Thesis Proposal
Feature Learning and Graphical Models for Protein Sequences

Subhodeep Moitra

Apr 29, 2014

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dr Christopher Langmead, Chair
Dr Jaime Carbonell
Dr Bhiksha Raj
Dr Hetunandan Kamisetty

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

Abstract

Machine learning methods rely heavily on using and learning good features. We study three problems in the context of protein sequences - (1) drug cocktail design (2) studying allostery in GPCRs and (3) generative modeling of protein families. We show that the core challenges underlying each of these tasks relates to effective feature selection, feature interpretation and feature learning, respectively. We address the drug cocktail design problem by providing solutions for feature selection in the context of large scale data. We employ structure learning in markov random fields and interpret the features learned from a biological perspective for studying allostery in GPCRs.

We investigate deep architectures for unsupervised feature learning of latent representations in protein families. We show preliminary results using Restricted Boltzmann Machines (RBMs). We propose to build a deep architecture from RBMs using Deep Boltzmann Machines (DBMs). Additionally, we propose Locally Connected Deep Boltzmann Machines (LC-DBMs) employing sparse structure learning for trading off model agnosticism with prior knowledge.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning methods rely heavily on using and learning good features. From predictive tasks such as regression and classification to descriptive ones such as clustering and probability density estimation ; the importance of high-quality features is incontrovertible. Without good features even the most advanced machine learning algorithms are incapable of being useful. Simpler algorithms often outperform their advanced counterparts when given better features. In fact, the bulk of engineering effort in practice is to design and fine tune data pre-processing pipelines.

The importance of good features holds true across application domains and is especially true in computational biology. With the advent of whole genome sequencing and personalized medicine, we are being deluged by vast amounts of data. This data includes raw sequence reads, protein crystal structures, epigenetic effects, etc. It is not practical nor feasible to rely completely on the expert opinions of biologists for selecting or hand engineering features.

In this thesis, we study three problems in the context of protein sequences - (1) drug cocktail design (2) studying allostery in GPCRs and (3) modeling protein fold families. We show that the core challenges underlying each of these tasks relates to effective *feature selection*, *feature interpretation* and *feature learning*.

In chapter 2, we tackle 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 3, 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 4, we study the problem of unsupervised feature learning. We attempt to address the task of learning a generative model for protein families with a rich feature representation. A good representation can be understood as one that models the posterior distribution of the latent factors for an observed input. We focus on deep architectures which comprise of units of non-linear transformations of data arranged hierarchically in layers. We hypothesise that deep architectures are a good candidate for modeling fold families given the complex factors underlying the evolutionary processes in proteins. We employ energy based graphical models for unsupervised feature learning of latent representations in protein families. We show preliminary results with Restricted Boltzmann Machines (RBMs). In chapter 5, We propose to build a deep architecture from RBMs using Deep Boltzmann Machines (DBMs). Additionally, we propose Locally Connected Deep Boltzmann Machines (LC-DBMs) employing sparse structure learning for trading off model agnosticism with prior knowledge.

Chapter 2

Drug Cocktail Design

In this chapter, we consider the problem of drug cocktail design against HIV-1 infection. e 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. We accomplish this by employing several feature reduction strategies.

2.1 Background

HIV-1 (Human Immunodeficiency virus) is a retrovirus that causes AIDS in humans. HIV-1 is a more virulent and infectious strain of HIV. It infects only humans though it is likely to have originated by cross-species transmission from chimpanzees. 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. The function of the reverse transcriptase is to convert the RNA into DNA for insertion into the human genome. Protease becomes active in the later stages of the viral replicative cycle and is responsible for cleaving the newly budded HIV-1 virion. Figure 2.1 and figure 2.2 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 - Protease Inhibitors (PI), Nucleoside Reverse Transcriptase Inhibitors (NRTI) and 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 the age of high throughput sequencing, viral genomes can be rapidly sequenced from the blood serum of the infected patient. This allows us to specify a regimen of drug treatment that personalizes the treatment for the patient. Only drugs that are necessary against the viral load present in the patient should be administered. The key resource needed for any such drug cocktail design scheme is a model of the viral sequence-fitness landscape. We work with a HIV-1 sequence dataset [40] 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, Bonhoeffer et al. [12] 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 [55] had but were unable to scale because of the size of the dataset. For validation they report the %-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 [24, 25] 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 [56]. 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 [21]. We are able to report the RMSE and this provides a concrete measurement of the predictive utility of our method. An advantage of using a lasso regression model is that the explanatory features are sparse which allows for interpretability. We further validate our model using a KL-divergence based metric to compare the distributions between actual and predicted RC values.

Finally, 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. [20] 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 [27]. We find that our cocktail design method compares favorably against a variety of competing baseline strategies under a limited budget setting.

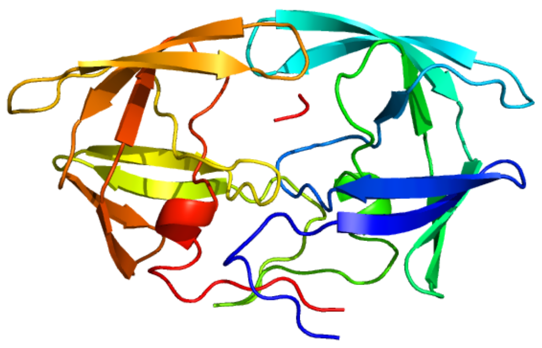


Figure 2.1: HIV-1 Protease

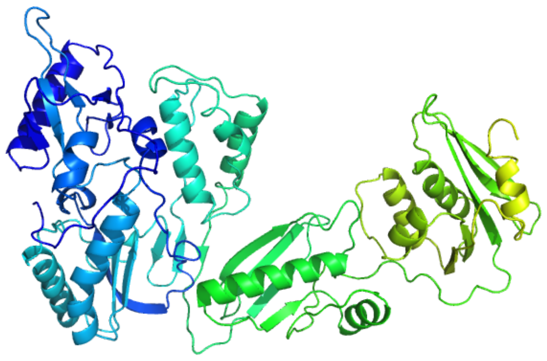


Figure 2.2: HIV-1 Reverse Transcriptase

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

2.2 Methods

In this section we describe the methods used in this study. We detail the RC value tranformation using Gremlin; lasso regression; the various feature reduction schemes involving marginal regression, El Ghaoui and strong rules; sparse matrices ; the drug cocktail design strategy. The experiments and results are discussed in 2.3.

2.2.1 Gremlin Contact Map

The RC values are approximately proportional to a monontonic function of the concentration of viral particles in the patient [24]. 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.

Gremlin [21] is an accurate method to predict the contact map of a protein based on just the sequence alignment of the protein. This allows us to leverage Gremlin to be used as an independent validation technique for selecting amongst candidate monotonic function transforms. We can create a weighted alignment of protein sequences in which abundance of a particular protein sequence is proportional to its abundance in-vivo. This alignment can subsequently be used to predict the contact map of the protein using Gremlin. A high contact map accuracy would indicate that the alignment is of high quality. This arises due of the fact that statistical correlations in the alignment are more pronounced when two amino acids interact in three dimensional space. Extending this argument, one can choose amongst from a set of feature transformations by comparing their contact map accuracies. We successfully use this approach to choose ”square root” as the appropriate monotonic transform for the RC data.

2.2.2 Lasso Regression

Lasso is a shrinkage regression and selection model [55]. 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 [39] since it supports sparse design matrices.

2.2.3 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 [12]. We restrict the possible single and pairwise amino acid features to the ones that appeared atleast 10 times in the dataset. This shrinks the possible set of features from 517660 to 35900046.

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 $\{i < j \in (1, \ldots p) : s_1, s_2, \ldots, s_p, s_{ij}\}$. 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 [10] 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 [56] was proposed by Tibshirani et al. as a techinique 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 2.2.3). 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.

2.2.4 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)

2.3 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.

2.3.1 Data description

The protein sequence dataset was obtained from Monogram Inc [40]. 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.

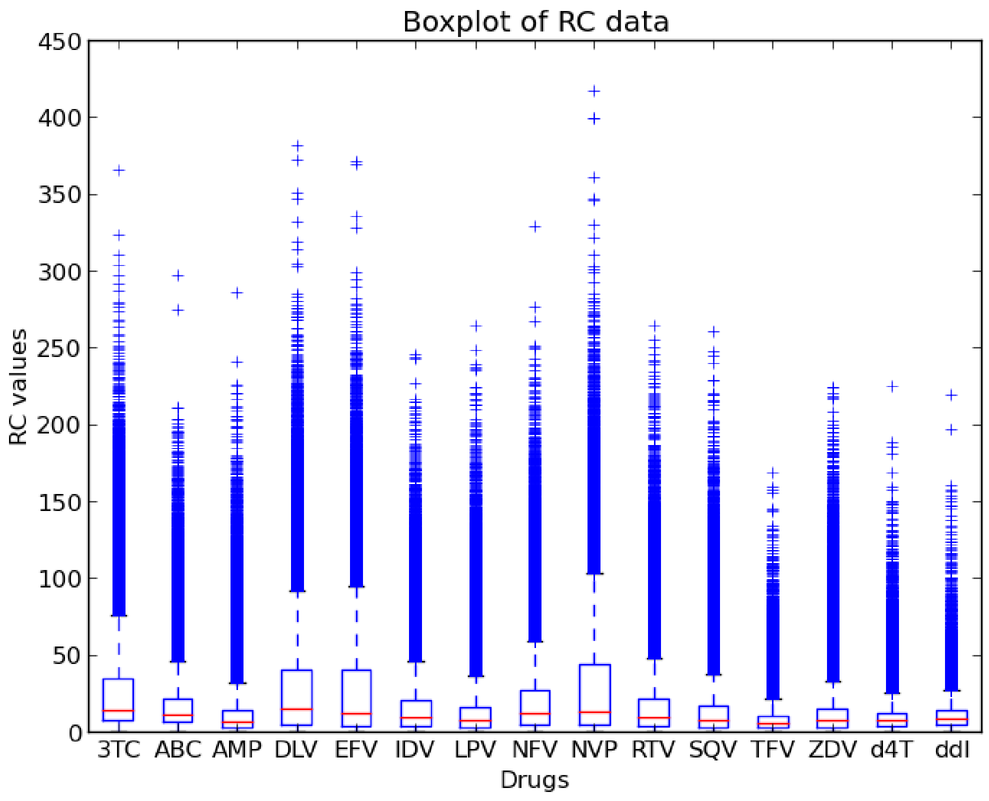


Figure 2.4: A boxplot of the RC data distribution

2.3.2 Feature Transformation - Contact Map Analysis

The motivation and methods for the feature transformation scheme is described in section 2.2.1. Figure 2.5 shows the contact map precision after applying gremlin to a 10K protease alignment. The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI). The X axis lists all the monontonic function transformations used (norc - no weighting, raw-untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \dots 10\}$, $\log$ and $\log\log$). 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. Cube root is better in some cases. 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 [12] 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.

2.3.3 Feature Reduction Strategies

The various feature reduction strategies as described in section 2.2.3 were applied to a 20K sequence validation dataset. The validation RMSE from these filtering schemes is plotted in figure 2.6, figure 2.7 and figure A.11 for protease inhibitors (PI), NRTIs and NNRTI drugs respectively. An interesting result falls out from this analysis. For drug class of PIs , the regression model trained on protease does better (bluer) than RT. Similarly, for the drug class of NRTI and NNRTI, the regression model trained on RT does better than protease. This adds further confidence that our regression model is capturing the signal in the dataset. The same results are also shown as bar plots in appendix figure A.12, figure A.13 and figure A.14.

The `s20` feature selection created by applying strong rules [56] to curate a set of 20K features does the best when compared to all the other strategies. Based on this evidence, we pre-select the `s20` features from which lasso subsequently learns a regression model. This leads to considerable speed and memory savings and is instrumental in helping us to scale up our regression models.

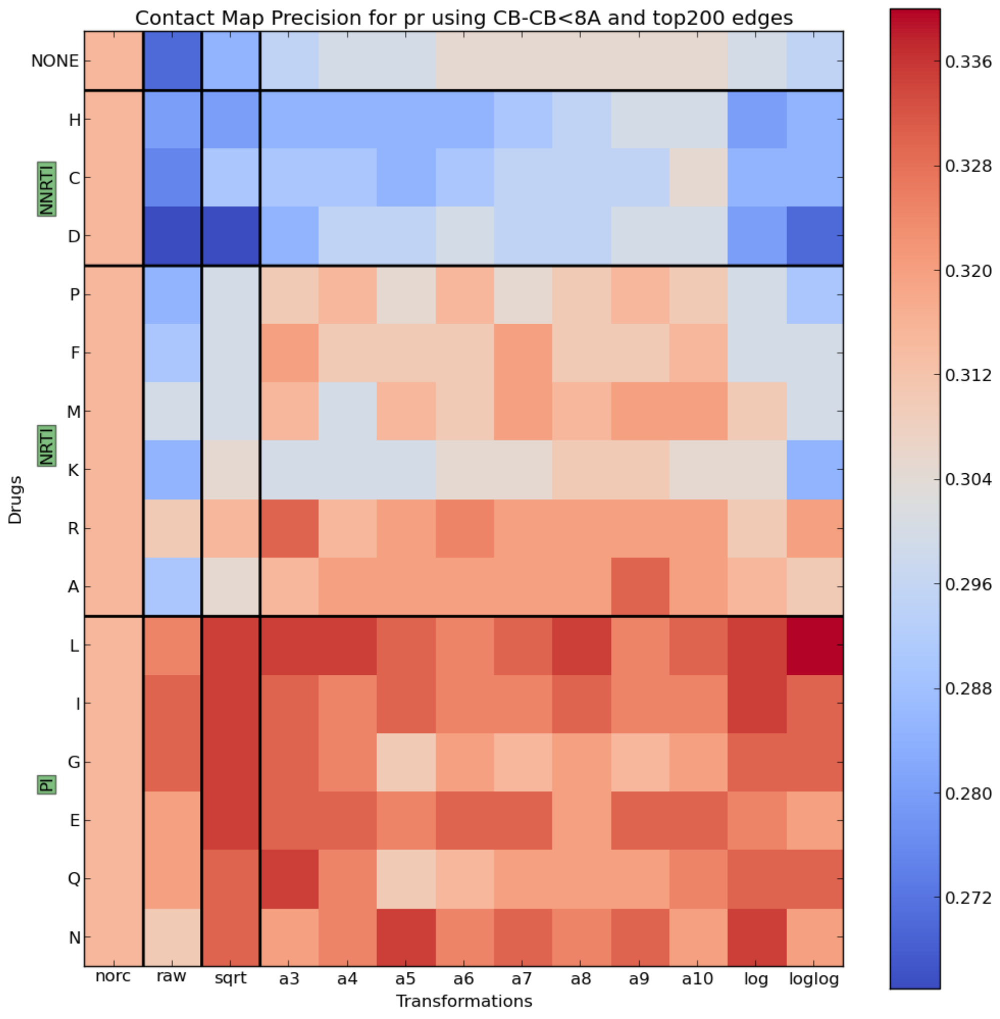


Figure 2.5: **Contact Map Analysis**: The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI). The X axis lists all the transformations used (norc - no weighting, raw- untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \ldots 10\}$, $\log$ and $\log \log$) 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.

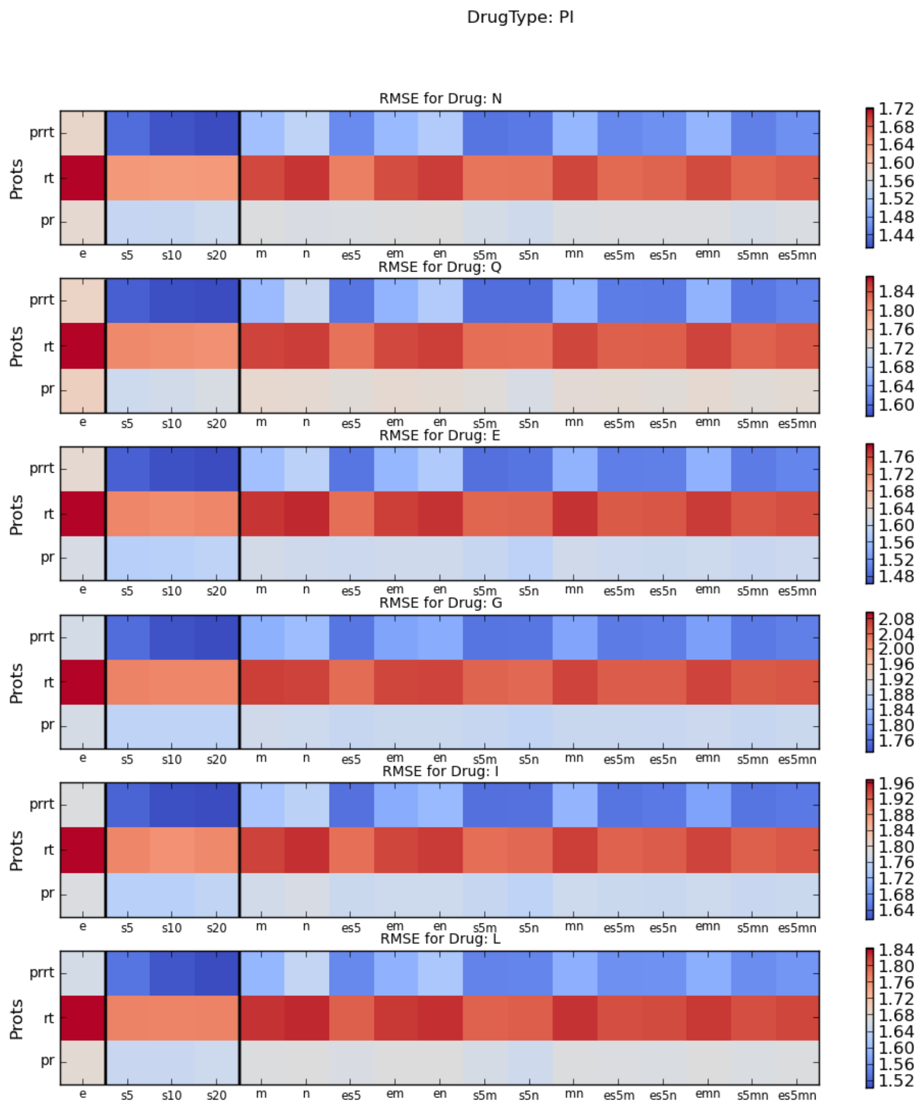


Figure 2.6: **Feature Selection plots for Protease Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs.

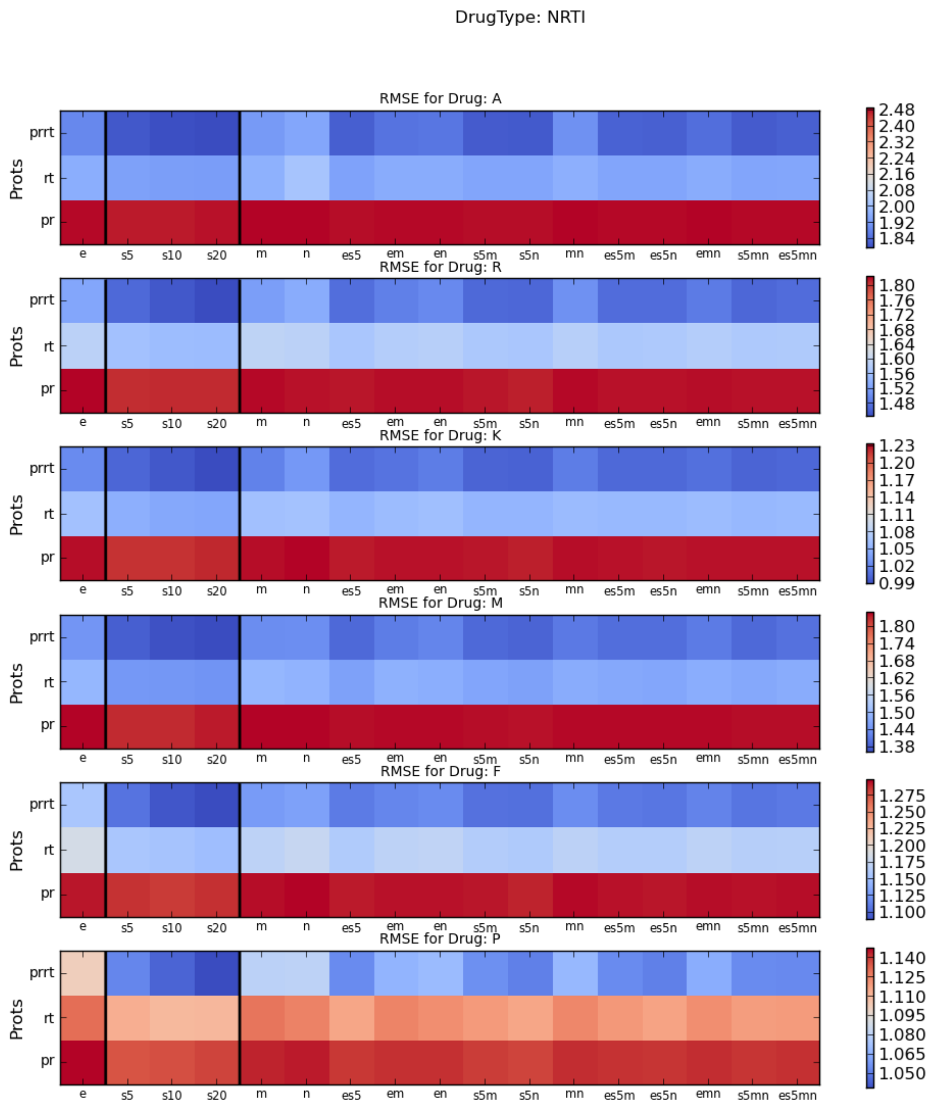


Figure 2.7: **Feature Selection plots for Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the validation RMSE. The plots are grouped according to the different drugs.

2.3.4 Lasso Regression Model

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 2.1. 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 std. dev itself acts as a baseline method which predicts the mean everytime. See figure 2.8 for a scatter plot of predicted vs actual RC values from the drug E regression model.

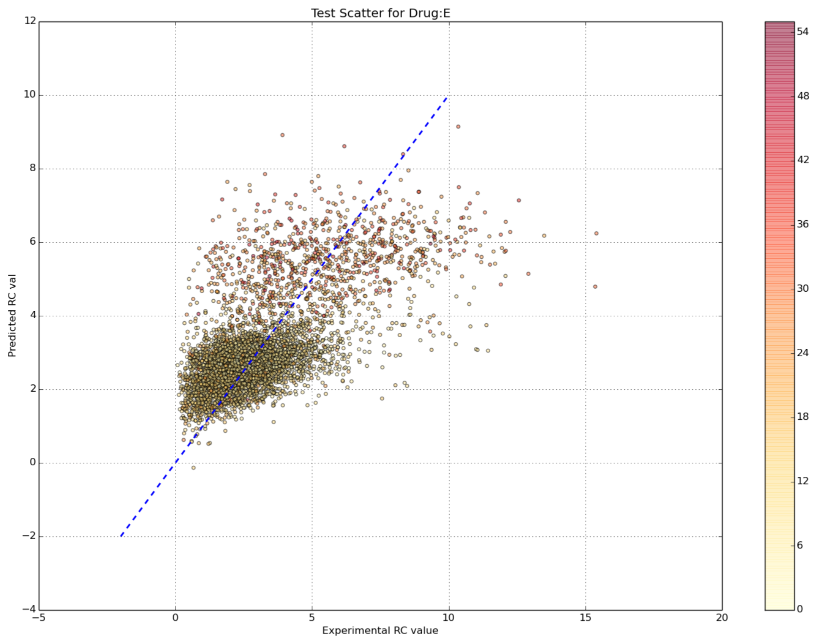


Figure 2.8: Scatter plots using test data for Elastic Net Model trained on 50K training sequences using S20 strong rules features. The color denotes edit distance from the root of a phylogenetic tree of the test sequences.

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 2.1: RMSE Training and Testing Errors using Lasso

2.3.5 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 2.9 for the different KL divergence comparisons. 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. Indeed, Pred-Real has the lowest KL divergence when compared to the other cases. This further instils confidence that our models are extracting valuable signal from the data and predicting the right outputs.

2.3.6 Cocktail Design

In section 2.2.4 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 2.10 shows the comparison between the different cocktail design strategies using regret 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 however the trend generally holds. This allows us to conclude that our cocktail design method can be used for personalized

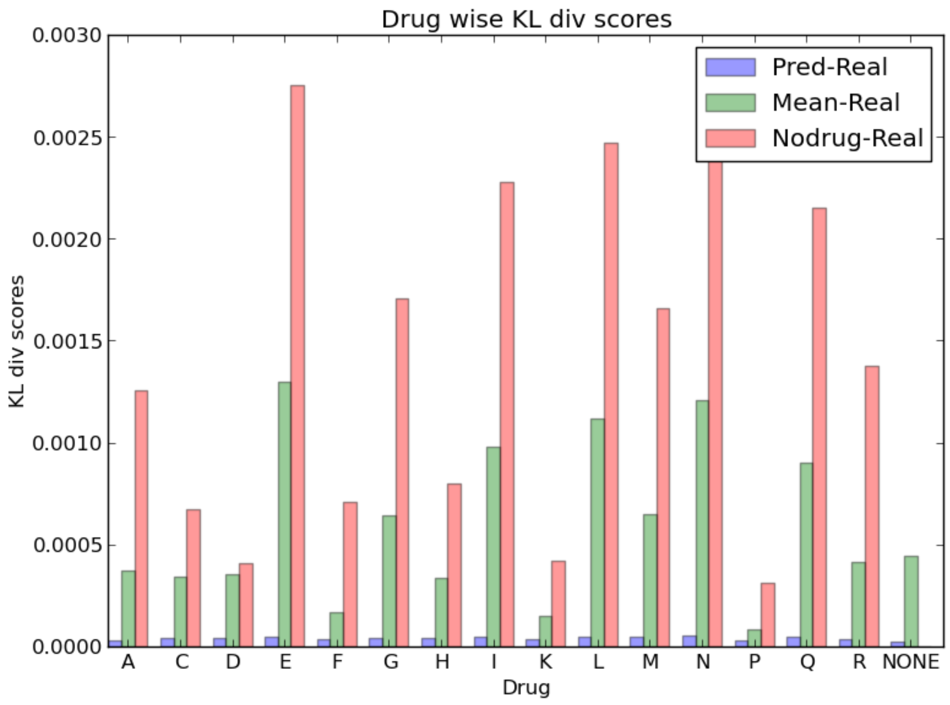


Figure 2.9: The RC values are used to compute a weighted histogram of the residue types at each position. This gives a distribution over amino acids based on the weighting scheme. We compute a KL divergence between pairs of distributions. Pred-Real:Predicted RC and Actual RC ; Mean-Real: Mean RCand actual RC ; Nodrug-Real: Drug Free RC and Actual RC. A low KL divergence indicates that the distributions agree with each other.

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 drugwise 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.

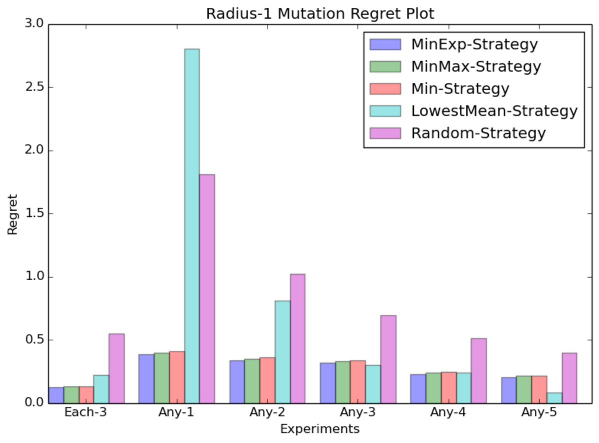


Figure 2.10: Radius 1 Regret

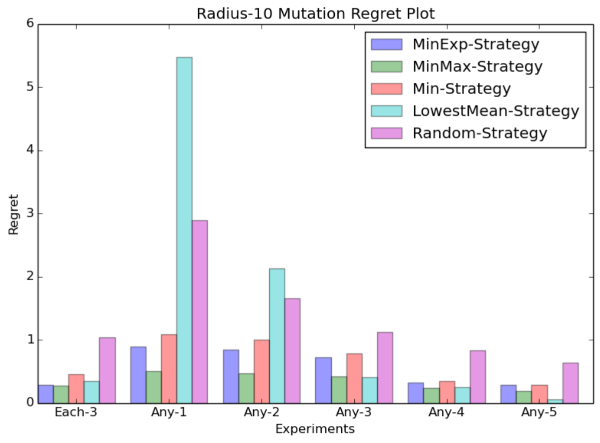


Figure 2.11: Radius 10 Regret

Figure 2.12: 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.

2.4 Conclusion

We work with a HIV-1 sequence dataset [40] 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 [56]. 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 [21]. We are able to report the RMSE and this provides a concrete measurement of the predictive utility of our method.

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 3

Biological Analysis of GPCR Features

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.

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 [2] ,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 [33] 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.

3.1 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 [51]. An estimated 30% of known drug compounds target these receptors [37]. The GPCR family is divided into five distinct classes, class A - E [9]. 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 [38]. 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 [3] . 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 [23, 41] . 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) [8] , Statistical Coupling Analysis (SCA) [31, 49] , Explicit Likelihood of Subset Co-variation (ELSC) [6] , Graphical Models for Residue Coupling (GMRC) [53] , and Generative REgularized ModeLs of proteINs (Gremlin) [2]. 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 [2] 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 [49] and GMRC [53] 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 [19] . 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 [33] for details regarding methods and results.

3.2 Methods

3.2.1 Gremlin

We employed Gremlin [2] to learn a Markov Random Field (MRF) model Fig 3.1 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:

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

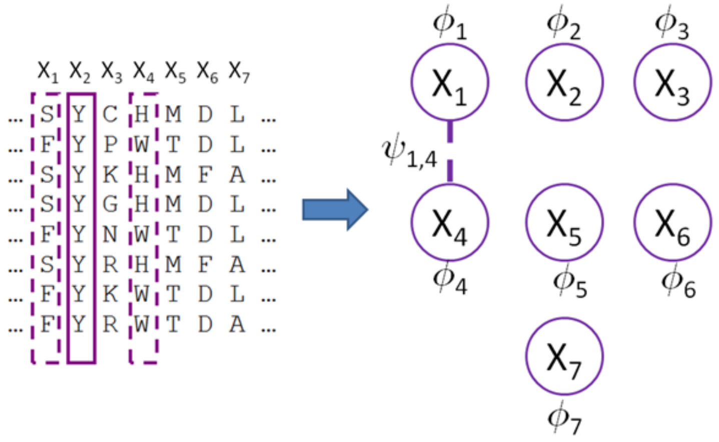


Figure 3.1: Shown in the figure is a cartoon figure of a multiple sequence alignment (MSA) and a corresponding Markov random field (MRF). There is one node in the MRF for each column in the MSA. The column-wise conservation statistics in the MSA are encoded by node potentials (ϕ_i). Similarly, the co-variation 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.

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 3.1 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' 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.

3.2.2 Dataset Description and Preparation

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

3.3 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 B.2.1 for a comparison of gremlin results with SCA and GMRC. See [33] for details regarding methods and results.

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 3.1: **Comparison of edge distribution from control set and Gremlin**

3.3.1 Bovine Rhodopsin Analysis

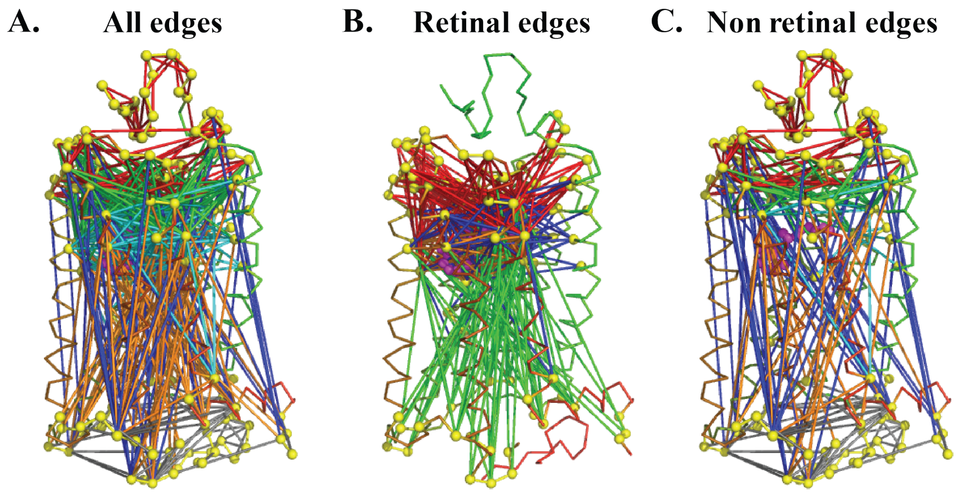
Our analysis revealed that most Gremlin edges involve residues in the RT ligand pocket; as compared to those between or within the residues belonging to EC, IC and TM domains outside of the RT pocket Fig 3.2. The RT pocket is located in the TM domain, at the interface with the EC domain. To quantify 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 3.1. 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.

3.3.2 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 (Table B.2). 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 3.3. 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 $\sim$ 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 3.3) and make key interactions that stabilize RT .



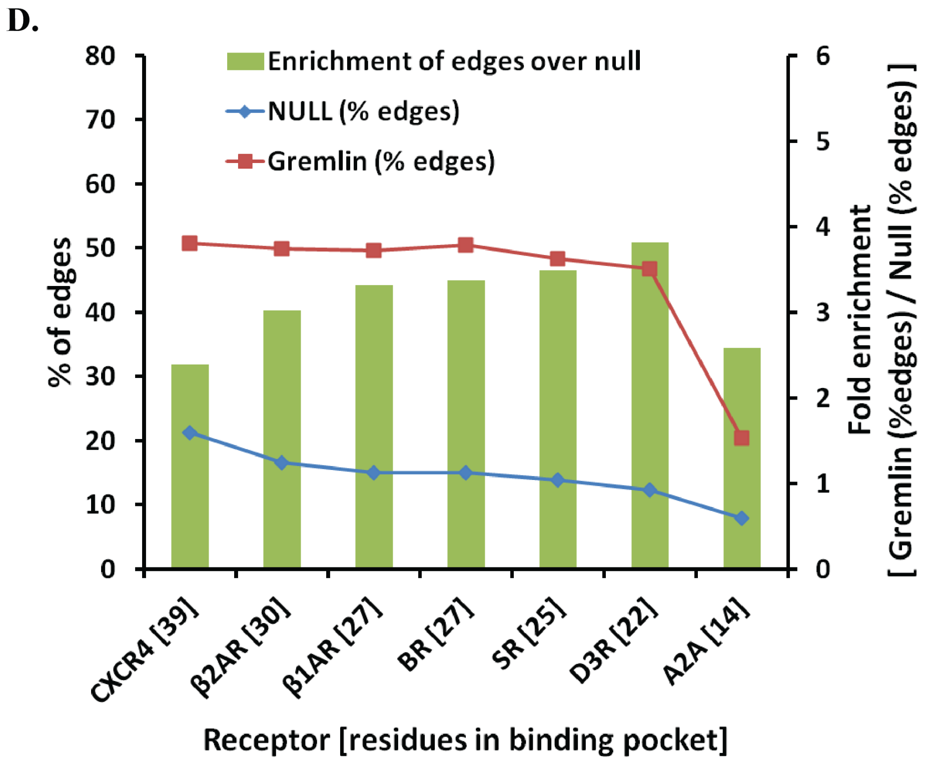


Figure 3.2: Mapping of (A) all (B) RT and (C) non RT edges identified by Gremlin (at =38) mapped onto the bovine rhodopsin structure (PDB ID: 1U19). The edges are EC-EC (red), EC-TM (green), EC-IC (blue), TM-TM (cyan), IC-TM (orange), IC-IC (grey40), EC-RT (red), RT-TM (blue), IC-RT (green) and RT-RT (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.

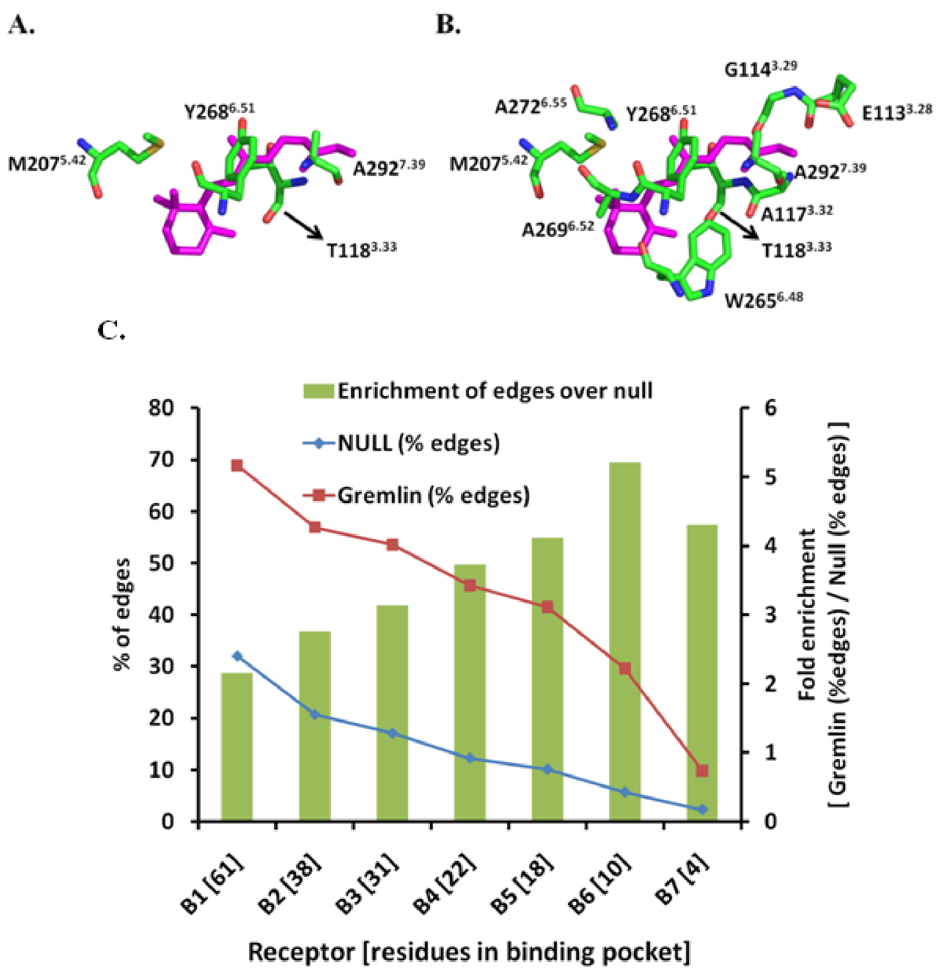


Figure 3.3: The spatial organization of residues in the minimal binding pocket (A) B7 and the larger pocket (B) B6 as present in the rhodopsin structure (PDB id 1U19). Rhodopsin numbering along with Ballesteros-Weinstein numbering (superscript) is given for comparison with other GPCRs. For clarity only 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.

3.3.3 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.

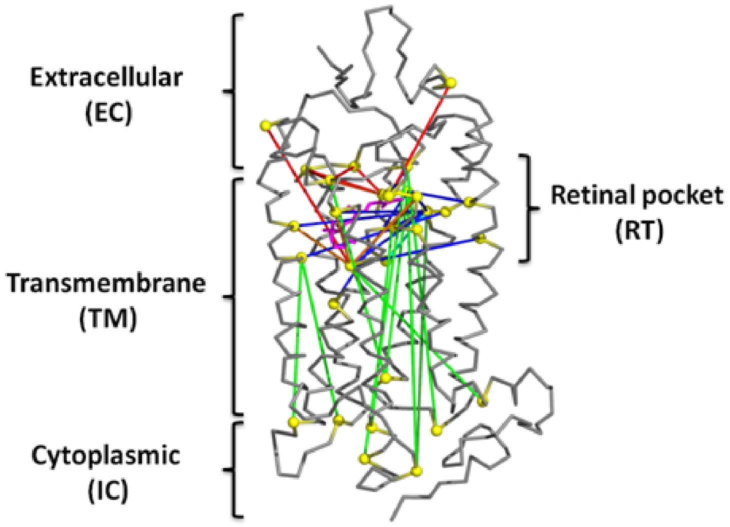


Figure 3.4: Persistent edges at penalty 140 for the top 10 residues are mapped onto the rhodopsin structure (PDB id 1U19). The residues forming the edges are represented as yellow spheres. The edges are TM-TM (cyan), IC-TM (dark green), EC-RT (red), RT-TM (blue), IC-RT (green) and RT-RT (orange), where 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 3.4.

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 3.2. 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 .

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 3.2: **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 4

Feature Learning with Deep Architectures

In this chapter, we move towards unsupervised feature learning of data distributions. Specifically, we consider the problem of modeling protein families using Deep Boltzmann Machines (DBMs) and its variants.

4.1 Background

Evolutionarily related proteins known as a protein family, generally have similar sequences, structures, and functions. Thus, the statistical patterns within the sequences comprising a protein family can provide insights into the constraints that determine structure and function. Generative models of protein families are often learnt from multiple sequence alignments [2, 22, 26]). The popularity of generative graphical 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., [22]) and to design new protein sequences (e.g., [2, 54]). 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.

We desire a good feature representation for protein families. A good representation can be understood as one that models the posterior distribution of the latent factors for an observed input.

We focus on deep architectures which comprise of units of non-linear transformations arranged hierarchically in layers. Such deep architectures are known to have the representative power for modeling complex distributions and functions [5]. Given the complex factors underlying the evolutionary constraints placed on protein sequences ; it is likely that deep architectures are a good candidate for modeling the complexities of protein family distributions.

4.2 Restricted Boltzmann Machines (RBMs)

Boltzmann machines [15] are energy based graphical models. See figure 4.1 for an illustration of a Boltzmann machine. It consists of a visible layer (V) and a Hidden Layer (H). The hidden and visible units are typically binary variables but they can be generalized to multinomial or other exponential family distributions as well [57]. A Boltzmann machine is fully connected and this makes exact inference intractable.

A Restricted Boltzmann Machine (RBM) is a modification to the structure of the Boltzmann machine that imposes some restrictions on the allowed connections. In particular, connections between hidden to hidden and between visible to visible layer units are not allowed ; thus forming a bipartite graph. See figure 4.2 for an illustration of a RBM. This restriction in structure allows for easier inference. Learning is simplified via the Contrastive Divergence (CD) algorithm [13] which is a form of approximate gradient descent.

RBMs have been used successfully in a number of different applications ranging from Collaborative Filtering [44] to modelling motion style [52]. They have been demonstrated to be capable of representing complex functions of data [28]. While Restricted Boltzmann machines are individually useful their role is critical in forming the building blocks of several deep architectures such as Deep Belief Networks, Deep Boltzmann machines and Stacked Autoencoders [14, 16, 42].

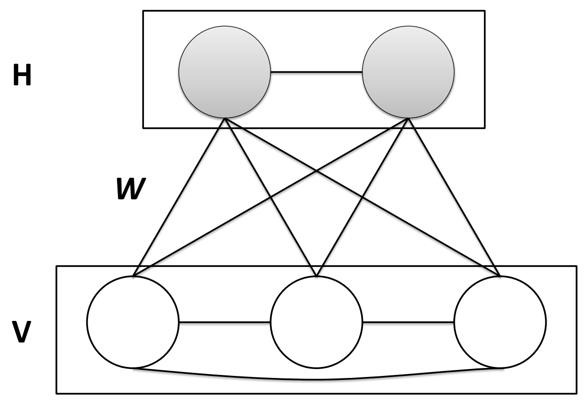


Figure 4.1: General Boltzmann Machines

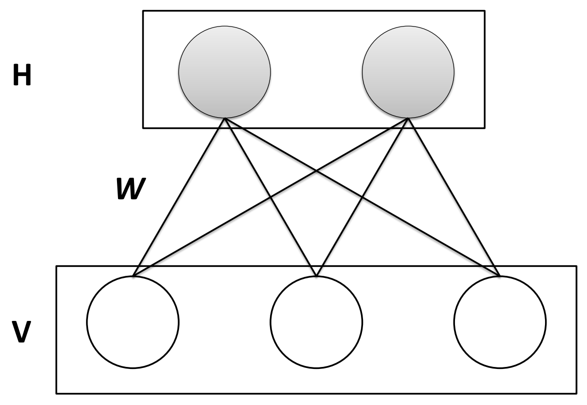


Figure 4.2: Restricted Boltzmann Machines

Figure 4.3: General and Restricted Boltzmann machines

Likelihood of a state (v, h) is

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

The function $Z(\theta)$ is known as the partition function. The function $E(v, h; \theta)$ is the energy function and is defined as

$$E(v, h; \theta) = -v^T W h - v^T c - h^T b$$

where v and h correspond to the visible and hidden variables respectively and $\theta = (W, b, c)$ are the parameters. A lower energy state corresponds to higher probability under the model. Conditioning on the visible variables makes the hidden variables independent and vice versa.

Inference can be done as

$$p(h_i|v) = \text{sigm}(b_i + W_i^T v)$$

where sigm is the sigmoid function $\sigma(x) = e^{-x}/(1 + e^{-x})$; b_i and W_i are the parameters associated with the ith hidden unit h_i.

Probability density of an input is calculated as

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

Sampling from the model can be done with gibbs sampling by alternating between $v^i \sim p(v|h^{i-1})$ and $h^i \sim p(h|v^i)$.

Learning in the RBM is done via the contrastive divergence algorithm which is a form of approximate gradient descent.

$$\frac{\partial}{\partial W} \langle \log p(v_{data}; \theta) \rangle = \langle vh^T \rangle_{data} - \langle vh^T \rangle_{model}$$

The $\langle vh^T \rangle_{data}$ component is obtained by taking samples of h by clamping the visible variables down with training data. The $\langle vh^T \rangle_{model}$ component is obtained by taking samples from the model usually done by running several iterations of gibbs sampling. It is often sufficient empirically to run just a few iterations and this method is known as CD-k [17].

Evaluation can be done by computing a cross-entropy against the empirical data distribution $(\hat{p})$ and the inferred data distribution $\tilde{p}$.

$$H(\hat{p}, \tilde{p}) = -\sum_{v \in V} \hat{p}(v) \log \tilde{p}(v))$$

Another way to evaluate the model is by computing the imputation error $\mathcal{E}$

$$\mathcal{E}(p) = \frac{1}{|V|} \sum_i p(v_i | v_{-i})$$

where V is the number of units in the visible layer and v_i is the ith visible unit.

4.3 Deep Boltzmann Machines (DBMs)

A Deep Boltzmann Machine (DBM) like a RBM is an undirected energy based graphical model [42]. Similar to a RBM it contains hidden and visible variables that model the latent probability distribution of the visible variables. Unlike a RBM there are multiple hidden layers. See figure 4.4 for an illustration of a DBM. This theoretically allows greater flexibility and representative power in modelling complex data distributions. A DBM is capable of generative modeling (sampling,probability density estimation) and can also be used for predictive tasks (classification/regression) when converted into a deep neural network with discriminative fine tuning [42].

DBMs have been used in variety of tasks such as image classification, object recognition and generative modeling[42, 48]. DBMs have been used for multi-modal datasets ; such as joint image and text data [47]. They have also been demonstrated to encode a kernel space at the highest layers [35].

A Deep Boltzmann Machine (DBM) is not the same as a Deep Belief Network (DBN). A DBN is a hybrid directed/undirected graphical model [16]. In a DBN, the topmost layer is an undirected RBM ; lower layers are all directed. Theoretically, a DBN is a generative model however its most common use is for prediction by converting the DBN into a deep neural network with an output layer [4]. For the task of generative protein family models we are interested in evaluating our generative models via cross entropy and imputation error ; this requires inferring the states of the hidden variables given test data. Calculating $p(h|v)$ is hard in a DBN because of the "explaining away" effect exhibited in directed bayes nets [36]. The DBM because of its undirected nature is more amenable to calculating $p(h|v)$ by allowing top down influences in variational inference updates. In most aspects a DBM is similar to a DBN, but for the purpose of evaluating the generative model a DBM is superior.

A DBM has certain limitations. The presence of multiple hidden layers make exact inference intractable. Learning a DBM is also challenging for the same reason. However, recent work has provided a number of ways in which the learning can be made more efficient via the centering trick, greedy pre-training and variational inference [11, 34, 43].

Likelihood of a state $(v \in V, h^1 \in H_1, h^2 \in H_2)$ is

$$p(v, h^1, h^2; \theta) = \frac{\exp^{-E(v,h^1,h^2;\theta)}}{\sum_{v,h^1,h^2} \exp^{-E(v,h^1;h^2;\theta)}} = \frac{\exp^{-E(v,h^1,h^2;\theta)}}{Z(\theta)}$$

The function $Z(\theta)$ is known as the partition function. The function $E(v, h^1, h^2; \theta)$ is the energy function and is defined as

$$E(v, h^1, h^2; \theta) = -v^T W h^1 - h^{1T} U h^2 - v^T b - h^{1T} c^1 - h^{2T} c^2$$

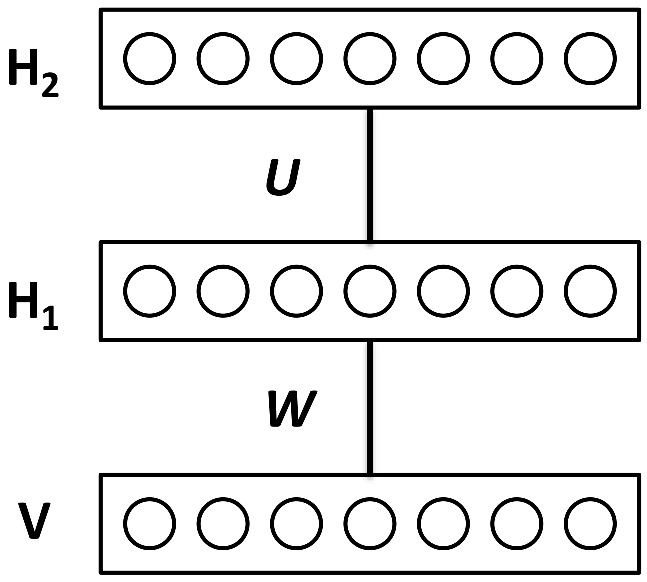


Figure 4.4: Deep Boltzmann Machine - An undirected graphical model where the each layer has no connections within itself but has dense connections across layers

where v, h^1 and h^2 correspond to the variables in the visible and first and second hidden layers respectively and $\theta = (W, U, b, c^1, c^2)$ are the parameters. A lower energy state corresponds to higher probability under the model. Conditioning on the variables in the adjacent layers makes the variables independent.

Inference can be done by conditioing on the adjacent layers.

$$\begin{aligned} v &\sim \text{Binomial}(\sigma(Wh^1 + b)) \\ h^1 &\sim \text{Binomial}(\sigma(W^T v + Uh^2 + c^1)) \\ h^2 &\sim \text{Binomial}(\sigma(U^T h^1 + c^2)) \end{aligned}$$

Probability density of an input is calculated as

$$\begin{aligned} p(v;\theta) &= p(v|h^1)p(h^1) \\ &= E_{p(h^1)}[\text{Binomial}(\sigma(Wh^1+b))] \\ &\approx E_{\bar{p}(h^1)}[\text{Binomial}(\sigma(Wh^1+b))] \end{aligned}$$

where $\bar{p}(h^1)$ is an approximate marginal of the first hidden layer variables obtained by an approximate inference strategy such as gibbs sampling or annealed importance sampling [42].

Sampling from the model can be done by alternating gibbs sampling between the different layers.

Learning in the DBM is done via the contrastive divergence algorithm which is a form of approximate gradient descent.

$$\begin{aligned} \frac{\partial}{\partial W}\langle \log p(v_{data};\theta)\rangle &= \langle vh^{1T}\rangle_{data} - \langle vh^{1T}\rangle_{model} \\ \frac{\partial}{\partial U}\langle \log p(v_{data};\theta)\rangle &= \langle h^1h^{2T}\rangle_{data} - \langle h^1h^{2T}\rangle_{model} \end{aligned}$$

The $\langle vh^T\rangle_{data}$ component is obtained by taking samples of h by clamping the visible variables down with training data using variational mean field inference. The $\langle vh^T\rangle_{model}$ component is obtained by taking samples from the model usually done by running several iterations of persistent chain block gibbs sampling.

Evaluation can be done by computing a cross-entropy against the empirical data distribution $(\hat{p})$ and the inferred data distribution $\tilde{p}$.

$$H(\hat{p},\tilde{p}) = -\sum_{v\in V}\hat{p}(v)\log\tilde{p}(v))$$

Another way to evaluate the model is by computing the imputation error $\mathcal{E}$

$$\mathcal{E}(p) = \frac{1}{|V|}\sum_i p(v_i|v_{-i})$$

where V is the number of units in the visible layer and v_i is the ith visible unit.

4.4 Locally Connected Deep Boltzmann Machines (LC-DBMs)

A Locally Connected Deep Boltzmann Machine (LC-DBM) is similar to a deep boltzmann machine but has additional restrictions. The connections between the visible layer and the first hidden layer are required to be sparse. Compare figure 4.5, a deep boltzmann machine with figure 4.6, a local 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 modeling load of complex data distributions. This acts as a trade-off between allowing rich representations and statistical/computational efficiency.

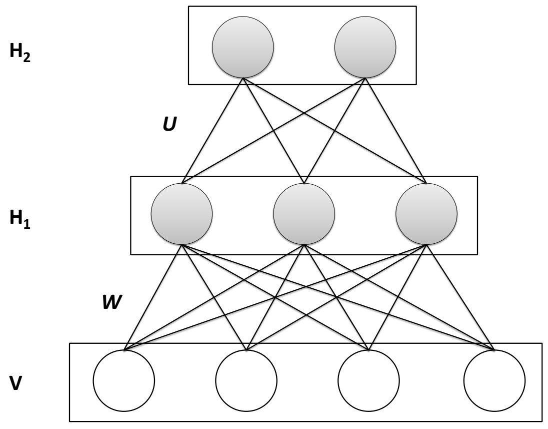


Figure 4.5: Deep Boltzmann Machine

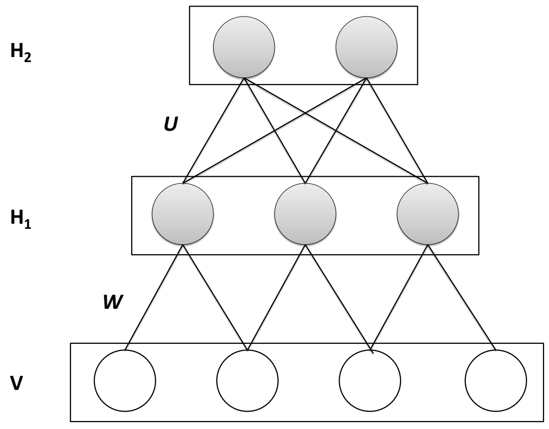


Figure 4.6: Local Deep Boltzmann Machine

Figure 4.7: A locally connected deep boltzmann machine has sparse connection in the lowermost layer but dense connections in higher layers. The sparsity in the structure arises out of a sparsity inducing learning procedure.

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 stuctures of videos and proteins, the 1D sequencial structures of text or protein sequences. This is a commonly used approach in convolutional neural networks [29]. The LC-DBM draws motivation from this idea in the context of DBMs. See figure 4.8 for an illustration of a convolutional neural network applied to a 2D image patch.

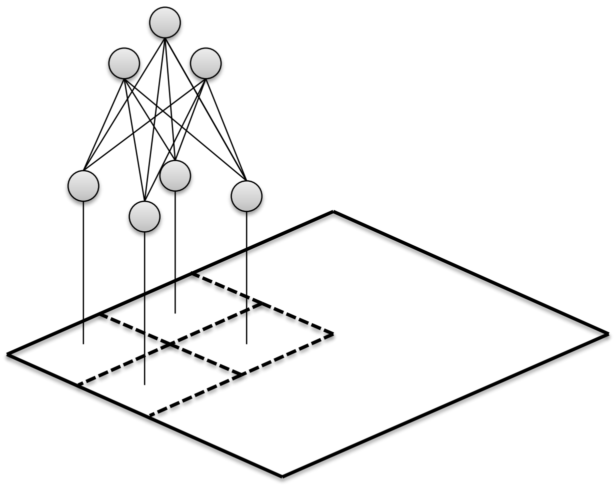


Figure 4.8: A convolutional neural network used in computer vision has a local structure property at the lower layers and dense connections at higher layers. This approach allows incorporating spatial coherence into the statistical structure, an injection of prior knowledge.

The LC-DBM can perform all the tasks possible of a DBM. It is capable of generative modeling (sampling,probability density estimation) and can also be used for predictive tasks (classification/regression) when converted into a deep neural network with discriminative fine tuning [42].

4.4.1 Topology Learning

Learning a sparse structure structure/topology of the connections between the visible layer and the first hidden layer can be done via two different approaches (1) Directly induce sparsity while training a RBM (2) Solving a different sparse problem and then importing the topology into an RBM.

- *Group L1 structure learning*: As previously discussed Gremlin [2] applies Group L1 topology learning [45] to protein families by learning a sparse Markov Random Field (MRF). A simple strategy would be to convert the topology learned via Gremlin into a RBM. See figure 4.9 for an illustration of a conversion process. Keeping the topology fixed, the parameters of the RBM can be retrained using CD. Thus one can go down from $O(n^2)$ potential hidden nodes to $O(n)$.

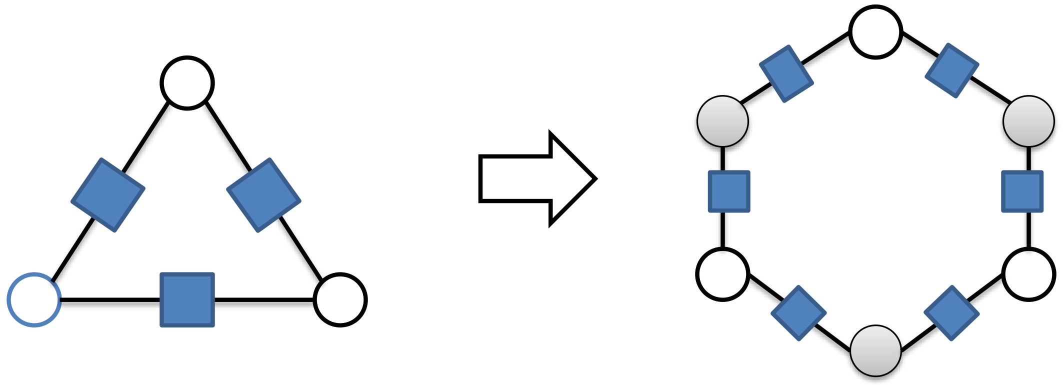


Figure 4.9: The figure on the left is a MRF with pairwise edge potentials (blue). This can be converted to a RBM with binary hidden nodes (grey) ; one or more for each edge factor. Note that this is not a lossless conversion. Suppose the variables are multinomials with M types, then the MRF edge potential is a $M \times M$ matrix whereas the RBM edge potential is a $M \times 1$ vector.

- *Sparse Restricted Boltzmann Machines (SpRBM)* were developed by Lee et.al [30] and adds a sparsity inducing penalty.

$$p(v,h) = \frac{1}{Z}\exp(v^TWh + v^Tc + h^Tb) + \lambda\sum_{j=1}^{|H|}(\rho\log q_j + (1-\rho)\log(1-q_j))$$

  where $q_j = 1/N\sum_N p(h_j = 1|v_n)$ is the average activation of hidden unit h_j, ρ is the desired target sparsity and λ is the strength of the penalty and N are the number of training samples. This encourages both *lifetime sparsity* and is likely to introduce *population sparsity* as well.

- *Cardinality Restricted Boltzmann Machine (CaRBM)* - were developed by Swersky et.

al [50] and incorporates a counting potential ψ_k.

$$p(v,h) = \frac{1}{Z}\exp(v^T W h + v^T c + h^T b) \cdot \psi_k\left(\sum_{i=1}^{|H|} h_i\right)$$

where ψ_k is the counting potential given by $\psi_k(c) = 1$ if $c \leq k$ and 0 otherwise . The counting potential encourages sparsity in the hidden layer. The authors describe an efficient dynamic programming/sum-product algorithm for training the CaRBM using contrastive divergence training.

Once a topology has been identified, the parameters can be re-learned using the new topology with RBM-CD learning. The training of the remaining higher layers proceeds using Persistent Contrastive Divergence in a DBM 4.3. Learning, sampling, inference and evaluation is also the same as in a DBM 4.3.

4.5 Modeling Protein Families with Deep Architectures

We consider the problem of modeling protein families with the aforementioned deep architectures. Figure 4.10 illustrates a small multiple sequence alignment (MSA) of a protein. Each column in the MSA corresponds to a position in the protein's sequence. Generative models of protein families typically encode the probability of observing each of the twenty amino acid types at each column, as well as any correlation between pairs of columns. Both Hidden Markov Models (e.g., [22, 26]) and Markov Random Fields (e.g., [2, 54]) have been used to model protein families previously. The nodes in such graphical models correspond to multinomial random variable, one for each column of the MSA, and the edges specify the conditional independencies between the columns. In practice, Markov Random Fields have proven superior to HMMs for a variety of inference tasks, including function prediction and design due to the fact that HMMs make assumptions about the conditional independencies among the columns that are not valid for most proteins.

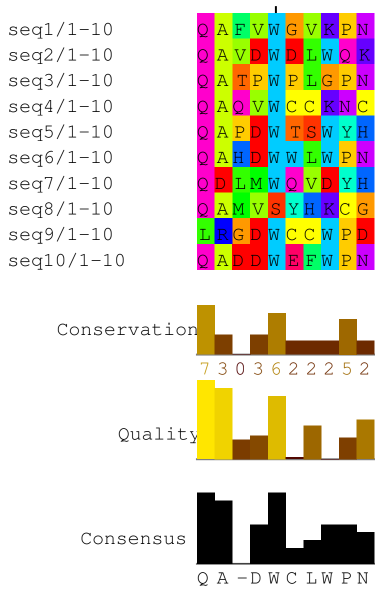


Figure 4.10: Protein multiple sequence alignment. The rows correspond to the protein sequences in the alignment. The columns correspond to the positions in the protein.

Figure 4.11 illustrates a candidate deep architecture for modeling protein families. The deep architecture in consideration is a LC-DBM 4.4. There is a visible softmax layer encoding amino acids as multinomials ; there are two binary hidden layers. The local topology of the lowest layer is learned via one of the sparsity inducing methods discussed in section 4.4.1.

The parameters of the lowest RBM are learned using RBM CD learning 4.2 . Note that the visible layer is a softmax layer and the CD learning rules need to be modified for this case. The higher layers are trained using the DBM training procedure 4.3. Sampling, Inference and Evaluation is also the same as in a DBM 4.3.

See Appendix C.1 for learning rules in the softmax case.

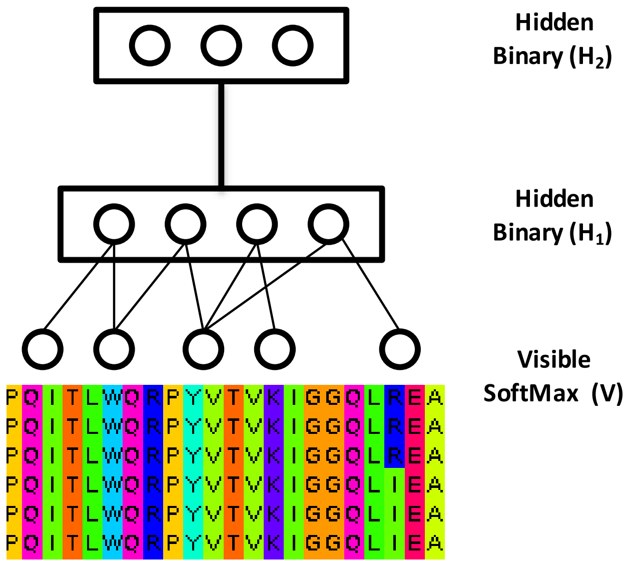


Figure 4.11: The figure illustrates a candidate deep architecture for modeling protein families using a LC-DBM 4.4. There is a visible softmax layer encoding amino acids as multinomials ; there are two binary hidden layers. The local topology of the lowest layer is learned via one of the sparsity inducing methods discussed in section 4.4.1

4.6 Experiments and Results

In this section we present results from experiments with simulated data and a small HIV-1 protease dataset using RBMs. We also propose future experiments for DBMs, LC-DBMs with the goal of studying the effect of deep architectures on representation power and computational efficiency of these models in the context of learning generative protein families.

4.7 Experiments using RBMs

We conducted experiments on both simulated data and on real protein sequences. The motivation for working with simulated data is that we can control for statistical properties in the data. This given us some feedback about understanding and introspecting the weights that our model is learning. This is similar to examining the weights that neural nets or boltzmann machines learn when fed inputs from a known boolean function.

4.7.1 Simulated Proteins

We generated a simulated dataset consisting of 10 visible variables and 700 data points. These are discrete variables that can take 20 possible values. We chose 20 because the number of possible amino acids in proteins is 20. The dataset was split into 500 train, 100 valid and 100 test. We added some statistical bias for some of the variable. Variables 1,2 and 5 are conserved and contain the amino acid Q,A and W 80% of the time. Columns 9-10 and 4-8 have covarying amino acid pairs for e.g. notice the frequent occurrence of (Y,H) and (P,N) pairs in columns 9-10. These pairs occur roughly 40% of the time. Columns 3,6,7 are uniformly distributed. See Figure 4.12 and figure 4.13 for a sample from the simulated dataset.

We then proceeded to train RBMs over this simulated dataset. The importance of hyperparameter search is well understood in the deep learning community. By choosing to use more flexible feature representations we often have to be responsible in choosing the right combinations of

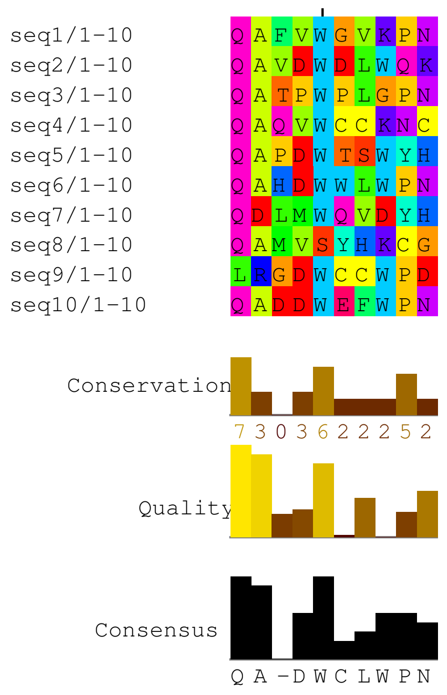


Figure 4.12: A subset of test sequences

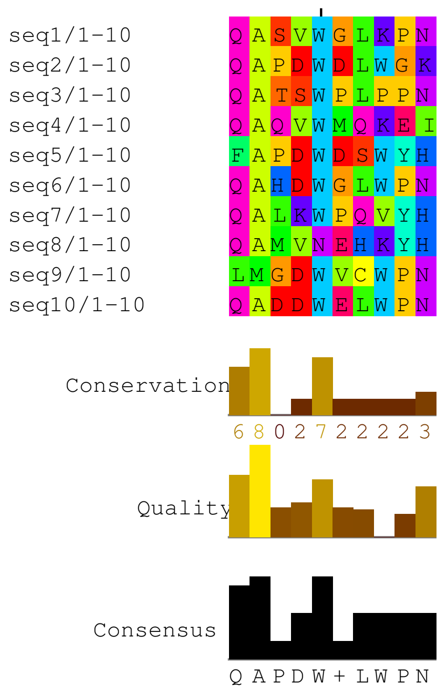


Figure 4.13: Reconstruction with best model

Figure 4.14: Simulated dataset where columns 1,2 and 5 are conserved. Columns 9-10 and 4-8 co-vary with each other. Columns 3,6 and 7 are uniformly distributed.

hyperparameters. Hinton et al. [17] discuss several factors that affect learning in RBMs. These include the learning rate, initial value of the weights, the number of hidden units, minibatch-size, etc. Often this involves trying all combinations of hyperparameters and making sure that we don't overfit by doing model selection on an evaluation dataset. Newer techniques on using bayesian optimization schemes for hyperparameter search [46] seem promising and might be to interesting to pursue in the future.

Training Strategies

We trained an RBM where the visible layer contains softmax variables (amino acids) and the hidden layer contains binary variables. A number of modeling choices were made based on best practices recommended in literature [17]. Training was done in minibatches using contrastive divergence. Batchsizes of 10 and 20 were tried. A number of different varieties of contrastive divergence exist; we chose to try CD-1 and CD-k for different values of k - 1,2,5 and 10. We also tried different learning rates - 0.1, 0.01 and 0.001. Combinatorially an n-length amino acid protein fragment can have 20^n possible combinations; whereas binary visible units can have 2^n. In practice, we often need many lesser units to model protein distributions. Additionally having an under-complete representation can enforce sparsity and learn the truly important statistical patterns in the data; which is the entire motivation behind sparse-autoencoders. It with this sparsity inducing ideology that we chose to restrict the number of units in the hidden layer between 10,20,30,40 and 50.

We used early stopping as a regularity measure using the validation cross-entropy as a control metric. The cross-entropy is between test data and model reconstruction. The best model was selected using the best validation found from a parameter scan across all the parameters discussed previously. Table 4.1 contains the training, testing and validation cross-entropy. The best model had 50 hidden units, learning rate of 0.1, CD-1 learning rule and batchsize of 20. The best model has a test cross-entropy of 10.0935. Table 4.1 shows a partial parameter scan keeping all the other parameter scan variables fixed to the best values. Figure 4.15 and figure 4.16 show the learning progress of the best model using psuedo-likelihood and cross-entropy.

The main lesson learned from the simulated experiments is that good learning in RBMs on proteins depends on having a large number of hidden nodes. Too many would ofcourse be overfitting and can be controlled with cross-validation using an evaluation set. There is also merit in restricting the number of hidden nodes as under-completeness leads to good feature coding at the cost of good reconstruction.

Hidden Units	Train cross-entropy	Validation cross-entropy	Test cross-entropy
10	13.9043	14.7229	14.0957
20	11.5798	13.8341	13.3881
30	10.0895	13.4879	13.7031
40	3.2286	12.2551	12.3070
50	1.9047	10.2341	10.0935

Table 4.1: Parameter Scan over the number of hidden units

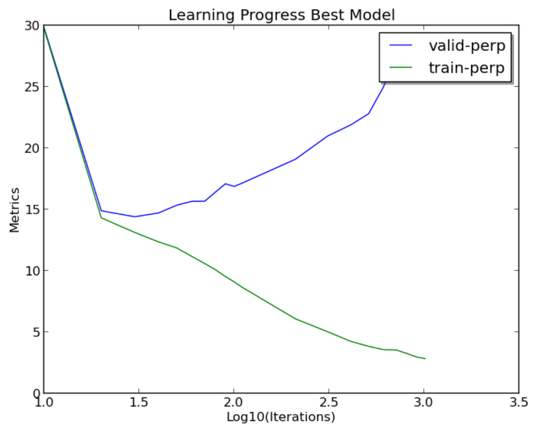


Figure 4.15: cross-entropy of the best model

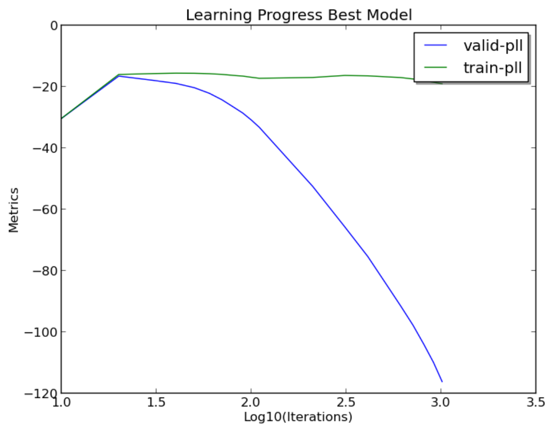


Figure 4.16: Psuedo-LL of the best model

Figure 4.17: Monitoring learning progress for the best RBM model

Train cross-entropy	**Validation cross-entropy**	**Test cross-entropy**
17.7886	11.0500	20.7563

Table 4.2: Training, validation and testing error when training a RBM on a small protease alignment

4.7.2 Small Protease Alignment Experiments

We took an alignment of HIV-1 protease sequences. The alignment contains 99 columns and 3500 sequences. We split the alignment into 2500 train, 500 test and 500 validation. We trained the RBM using 99 hidden binary units. Table 4.2 shows the training and testing error on this dataset.

Chapter 5

Proposed Work and Timeline

5.1 Completed Work

In chapter 2, we tackle feature selection in the context of large scale data. We address the problem of drug cocktail design against HIV-1 infection. They key challenge in this approach was scaling up our regression models to this large dataset and selecting discriminatory features.

In chapter 3 and published in Moitra et al. [33], we study the problem of feature interpretation in an unsupervised setting. We explore signal transduction in G protein coupled receptors (GPCRs. 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$^{7.39}$).

In chapter 4, we study the problem of unsupervised feature learning. We attempt to address the task of learning a generative model for protein families with a rich feature representation. We show preliminary results with Restricted Boltzmann Machines (RBMs) in section 4.7.

The RBMs are the fundamental units of deep architectures. While RBMs themselves are useful in modeling protein families, real representative power is obtained by stacking the RBM units into deeper architectures to form Deep Boltzmann Machines (DBMs). In the next section, we describe proposed experiments for DBMs and LC-DBMs.

5.2 Proposed Experiments

We would like to propose the following experiments:

5.2.1 Deep Boltzmann Machines (DBMs)

1. ***Datasets:***
   - Simulated proteins as described in section 4.7.1
   - Large protein sequence datasets $\sim$ 50,000 non-redundant sequences such as those curated by Baker lab - `http://gremlin.bakerlab.org/`.
2. ***Learning:*** These are the major steps in the learning algorithm
   - *Contrastive Divergence* : Train Softmax RBMs
   - *Greedy Stacking* : Layerwise stacking of pretrained RBMs
   - *Variational Inference* : Solve fixed point equations to fine tune stacked model
3. ***Evaluation:***
   (a) *Generative metrics* : By splitting the dataset into train, test and eval we can calculate pseudo log-likelihood, cross entropy and imputation error as detailed in 4.3.
   (b) *Baseline :* Compare with other generative models such as HMMs and CRFs
   (c) *Hyper-parameters :* Generative metrics should improve and then deteriorate on adding additional layers and nodes in a layer.
4. ***Risks and Mitigation:***
   (a) *Slow Learning :* Learning in the joint DBM model can be slow for a variety of reasons including (1) stochastic noise in updates (2) poor conditioning. Greedy pre-training, persistent CD and gibbs sampling with top down influences are common tricks to deal with the stochastic noise [42]. The poor conditioning can be ameliorated to some extent by using the centering trick [36] and sufficient amounts of training data.

Further, vectorizing the matrix operations using GPUs and CUDAMAT can lead to significant speedups [32]. As shown by Montavan et al. [36], experiments with DBMs using 28×28 image patches from the MNIST dataset using 60,000 samples and 400 and 200 hidden nodes have been known to scale well. Protein sequence models are of similar size and these will likely scale too.

(b) *Poor Generative performance :* It is possible that DBMs might have inferior generative performance. The main cause of poor generative ability would be overfitting to the training data. Strategies to limit overfitting might be helpful in this context such as aggressively limiting the number of hidden nodes and adding a L2 penalty term. Also, given satisfactory preliminary experiments using RBMs we are hopeful that this scenario is less likely to arise.

5.2.2 Locally Connected Deep Boltzmann Machines (LC-DBMs)

1. ***Datasets:*** Same as in section 5.2.1
2. ***Learning:*** The following are the sparse structure learning strategies. The rest is the same as in section 5.2.1.
   - *Gremlin to RBM* conversion
   - *SpRBM/CaRBM* : These sparse learning strategies can be optionally pursued, though a gremlin to RBM conversion should be sufficient in defining the structure of the lowest layer.
3. ***Evaluation:*** Same as in section 5.2.1. With the addition of a qualitative introspection of the weight matrices at the lowest layer.
4. ***Risks and Mitigation:*** Same as in section 5.2.1.

5.3 Timeline

We would like to propose the following broad timeline

1. ***May 2014*** : Complete DBM experiments
2. ***June - Aug 2014***: Internship at Google
3. ***Sep - Dec 2014*** : Complete LC-DBM experiments
4. ***Jan 2015*** : Thesis Defence.

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 [40] contained 71727 sequences in total. The protease alignment had a width of 99 columns while the reverse yranscriptase 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 [12] 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 [12]. See 2.4 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 ∼15%. These X characters are considered as missing values that need to be imputed, further discussed in section A.1.1.

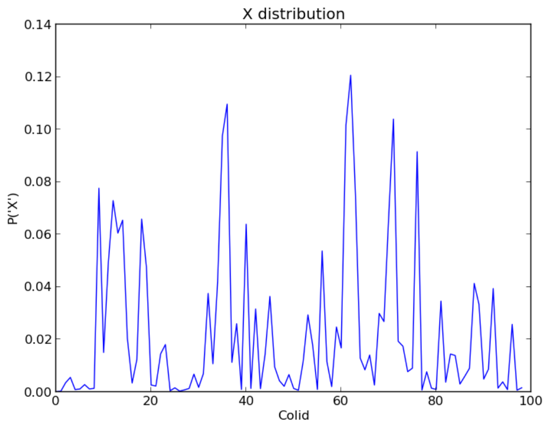


Figure A.1: Protease

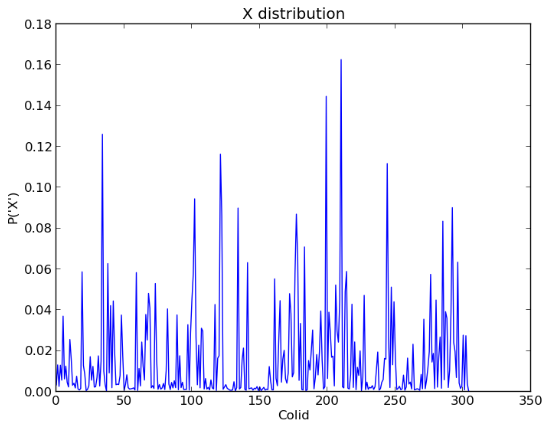


Figure A.2: Reverse Transriptase

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.

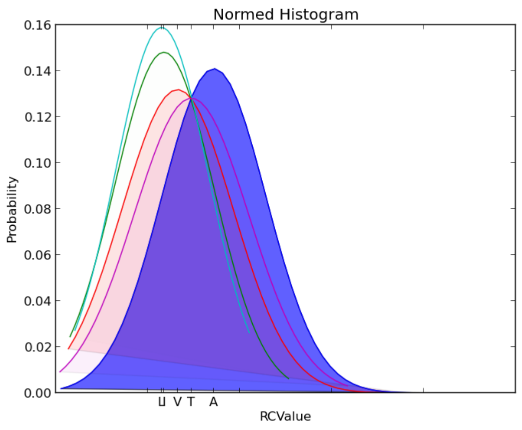


Figure A.4: Col 71 Protease

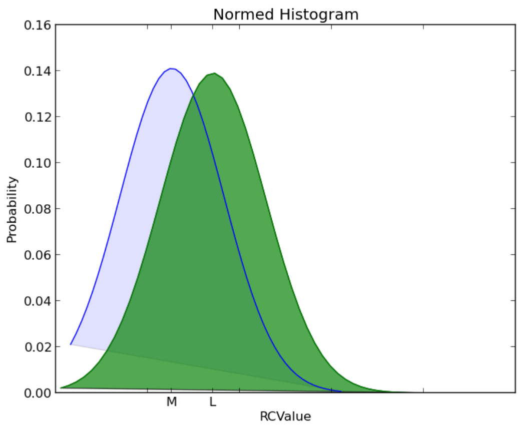


Figure A.5: Column 90 Protease

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.

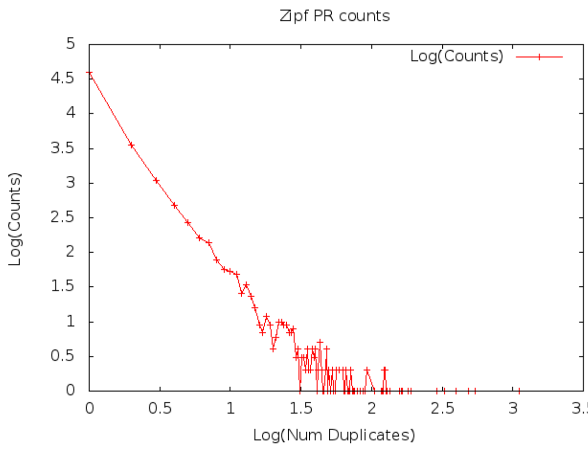


Figure A.7: Zipf Protease

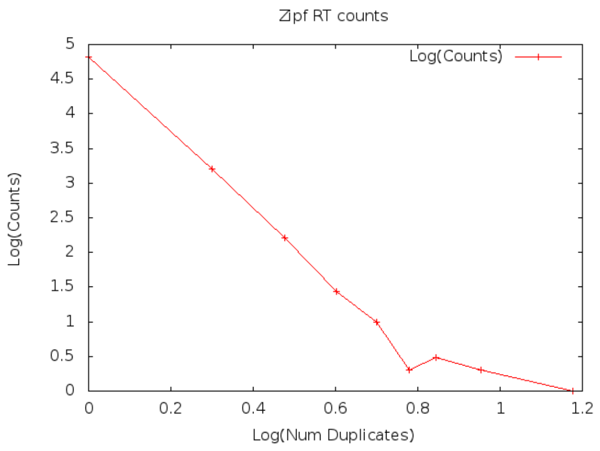


Figure A.8: Zipf Reverse Transcriptase

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

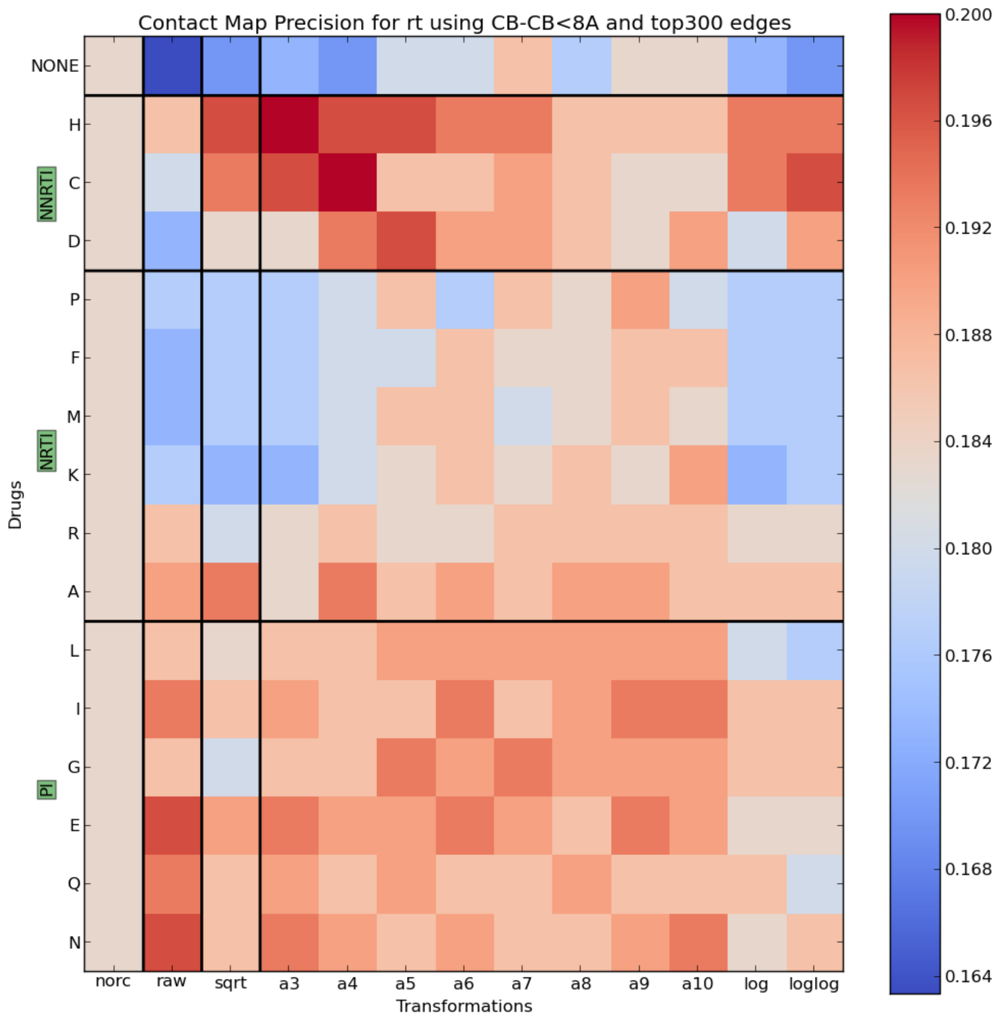


Figure A.10: **Contact Map Analysis Reverse Transcriptase**: The 15 different drugs on the X axis organized according to drug type (PI, NRTI, NNRTI). The X axis lists all the transformations used (norc - no weighting,raw- Untransformed RC, sqrt - square root, ak - $RC^{1/k}$ for $k \in \{1 \ldots 10\}$,log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

A.1.3 Feature Reduction Strategies

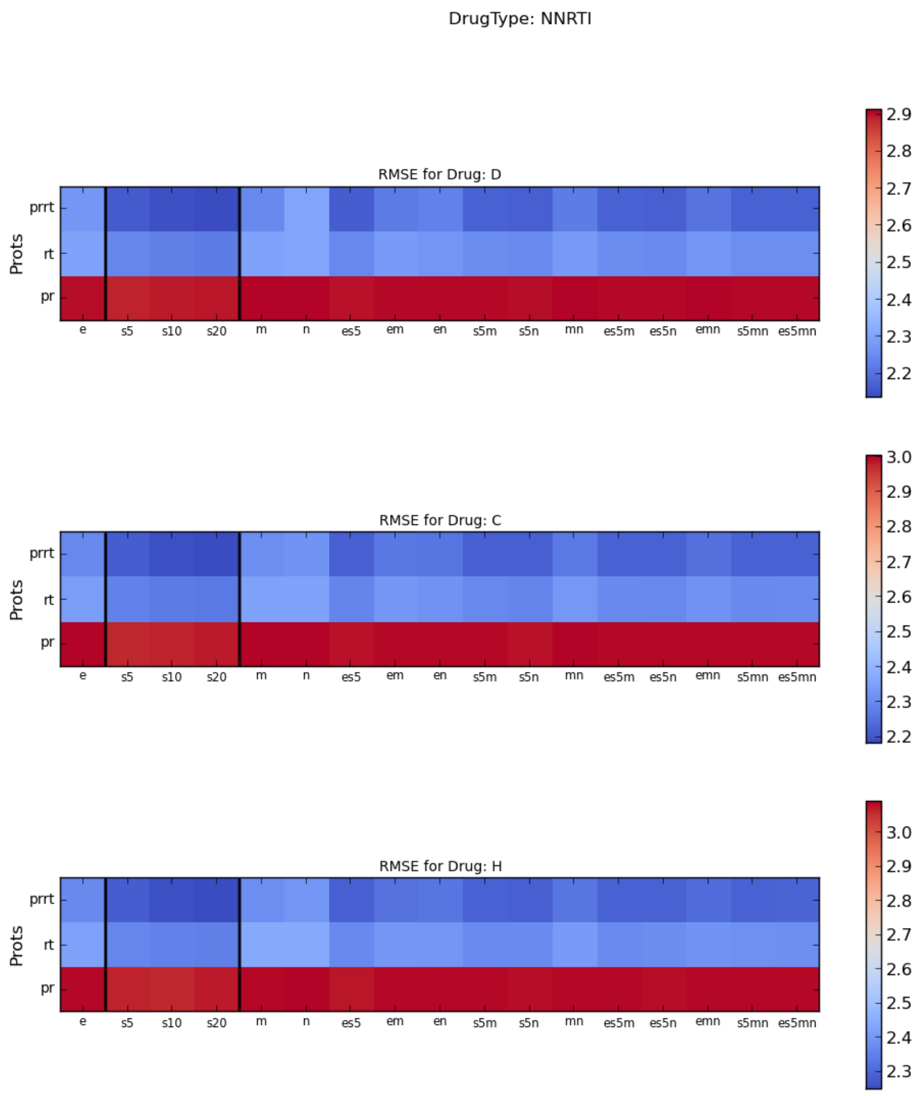


Figure A.11: **Feature Selection plots for Non Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT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

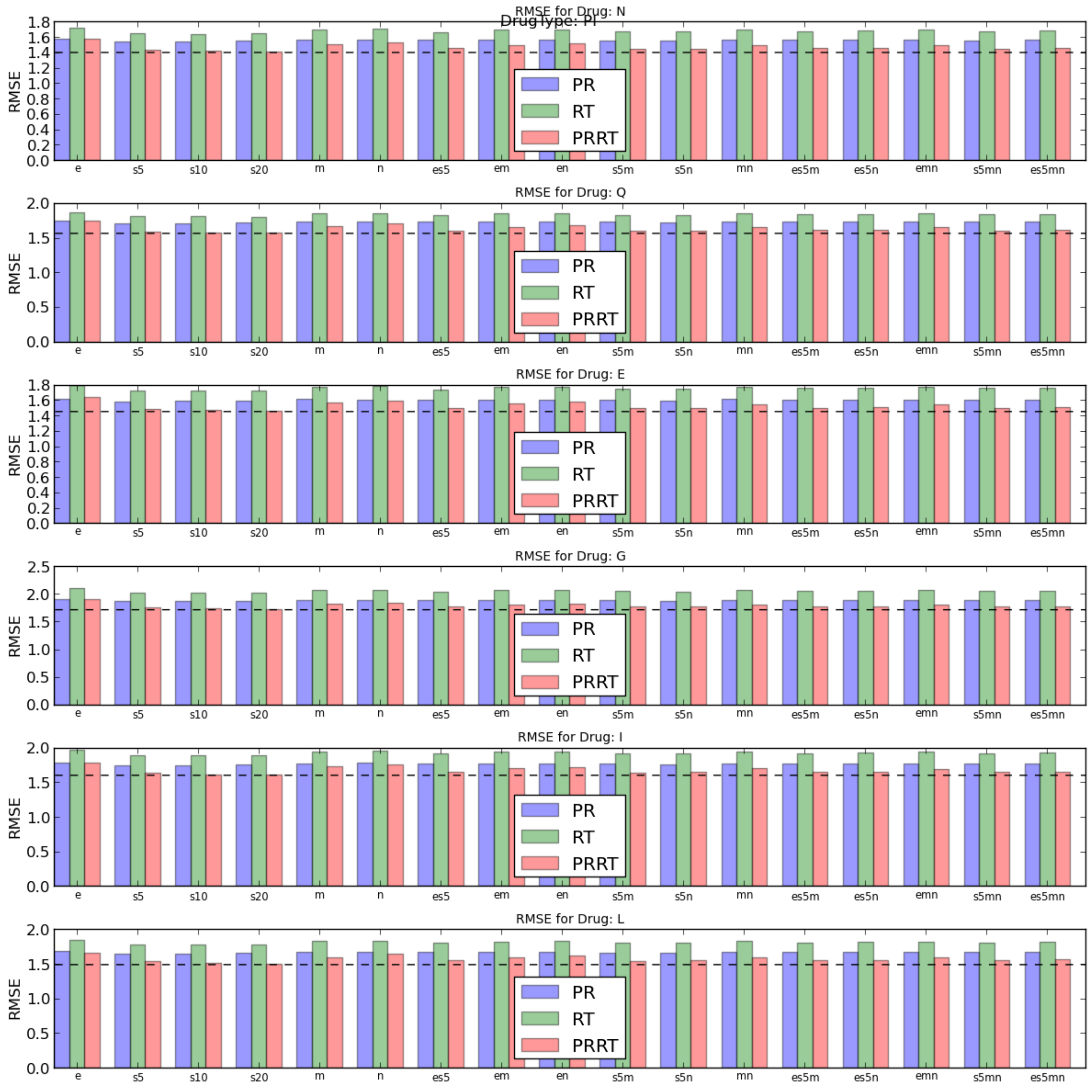


Figure A.12: **Feature Selection plots for Protease Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment. The colorbar is the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

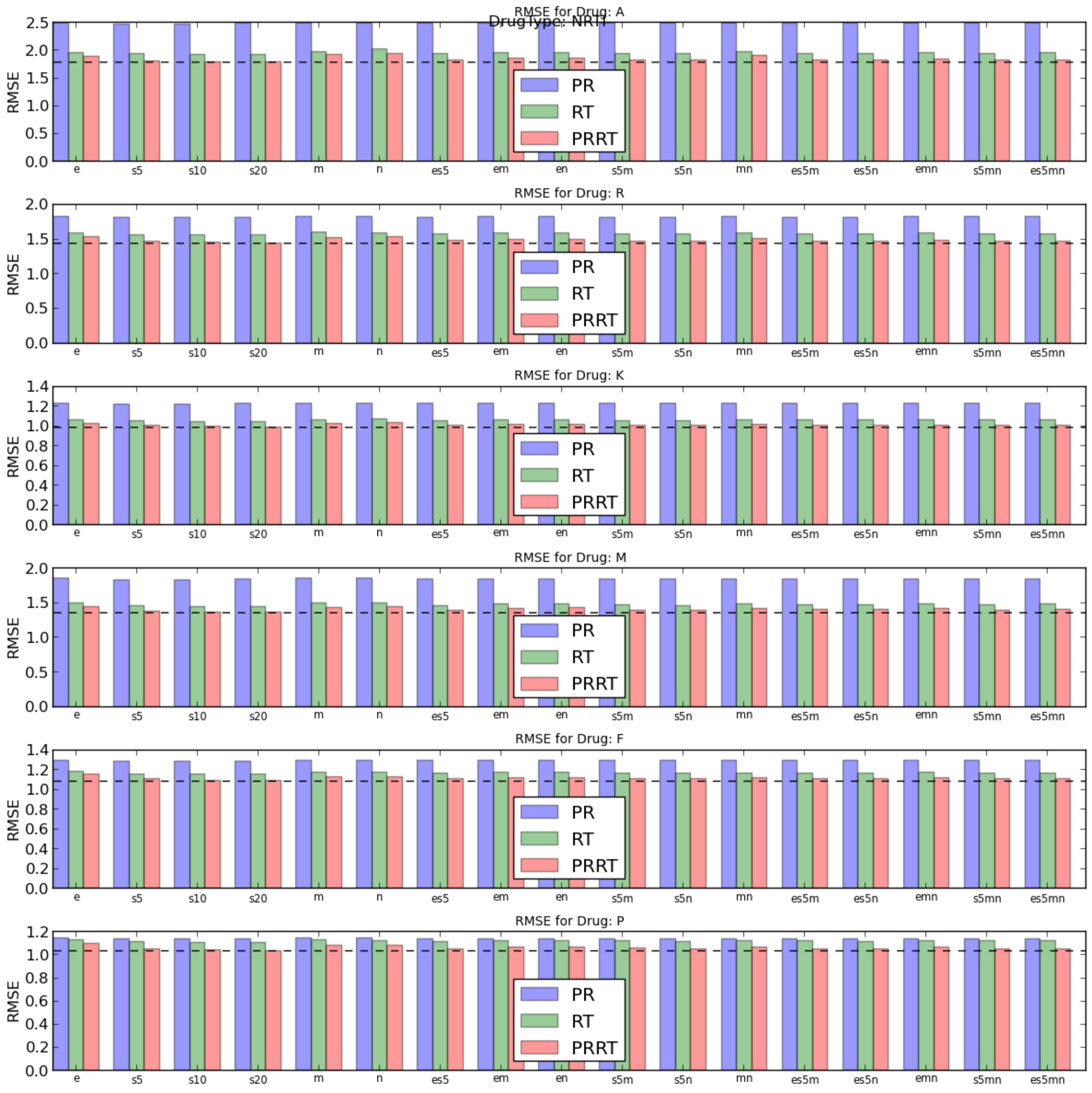


Figure A.13: **Feature Selection plots for Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT alignment.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

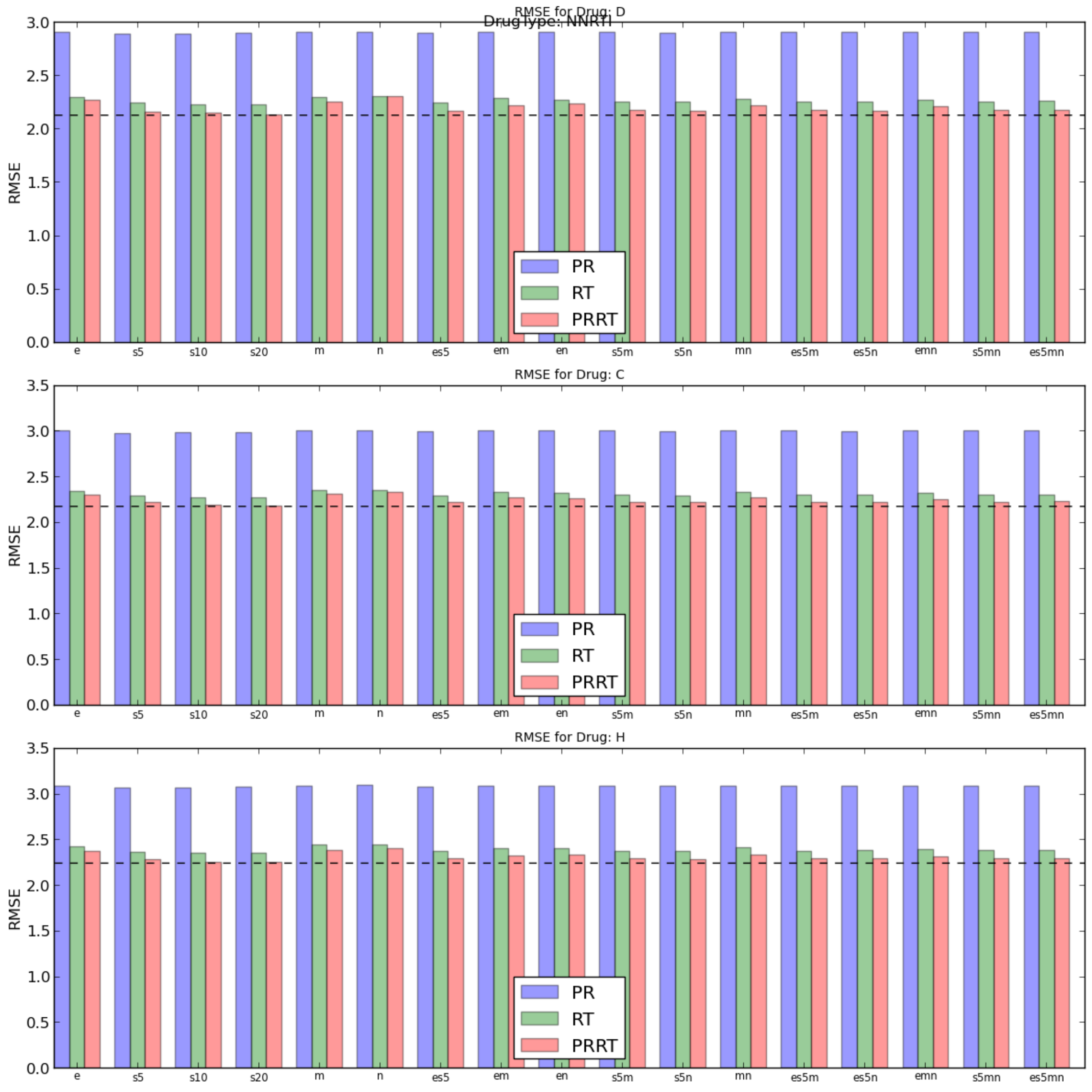


Figure A.14: **Feature Selection plots for Non Nucleoside Reverse Transcriptase Inhibitor:** X axis contains the different feature selection schemes(e-elghaoui,n-new gremlin,s-strong rules,m-marginal regression) and their combinations. Y axis has lasso regression models trained on a PR, RT and PR+RT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

A.1.4 Cocktail Design

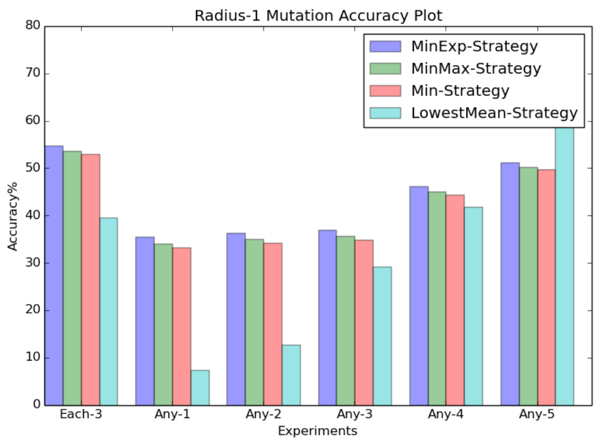


Figure A.15: Radius 1 Accuracy

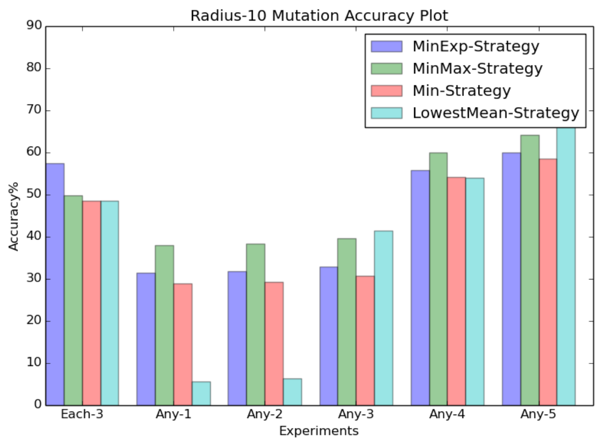


Figure A.16: Radius 10 Accuracy

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

Biological Analysis of GPCR Features

B.1 Methods

B.1.1 Multiple sequence alignment

The authors of the SCA study [49] obtained the class A GPCR alignment from GPCRDB [18] and TinyGRAP [7] 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 .

B.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.

B.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).

B.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 $5\mathring{A}$ 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 3.1). 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 [6]).

B.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.

B.2 Results and Discussion

B.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 B.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 B.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 B.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 net-*

B1	B2	B3	B4	B5	B6	B7
M1, G3, L31, Q36, F37, M44, M86, G90, T93, T94, T97, S98, F103, E113, G114, A117, T118, G121, E122, L125, P171, L172, Y178, I179, P180, E181, S186, C187, G188, I189, Y191, T193, P194, H195, E196, E197, N200, F203, V204, M207, F208, H211, F212, F261, W265, Y268, A269, A272, F273, I275, H278, Q279, S281, F283, P285, M288, T289, A292, F293, A295, K296	M86, T94, T97, S98, F103, E113, G114, A117, T118, G121, E122, L125, Y178, I179, P180, E181, S186, C187, G188, I189, Y191, F203, V204, M207, F208, H211, F212, F261, W265, Y268, A269, A272, P285, M288, T289, A292, F293, K296	T94, T97, S98, E113, G114, A117, T118, G121, E122, Y178, I179, P180, E181, G188, I189, Y191, F203, V204, M207, F208, H211, W265, Y268, A269, A272, P285, M288, T289, A292,	T94, E113, G114, A117, T118, G121, E122, P180, I189, F203, V204, M207, F208, H211, W265, Y268, A269, A272, M288, T289, A292, K296	E113, G114, A117, T118, G121, E122, P180, I189, M207, F208, H211, W265, Y268, A269, A272, M288, A292, K296	E113, G114, A117, T118, M207, W265, Y268, A269, A272, A292	T118, M207, Y268, A292

work - 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 C

Feature Learning with Deep Architectures

C.1 Learning Rules for Multinomial RBM

Figure 4.2 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:

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

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, [57] 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:

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

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

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. 3.1

[2] 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. 3, 3.1, 3.2.1, 4.1, 4.4.1, 4.5

[3] 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. 3.1

[4] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 4.3

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. 2013. 4.1

[6] 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. 3.1, B.1.4

[7] 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. B.1.1

[8] 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. 3.1

[9] 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. 3.1

[10] 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. 2.2.3

[11] Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. Joint training deep boltzmann machines for classification. *arXiv preprint arXiv:1301.3568*, 2013. 4.3

[12] 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. 2.1, 2.2.3, 2.3.2, A.1.1

[13] Geoffrey E Hinton. Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET, 1999. 4.2

[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 4.2

[15] Geoffrey E Hinton and Terrance J Sejnowski. Learning and relearning in boltzmann machines. *Cambridge, MA: MIT Press*, 1:282–317, 1986. 4.2

[16] 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. 4.2, 4.3

[17] Georey Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010. URL http://www.cs.toronto.edu/~{}hinton/absps/guideTR.pdf. 4.2, 4.7.1, 4.7.1

[18] 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. B.1.1

[19] 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. 3.1

[20] 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. 2.1

[21] 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. 2.1, 2.2.1, 2.4

[22] 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. 4.1, 4.5

[23] 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. 3.1

[24] 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. 2.1, 2.2.1

[25] 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. 2.1

[26] 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. 4.1, 4.5

[27] Volodymyr Kuleshov and Doina Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*, 2010. 2.1

[28] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008. 4.2

[29] Yann LeCun et al. Generalization and network design strategies. *Connections in Perspective. North-Holland, Amsterdam*, pages 143–55, 1989. 4.4

[30] 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. 4.4.1

[31] S. W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286(5438):295–299, Oct 1999. 3.1

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

[33] 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. 3, 3.1, 3.3, 5.1

[34] 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. 4.3

[35] Grégoire Montavon, Mikio L Braun, and Klaus-Robert Müller. Kernel analysis of deep networks. *The Journal of Machine Learning Research*, 12:2563–2581, 2011. 4.3

[36] Grégoire Montavon, Mikio L Braun, and Klaus-Robert Müller. Deep boltzmann machines as feed-forward hierarchies. In *International Conference on Artificial Intelligence and Statistics*, pages 798–804, 2012. 4.3, 4a

[37] 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. 3.1

[38] 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. 3.1

[39] 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. 2.2.2

[40] Christos Petropoulos. Data management plan for monogram bioscience. Nature Precedings, 2011. URL `http://dx.doi.org/10.1038/npre.2011.5668.1`. 2.1, 2.3.1, 2.4, A.1.1

[41] 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. 3.1

[42] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International*

Conference on Artificial Intelligence and Statistics, pages 448–455, 2009. 4.2, 4.3, 4.3, 4.4, 4a

[43] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010. 4.3

[44] 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. 4.2

[45] Mark W Schmidt, Kevin P Murphy, Glenn Fung, and Rómer Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, volume 1, page 1. Citeseer, 2008. 4.4.1

[46] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012. 4.7.1

[47] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239, 2012. 4.3

[48] Nitish Srivastava, Ruslan R Salakhutdinov, and Geoffrey E Hinton. Modeling documents with deep boltzmann machines. *arXiv preprint arXiv:1309.6865*, 2013. 4.3

[49] 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. 3.1, B.1.1

[50] Kevin Swersky, Daniel Tarlow, Ilya Sutskever, Ruslan Salakhutdinov, Richard S Zemel, and Ryan P Adams. Cardinality restricted boltzmann machines. In *NIPS*, pages 3302–3310, 2012. 4.4.1

[51] Shigeki Takeda, Shiro Kadowaki, Tatsuya Haga, Hirotomo Takaesu, and Shigeki Mitaku. Identification of g protein-coupled receptor genes from the human genome se-

quence. {*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`. 3.1

[52] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009. 4.2

[53] 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. 3.1

[54] 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. 4.1, 4.5

[55] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 2.1, 2.2.2

[56] 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. 2.1, 2.2.3, 2.3.3, 2.4

[57] 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. 4.2, C.1