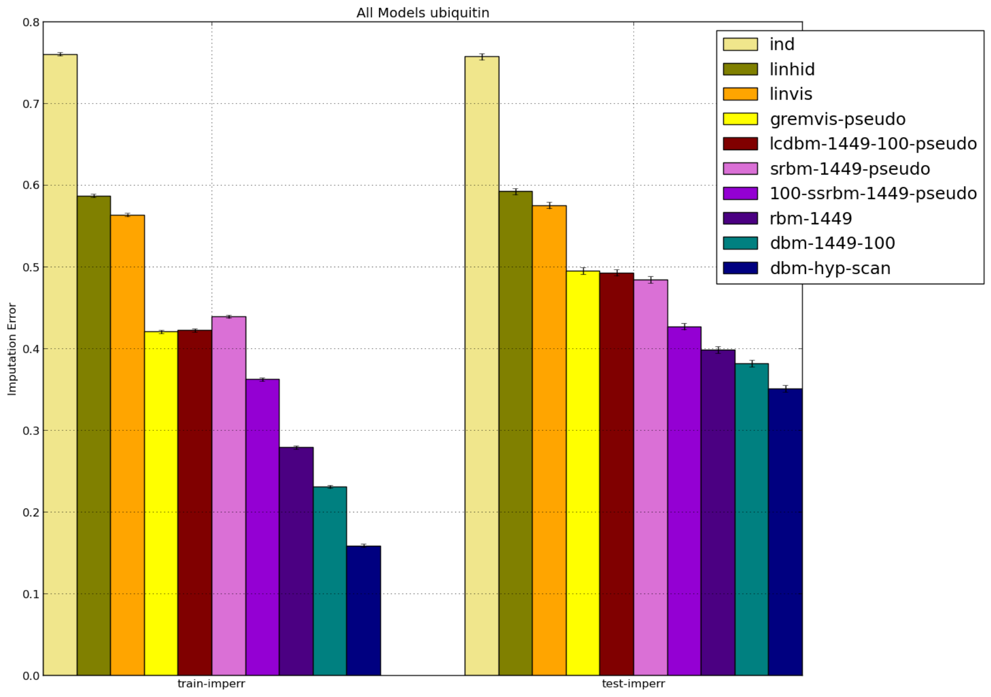


Figure 6.13: Comparing MRFs and Boltzmann Machines for ubiquitin: Imputation error improves with relaxations in model assumptions and addition of latent variables

6.6.5 Discussion of all Generative Models

Figure 6.13 shows a summary of the MRF models discussed in chapter 4 and the different Boltzmann machines discussed in this chapter.

We note that all models have better training and testing errors when compared to the strawman independent model *ind*. This shows that all models are able to discover contextual information present in the protein family. Tree structured graphical models, *linvis* and *linhid* have the next best performance. Exact inference is possible in these models using the sum-product inference algorithm. Additionally these models have similar training and test errors showing very little over-fitting.

The GREMLIN algorithm represented by *gremvis-pseudo* extends the above models to general graph features, significantly boosting the modelling capacity. Long distance features are now enabled. Due to the presence of cycles, inference is no longer exact and pseudo log likelihood is used as an approximation.

SRBMs are direct maps from general MRFs to RBMs. Not surprisingly, the SRBM model represented by *srbm-1449-pseudo* model has very similar performance to *gremvis-pseudo*. LC-DBMs and SSRBMs are models that attempt to rescue SRBMs with additional hidden nodes. The LC-DBM model adds a second hidden layer to the hidden layer of the SRBM. Unfortunately, the LC-DBM (*lcdbm-1449-100-pseudo*) model fails to make an improvement and has similar performance to the *srbm-1449-pseudo* and *gremvis-pseudo*. This might suggest the existence of an information bottleneck in the representations of the first hidden layer in a SRBM.

The SSRBM represented by *100-ssrbm-1449-pseudo* also attempts to rescue SRBMs but unlike an LC-DBM it adds a hidden layer directly to the visible layer. These models are successful in turbo-charging the SRBMs with even modest number of hidden units. Along with evidence from the LC-DBMs, this suggests that dense connections are imperative for gain in imputation error.

RBMs and DBMs have the best overall performance. They allow arbitrary combinations of features and have efficient inference algorithms. However, the gap between training and testing error continues to widen indicating some over-fitting. The VBM (Visible General Boltzmann Machine) model represented by *vbm* trained using pseudo log likelihood has comparable performance to RBMs and DBMs. However, it has an abysmally low training error, indicating a near perfect memorization of the training data. This is evidence of severe over-fitting. Also, it goes to show that an over-abundance of parameters can bring down the test error.

The absolute best imputation error was reported by a DBM model represented by *dbm-hyp-scan* obtained by hyper-parameter scanning. It had a 1000-500 architecture and uses "dropout" among others. In general, the best hyper-parameters are not predictable across datasets. This necessitates expensive hyperparameter scanning which maybe computationally infeasible given a limited budget.

To summarize, by relaxing model assumptions and adding latent variables, we can learn increasingly complex generative models that continue to improve the imputation error with increments in representational power.

Similarly, a summary of all generative models for PDZ family is provided in appendix D.2.1.

6.6.6 Large Scale experiments

In the third part, we study 10 additional protein families. These protein families were selected to have variety in the structural type of the family. See table 6.11 for a list of the protein families. Two extreme cases were also studied. PF12569 contained a 517 residue protein family. PF11427 had the largest number of sequences (54866 sequences) and is additionally discussed in section 6.6.7. The datasets were split according to 60%, 20% and 20%; train, valid and test split, respectively.

We learned RBM and DBM models for all these protein families. A *rbm-1000* and a *dbm-1000-500* was chosen for all the families. We used the Contrastive Divergence (CD) [30] algorithm for training all the models. The other hyperparameters were the fixed as in section 6.6.2.

We report the test imputation error of all datasets in figure 6.14. The Boltzmann Machines beat the sparse MRF model in all datasets except PF11427 with about a 6% gain averaged across

Pfam id	**Sequences**	**Length**	**Clan**	**Scop description**
PF12569	7321	**517**	CL0020	Alpha-alpha-superhelix
PF11427	**54866**	50	CL0123	Helix-turn-helix
PF03459	10264	64	CL0021	All-beta + barrel
PF13693	10626	78	CL0123	Helix-turn-helix
PF13710	10688	63	CL0070	Ferrodoxin like
PF13899	17321	82	CL0172	Thioredoxin like
PF13920	15588	50	CL0229	Zinc finger
PF13459	19491	65	CL0344	Dehydrogenase like
PF13927	21858	75	CL0011	Immunoglobulin
PF13833	15426	54	CL0220	EF hand

Table 6.11: 10 additional families - Variety in depth, width and structural type

datasets. The gain is dataset dependent with about a standard deviation of about 2% points. The maximum gain was observed for the ubiquitin protein family (PF12569) with about 12% points.

We compared the gain in imputation error for Boltzmann Machines over the Gremlin (MRF) model as a function of sequence/length. Figure 6.15 shows a scatter plot of these two variables. A higher gain is observed in the lower sequence/length region and this manifests itself it as an inverse correlation between these two variables. PF11427 was observed in figure 6.14 to have no improvement in performance when compared to Gremlin. It also happens to be the rightmost point in the scatter plot with a very high sequence/length ratio of 1097 sequences per residue. On the other hand PF12569 had the maximum gain and it has the lowest sequence/length ratio of around 14 sequences per residue.

We also investigated if the imputation errors were important from a biophysical/evolutionary perspective. Blosum matrices [19] encode such information in a substitution matrix. Higher scores correspond to successful matches. We chose the Blosum90 matrix since our MSAs have a maximum of 90% sequence similarity. We report the test Blosum90 of all datasets in figure 6.16.

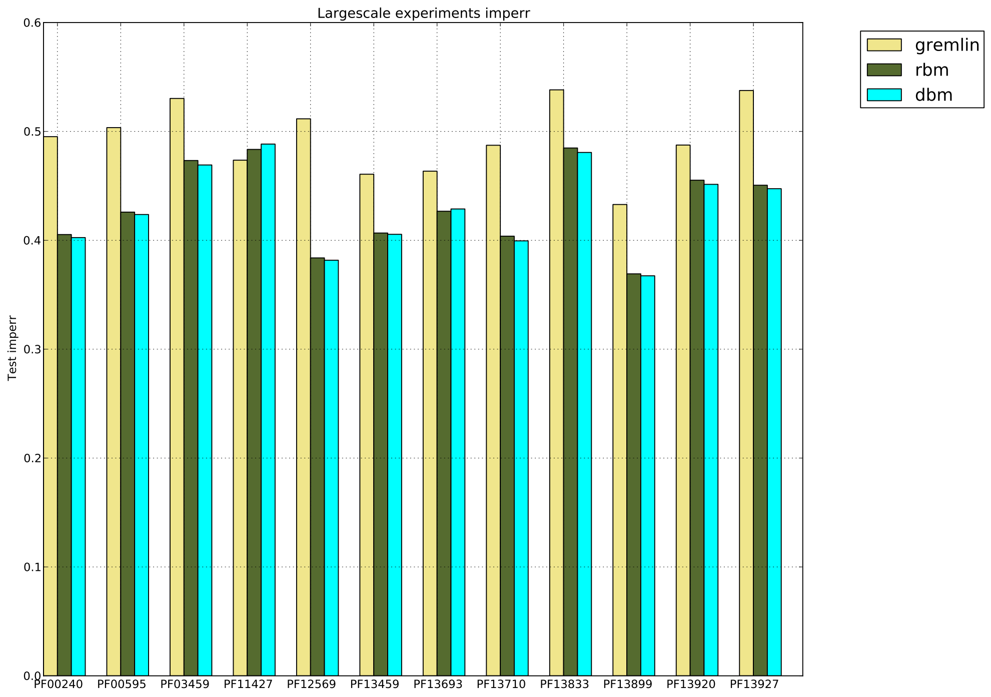


Figure 6.14: Largescale Imputations: Imputation error for all datasets comparing Sparse MRFs (Gremlin), RBMs and DBMs. Boltzmann machines beat Gremlin by about 6 percentage points

The Boltzmann Machines beat the sparse MRF model in all datasets with about 0.5 blosum90 units averaged across datasets. Interestingly, the Blosum90 scores for the PF11427 dataset were partially higher using the Blosum90 metric, indicating the imputation mistakes that were made were less important from a biophysical/evolutionary perspective.

The imputation error and Blosum90 metrics across different protein families provide further evidence of the superior performance of Boltzmann Machines over sparse MRFs.

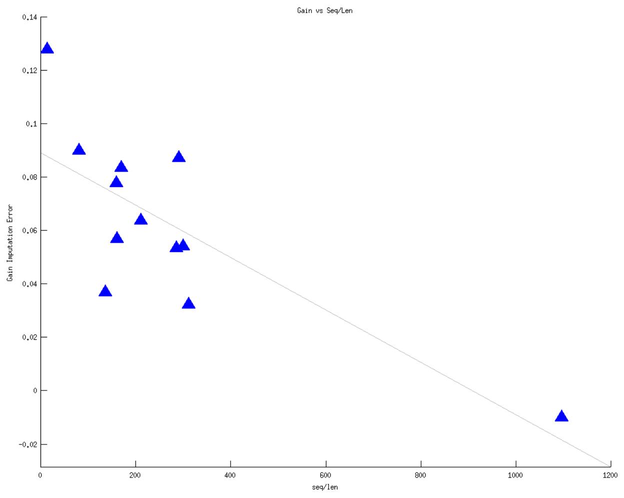


Figure 6.15: Scatter plot of Gain vs Seq/Len ratio. The gain is defined as difference in the imputation error between the top Boltzmann machine and the gremlin model. The inverse relationship shows that Boltzmann Machines outperform MRFs in the low sequence count region.

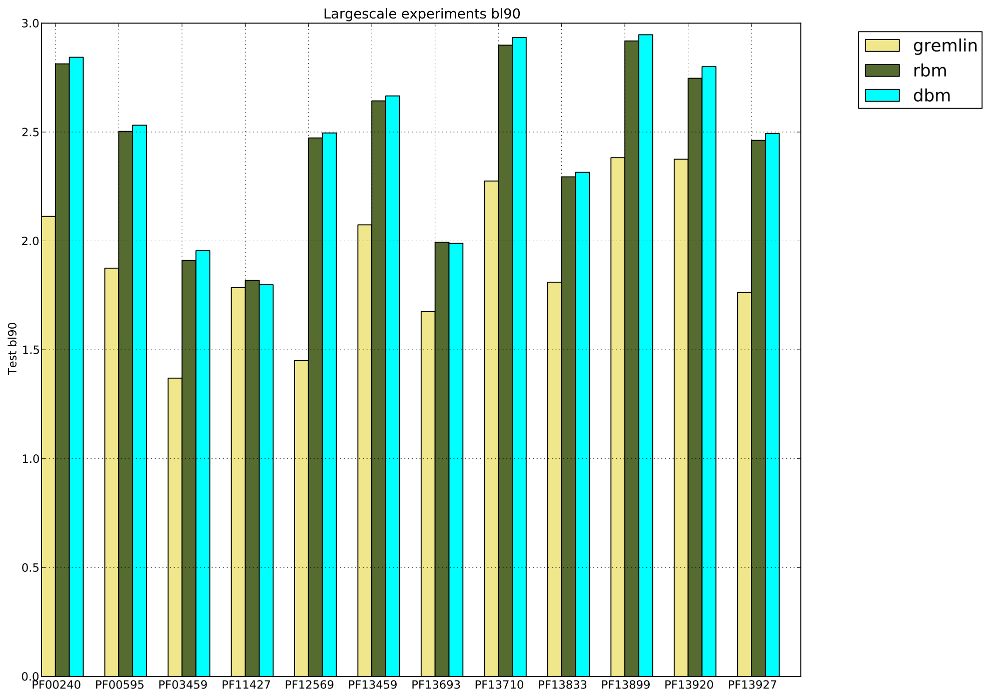


Figure 6.16: Largescale Imputations using Blosum90 scoring matrix: for all datasets comparing Sparse MRFs (Gremlin), RBMs and DBMs. Boltzmann machines beat Gremlin by about 0.5 blosum90 units

Multicolumn Imputation Metrics

We examined if the RBM and DBM models are robust to multicolumn imputations. We examine two cases :

- *Missing Completely at Random (MCAR):* Pick multiple columns at random.
- *Missing with Structure (Block):* Sweep blocks of adjacent columns in the MSA.

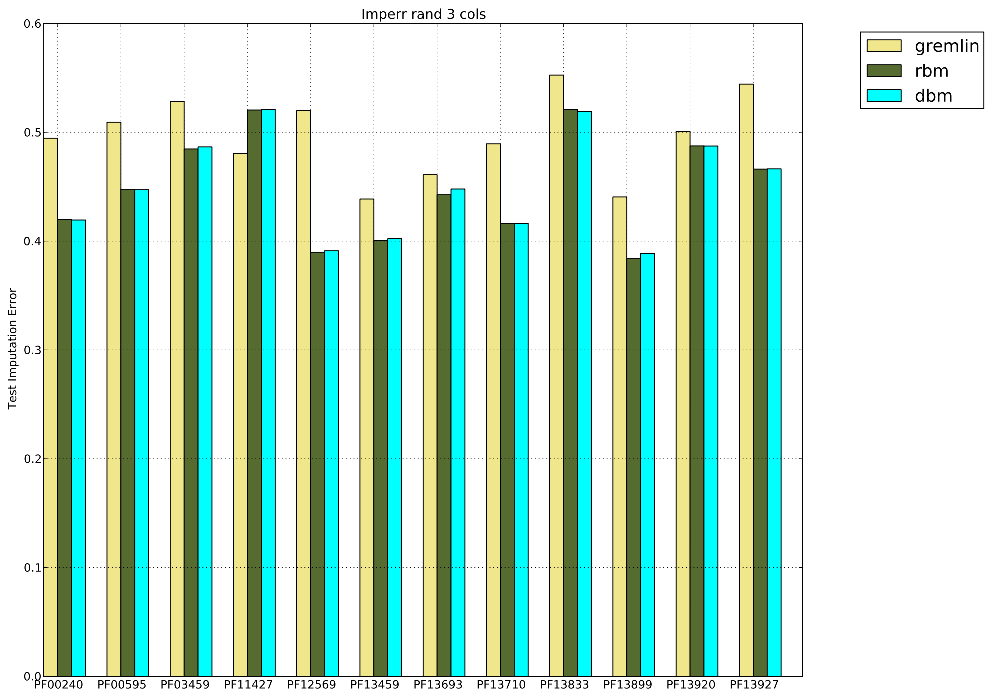


Figure 6.17: Largescale Random Multicolumn Imputations: Multicolumn imputations for all datasets holding out 3 columns chosen at random. Boltzmann machines beat Gremlin by about 5 percentage points

We report the test imputation error of all datasets under the MCAR scenario holding out 3 columns in figure 6.17. The Boltzmann Machines beat the sparse MRF model in all datasets except PF11427 with about a 5% gain averaged across datasets. The gain is dataset dependent with about a standard deviation of about 2% points. Further imputation errors were also run holding out 2,5 and 10 columns and showed similar trends. Scores deteriorated with increasingly held out columns but gain over Sparse MRFs were largely observed. See appendix D.2.2 for details.

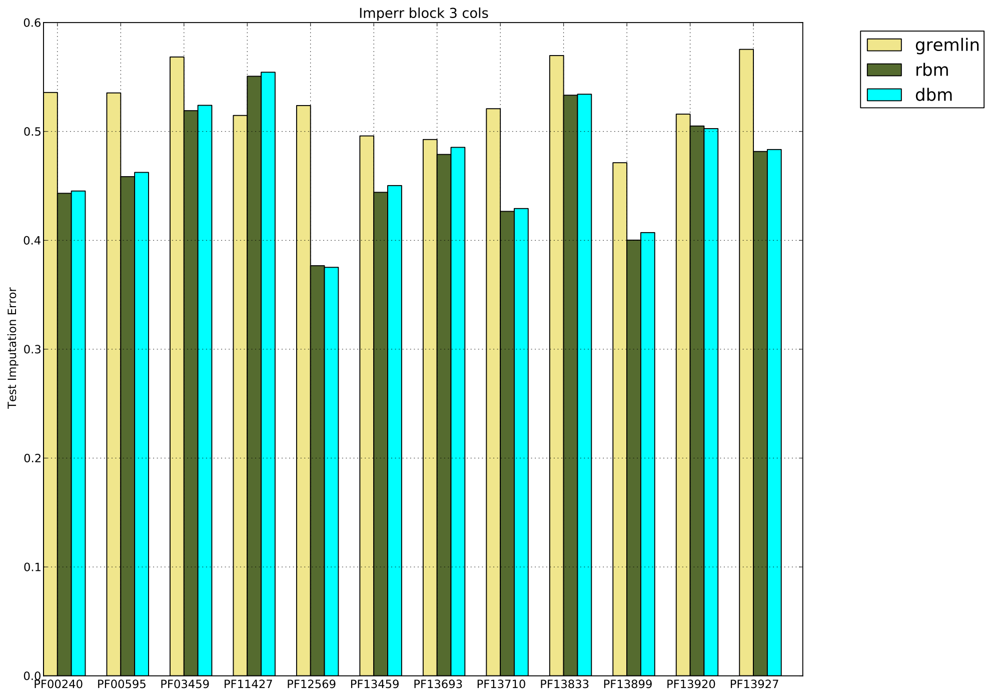


Figure 6.18: Largescale Block Multicolumn Imputations: Multicolumn imputations for all datasets holding out 3 adjacent columns in a block and sweeping the MSA. Boltzmann machines beat Gremlin by about 5 percentage points

We report the test imputation error of all datasets under the block scenario holding out 3 columns in figure 6.18. The Boltzmann Machines beat the sparse MRF model in all datasets except PF11427 with about a 6% gain averaged across datasets. The gain is dataset dependent with about a standard deviation of about 2% points. Further imputation errors were also run holding out 2,5 and 10 columns and showed similar trends. Scores deteriorated with increasingly held out columns but gain over Sparse MRFs were largely observed. See appendix D.2.2 for details.

The multicolumn imputation errors across different protein families demonstrate the robustness of Boltzmann Machines for designing and imputing chunks of amino acids.

Imputation of Important Residues

In the previous sections we evaluated the Boltzmann Machines using single and multicolumn imputation error for any residue in the MSA. Here we consider functionally important residues. We consider three types of functionally important residues:

- *Surface Residues:* The solvent exposed surface of the protein.
- *Core Residues:* In the hydrophobic interior/core of the protein in complex.
- *Binding Residues:* Found at the binding interface of the protein multimer or those residues bound to a ligand.

We extracted the structural information from PDB structures associated with these families (`http://gremlin.bakerlab.org/`). The surface and core residues were chosen using a solvent exposure cutoff of $2.5\AA^2$. The ligand residues were chosen using a $10\AA$ radius from the pocket. The extracted PDB residues were mapped onto the pfam columns by running ClustalW2 with default parameters for protein alignments [85]. Table 6.12 lists the protein families and the associated pdbids and ligands. Table 6.13 lists the protein families and the number of residues extracted according to each category. In some cases, the number of core residues were very limited (PF11427, 13693) had only 3 residues according to this threshold.

We report the test imputation error of all datasets for the bound, core and surface residues holding out 3 columns in figures 6.19, 6.21 and 6.20 . The Boltzmann Machines beat the sparse MRF model in all datasets except PF11427 with about a 5% gain averaged across datasets. The gain is dataset dependent with about a standard deviation of about 2% points. Further imputation errors

Pfamid	**Pdbid**	**Chain**	**BindingChains**	**Ligands**
PF12569	2XPI	A	E	AUC
PF03459	1H9M	A	B	MOO
PF13693	4ICH	A	B	B3P;BMR;BR;EDO
PF13710	2F1F	A	B	1PE;MG;P33
PF13899	3F9U	A	B	EDO;NO3
PF11427	1TC3	C	A;B	None
PF13920	4AYC	A	B	CL;CPQ;GOL;SO4;ZN
PF13459	1IQZ	A	None	SF4; SO4
PF13927	2ID5	A	B;D	MAN;NAG
PF13833	3TZ1	A	B	CA
PF00240	3V6C	B	A	CL;GOL;ZN
PF00595	4F8K	A	B	None

Table 6.12: Extracting Important Residues Positions from Protein Families. Associated PDB structures, ligands and binding interfaces.

were also run holding out 2,5 and 10 columns and showed similar trends. Scores deteriorated with increasingly held out columns but gain over Sparse MRFs were largely observed. See appendix D.2.2 for details.

dataset	**core**	**bound**	**surface**
PF00240	8	44	51
PF00595	15	3	66
PF03459	10	30	54
PF11427	2	20	48
PF12569	76	61	100
PF13459	10	51	55
PF13693	8	44	70
PF13710	7	34	56
PF13833	2	38	51
PF13899	7	56	75
PF13920	7	48	42
PF13927	7	30	68

Table 6.13: Additional Families - Number of important residues according to each category

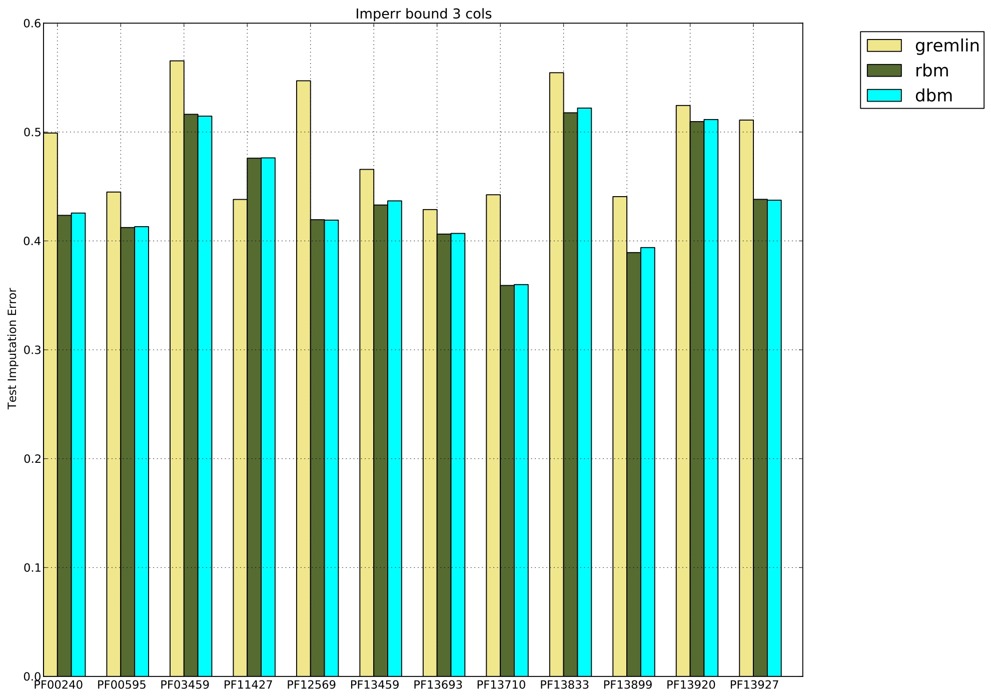


Figure 6.19: Largescale Bound Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 3 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

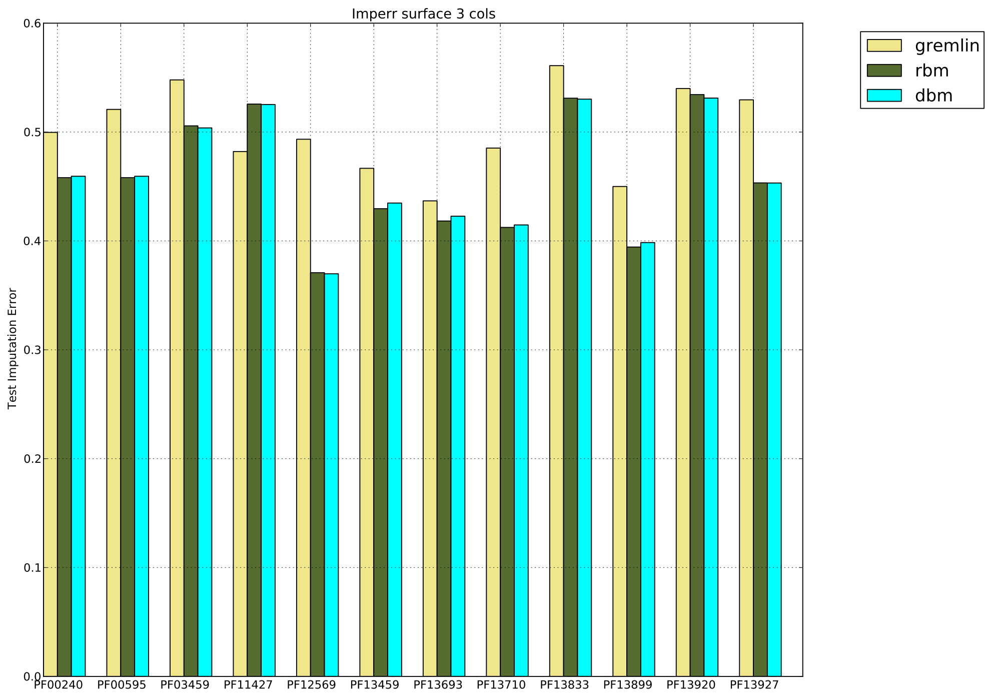


Figure 6.20: Largescale surface Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 3 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

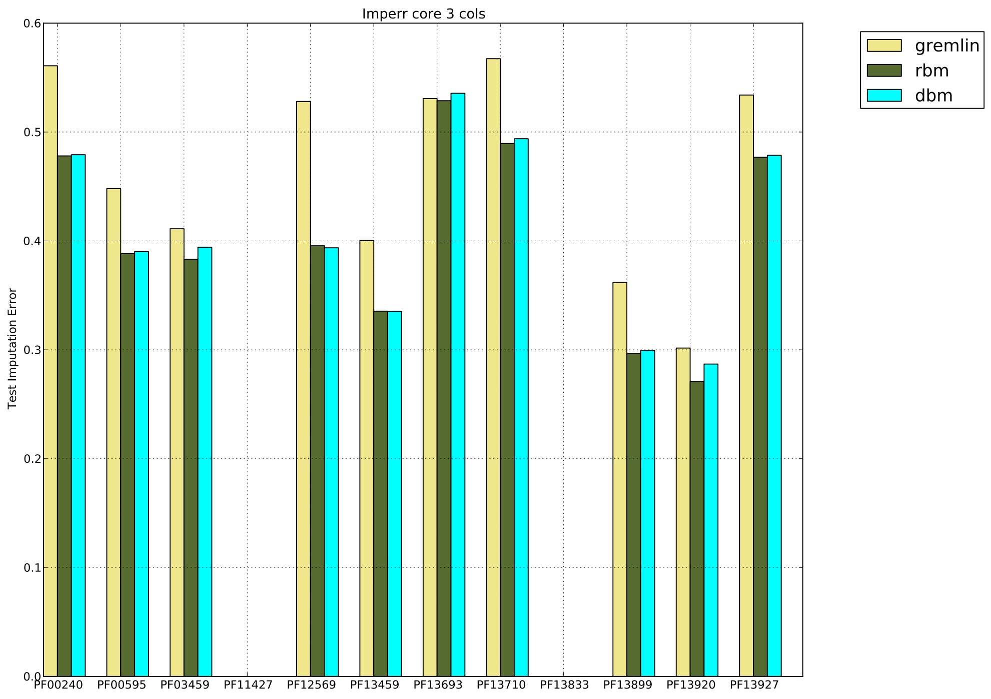


Figure 6.21: Largescale core Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 3 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

metric	gremlin	rbm	dbm	rbmhyp	dbmhyp
imp	0.4736	0.4835	0.4884	0.4609	0.4573
imp2	0.4941	0.5287	0.5317	0.4840	0.4845
imp3	0.5147	0.5507	0.5544	0.4993	0.4995
imp5	0.5595	0.5857	0.5917	0.5247	0.5244
imp10	0.6042	0.6269	0.6347	0.5564	0.5551

Table 6.14: Block Imputation for PF11427. RBM and DBM models using default hyperparameters are represented by *rbm* and *dbm*. An alternate RBM and DBM model with a different architecture, represented by *rbmhyp* and *dbmhyp*, is able to obtain moderate gains over the *gremlin* model.

6.6.7 Investigating PF11427

The Boltzmann Machine models beat the Gremlin model in all datasets except for the PF11427 dataset. This was the dataset with the largest number of sequences (54866 sequences). It also has the highest sequence/length ratio of 1097 sequences per residue. We investigated if an alternate Boltzmann Machine topology could help improve the performance. We tried a *rbm-100* and a *dbm-100-100* architecture. We report the results of block imputations for PF11427 in table 6.14 using blocks of 1,2,3,5 and 10.

In all cases, the rbm and dbm with the newer architecture represented by *rbm-hyp* and *dbm-hyp* modestly beat the gremlin model. We conjecture that with more detailed hyperparameter scanning it might be possible to obtain other Boltzmann Machines that might beat the Gremlin model by a larger margin. However, the computational costs associated with such a hyperparameter scans might outweigh the benefits of the obtained gains. If the best model is indeed desired, then alternate hyperparameter search techniques using Bayesian Optimization might be more appropriate than exhaustive scanning.

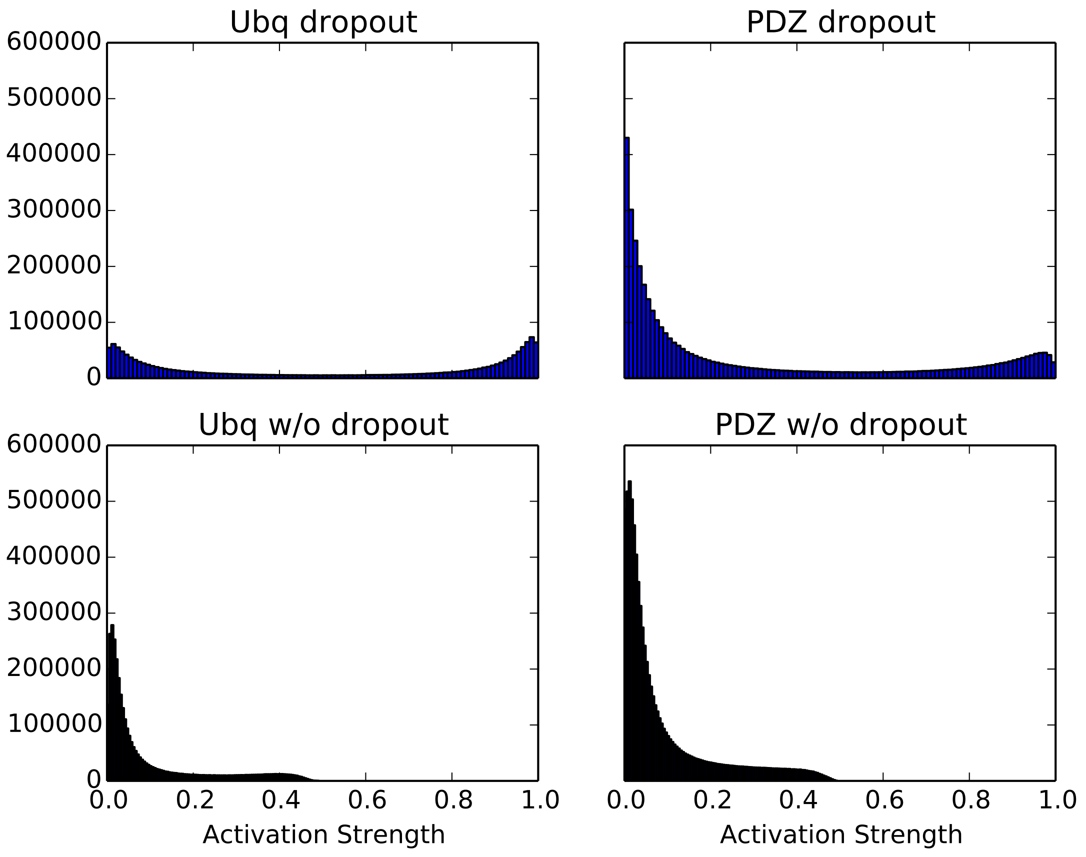


Figure 6.22: Histograms of the activation of the hidden nodes with and without dropout. The x-axis is a measure of the activation strength. The y axis counts the number of nodes with a particular activation strength.

Model Analysis

We introspect the properties of our models to gather insights into the parameters and properties of the latent representations.

Dropout: We calculated the average activations of the models with and without dropout. We see in the histograms in Figure 6.22 that without dropout (bottom row) the vast majority of nodes have very low activation strengths. That is, a small minority of the nodes are responsible for representing the features of the distribution. In contrast, when dropout is used (upper row), there

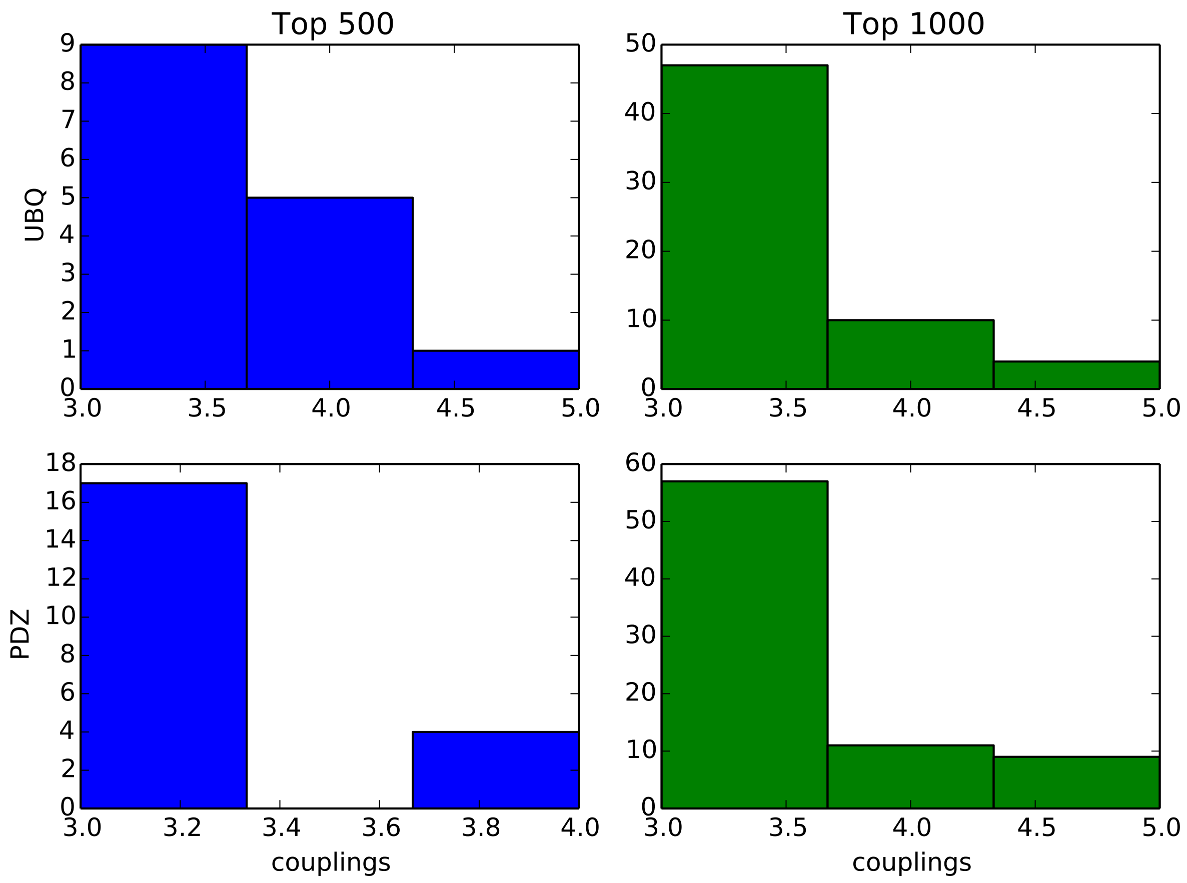


Figure 6.23: Number of strong connections for top 500 and top 1000 edge weights. The x-axis measures the number of interactions modelled by a hidden node. The y-axis is a count of the number of hidden nodes that model k interactions.

are many more nodes with high activation strengths. This demonstrates that dropout has a role to play in regularizing the activations, and reducing the model's tendency to over-fit (although not eliminating it).

Strong Activations: Markov Random fields have limited representative power, since they are limited to pairwise edge potentials between visible variables. HMRFs and Boltzmann Machines allow for more subtle interactions via the hidden nodes. We extracted the top 500 and top 1000 weights from our models ($\ll 1\%$ of the total number of weights) to see if they show evidence that some hidden nodes are learning k-ary interactions. As seen in Figure 6.23, there are a significant

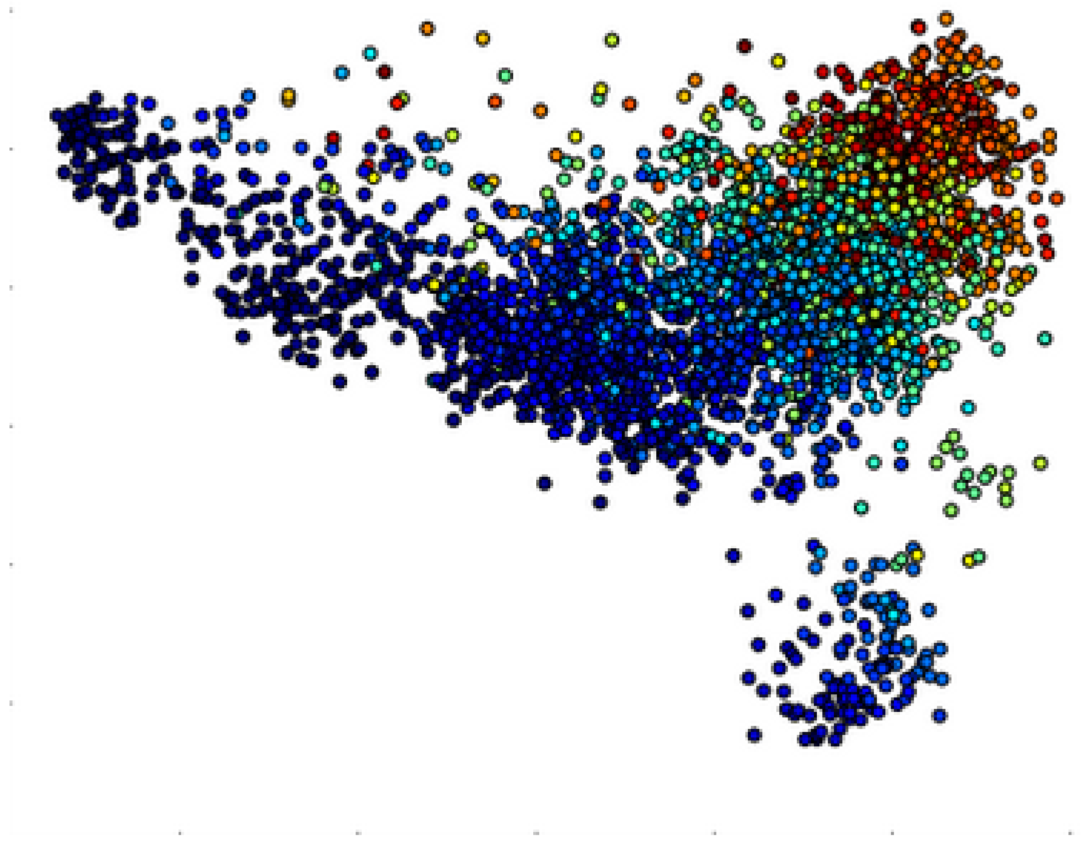


Figure 6.24: Projecting hidden layer representations onto two dimensions using PCA and coloring by the number of gaps for ubiquitin. An interesting progressive coloring pattern emerges.

number of 3,4, and 5-way couplings (couplings < 3 are not shown). The ability for Boltzmann Machines to capture these k-ary interactions explains their superior performance over HMRFs.

Hidden representation analysis: We extracted the representations of the ubiquitin training data from the hidden layers of our models and then perform Principal Components Analysis to examine the main patterns in the data (Figure 6.24). We colored the projected instances via the number of gaps in the aligned sequence (a measure of evolutionary distance from the consensus sequence). The projected sequences show evidence of clustering that roughly corresponds to the percentage of gaps.

6.6.8 Chapter Summary

In chapter 4, we introduced a framework for unsupervised learning of protein families using Markov Random Fields. We made a distinction between Visible Markov Random Fields and those with latent variables also known as Hidden Markov Random Fields. We observed that despite the promise of latent variables in modelling complex distributions, they empirically fail to learn good models due to poor inference in non-tree structured graphs.

In this chapter, we introduced a different way to model latent variables by using Boltzmann Machines. In particular, we examine specific architectures of Boltzmann Machines characterized by layers of visible and hidden variables arranged in a p-partite graph structure. These architectures allow for efficient inference due their ability to factorize given adjacent layers. These architectures are known as Restricted Boltzmann Machines (RBM) which contain a single hidden layer and Deep Boltzmann Machines (DBMs) which contain multiple layers.

We train and evaluate RBMs and DBMs for the ubiquitin and the PDZ protein family using a number of different architectures and hyperparameter settings. We also employ the dropout technique, a model averaging technique which works by dropping nodes at random at each learning step and stitches them together at test time. We find that the RBM and DBM models far outperform the Markov Random Fields described in the previous chapters with respect to test metrics. Additionally, the dropout technique has a pronounced effect on the test error with the best models being obtained thusly.

RBMs and DBMs are densely connected Boltzmann Machines with a very large number of parameters. We hypothesise that it might be possible to learn Boltzmann Machines that have comparable performance without needing excessive parameters. Towards this end, we introduce an approach to learn Boltzmann Machines with sparse edge topologies. This is achieved by bootstrapping a RBM from a sparse MRF via a Cholesky decomposition. The corresponding RBM model is known as a Sparse Restricted Boltzmann Machine (SRBM)

SRBMs have similar test error as MRFs showing a successful mapping of MRF parameters to RBMs. Additional hidden nodes are added to the SRBM to create a Sparse Semi Restricted Boltzmann Machine (SSRBM). This causes the test error to drop sharply and hence it turbocharges the SRBM. This shows that the additional nodes help in modelling aspects which the SRBM alone cannot model. However, the SSRBMs perform perform slightly worse that their corresponding RBMs. This might suggest that RBMs by themselves can learn good representations and additional MRF prior information is not needed.

We introduced the Locally Connected Deep Boltzmann Machine (LC-DBM) which adds a second hidden layer to the sparse SRBM hidden layer. The LC-DBM model was motivated by the hypothesis that higher layers can model complex aspects of the distribution. We note that test error of LC-DBM model is lower than the corresponding SRBM in some cases. However these gains are fairly modest. In general, it appears that the second hidden layer has no net positive effect. Moreover, continually adding units to the second hidden layer does not improve the test error. This suggests that there is a fundamental information bottleneck in the first hidden layer created by a sparse edge matrix. This theory is given further credibility when taken in conjunction with SSRBM results; wherein the test error drops sharply after adding even a few dense hidden units.

Finally, we run a large scale study with ten additional protein families selected to belong to variety of structural folds. We also test two extreme cases of protein families with large number of sequences and length. We test these models with additional multicolumn imputation metrics and also over functionally important regions of these proteins. We note that for almost all models, the Boltzmann Machines outperform the MRFs using these battery of metrics. We also note that there is an inverse relationship between the gain obtained and the sequence/length ratio of the protein family suggesting that Boltzmann Machines can be especially effective in the low sequence count regime.

Chapter 7

Conclusions and Future Work

7.1 Summary of Contributions

The aim of this thesis was to learn and evaluate effective predictive and generative models of protein sequence families. We achieve this by (1) effective large-scale feature selection for predictive models and (2) learning rich feature representations via undirected graphical models for generative models.

Predictive models via drug design: In the first part of the thesis (chapter 3), we focused on large scale feature selection for learning predictive models of protein sequence families. We did so through the lens of drug cocktail design against HIV-1 infection. The core of the cocktail design problem relied on scaling a lasso regression model to large multiple sequence alignment containing 70,000 sequences. In the context of this application, we evaluated and discussed several large scale feature selection strategies for predictive models of protein sequence families.

We were able to scale the lasso regression model to the entire HIV-1 dataset which the previous authors Hinkley et.al [29] failed to address. We solved the lasso scaling problem using a variety of strategies including sparse matrix vectorization and feature reduction strategies such as strong rules [87]. We were able to report the RMSE and provided a concrete measurement of the predictive utility of our method. The previous authors do not report the RMSE, R^2 or the p-values of their model and this renders their results incomparable with standard regression models.

Finally, we leveraged the lasso regression model in a method for designing drug cocktails robust to HIV mutations. We validated our predicted drug cocktails on a held out test data set using regret analysis. We found that our cocktail design method can be used for personalized medicine under a limited budget (upto 3 drugs) and is robust to viral mutations. We additionally suggested that for larger budgets it is beneficial to use overall drug statistics than personalized cocktail approaches.

Generative models via Markov Random Fields: In the second part of the thesis (chapter 4) we switched our focus to generative models of protein sequence families. Generative models model the probability density of the data through the features in its model. Good features leads to better models. The crux of the problem is allowing the generative models to have a rich enough feature representation space such that they can model arbitrary distributions of protein sequence families, yet remain computationally tractable and learneable.

We presented a framework for modelling protein families as a series of increasingly complex generative models. These models belong to the class of undirected graphical models known as Markov Random Fields (MRFs). We show that by adding edges between variables in the MRF we can relax the assumptions made about the model distribution. Additionally, we made a distinction between MRF models that are completely visible and those with unobserved latent variables. The introduction of latent variables offers the flexibility of learning distributions that cannot be explained by the visible variables alone. Often, this comes at the cost of model complexity and computational tractability.

We evaluated these models using generative metrics such as imputation error and test log likelihood. It was observed that all MRF models beat a completely independent model and were hence able to model contextual information. Also, VMRFs trained with pseudo log likelihood had the best imputation error scores, showing further gains with added edges. However, we find that non-tree HMRFs do not learn well and suffer from poor inference. We note that Hidden Markov Random Fields offer the promise of modelling complex distributions, but are unable to do so in practice for all but tree-structured graphs. This motivated the need to commission other latent variable architectures. This led us to explore Boltzmann Machines, in the next chapter.

Feature Interpretation via GPCRs In the third part of the thesis (chapter 5), we considered extending the utility of the generative models beyond just improving the test error. We did so by introspecting and interpreting the learned features beyond for biological significance. Similar to the first part of the thesis we approach this task through the lens of a target application. In particular, we examine signal transduction in G protein coupled receptors (GPCRs) ; which relay signals across cell membranes.

We identify networks of co-evolving residues from multiple sequence alignments by learning the topology of a Markov Random Field trained on GPCR sequences. We find that pairwise interactions containing residues in the ligand binding pocket are enriched. An analysis of these interactions reveals a minimal GPCR binding pocket containing four residues (T118$^{3.33}$, M207$^{5.42}$, Y268$^{6.51}$ and A292$^{7.39}$). Additionally, the ten residues predicted to have the most long-range interactions, are also part of the ligand binding pocket. This suggests that the activation in rhodopsin (a canonical GPCR) involves these long-range interactions between extracellular and intracellular domain residues mediated by the retinal domain.

Generative models via Boltzmann Machines : In the last part of the thesis (chapter 6), we introduced a different way to model latent variable models by using Boltzmann Machines. In particular, we examine specific architectures of Boltzmann Machines characterized by layers of visible and hidden variables arranged in a p-partite graph structure. These architectures allow for efficient inference due their ability to factorize given adjacent layers. These architectures are known as Restricted Boltzmann Machines (RBM) which contain a single hidden layer and Deep

Boltzmann Machines (DBMs) which contain multiple layers.

We evaluated RBMs and DBMs using a number of different architectures and hyperparameter settings. We also employed “dropout” , a model averaging technique which works by dropping nodes at random and stitches them together at test time. We found that the RBM and DBM models far outperformed the Markov Random Fields described in the previous chapter with respect to test metrics. Additionally, the dropout technique helps in learning the best models.

RBMs and DBMs are densely connected Boltzmann Machines with a very large number of parameters. We hypothesise that it might be possible to learn Boltzmann Machines that have comparable performance without needing excessive parameters. Towards this end, we introduce an approach to learn Boltzmann Machines with sparse edge topologies. This is achieved by bootstrapping a RBM from a sparse MRF via a Cholesky decomposition. The corresponding RBM model is known as a Sparse Restricted Boltzmann Machine (SRBM)

SRBMs have similar test error as MRFs showing a successful mapping of MRF parameters to RBMs. Additional hidden nodes are added to the SRBM to create a Sparse Semi Restricted Boltzmann Machine (SSRBM). This causes the test error to drop sharply and hence it turbocharges the SRBM. This shows that the additional nodes help in modelling aspects which the SRBM alone cannot model. However, the SSRBMs perform perform slightly worse that their corresponding RBMs. This might suggest that RBMs by themselves can learn good representations and additional MRF prior information is not needed.

Finally, we introduced the Locally Connected Deep Boltzmann Machine (LC-DBM) which adds a second hidden layer to the sparse SRBM hidden layer. The LC-DBM model was motivated by the hypothesis that higher layers can model complex aspects of the distribution. We note that test error of LC-DBM model is lower than the corresponding SRBM in some cases. However these gains are fairly modest. In general, it appears that the second hidden layer has no net positive effect. Moreover, continually adding units to the second hidden layer does not improve the test error. This suggests that there is a fundamental information bottleneck in the first hidden layer created by a sparse edge matrix. This theory is given further credibility when taken in conjunction with SSRBM results; wherein the test error drops sharply after adding even a few dense hidden units.

7.2 Future Directions

There are several ways the ideas mentioned in this thesis can be extended and developed upon, particularly related to Boltzmann Machines.

Protein Design : In this thesis we demonstrated that Boltzmann Machines such as RBMs and DBMs can be used to learn effective generative models of protein sequence families. A useful property of generative models is that it is possible to sample from these models. Novel protein sequences can be generated from the model by running Gibbs sampling chains. This can be useful for the task of protein design. Constraints can be specified which fix certain amino acid positions while sampling the other positions to “design” a novel sequence subject to constraints. A real world example is the design of an influenza vaccine, wherein the stem of an antibody is constrained to be fairly conserved while other positions can be designed for.

Boltzmann Machines of protein structures: So far we have only discussed generative models of protein sequences. Markov random fields with fixed physical force fields [39, 93] can be used to model protein structure. The quantities to be modelled are backbone (ϕ) and side chain (ψ) angles. We can envision modelling these angles using the different Boltzmann Machines as well. The angles can be discretized into rotamers [11] and modelled via softmax visible units. If on the other hand, real valued angles are desired, we can use Gaussian visible units to model the angles. The partition function of this distribution can also be calculated using Annealed Importance Sampling (AIS) and can be instrumental in determining physical quantities such as the enthalpy and the entropy of the equilibrium protein distribution.

Multimodal joint sequence and structure model : Srivastava *et.al* [77] learned a joint image and text Deep Boltzmann Machine using flickr data. The images were modelled using Gaussian visible units whereas the text was modelled using replicated softmax units. These Boltzmann Machines were tied together using higher level hidden layers to form a Deep Boltzmann Machine. Similarly, we can learn a joint sequence and structure of proteins with the higher levels acting as a bridge between the two modalities.

Universal protein family model : A particularly ambitious goal is to have a single universal model of all protein sequence families. So far our sequence models have fixed lengths. As a consequence a separate model needs to be learned for every new protein family. A universal protein family should allow for variable length protein families. This can be achieved by considering template based models which share parameters. Another approach is to have Convolutional Max-Pooling units [48]. The universal protein family is likely to learn useful high level representations that are common across all proteins.

Appendix A

Drug Cocktail Design

A.1 Experiments and Results

In this section we describe the dataset and data preparation protocols. We also describe the experimental setup and results from contact map based feature transformation, feature reduction, lasso regression model and drug cocktail design studies.

A.1.1 Data Description and Preparation

The dataset obtained from Monogram Inc [67] contained 71727 sequences in total. The protease alignment had a width of 99 columns while the reverse transcriptase alignment had a width of 305 columns. The sequences were annotated with Replication Coefficient (RC) values in presence and absence of drugs. RC is a non-negative real value denoting approximately the infectivity of a viral sequence. See [29] for clinical methods to generate the RC values. In total there were 20 drugs. Not all sequences had all the drugs. We consider only the 15 drugs as used in GKRR study [29]. See 3.4 for the distribution of RC values.

On analyzing the sequences, we found 9 sequences that were badly aligned. 618 had long indels. Addditionally, 9 sequences did not have all the 15 drug used in the GKRR study. These anomalous sequences were removed from the dataset to keep the dataset consistent with the GKRR study. After cleaning, there were 71,091 HIV protease and RT sequences each annotated with 16 real values (15 drugs + 1 non-drug).

We additionally note that a large number of sequences ($>$ 67,000) had X characters in them ; most likely arising out of inaccuracies in the next-gen sequencing method. Figure A.1 and A.2 show the distribution of X characters in the protease and RT alignments respectively. Most of these positions have X character occurrence rate of $<$5% even though there are some $\sim$15%. These X characters are considered as missing values that need to be imputed, further discussed in section A.1.1.

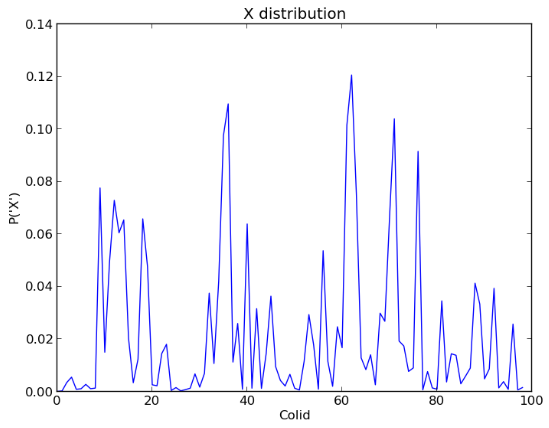


Figure A.1: Protease

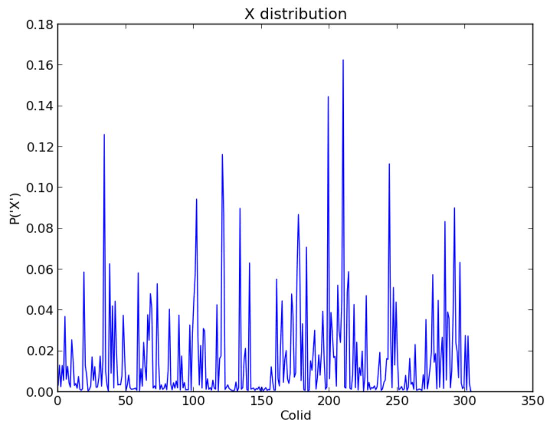


Figure A.2: Reverse Transriptase

Figure A.3: Distributions of X characters in the raw alignments of protease and reverse transcriptase. These X characters correspond to undetermined amino acids at a position usually arising out of inaccuracies with the next gen sequencing process.

The dataset was split into a training + validation set of 50,000 sequences and a test set of 21,091 sequences. The test set was not touched until after all regression models were learned.

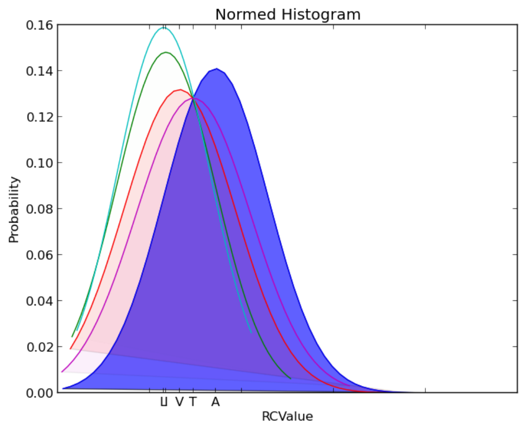


Figure A.4: Col 71 Protease

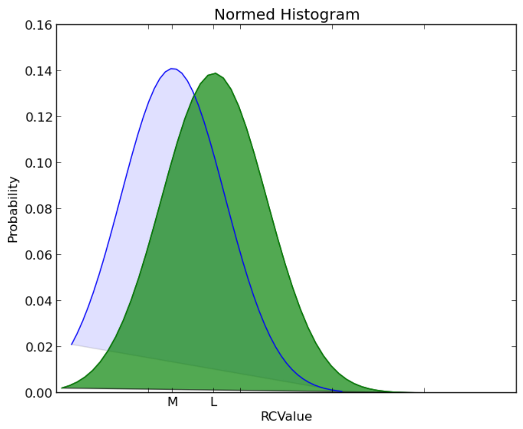


Figure A.5: Column 90 Protease

Figure A.6: Visualizing the differences in single amino acid features. Each Gaussian corresponds to the RC value distribution conditioned on the amino acid at a particular column in the Protease MSA. The shade depth corresponds to the abundance of the particular amino acid type at that position. Column 71 and Column 90 have especially strong signals.

Imputation of X characters

X characters were replaced using a careful imputation protocol. The imputation protocol involved taking a k-NN vote using edit distance around a candidate sequence. k was set at 5 based on empirical evidence. If all neighbours also had X characters, we defaulted to using the canonical NL4-3 sequence. Figure A.7 and A.8 show a log-log zipf plot of the duplicate count of the imputed sequences. The X axis shows the duplicate count. The Y axis shows the frequency of occurrence of that duplicate. The axes are log-log. We can see that protease has atleast one duplicate which occurs 1000 times(rightmost point on X axis). Whereas RT has much fewer duplicates. When Protease and RT are combined into a joint alignment the number of duplicates becomes zero . Additionally, we also considered a simpler imputation protocol by replacing all the X characters with the consensus NL4-3 sequence, however we found that it introduced a lot of duplicates in the alignment and hence this approach was rejected.

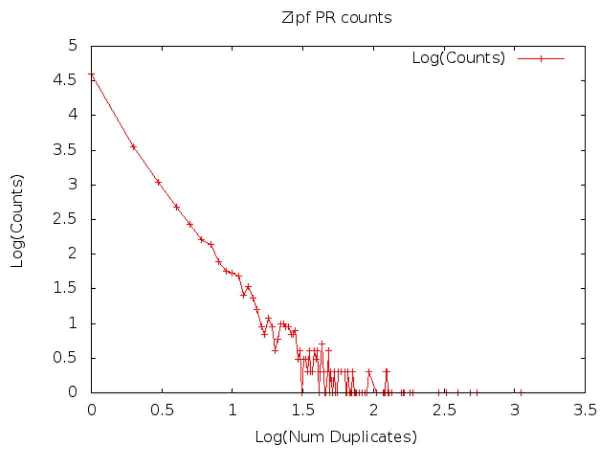


Figure A.7: Zipf Protease

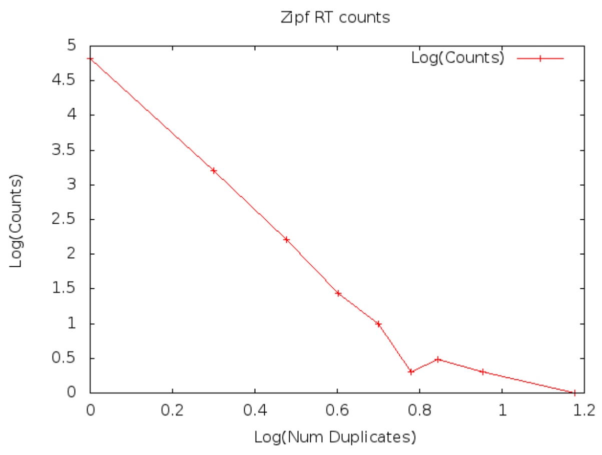


Figure A.8: Zipf Reverse Transcriptase

Figure A.9: Zipf Plots of duplicates in protease and reverse transcriptase. The X axis shows the duplicate count. The Y axis shows the frequency of occurrence of that duplicate. We can see that protease. The axes are log-log. We can see that protease has atleast one duplicate which occurs 1000 times(rightmost point on X axis). Whereas RT has much fewer duplicates. When Protease and RT are combined into a joint alignment the number of duplicates becomes zero

A.1.2 Feature Transformation - Contact Map Analysis

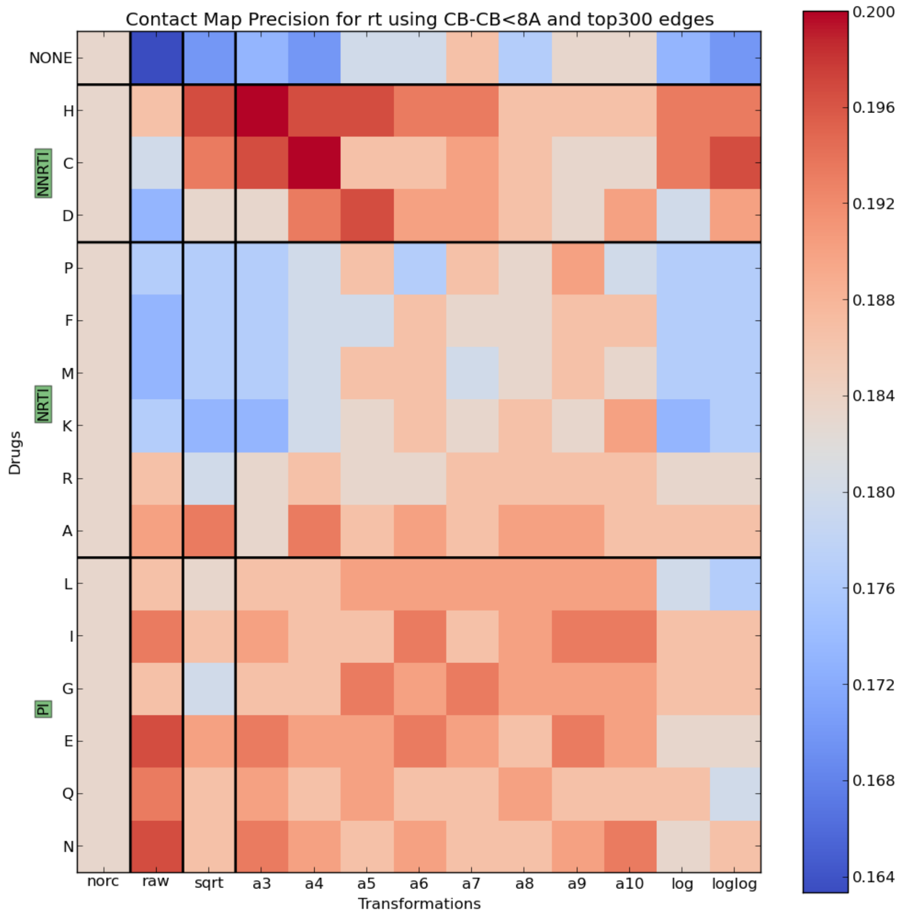


Figure A.10: **Contact Map Analysis Reverse Transcriptase**: The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI). The X axis lists all the transformations used (norc - no weighting,raw- Untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \ldots 10\}$,log and loglog). The colorbar corresponds to the precision for contact map recovery using the Gremlin method. The top 300 edges were chosen for visualization purposes.

A.1.3 Feature Reduction Strategies

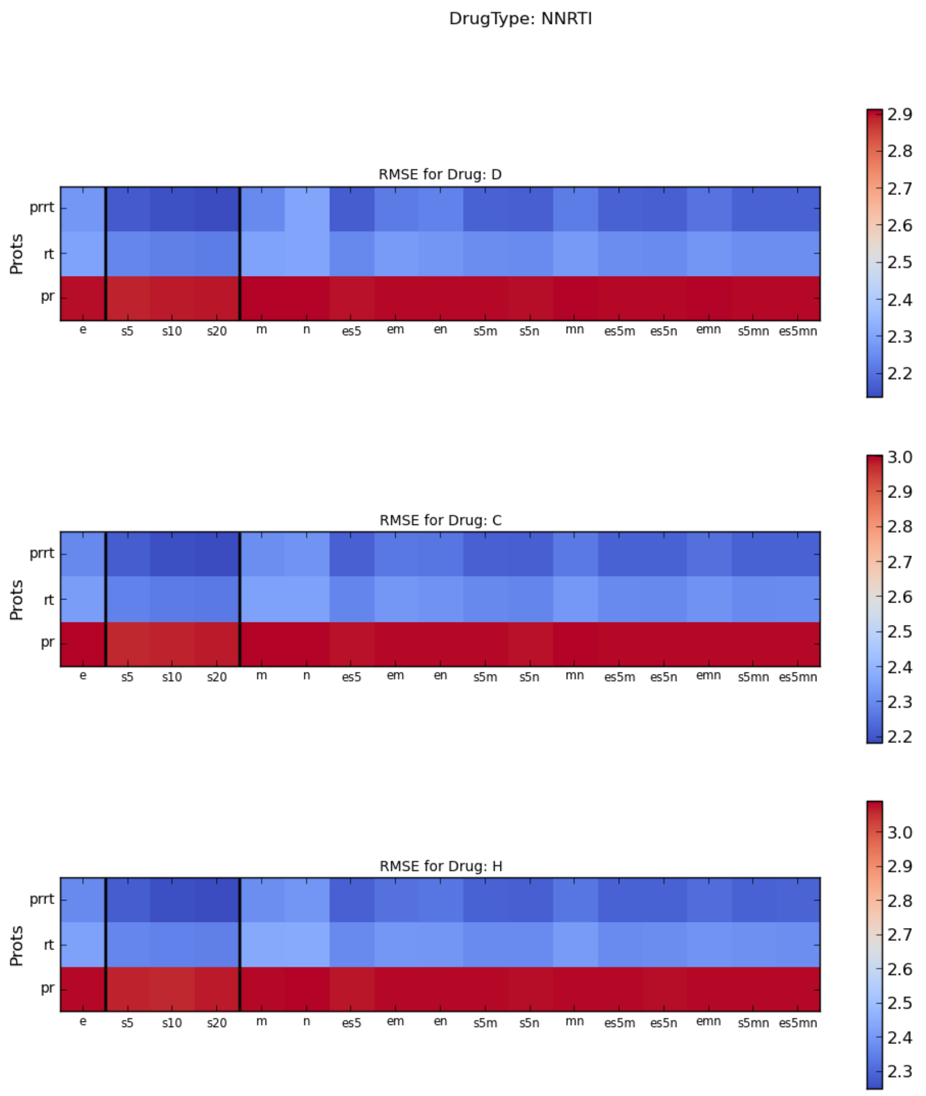


Figure A.11: **Feature Selection plots for Non Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs

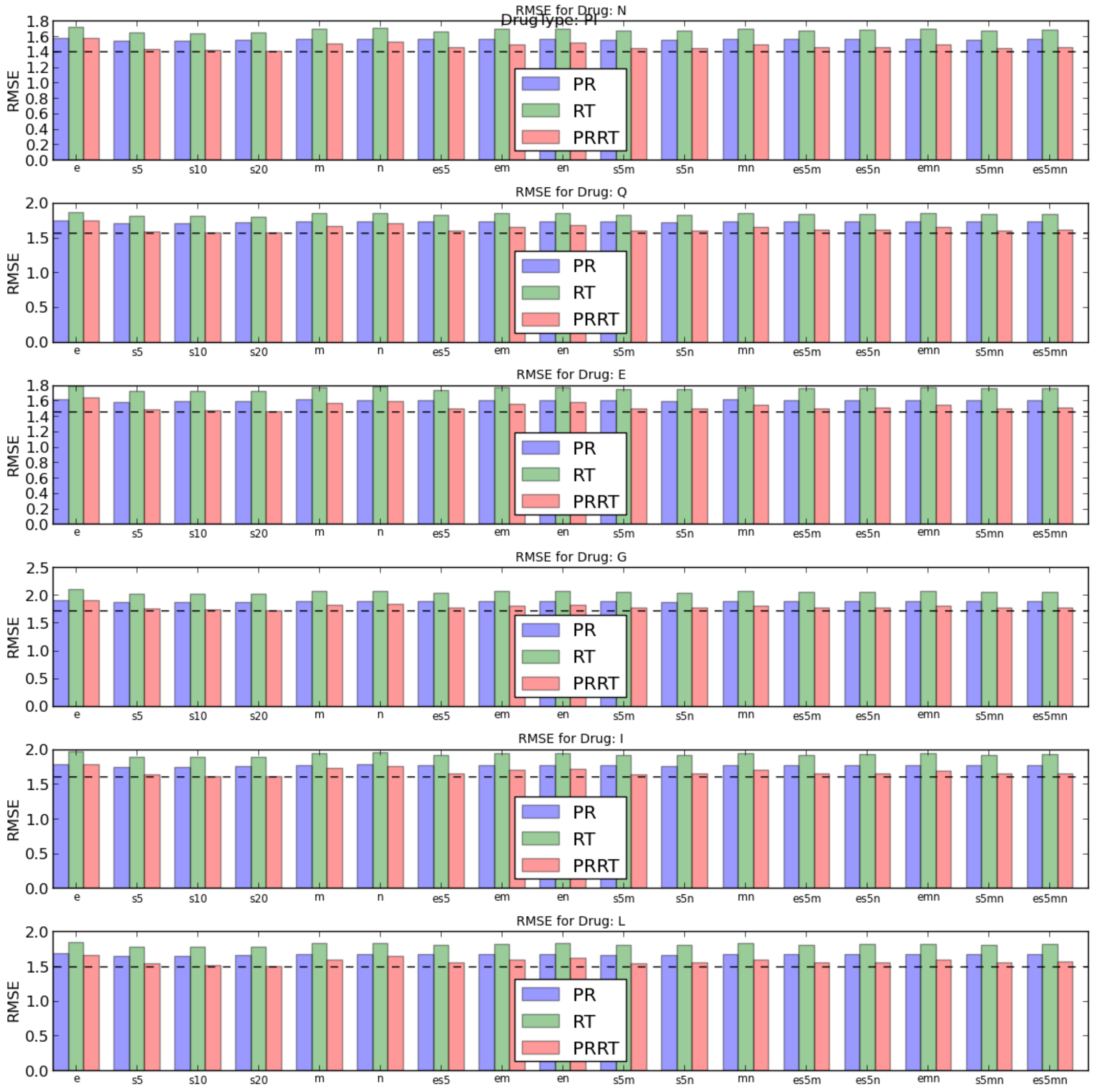


Figure A.12: **Feature Selection plots for Protease Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs

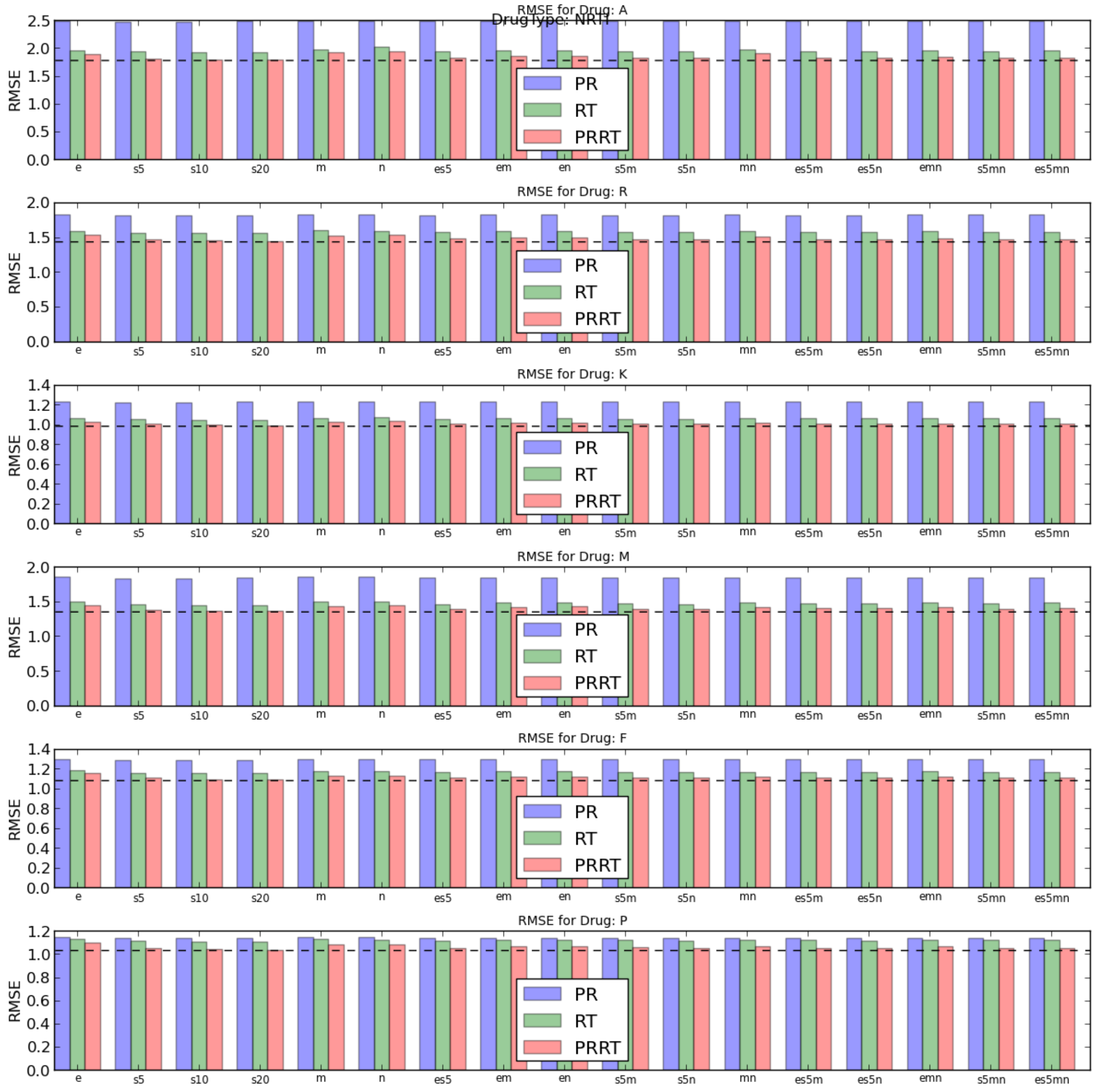


Figure A.13: **Feature Selection plots for Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs

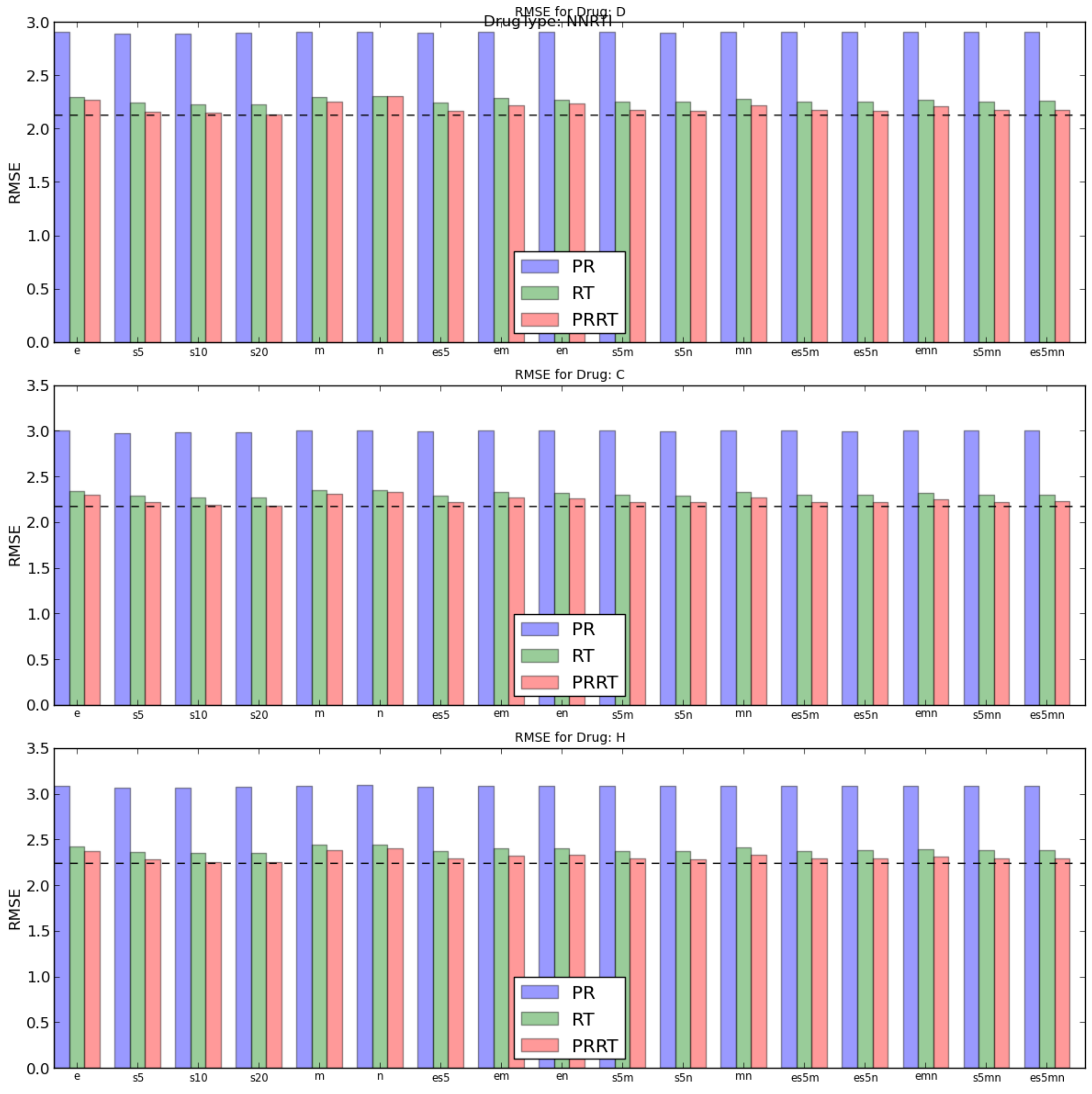


Figure A.14: **Feature Selection plots for Non Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs

A.1.4 Cocktail Design

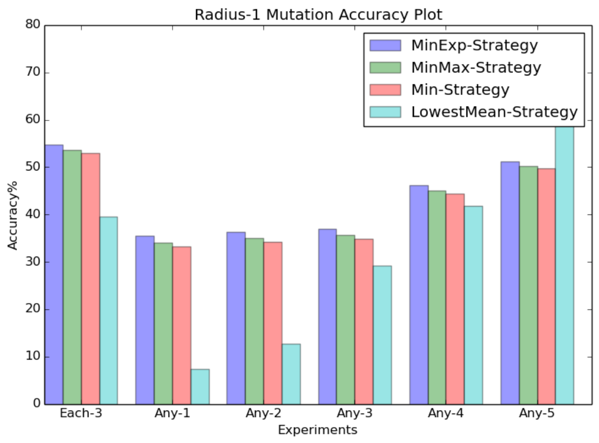


Figure A.15: Radius 1 Accuracy

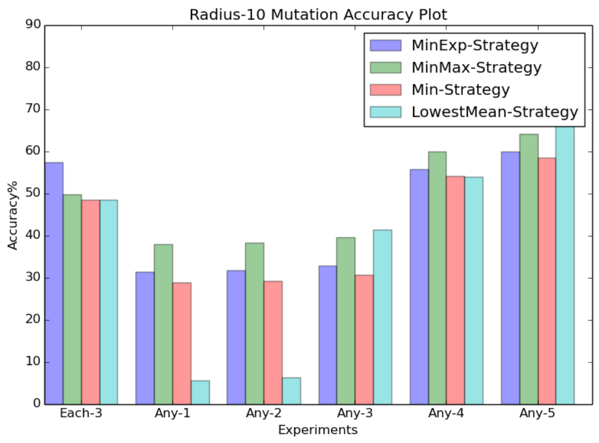


Figure A.16: Radius 10 Accuracy

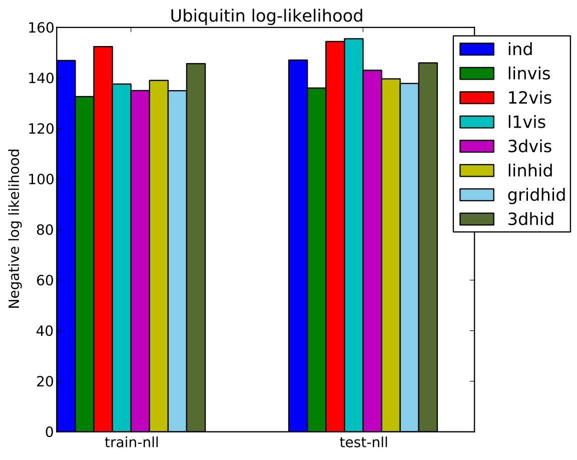
Figure A.17: The X axis contains shows the different drug combination choices. Each-3 refers to a choice of a single from each drug type (PI,NRTI,NNRTI). Any-k refers to a choice of any k drugs from among the 15 drugs in the study. The Y axis hows the accuracy in retrieving a drug combination with the best possible outcome. The plot on the left (Radius 1) allows only point mutations while plot on the right (Radius 10) allows the test sequence to mutate up to 10 mutations. MinExp, MinMax and Min Strategies are all derived from our lasso regression model. We observe that for limited budgets (upto 3 drugs) our strategies outperform the competing baseline. This suffers with increasing mutational load however the trend more or less holds.

Appendix B

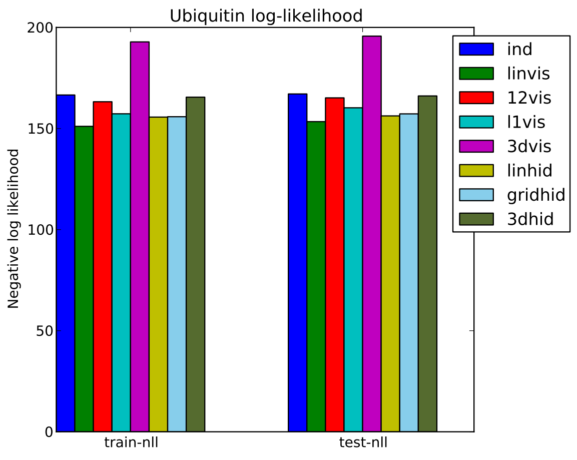
Markov Random Fields

B.1 Results and Discussion

Includes additional results for Markov Random fields.



(a) Test negative log likelihood for ubiquitin



(b) Test negative log likelihood for PDZ

Figure B.1: Negative log likelihood for ubiquitin and PDZ

Figure B.1 shows the negative log likelihood for ubiquitin and PDZ respectively. Similar to imputation error 4.2, tree-structured models such as `linvis` and `l1hid` have the best performance. On the other hand, loopy models such as `l1vis, l2vis` and `3dvis` do poorly. This is likely due to the poor convergence of loopy belief propagation in these graphs. Also note the large difference in training and testing error for loopy graphs showing that the model has overfitted. This suggests that optimizing the log-likelihood directly for loopy graphs is difficult and motivates the need for latent variable models with special structures such as Boltzmann Machines.

Models trained with Log Likelihood and Loopy BP

We report the results for the PDZ family (PF00595) in this section.

Table B.1 shows the train and test imputation error for the PDZ dataset for models trained with log likelihood and loopy belief propagation. All models do better than *ind* (completely independent). This shows that there is contextual information to be captured and most models are able to capture some aspect of it. The *linvis* and *linhid* models do fairly well. This can be attributed to exact inference in tree-structured graphs. They also have similar training and testing error, indicating that the models are not overfitting.

gremvis and *gridhid* are the next best. Surprisingly, *gremvis* which is closely related to the GREMLIN model did not have the best scores overall when compared to *linvis* (0.592 vs 0.567). We attribute this to the fact that GREMLIN internally uses pseudo log-likelihood to train the models. In the following section, we will show that pseudo log-likelihood indeed helps in better inference properties for VMRFs. We chose to use log-likelihood as the objective function to keep our models comparable with latent variable models. HMRFs and Boltzmann Machines (discussed in chapter 6) contain latent variables and cannot use pseudo log-likelihood since conditioning on the other visible variables connects the graph connected via hidden nodes.

Extremely loopy graphs such as *12vis, 3dvis* and *3dhid* perform poorly. They also have a larger difference between train and test error indicating overfitting. We arrived at the conclusion, the 3D topology is not sufficient to provide features for a better generative model using

model-name	**train-imperr**	**test-imperr**
ind	0.7007	0.7019
3dhid	0.6298	0.6315
3dvis	0.6099	0.6178
12vis	0.5966	0.6035
gridhid	0.5874	0.5978
gremvis	0.5823	0.5926
linhid	0.5805	0.5838
linvis	0.5585	0.5672

Table B.1: Imputation error PDZ : Models trained using log likelihood and loopy belief propagation

model-name	**train-imperr**	**test-imperr**
12vis-pseudo	0.5188	0.5378
gremvis-pseudo	0.4657	0.5035
3dvis-pseudo	0.4449	0.4873

Table B.2: Imputation error PDZ : Models trained using pseudo log likelihood

log-likelihood as the objective function. The extra cycles it introduces stymies loopy belief propagation from finding correct beliefs.

Models trained with Pseudo Log Likelihood

Table B.2 shows the train and test imputation error for the PDZ dataset for models trained with pseudo log likelihood. The counterparts of the *12vis, gremvis* and *3dvis* are referred to as *12vis-pseudo*, *gremvis-pseudo* and *3dvis-pseudo*, respectively. We note that these models have much better imputation error scores. This shows that pseudo log likelihood helps in better inference and hence better learning of parameters.

Additionally, the *gremvis-pseudo* model has better scores than *12vis-pseudo* (0.503 vs 0.537).

Similarly, the *3dvis-pseudo* model has better scores than *gremvis-pseudo* (0.487 vs 0.503). This seems to suggest that models with more features (denser edges) tend to have better imputation error scores. Note that the gap between the training and testing error is also widening from *linvis* to *3dvis-pseudo*, indicating that some amount of overfitting might be occuring.

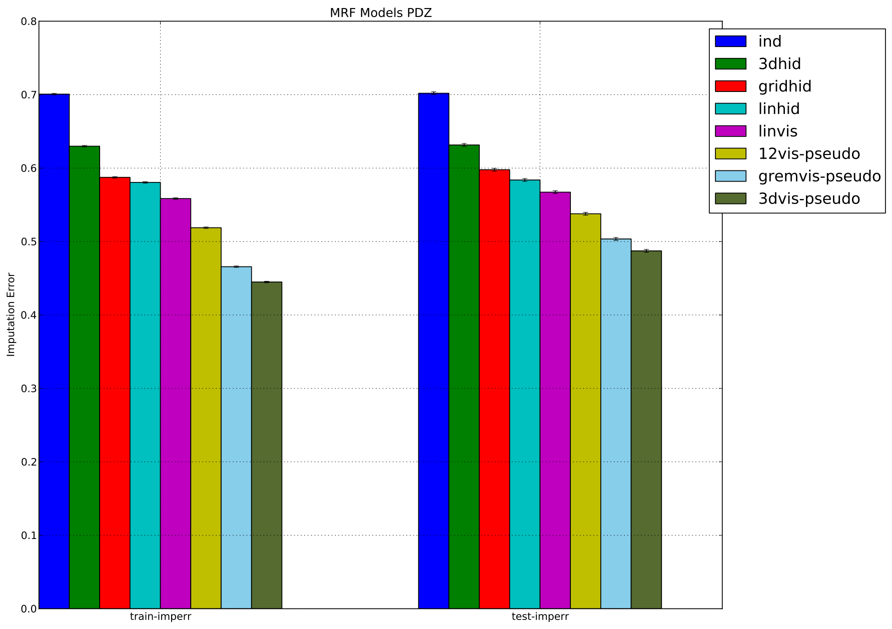


Figure B.2: Imputation error for PDZ for MRF like models

Appendix C

Biological Analysis of GPCR Features

C.1 Methods

C.1.1 Multiple sequence alignment

The authors of the SCA study [81] obtained the class A GPCR alignment from GPCRDB [36] and TinyGRAP [20] databases and manually adjusted the sequences using structure-based sequence alignments. The final MSA has 940 sequences and 348 residue positions covering the entire length of bovine rhodopsin without any gaps. We used this MSA here. As a pre-processing step, we selected the top 1000 candidate edges using a mutual information metric on which the structure learning approach would be subsequently run. This pre-processing step was done purely for computational reasons. Later versions of GREMLIN can avoid this pre-processing step by scaling up to larger sized proteins by parallelizing the computations using a Map-Reduce framework .

C.1.2 Model Selection

GREMLIN uses a single parameter λ which determines the sparsity of the MRF (i.e. the number of edges) and the likelihood of the sequences in the MSA under the model. Higher values of

λ will produce sparser models. In general, a dense graph will yield higher likelihoods than a sparse graph. However, maximizing the likelihood of the MSA is likely to over-fit the data. Thus, the regularization parameter,λ, controls the trade-off between goodness-of-fit to the data and the tendency to over-fit. As in previous work, we used a permutation-based method to select λ. Briefly, we randomly permute the columns of the MSA in order to destroy all correlations between columns while retaining the column-wise distribution of amino acids. We then run GREMLIN on the permutated MSA using different values of λ. The smallest λ yielding zero edges on the permuted MSA is selected. This is a conservative estimate designed to minimize the number of false positive edges. In our experiments the optimal λvalue was 38. We used GREMLIN to learn models from the un-permuted MSA using penalties of **38**, or higher. We consider such edges as the most "robust". The analysis of GPCRs described here is based on these robust edges unless otherwise stated.

C.1.3 GPCR structures files

As of January 2011, there were a total of 43 structures representing seven different GPCRs deposited in the PDB. Only class A GPCRs have been crystallized so far. The GPCRs for which structural information is available are bovine rhodopsin (BR; 18 structures including opsin), squid rhodopsin (SR; 2 structures) turkey $\beta1$ adrenergic receptor ($\beta1$AR; 6 structures), human $\beta2$ adrenergic receptor ($\beta2$AR; 10 structures), human A2A adenosine receptor (A2A; 1 structure), human chemokine receptor CXCR4 (5 structures) and human dopamine D3 receptor (D3R; 1 structure).

C.1.4 Ligand Binding Pockets

The residues in the ligand pocket of the different GPCR crystal structures available to date were defined as those which have at least one atom within $5\AA$ of the respective ligand. Scripts were written to extract residues within a ligand binding pocket using this cut-off distance from crystal structures. We mapped the ligand binding pockets of the different GPCRs onto bovine rhodopsin for comparison. Pair-wise sequence/structure based alignments between rhodopsin (PDB ID: 1U19) and other GPCR structures were generated using the 'salign' module in the MODELLER software. All ligand binding pockets discussed in this paper are mapped onto the structure of bovine rhodopsin.

In addition to comparing ligand binding pockets directly (i.e. extracting $5\AA$ residues in PDB ID: 1F88 for rhodopsin to identify the RT ligand binding pocket), we also generated the following combined sets of pocket residues to investigate similarities and differences between ligand binding pockets of different GPCRs . For each of the 7 GPCRs, we defined a common ligand binding pocket by combining the ligand binding pockets from all available crystal structures for the respective receptor (Table 5.1). Thus, for bovine rhodopsin, the common ligand pocket is the combination of all RT binding pockets of 12 different structures. [Note: Rhodopsin PDBs excluded are 1JFP and 1LN6, because these represent structure models from NMR structures of protein fragments. 2I36, 2I37, 3CAP and 3DQB were also excluded because these are opsin structures and have no RT in them.] In analogous fashion, common pockets were created for squid rhodopsin (SR), turkey 1 adrenergic receptor ($1AR$), human 2 adrenergic receptor (2AR), human A2A adenosine receptor (A2A), human chemokine receptor CXCR4 and human dopamine D3 receptor (D3R).

Finally, to generalize across different GPCRs, we derived additional ligand pockets B1, B2, B3, B4, B5, B6 and B7 representing common sets of residues present in at least one, two, three, four, five, six and seven receptor ligand binding pockets, respectively.

Definition of long-range interactions: A long-range interaction is defined as a statistical coupling between two amino acids that are separated by at least 8 amino acids in the sequence (a definition used in CASP [17]).

C.1.5 Control Set

GREMLIN derived robust edges were checked for statistically over- or under-represented patterns amongst couplings observed. These tests were not done to validate the efficacy of GREMLIN in terms of modelling the protein family, but to get structural and biological insights into the nature of couplings that the model learns. For this purpose we compared the edges that GREMLIN returns against a control distribution of edges. The control distribution is created by drawing edges from a random graph. We classified the edges into one of the following categories: EC-EC, EC-IC, EC-TM, EC-RT, IC-IC, IC-RT, IC-TM, TM-TM, RT-TM and RT-RT. Here, RT stands for the ligand binding domain in rhodopsin (PDB ID: 1F88). To define the control distribution, we enumerated all possible edges coupling any two amino acids in rhodopsin (PDB ID: 1U19) and assigned these edges into the previously defined categories. We defined a control distribution of a category as the probability of randomly picking an edge in that category from the control dataset. To check for statistical significance, we enumerated the edges returned by GREMLIN in each category and compared the fraction of edges in this category against the control distribution. A p-value was calculated by a one-sided binomial test for statistical significance of GREMLIN categories against categories of the control distribution.

C.2 Results and Discussion

C.2.1 Comparison with SCA and GMRC

Since we applied GREMLIN to the same MSA previously studied by the SCA and GMRC methods, we can directly compare the residues predicted by the three methods. There are listed in Table C.1. The GREMLIN residues correspond to those obtained at a penalty of $\lambda = 38$. In the SCA study, the authors focused on K296$^{7.43}$, since this is moderately conserved residue and a key determinant of ligand interaction in GPCRs. The common residues between GREMLIN and SCA forming edges with K296$^{7.43}$ are T93$^{2.60}$, A117$^{3.32}$, G121$^{3.36}$ and F293$^{7.40}$. There are

GREMLIN	SCA	GMRC
M44, L72, N73, G90, T93, G114, A117, G121, W175, Y178, C185, D190, S202, H211, A269, P291, A292, F293	I54, T58, N73, N78, F91, T92, T93, E113, A117, G121, E122, I123, L125, V129, E134, Y136, F148, A164, F212, I213, I219, M257, F261, W265, Y268, F293, F294, A295, S298, A299, N302, F313, M317	L57 - A82, F313 - R314, I305 - Y306, N302 - I304, C264 - A299

Table C.1: **Comparison of edges reported in SCA and GMRC studies with GREMLIN.** Short range edges are italicized while bold residues are common edges between SCA and GREMLIN. Edges from GRMC are not shared by SCA or GREMLIN.

no statistically coupled residues involving K296$^{7.43}$ in the GMRC study (Table C.1). There are only 5 edges in GMRC that are identified to be statistically significant and none of the residues that are identified have any edges in GREMLIN at a penalty of $\lambda = 38$. GMRC also shares no common edges with SCA. Only two out of five edges in GMRC study qualify as long-range and the residues involved (A82$^{2.49}$, C264$^{6.47}$ and A299$^{7.46}$) are strategically located in the middle of TM helices. This might be an artefact of the topology learning heuristic used by GMRC when compared with the other methods. It is important to note that in the GMRC study, the authors considered a sub-class of the original MSA involving only amine (196 sequences), peptide (333 sequences) and rhodopsin (143 sequences) that represents the bulk of the sequences (672 out of a total of 948 sequences).

In the SCA study, the residues statistically coupled to K296$^{7.43}$ were classified further into three classes: (1) *Immediate neighbours* - F293$^{7.40}$, L294$^{7.41}$, A295$^{7.42}$, A299$^{7.46}$, F91$^{2.56}$, E113$^{3.28}$, (2) *Linked network* - F261$^{6.44}$, W265$^{6.48}$, Y268$^{6.51}$, F212$^{5.47}$ and (3) *Sparse but contiguous network* - G121$^{3.36}$, I123$^{3.38}$, L125$^{3.40}$, I219$^{5.54}$, F261$^{6.44}$, S298$^{7.45}$, A299$^{7.46}$, N30267.49. These categories were formulated on mapping the residues onto the rhodopsin structure. Residues in the immediate neighbour category are in the vicinity of K296$^{7.43}$ and are mainly involved in helix packing interactions except for E113$^{3.28}$. E113$^{3.28}$ forms a salt bridge interaction with the protonated Schiff base on K296$^{7.43}$ and is an important interaction identified by SCA. In the GREMLIN model, E113$^{3.28}$ and K296$^{7.43}$ arent connected by an edge, but they do share three common neighbours: M44, L72, and F293, and are thus indirectly correlated.

The linked network residues in SCA are parallel to the membrane and form an aromatic cluster around the β-ionone ring of RT in rhodopsin. The residues in the sparse but contiguous network are distant from K296$^{7.43}$ and form helix packing interactions toward the IC side. The SCA method performs a perturbation on a particular amino acid only if the corresponding sub-alignment size is beyond a certain cutoff in order to calculate $\triangle\triangle G_{stat}$ values. GREMLIN on the other hand makes no such distinction. Hence it is possible that SCA detects edges even if a position is fairly conserved whereas GREMLIN ignores them. This could be a source of difference between GREMLIN and SCA edge couplings. Overall, compared to SCA and GMRC, GREMLIN seems to identify couplings that are more extensive (i.e., involving EC, TM, RT and IC) and are part of experimentally functional switches and structural micro-domains that are critical of activation as discussed above.

Appendix D

Boltzmann Machines

D.1 Learning Rules for Multinomial RBM

Figure 6.2 illustrates a typical RBM. Let $V = \{V_1, ..., V_M\}$ and $H = \{H_1, ..., H_N\}$ be sets of variables comprising the visible/observed and hidden layers of the RBM, respectively. V and H form a bipartite graph. Each variable can take on multiple values: $V_i \in \{1..K\}$ and $H_i \in \{1..L\}$. The energy of a configuration $E(V, H)$ is defined as:

$$E(V = v, H = h; \theta = \{W, b, c\}) = -\sum_{i=1}^{M} b_i^{v_i} - \sum_{j=1}^{N} c_j^{h_j} - \sum_{i,j} W_{ij}^{v_i h_j} \tag{D.1}$$

Where the parameters are $\theta = \{W, b, c\}$. The distribution factorizes according to the Boltzmann distribution:

$$p(v, h) = \frac{\exp^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}$$

Note that not all parameters are free; some parameters must be zero to keep the model identifiable. Without loss of generality, let $b^K = c^k = W_{ij}^{K \cdot} = W_{ij}^{\cdot L} = 0$. Also note that the structure of the RBM implies the following conditional factorization: $P(h|v) = \prod_{j=1}^{N} P(h_j|v)$. This conditional independence lets us write the conditional probability as:

$$P(h_j|v) = \frac{e^{c_j^{h_j} + \sum_i W_{ij}^{v_i h_j}}}{\sum_{h_j} e^{c_j^{h_j} + \sum_i W_{ij}^{v_i h_j}}}$$

which is known as the "softmax function". The softmax has been successfully used in the deep learning community in the past in several contexts such as in language models. Further, [90] shows that RBM model factorizes cleanly for any exponential distribution.

Training the softmax RBM is done via Contrastive Divergence (CD), which is a form of approximate gradient descent. Specifically, we wish to minimize the negative log-likelihood cost function $-\log(P(v))$. Define the "Free Energy" of a visible sample as $F(v) = -\log \sum_h e^{-E(v,h)} = -\log Z_v$. Then the gradient of the cost function can be written as:

$$\frac{-\partial \log P(v)}{\partial \theta} = E_{\tilde{v}}\left[\frac{\partial F(\tilde{v})}{\partial \theta}\right] - \frac{\partial F(v)}{\partial \theta}$$

This results in the following update rules for the parameters θ:

$$\begin{aligned}
\frac{-\partial \log P(v)}{\partial W_{ij}^{kl}} &= E_{\tilde{v}}\left[P(h_j = l|\tilde{v})1_{\tilde{v_i}=k}\right] - P(h_j = l|v)1_{v_i} = k \\
&= E_{\tilde{v}}\left[\text{softmax}(h_j = l|\tilde{v})1_{\tilde{v}_i=k}\right] - \text{softmax}(h_j = l|v)1_{v_i=k} \\
\frac{-\partial \log P(v)}{\partial c_j^l} &= E_{\tilde{v}}\left[\text{softmax}(h_j = l|\tilde{v})\right] - \text{softmax}(h_j = l|v) \\
\frac{-\partial \log P(v)}{\partial b_j^k} &= E_{\tilde{v}}\left[1_{\tilde{v}_i=k}\right] - 1_{v_i=k}
\end{aligned}$$

where 1_x is the indicator variable for condition x and $\tilde{v}$ is a sample from the model.

D.2 Experiments and Results

We report the imputation errors calculated on the training and the test sets in Table D.1. We also list the MRF results from section 4.4 for comparison. The results show that both Boltzmann Machines outperform the next-best models by around 10% points.

Sparse Boltzmann Machines

In this section we discuss the results from the next set of experiments pertaining to the Sparse Boltzmann Machines. We discuss evaluation metrics for sparse Boltzmann machines boot-

model-name	**train-imperr**	**test-imperr**
ind	0.7007	0.7019
linhid	0.5805	0.5838
linvis	0.5585	0.5672
12vis-pseudo	0.5188	0.5378
gremvis-pseudo	0.4657	0.5035
3dvis-pseudo	0.4449	0.4873
rbm-hyp-scan	0.1736	0.3813
dbm-hyp-scan	0.2690	0.3843

Table D.1: Imputation error PDZ PF00595

Model	**train-imperr**	**test-imperr**
rbm-50	0.5288	0.5386
rbm-100	0.4827	0.5015
rbm-500	0.4681	0.4908
rbm-1000	0.4577	0.4849
rbm-1701	0.4456	0.4719
vbm	0.1143	0.4127
dbm-1701-50	0.3724	0.4318
dbm-1701-100	0.3770	0.4343

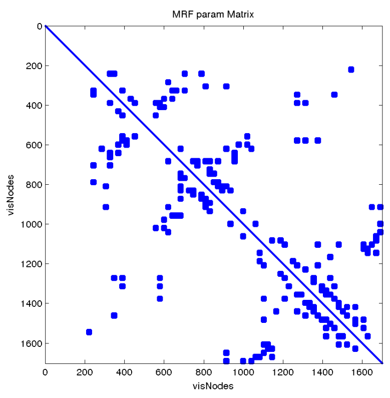
Table D.2: Imputation error PDZ RBM and DBM

strapped from MRFs containing sparse edges.

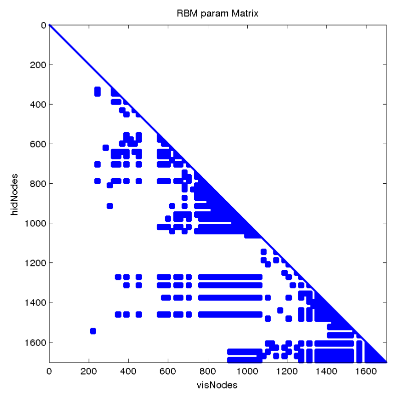
Table D.2 shows the imputation errors calculated for baseline models of RBMs and DBMs. The DBM model *dbm-1701-50* has the best test error.

Bootstrap from Long Range Interactions

This section discusses the results from long range interactions MRF bootstrap. Figure D.1 shows the edges for the PDZ family and subsequent mapping to a RBM. Note the absence of any edges



(a) PDZ MRF param matrix with long range edges



(b) SRBM bootstrapped from PDZ MRF param matrix with long range edges

Figure D.1: Cholesky decomposition of PDZ MRF param matrix with long range edges

Model	**train-imperr**	**test-imperr**
srbm-1701	0.5706	0.5826
10-ssrbm-1701	0.5404	0.5533
50-ssrbm-1701	0.5323	0.5448
100-ssrbm-1701	0.5309	0.5431
500-ssrbm-1701	0.4969	0.5138

Table D.3: Imputation error PDZ SRBM and SSRBM (long edges)

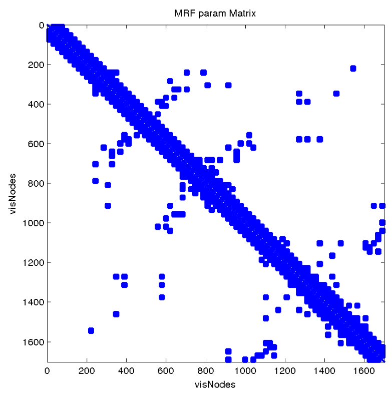
within three residue positions.

Table D.3 shows imputation errors for SRBM and SSRBM models. The number of additional hidden nodes in the SSRBM layer are varied.

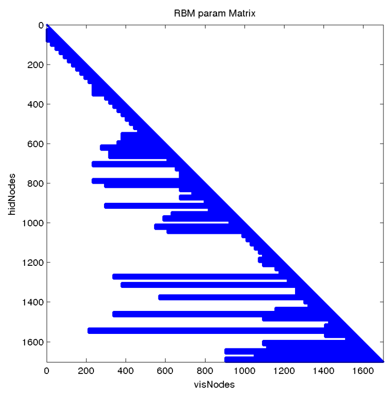
Table D.4 shows the imputation errors for the LC-DBM model when varying the number of hidden nodes in the second hidden layer.

Model	**train-imperr**	**test-imperr**
srbm-1701	0.5706	0.5826
lcdbm-1701-50	0.5801	0.5992
lcdbm-1701-100	0.5663	0.5852
lcdbm-1701-250	0.5543	0.5725
lcdbm-1701-500	0.5635	0.5834
lcdbm-1701-1000	0.5597	0.5797

Table D.4: Imputation error PDZ SRBM and LC-DBM (long edges)



(a) PDZ MRF param matrix with short and long range edges



(b) SRBM bootstrapped from PDZ MRF param matrix with short and long range edges

Figure D.2: Cholesky decomposition of PDZ MRF param matrix with short and long range edges

Bootstrap from Short Plus Long Range Interactions

This section discusses the results for sparse Boltzmann machines bootstrapped from short plus long range edges. Figure D.2 shows the edges for the PDZ family and subsequent mapping to a RBM. The RBM has many more parameters than the long range bootstrapped RBM (figure D.1).

Table D.5 shows imputation errors for SRBM and SSRBM models. The number of additional hidden nodes in the SSRBM layer are varied.

Table D.6 shows the imputation errors for the LC-DBM model when varying the number of

Model	**train-imperr**	**test-imperr**
srbm-1701	0.5001	0.5203
10-ssrbm-1701	0.4916	0.5118
50-ssrbm-1701	0.4907	0.5132
100-ssrbm-1701	0.4872	0.5085
500-ssrbm-1701	0.4832	0.5056

Table D.5: Imputation error PDZ SRBM and SSRBM (short plus long edges)

Model	**train-imperr**	**test-imperr**
srbm-1701	0.5001	0.5203
lcdbm-1701-50	0.5097	0.5397
lcdbm-1701-100	0.5050	0.5348
lcdbm-1701-250	0.4968	0.5262
lcdbm-1701-500	0.4993	0.5299
lcdbm-1701-1000	0.4979	0.5288

Table D.6: Imputation error PDZ SRBM and LC-DBM (short plus long edges)

hidden nodes in the second hidden layer.

D.2.1 Summary of all Generative Models

In this section we provide a summary of all MRFs and Boltzmann Machines for the PDZ family. Figure D.3 shows a summary of the MRF models discussed in chapter 4 and the different Boltzmann machines discussed in this chapter for the PDZ family.

D.2.2 Additional Results Large Scale Imputation Errors

Additional results from largescale experiments holding out 10 columns at a time are shown in figures D.8, D.4, D.6, D.7 and D.5 respectively.

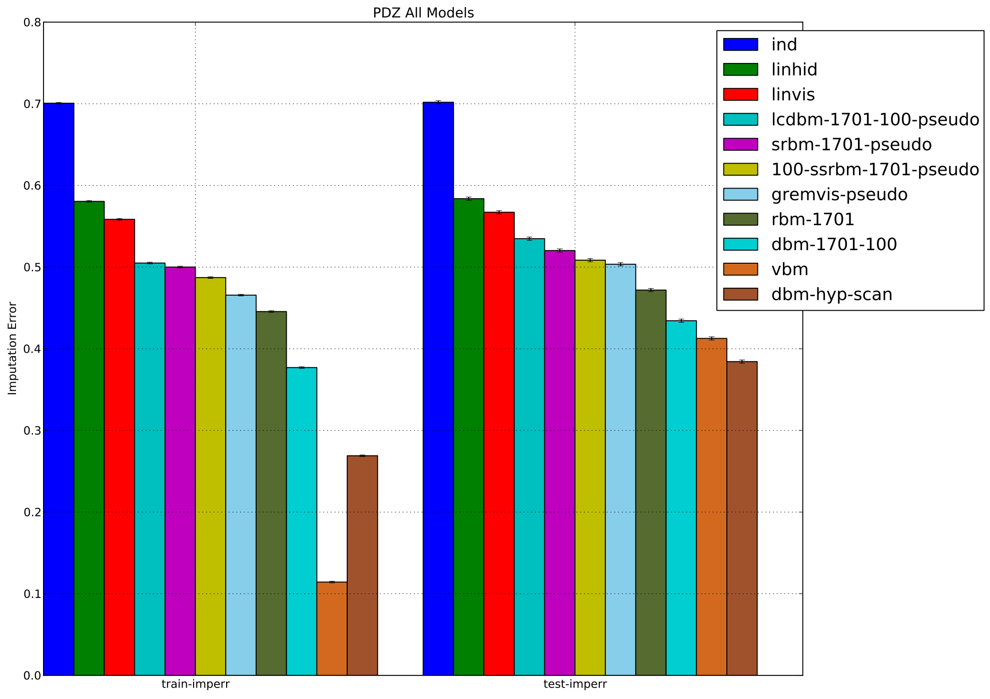


Figure D.3: Comparing MRFs and Boltzmann Machines for PDZ: Imputation error improves with relaxations in model assumptions and addition of latent variables

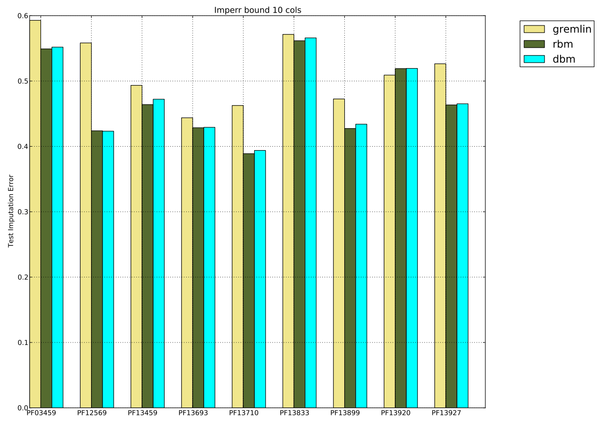


Figure D.4: Largescale Bound Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 10 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

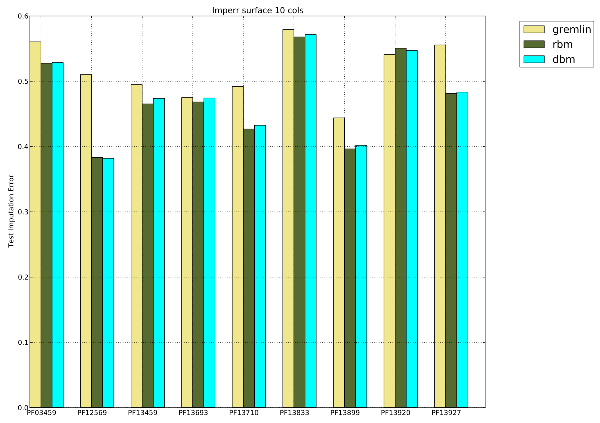


Figure D.5: Largescale surface Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 10 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

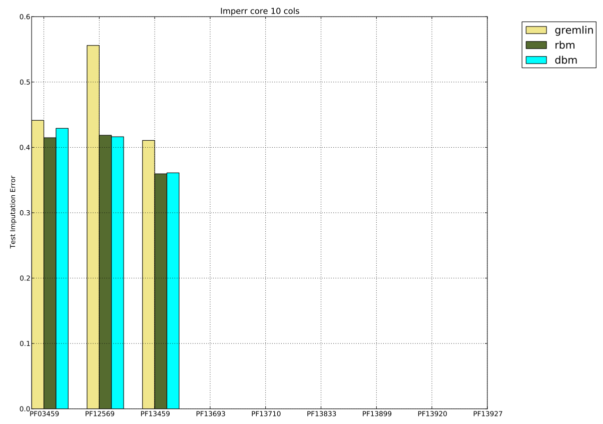


Figure D.6: Largescale core Residues Multicolumn Imputations: Multicolumn imputations for all datasets holding out 10 columns using binding residues. Boltzmann machines beat Gremlin by about 5 percentage points

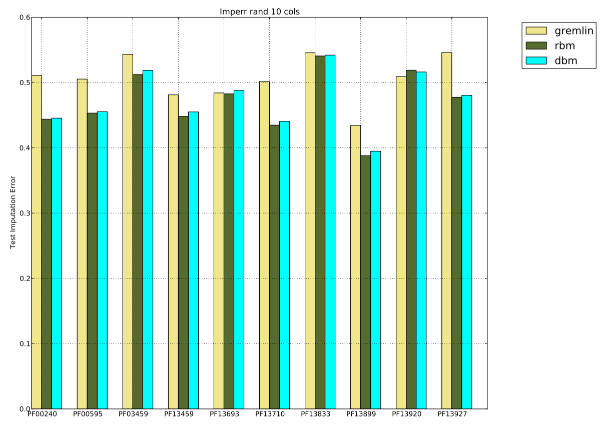


Figure D.7: Largescale Random Multicolumn Imputations: Multicolumn imputations for all datasets holding out 10 columns chosen at random. Boltzmann machines beat Gremlin by about 5 percentage points

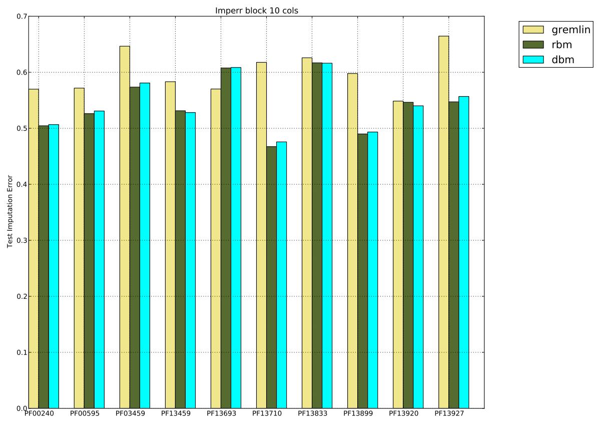


Figure D.8: Largescale Block Multicolumn Imputations: Multicolumn imputations for all datasets holding out 10 adjacent columns in a block and sweeping the MSA. Boltzmann machines beat Gremlin by about 5 percentage points

Bibliography

[1] S. Ahuja and S. O. Smith. Multiple switches in G protein-coupled receptor activation. *Trends Pharmacol. Sci.*, 30(9):494–502, Sep 2009. 5.1

[2] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997. 2.2.2

[3] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996. 6.4.2

[4] S. Balakrishnan, H. Kamisetty, J.C. Carbonell, S.I. Lee, and Langmead C.J. Learning Generative Models for Protein Fold Families. *Proteins: Structure, Function, and Bioinformatics*, 79(6):1061?1078, 2011. 1, 2.4, 4, 4.2.5, 5, 5.1, 5.2.1, 6, 6.1.3, 6.4, 6.4.2

[5] J. A. Ballesteros, L. Shi, and J. A. Javitch. Structural mimicry in G protein-coupled receptors: implications of the high-resolution structure of rhodopsin for structure-function analysis of rhodopsin-like receptors. *Mol. Pharmacol.*, 60(1):1–19, Jul 2001. 5.1

[6] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 4, 6.1

[7] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36. Springer, 2011. 6.3

[8] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007. 6.3.1

[9] Julian Besag. Statistical analysis of non-lattice data. *The statistician*, pages 179–195, 1975. 4.2.3, 6.1.3

[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 6.4.2

[11] Adrian A Canutescu, Andrew A Shelenkov, and Roland L Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein science*, 12(9):2001–2014, 2003. 7.2

[12] KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Parallel tempering is efficient for learning restricted boltzmann machines. In *IJCNN*, pages 1–8. Citeseer, 2010. 6.2

[13] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011. 6.2

[14] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008. 4

[15] UniProt Consortium et al. The universal protein resource (uniprot). *Nucleic acids research*, 36(suppl 1):D190–D195, 2008. 2.2.2, 4.4.1, 6.6.1

[16] Timothy A Davis and William W Hager. Modifying a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(3):606–627, 1999. 6.4.2, 6.4.2

[17] J. P. Dekker, A. Fodor, R. W. Aldrich, and G. Yellen. A perturbation-based method for calculating explicit likelihood of evolutionary co-variance in multiple sequence alignments. *Bioinformatics*, 20(10):1565–1572, Jul 2004. 5.1, C.1.4

[18] Ken A Dill and Justin L MacCallum. The protein-folding problem, 50 years on. *Science*, 338(6110):1042–1046, 2012. 3.2.1

[19] Sean R Eddy. Where did the blosum62 alignment score matrix come from? *Nature biotechnology*, 22(8):1035–1036, 2004. 6.6.6

[20] Øyvind Edvardsen, Anne Lise Reiersen, Margot W Beukers, and Kurt Kristiansen. tgrap, the g-protein coupled receptors mutant database. *Nucleic acids research*, 30(1):361–363, 2002. C.1.1

[21] SM Ali Eslami, Nicolas Heess, Christopher KI Williams, and John Winn. The shape boltzmann machine: a strong model of object shape. *International Journal of Computer Vision*, 107(2):155–176, 2014. 6.4, 6.4.1

[22] R. D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. Sonnhammer, S. R. Eddy, and A. Bateman. The Pfam protein families database. *Nucleic Acids Res.*, 38(Database issue):D211–222, Jan 2010. 5.1

[23] R. Fredriksson, M. C. Lagerstrom, L. G. Lundin, and H. B. Schioth. The G-protein-coupled receptors in the human genome form five main families. Phylogenetic analysis, paralogon groups, and fingerprints. *Mol. Pharmacol.*, 63(6):1256–1272, Jun 2003. 5.1

[24] Yoav Freund and David Haussler. *Unsupervised learning of distributions of binary vectors using two layer networks*. Computer Research Laboratory [University of California, Santa Cruz], 1994. 6.2, 6.3

[25] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. 6.4

[26] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010. 3.2.3

[27] Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 548–556, 2013. 6.3.1

[28] Pinar Heggernes, SC Eisestat, Gary Kumfert, and Alex Pothen. The computational complexity of the minimum degree algorithm. Technical report, DTIC Document, 2001. 6.4.2

[29] T. Hinkley, J. Martins, C. Chappey, M. Haddad, E. Stawiski, J. M. Whitcomb, C. J. Petropoulos, and S. Bonhoeffer. A systems analysis of mutational effects in HIV-1 protease and reverse transcriptase. *Nat. Genet.*, 43(5):487–489, May 2011. 3.1, 3.2.3, 3.3.2, 3.4, 7.1, A.1.1

[30] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL `http://dx.doi.org/10.1162/089976602760128018`. 6.2.1, 6.6.2, 6.6.6

[31] Geoffrey E Hinton and Terrance J Sejnowski. Learning and relearning in boltzmann machines. *Cambridge, MA: MIT Press*, 1:282–317, 1986. 6.1

[32] Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 448–453. IEEE New York, 1983. 6.1

[33] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL `http://dx.doi.org/10.1162/neco.2006.18.7.1527`. 4, 6.2, 6.3, 6.3.1

[34] Georey Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010. URL `http://www.cs.toronto.edu/~{}hinton/absps/guideTR.pdf`. 6.2.1, 6.5, 6.6.2

[35] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982. 6.1

[36] Florence Horn, J Weare, Margot W. Beukers, S Hörsch, Amos Bairoch, W Chen, Øyvind Edvardsen, Fabien Campagne, and Gert Vriend. Gpcrdb: an information system for g protein-coupled receptors. *Nucleic Acids Research*, 26(1):275–279, 1998. C.1.1

[37] J. Hwa, P. Garriga, X. Liu, and H. G. Khorana. Structure and function in rhodopsin: packing of the helices in the transmembrane domain and folding to a tertiary structure in the intradiscal domain are coupled. *Proc. Natl. Acad. Sci. U.S.A.*, 94(20):10571–10576, Sep 1997. 5.1

[38] Alexander T Ihler, John Iii, and Alan S Willsky. Loopy belief propagation: Convergence and effects of message errors. In *Journal of Machine Learning Research*, pages 905–936, 2005. 6, 6.2.1

[39] Hetunandan Kamisetty, Arvind Ramanathan, Chris Bailey-Kellogg, and Christopher James Langmead. Accounting for conformational entropy in predicting binding free energies of protein-protein interactions. *Proteins: Structure, Function, and Bioinformatics*, 79(2):444–462, 2011. 7.2

[40] Hetunandan Kamisetty, Eric P Xing, and Christopher J Langmead. Approximating correlated equilibria using relaxations on the marginal polytope. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1153–1160, 2011. 3.1

[41] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence-and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, 2013. 3.2.1, 3.4, 4, 6.4.2

[42] Kevin Karplus, Kimmen Sjlander, Christian Barrett, Melissa Cline, David Haussler,

Richard Hughey, Liisa Holm, Chris Sander, Ebi England, and Ebi England. Predicting protein structure using hidden markov models. In *Proteins: Structure, Function, and Genetics*, pages 134–139, 1997. 1, 4

[43] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998. 4.3, 6

[44] J. Klein-Seetharaman. Dual role of interactions between membranous and soluble portions of helical membrane receptors for folding and signaling. *Trends Pharmacol. Sci.*, 26(4): 183–189, Apr 2005. 5.1

[45] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 4.2.3, 4.3.4

[46] R. D. Kouyos, V. von Wyl, T. Hinkley, C. J. Petropoulos, M. Haddad, J. M. Whitcomb, J. Boni, S. Yerly, C. Cellerai, T. Klimkait, H. F. Gunthard, and S. Bonhoeffer. Assessing predicted HIV-1 replicative capacity in a clinical setting. *PLoS Pathog.*, 7(11):e1002321, Nov 2011. 3.1, 3.2.1

[47] R. D. Kouyos, G. E. Leventhal, T. Hinkley, M. Haddad, J. M. Whitcomb, C. J. Petropoulos, and S. Bonhoeffer. Exploring the complexity of the HIV-1 fitness landscape. *PLoS Genet.*, 8(3):e1002551, 2012. 3.1

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 7.2

[49] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjlander, and David Haussler. Hidden markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994. 1

[50] Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 2010. 3.1

[51] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008. 4, 6.2

[52] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008. 6.3

[53] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006. 6.1

[54] Yann LeCun et al. Generalization and network design strategies. *Connections in Perspective. North-Holland, Amsterdam*, pages 143–55, 1989. 6.4.1

[55] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *NIPS*, volume 7, pages 873–880, 2007. 6.4.2

[56] S. W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–299, Oct 1999. 4.1, 5.1

[57] Alireza Makhzani and Brendan Frey. k-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013. 6.4.2

[58] James Martens and Ilya Sutskever. Parallelizable sampling of markov random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 517–524, 2010. 6.4.2

[59] Volodymyr Mnih. Cudamat: a cuda-based matrix class for python. *Department of Computer Science, University of Toronto, Tech. Rep. UTML TR*, 4, 2009. 6.6.2

[60] Subhodeep Moitra, Kalyan C Tirupula, Judith Klein-Seetharaman, and Christopher J Langmead. A minimal ligand binding pocket within a network of correlated mutations identified by multiple sequence and structural analysis of g protein coupled receptors. *BMC biophysics*, 5(1):13, 2012. 5, 5.1, 5.3

[61] Grégoire Montavon and Klaus-Robert Müller. Deep boltzmann machines and the centering

trick. In *Neural Networks: Tricks of the Trade*, pages 621–637. Springer, 2012. 6.4

[62] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 4.3

[63] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999. 4.2.2, 4.3.2

[64] J. P. Overington, B. Al-Lazikani, and A. L. Hopkins. How many drug targets are there? *Nat Rev Drug Discov*, 5(12):993–996, Dec 2006. 5.1

[65] K. Palczewski, T. Kumasaka, T. Hori, C. A. Behnke, H. Motoshima, B. A. Fox, I. Le Trong, D. C. Teller, T. Okada, R. E. Stenkamp, M. Yamamoto, and M. Miyano. Crystal structure of rhodopsin: A G protein-coupled receptor. *Science*, 289(5480):739–745, Aug 2000. 5.1

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011. 3.2.2

[67] Christos Petropoulos. Data management plan for monogram bioscience. Nature Precedings, 2011. URL `http://dx.doi.org/10.1038/npre.2011.5668.1`. 3.1, 3.3.1, 3.4, A.1.1

[68] A. J. Rader, G. Anderson, B. Isin, H. G. Khorana, I. Bahar, and J. Klein-Seetharaman. Identification of core amino acids stabilizing rhodopsin. *Proc. Natl. Acad. Sci. U.S.A.*, 101 (19):7246–7251, May 2004. 5.1

[69] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature methods*, 9(2):173–175, 2012. 4.4.1, 6.6.1

[70] Ruslan Salakhutdinov. Learning and evaluating boltzmann machines. Technical report,

Technical Report UTML TR 2008-002, Department of Computer Science, University of Toronto, 2008. 6.1.2

[71] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009. 6.2, 6.3, 6.3.1, 6.3.1

[72] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010. 6.3

[73] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007. 6.2

[74] Mark Schmidt. Ugm: Matlab code for undirected graphical models, 2008. 4.4.2

[75] Mark Schmidt. *Graphical model structure learning with l1-regularization*. PhD thesis, UNIVERSITY OF BRITISH COLUMBIA (Vancouver, 2010. 6.4

[76] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. 1986. 6.2

[77] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239, 2012. 6.3, 7.2

[78] Nitish Srivastava, Ruslan R Salakhutdinov, and Geoffrey E Hinton. Modeling documents with deep boltzmann machines. *arXiv preprint arXiv:1309.6865*, 2013. 6.3

[79] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 6.5, 6.6.2

[80] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the perceptronalgorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids*

Research, 10(9):2997–3011, 1982. 6

[81] G. M. Suel, S. W. Lockless, M. A. Wall, and R. Ranganathan. Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nat. Struct. Biol.*, 10 (1):59–69, Jan 2003. 5.1, C.1.1

[82] Shigeki Takeda, Shiro Kadowaki, Tatsuya Haga, Hirotomo Takaesu, and Shigeki Mitaku. Identification of g protein-coupled receptor genes from the human genome sequence. {*FEBS*} *Letters*, 520(13):97 – 101, 2002. ISSN 0014-5793. doi: http://dx.doi.org/10.1016/S0014-5793(02)02775-8. URL `http://www.sciencedirect.com/science/article/pii/S0014579302027758`. 5.1

[83] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE/ACM Trans Comput Biol Bioinform*, 5(2):183–197, 2008. 5.1

[84] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Protein design by sampling an undirected graphical model of residue constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(3):506–516, 2009. 1, 4

[85] Julie D Thompson, Toby Gibson, Des G Higgins, et al. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3, 2002. 6.6.6

[86] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 3.1, 3.2.1, 3.2.2

[87] Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012. 3.1, 3.2.3, 3.4, 7.1

[88] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008. 6.3.1, 6.3.1, 6.6.2

[89] Yuhao Wang and Jianyang Zeng. Predicting drug-target interactions using restricted boltzmann machines. *Bioinformatics*, 29(13):i126–i134, 2013. 6.2

[90] Max Welling, Michal Rosen-Zvi, and Geoffrey E Hinton. Exponential family harmoniums with an application to information retrieval. In *Nips*, volume 17, pages 1481–1488, 2004. D.1

[91] Timothy A Whitehead, Aaron Chevalier, Yifan Song, Cyrille Dreyfus, Sarel J Fleishman, Cecilia De Mattos, Chris A Myers, Hetunandan Kamisetty, Patrick Blair, Ian A Wilson, et al. Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing. *Nature biotechnology*, 30(6):543–548, 2012. 1, 4

[92] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969. 4.4.2, 6.6.2

[93] Chen Yanover and Yair Weiss. Approximate inference and protein-folding. In *Advances in neural information processing systems*, pages 1457–1464, 2002. 7.2

[94] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997. 4.4.2, 6.6.2