University of California
Santa Barbara

Mining Patterns and Networks from Sequence Data

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Honglei Liu

Committee in charge:

Professor Xifeng Yan, Co-Chair
Professor Kenneth S. Kosik, Co-Chair
Professor Ambuj Singh

September 2017

The Dissertation of Honglei Liu is approved.

Professor Ambuj Singh

Professor Kenneth S. Kosik, Committee Co-Chair

Professor Xifeng Yan, Committee Co-Chair

September 2017

Mining Patterns and Networks from Sequence Data

To my parents,

Shujie Yan and Dongfang Liu,

who provided me with unconditional love and support

Acknowledgements

My deepest gratitude goes to my Ph.D. advisor, Prof. Xifeng Yan. I'm grateful to have benefited tremendously from his broad knowledge in the field, his honesty and his attention to details. He is also a man with endless passion. He taught us to aim high and achieve big. I have always felt enlightened when he generously shared his invaluable advice for life and career. He inspired me in so many ways that helped shape who I am today. I'm truly thankful for his guidance.

I would also like to thank Prof. Kenneth S. Kosik, whom I have worked with since the start of my Ph.D. studies. Without his help, I wouldn't have accomplished the works presented in this dissertation. He helped me build a bridge between computer science and neuroscience. Furthermore, I want to thank Prof. Ambuj K. Singh for serving on my committee and providing insightful comments on my dissertation and defense.

I'm very lucky to have worked with many brilliant people. I owe my thanks to my collaborators: Dr. Fangqiu Han, Yu Su, Dr. Bian Wu, Prof. Huan Sun, Izzeddin Gur, Semih Yavuz, Dr. Hongjun (Robin) Zhou, Dr. Kristofer Bouchard, Daniel Bridges, Connor Randall, Prof. Paul Hansma, Prof. Yasser Roudi, Prof. Sara A. Solla and Dr. Ken Tovar. Not only did they help me significantly in every piece of my work, but also I learned a lot from them during our collaborations. I also want to thank my lab mates and friends who have inspired me and supported me during this Ph.D. journey.

During my two internships, it was a great pleasure to work with my fantastic mentors: Dr. Jocelyne Bruand and Oleksandr Voietsa. They were kind and patient to me. The knowledge and experiences I learned from them are lifelong assets.

Finally, I want to give special thanks to my girlfriend, Ying Yu, who loved and supported me throughout this long Ph.D. journey. It wasn't always happiness on this journey, but her companionship helped me get through these difficulties.

Curriculum Vitæ
Honglei Liu

Education

2012-2017	University of California, Santa Barbara, USA Ph.D. in Computer Science *Advisor*: Prof. Xifeng Yan *Research Areas*: sequence mining, active learning, deep learning
2010-2012	Northeastern University (NEU), Shenyang, China M.S. Computer Science
2006-2010	Northeastern University (NEU), Shenyang, China B.S., Computer Science

Publications

Yu Su*, **Honglei Liu***, Semih Yavuz, Izzeddin Gur, Huan Sun, Xifeng Yan, Global Relation Embedding for Relation Extraction. (*: Equal Contribution), *preprint arXiv*, 2017.

Honglei Liu, Bian Wu, Active Learning of Functional Networks from Spike Trains, In *SDM*, 2017.

Honglei Liu, Fangqiu Han, Hongjun Zhou, Xifeng Yan, Kenneth S. Kosik, Fast Motif Discovery in Short Sequences, In *ICDE*, 2016.

Xiaochun Yang, **Honglei Liu**, Bin Wang, ALAE: Accelerating Local Alignment with Affine Gap Exactly in Biosequence Databases, In *VLDB*, 2012.

Honglei Liu, Xiaochun Yang, Bin Wang, Rong Jin, Approximate Substring Query Algorithms Supporting Local Optimal Matching, *Journal of Frontiers of Computer Science and Technology*, 2011.

In Progress

Honglei Liu, Daniel Bridges, Connor Randall, Sara A. Solla, Bian Wu, Ken Tovar, Paul Hansma, Xifeng Yan, Kenneth S. Kosik, Kristofer Bouchard, Prediction and Validation of Inhibitory Connections from Spike Trains

Patents

Honglei Liu, Jocelyne Bruand, Off-target Detection Tool for Strings with Multi-level Cache, *US62/395,288*, pending.

Honglei Liu, Xiaochun Yang, Jiaying Wang, Bin Wang, Biological Sequence Local Comparison Method Capable of Obtaining Complete Solution, *CN102750461*, issued on April 22, 2015.

Honglei Liu, Xiangfei Meng, An Electric Automobile Battery Replacing Device, *CN202089042*, issued on Dec. 28, 2011.

Experience

Summer 2016	Intern, Facebook, Menlo Park, CA *Topic*: Correlation Finding and Indexing for Extremely Large Scale Time Series Data
Summer 2015	Bioinformatics intern, Illumina, San Diego, CA *Topic*: Fast Specificity Checking for Multiplex PCR Primer Design

Honors and Awards

VLDB Conference Travel Award, 2012

Outstanding Graduate Student, Liaoning Province, China, 2011

Jian Long Scholarship, Northeastern University, 2011

Suzhou Industrial Park Scholarship, Northeastern University, 2009

Outstanding Undergraduate Student, Northeastern University, 2009

2^{nd} Prize of National Undergraduate Electronic Design Contest, China, 2009

1^{st} Prize of the 4th National Undergraduate Intelligent Car Contest, China, 2009

Abstract

Mining Patterns and Networks from Sequence Data

by

Honglei Liu

Sequence data are ubiquitous in diverse domains such as bioinformatics, computational neuroscience, and user behavior analysis. As a result, many critical applications require extracting knowledge from sequences in multi-level. For example, mining frequent patterns is the central goal of motif discovery in biological sequences, while in computational neuronal science, one essential task is to infer causal networks from neural event sequences (spike trains). Given the wide application of pattern and network mining tools for sequence data, they are facing new challenges posted by modern instruments. That is, as large scale and high resolution sequence data become available, we need new methods with better efficiency and higher accuracy.

In this dissertation, we propose several approaches to improve existing pattern and network mining tools to meet new challenges in terms of efficiency and accuracy. The first problem is how to scale existing motif discovery algorithms. Our work on motif discovery focuses on the challenge of discovering motifs from a large scale of short sequences that none of existing motif finding algorithms can handle. We propose an anchor based clustering algorithm that could significantly improve the scalability of all the existing motif finding algorithms without losing accuracy at all. In particular, our algorithm could reduce the running time of a very popular motif finding algorithm, MEME, from weeks to a few minutes with even better accuracy.

In another work, we study the problem of how to accurately infer a functional network from neural recordings (spike trains), which is an essential task in many real world applications such as diagnosing neurodegenerative diseases. We introduce a statistical tool that could be used to accurately identify inhibitory causal relations from spike trains. While most of existing works devote their efforts on characterizing the statistics of neural spike trains, we show that it is crucial to make predictions about the response of neurons to changes. More importantly, our results are validated by real biological experiments with a novel instrument, which makes this work the first of its kind.

Furthermore, while most existing methods focus on learning functional networks from purely observational data, we propose an active learning framework that could intelligently generate and utilize interventional data. We demonstrate that by intelligently adopting interventional data using the active learning models we propose, the accuracy of the inferred functional network could be substantially improved with the same amount of training data.

Contents

Chapter 1

Introduction

1.1 Mining Patterns

Mining patterns from sequence data is an important problem in data mining research. In this dissertation, we focus on motif discovery which is a fundamental task to many biological problems such as antibody biomarker identification [1]. Recent advances in instrumental techniques make it possible to generate thousands of protein sequences at once, which raises a big data issue for the existing motif finding algorithms: They either work only in a small scale of several hundred sequences or have to trade accuracy for efficiency. In this dissertation, we demonstrate that by intelligently clustering sequences, it is possible to significantly improve the scalability of all the existing motif finding algorithms without losing accuracy at all. An anchor based sequence clustering algorithm (ASC) is thus proposed to divide a sequence dataset into multiple smaller clusters so that sequences sharing the same motif will be located into the same cluster. Then an existing motif finding algorithm can be applied to each individual cluster to generate motifs. In the end, the results from multiple clusters are merged together as final output. Experimental results show that our approach is generic and orders of magnitude faster than traditional motif finding algorithms. It can discover motifs from protein sequences in the scale that no existing algorithm can handle. In particular, ASC reduces the running time of a very popular motif finding algorithm, MEME[2], *from weeks to a few minutes* with even better accuracy.

1.2 Mining Networks

In addition to mining patterns from sequence data, many applications also require mining networks. For example, inferring functional networks from spike trains is a fundamental problem in neuroscience, which could help us understand how neuronal signals propagate in local networks. Spike trains recorded with Multi-electrode Arrays (MEAs) have been widely used to study behaviors of neural connections. Studying the dynamics of neuronal networks requires the identification of both excitatory and inhibitory connections. The detection of excitatory relationships can be inferred by characterizing the statistical relationships of neural spike trains. However, the identification of inhibitory relationships is more difficult: distinguishing endogenous low firing rates from active inhibition is not obvious. In this dissertation, we propose an *in silico* interventional procedure that could make predictions about the effect of stimulating or inhibiting single neurons on the other neurons, and thereby gives us the ability to accurately identify inhibitory causal relationships. In addition, we have developed a Neural Circuit Probe (NCP) that can deliver drugs transiently and reversibly on individually identified neurons to assess their contributions to the neural circuit behavior. With the help of NCP, three inhibitory connections identified by our *in silico* modeling were validated through real interventional experiments.

While the *in silico* interventional procedure we have proposed could help us identify inhibitory connections, we also need a unified framework to guide the interventional experiments and incorporate both observational and interventional data seamlessly. However, most of existing works focus on inferring the functional network purely from observational data, which could lead to undiscovered or spurious connections. We demonstrate that by adopting experimental data with interventions applied, the accuracy of the inferred network can be significantly improved. Nevertheless, doing interventions in real experiments is often expensive and must be chosen with care. Hence, in this dissertation, we design an active learning framework to iteratively choose interventions and learn the functional network. In particular, we propose two models, the variance model and the validation model, to effectively select the most informative interventions. The variance model works best to reveal undiscovered connections while the validation model has the advantage of eliminating spurious connections. Experimental results with both synthetic and real datasets show that when these two models are applied, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

Chapter 2

Fast Motif Discovery in Short Sequences

2.1 Introduction

Motif discovery, finding sequence patterns from a set of protein sequences, is a very basic problem in many critical biological and medical applications. It has been extensively studied for more than a decade. For example, it is widely used to identify transcription factor binding sites (TFBS) [3] and antibody biomarkers [1]. Studying TFBS could help us learn the mechanisms that regulate gene expression, while antibody biomarkers are useful for diagnosis of diseases.

Figure 2.1 shows an example of protein sequences. In these sequences, there are subseqeuces that are almost identical to each other with only a few mismatches. When aligning these subsequences together, we can extract a sequence pattern, which is also known as motif. A motif could be represented as a consensus string or a model describing probabilities of characters appearing at different positions. A data set usually contains more than one motifs which makes it hard to deal with even for a small number of sequences.

```
        APFSELREIMHSYRG
         PFSEEAYWHVGGMKA
LEWFESSGVPFSARS
 RGIGSTLKPFSATRD
        ATFSARWSNMVPDLR
         CFSELPFSVWTPKAC
         PFTEAGITADMWAWV
```

Figure 2.1: A Motif in Multiple Protein Sequences

Advances in instrumental techniques like Random Peptide Libraries (RPLs) [1] that generate massive sequences with complex alphabets, e.g., protein sequences, post a Big Data challenge for motif finding algorithms. For example, it takes MEME [2], a very popular motif finding algorithm based on Expectation Maximization (EM), almost two weeks to finish running for a data set with 10k sequences. Several algorithms such as DREME [4], MEME-ChIP [5] and STEME [6] adopted combinatorial approaches to improve the speed of MEME. Unfortunately, they can only work with DNA sequences (4 types of alphabets, A,G,C,T) as they either do exhaustive search or rely on index structure like suffix tree. A recently published algorithm MUSI [7], faster than MEME, can be applied to protein sequences (20 types of amino acids). Unfortunately, its accuracy is far below MEME according to our experiments.

To the best of our knowledge, there is no algorithm that could work with $> 10k$ sequences with a complex alphabet set and achieve comparable accuracy with MEME. Rather than developing another motif finding algorithm to outperform MEME, we propose a new strategy, that is pre-processing sequences with clustering, to divide the data into multiple small clusters and run existing motif finding algorithms on each cluster. This strategy has several advantages. First, it could be used in any existing motif finding algorithm. Second, by limiting the input to only a subset of sequences, those time-consuming algorithms may finish in a reasonable amount of time. Figure 2.2 shows a sketch of the clustering and motif discovery pipeline. Sequences are first grouped into clusters before a motif finding algorithm is applied. Motifs discovered from each cluster are merged together and delivered to users.

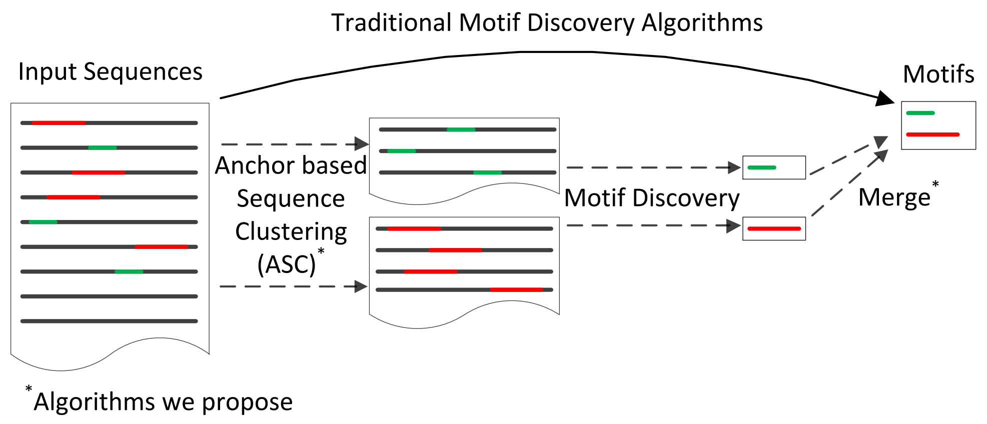

Traditional Motif Discovery Algorithms
Input Sequences
Motifs
Anchor based Sequence Clustering (ASC)*
Motif Discovery
Merge*
*Algorithms we propose

Figure 2.2: Clustering Sequences for Motif Discovery

Now the key problem is how to cluster the sequences. Could the problem be solved if we just arbitrarily divide the sequences into several clusters (Partitioning)? The answer is no since this method will miss most of low frequent motifs. For the same reason, randomly sampling a subset of sequences will not work either. Another straightforward approach is clustering sequences based on their overall similarities, for instance, K-means with edit distance as similarity measure. This will not work for the reason that motifs only appear in subregions of sequences. Table 2.1 compares the number of motifs found by aforementioned methods on a real data set. It is clear that direct sampling, partitioning, and K-means do not work well though they significantly reduce the runtime.

Can we compare sequences based on their most similar subsequences, e.g., longest common subsequence (LCS)? This solution is intuitive but not scalable: Calculating LCS of two sequences is nontrivial and the number of comparisons could be quadratic

Table 2.1: Comparisons between different methods on a real dataset with 11,642 protein sequences

Methods	# of motifs found	Runtime (Min.)
MEME	20	two weeks
Sampling	11	79
Partitioning	5	9
K-means	14	32
ASC	24	6

in order to select a cluster center. In this chapter, we propose an *anchor based sequence clustering* (ASC) algorithm that could efficiently identify sequences potentially containing the same motif. ASC could bypass the problem of directly calculating LCSs and identify cluster centers with only one scan. In this algorithm, each sequence is decomposed into a set of *anchors* which are similar to gapped q-grams in other literatures, but with variable shapes. Sequences are clustered based on anchors they contain to avoid pair-wise sequence comparisons. In particular, these anchors are not randomly selected. They are from the most significant ones in the dataset and then iteratively refined. Afterwards, a set of sequences are sampled from each cluster and provided to an existing algorithm to find motifs. Table 2.1 shows the superior performance of ASC. We are able to reduce the running time of MEME from weeks to less than 10 minutes and discover even more motifs than MEME.

The main contributions of this work are summarized as follows.

- We introduce a new strategy for speeding up motif discovery by pre-processing sequences with clustering. This strategy is generic for any existing motif finding algorithm and a post-processing pipeline consisting of sampling, filtering and merging is also built to discover significant motifs.

- We propose an anchor based sequence clustering (ASC) algorithm that could efficiently group sequences containing the same motif together. As far as we know, ASC is the first anchor (gapped q-grams with variable shapes) based clustering algorithm.

- We provide theoretical analysis for our anchor-based similarity measure and illustrate that it can help check if two sequences contain the same motif with high accuracy.
- We perform extensive experiments with both synthetic and real data sets. The results show that ASC can discover motifs from protein sequences in the scale that none of the existing algorithms can handle.

2.2 Preliminaries

Let $S = \{s_1, s_2, ..., s_N\}$ denote a set of N sequences over a fixed alphabet set $\Sigma = \{\beta_1, \beta_2, ..., \beta_{|\Sigma|}\}$ with $|\Sigma|$ symbols, e.g., $|\Sigma| = 20$ for protein sequences. Let $s[i, j]$ denote the subsequence between the i^{th} and j^{th} (both inclusive) character of sequence s. Usually, the input sequences have the same length, so we use l to represent the length of the sequences. Let w be the length of a motif m. We use a vector $\Theta = (\theta_1, \theta_2, ..., \theta_{|\Sigma|})$ to represent the background model which describes probabilities of seeing a character β_i appearing at any position in all the input sequences. θ_i is called the background probability of β_i, which can be calculated as the number of occurrences of β_i divided by the total number of characters $N \times l$. Table 4.1 summarizes some common notations that we are going to use in this chapter.

Intuitively, a motif is a sequence pattern that repeatedly appears in S. The colored subsequences in Figure 2.1 follow a common sequence pattern as they differ from each other with only a few mismatches. That is, their Hamming distance, which is the number of different characters when aligned together, is small.

Table 2.2: Notations

Notations	Description
S	A set of input sequences
N	Total number of input sequences
Σ	Alphabet set
l	Length of input sequences
m	A motif
w	Width of the motif
θ_i	Background probability of β_i

Definition 1 (Local Hamming Similarity) *Given two sequences s_1, s_2 and motif width w, the local hamming similarity between s_1 and s_2 under w is*

$$lh(s_1, s_2, w) = \max_{\substack{0\le i\le l_1-w \\ 0\le j\le l_2-w}} h(s_1[i, i+w-1], s_2[j, j+w-1]),$$

where l_1 and l_2 are the length of s_1 and s_2 respectively, and $h(s_1[i, i+w-1], s_2[j, j+w-1])$ is the hamming similarity defined as the number of exactly matched characters between $s_1[i, i+w-1]$ and $s_2[j, j+w-1]$).

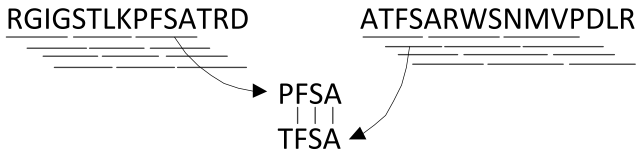


Figure 2.3: Local Hamming Similarity

Figure 2.3 shows the local hamming similarity between two sequences is 3 when $w = 4$.

A motif is a sequence pattern extracted from a set of similar subsequences. One way of representing motifs is using *consensus string.*

Definition 2 (Consensus String) *Given n similar subsequences $S = \{s_1, s_2, ..., s_n\}$, the consensus string is a sequence m whose i^{th} character $m[i] = \arg\max_{\beta\in\Sigma} f(i, \beta)$, where $f(i, \beta)$ is the number of times that character β appears in position i for all $s \in S$.*

For the example shown in Figure 2.1, the consensus string representation of the motif is *PFSE*. There are other representations such as position weight matrix (PWM) [2] (Section 2.5). Our proposed algorithm can work with all the representations.

An occurrence of a motif is a subsequence that matches the motif (approximately). As shown in Figure 2.1, all the colored subsequences are occurrences of the motif, *PFSE*. Given a motif, we usually need to check which sequences contain this motif. Assuming the motif m is represented as a consensus string, we define the occurrence of motifs as follows.

Definition 3 (Occurrence of Motifs) *Given a motif m of length w, a sequence s of length w is an occurrence of m if their hamming similarity is equal to or greater than t. We refer to the occurrence of a motif in a sequence as motif region.*

Similarly, we could say a sequence s contains an occurrence of motif m with similarity t if their local hamming similarity $lh(s, m, w) \geq t$. How to choose the value of t is vague. A statistical approach [8] can bypass this problem by assigning a p-value to each sequence. A sequence is considered significant when it has a very small p-value, e.g., less than 0.05.

Motif Discovery: Given a set of sequences S, the task of motif discovery is to identify significant subsets of S that contain motifs and extract them. In this chapter, we focus on the setting where each sequence is short and contains at most one motif. This is a widely used setting for motif discovery from peptide sequences [7]. Even under this simplified setting, none of the existing algorithms works well for a large number of sequences.

2.3 Anchor based similarity

As briefly discussed, clustering sequences based on motifs they contain has several advantages and could lead us to an efficient solution without developing yet another motif finding algorithm. However, the dilemma is given a sequence dataset, we do not know the motifs beforehand. Therefore, we need to develop a measure that is able to cluster the data without extracting motifs first.

Definition 4 (Anchor) *A q-anchor consists of q characters* $(\beta_{i_1}, \beta_{i_2}, ..., \beta_{i_q})$ *drawn from* Σ *with replacement, and* β_{i_j} *appears before* $\beta_{i_{j+1}}$ *with gaps between them.*

Definition 5 (Anchor based Similarity) *Given two sequences* s_1, s_2 *and their corresponding anchor sets* A_1, A_2, *the anchor based similarity between* s_1 *and* s_2 *is* $|A_1 \cap A_2|$ *which is the number of common anchors they share.*

For example, the set of 2-anchors for $PFSE$ is {*PF, FS, SE, P_S, F_E, P_ _E*}. The concept of anchor is similar to gapped q-gram [9] except that it has variable shapes. Instead of using local hamming similarity, we propose to represent a sequence as a vector of anchors and use the number of common anchors to cluster sequences. In order to make anchor-based clustering work, we need to build a connection between anchor-based similarity and sequences that contain a motif.

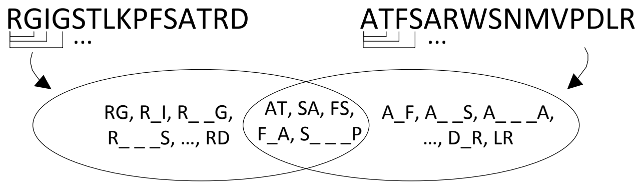

RGIGSTLKPFSATRD
...
ATFSARWSNMVPDLR
...
RG, R_I, R_ _G, R_ _ _S, ..., RD
AT, SA, FS, F_A, S_ _ _P
A_F, A_ _S, A_ _ _A, ..., D_R, LR

Figure 2.4: Common 2-Anchors

Figure 2.4 shows two sequences and their 2-anchors. Among all the five common 2-anchors, {*FS, SA, F_A*} are from motif regions shown in Figure 2.3. It is possible for two random sequences to contain some common anchors. However this probability is relatively small compared to the cases where two sequences contain the same motif.

We first analyze the probability of two sequences sharing at least d common anchors in these two cases as follows. Then we will show that when q-anchors are selected first, the chance for a random sequence of length l to contain d q-anchors is much smaller.

Here and throughout this section, we assume in a random sequence model, called *the background model*, all characters in the alphabet set Σ appear with equal probability. This assumption simplifies the presentation of theorems. We say a sequence pair contains an anchor if both sequences in the pair contain the anchor.

Theorem 1 *Given a pair of sequences of length l drawn from the background model, the probability that this sequence pair contains at least d common q-anchors is at most $\frac{1}{d|\Sigma|^q}\sum_{i=q}^{l}\binom{i-2}{q-2}(l-i+1)^2$. When $q=2$, this probability is upper bounded by $\frac{l^3}{3d\Sigma^{2l-2}}$.*

We prove this theorem by following a counting argument: we first focus on the number of common anchors shared by a sequence pair and compute the sum, denoted as T_{ca}, of this number among all sequence pairs; then the number of sequence pairs that contain at least d common anchors is at most $\frac{T_{ca}}{d}$; thus the probability of one sequence pair containing at least d common anchors will be upper bounded by $\frac{T_{ca}}{d}$ divides by the total number of possible sequence pairs T_s. We compute T_{ca} and T_s in the following two lemmas.

Lemma 1 *The total number of possible length l sequence pairs $T_s = |\Sigma|^{2l}$. Those sequence pairs appear with equal probability in the background model.*

Proof: Each position in length l has $|\Sigma|$ possible characters. Then the total number of possible length l sequence pairs $T_s = |\Sigma|^{2l}$, as one sequence pair has $2l$ positions. Note that all characters in alphabet set Σ appear with equal probability in the background model. Therefore all sequence pairs appear with equal probability. ■

Lemma 2 *Let S be the set of all possible sequences and let $N_q(s_1, s_2)$, s_1, $s_2 \in S$, be the number of common q-anchors shared by s_1 and s_2. Then $T_{ca} = \sum_{s_1, s_2 \in S} N_q(s_1, s_2) = |\Sigma|^q \cdot \sum_{i=q}^{l} \binom{i-2}{q-2}(l-i+1)^2$.*

Proof: Firstly, for each length k q-anchor, here we use a length 3 anchor A_B as an example, we count the number of possible positions in one sequence that contains this anchor. If a sequence s_1 contains anchor A_B, this anchor could appear in $l-k+1 = l-2$ positions in this sequence. Then if two sequences in a sequence pair both contain anchor A_B, this anchor could appear in $(l-k+1)^2$ positions pairs in this sequence pair. Secondly, there are $|\Sigma|^q$ possible choices for the characters in a q-anchor. Since the length of sequences is l, the length of a q-anchor including gaps could range from q to l. And for each possible length i, there are $\binom{i-2}{q-2}$ ways to place the $q-2$ characters other than the start and end of the q-anchor. Therefore the total number of anchor matches between all possible sequence pairs will be $|\Sigma|^q \cdot \sum_{i=q}^{l} \binom{i-2}{q-2}(l-i+1)^2$. ■

Now we are ready to show the proof of Theorem 1.

Proof: Clearly, the number of sequences pairs which contain at least d q-anchors is upper bounded by the number of common q-anchors T_{ca} divides by d. Then the probability of a sequence pair containing at least d q-anchor is $T_{ca}/(dT_s) = \frac{1}{d|\Sigma|^{2l-q}} \sum_{i=q}^{l} \binom{i-2}{q-2}(l-i+1)^2$. Specially, when $q = 2$, $T_{ca}/(dT_s) = \frac{1}{d|\Sigma|^{2l-q}} \sum_{i=q}^{l} \binom{i-2}{q-2}(l-i+1)^2 \leq \frac{l^3}{3d\Sigma^{2l-2}}$. ■

Corollary 1 *Given a sequence s of length l drawn from the background model and d random q-anchors, the probability of s containing all d anchors is $(\frac{\binom{l}{q}}{|\Sigma|^q \sum_{i=q}^{l} \binom{i-2}{q-2}})^d$. When $q = 2$, this probability is $(\frac{l}{2|\Sigma|^2})^d$.*

Proof: It is clear to see that sequence s contains at most $\binom{l}{q}$ different anchors. By Lemma 4 (Section 2.4.1), the number of possible q-anchors is $|\Sigma|^q \sum_{i=q}^{l} \binom{i-2}{q-2}$. Therefore the probability of s containing one random q-anchor is $\frac{\binom{l}{q}}{|\Sigma|^q \sum_{i=q}^{l} \binom{i-2}{q-2}}$. As these d anchors are independently selected, the probability of s containing all d anchors is $(\frac{\binom{l}{q}}{|\Sigma|^q \sum_{i=q}^{l} \binom{i-2}{q-2}})^d$.

■

Corollary 1 shows that the chance for a random sequence to contain multiple q-anchors from a motif could be very low. If we are able to extract a few q-anchors from potential motifs, we can quickly remove those sequences that do not contain any motif and also group those sequences that contain the same motif together. This property will be used in the design of anchor-based clustering in Section 2.4. Theorem 1 and Corollary 1 together show that the chance for two sequences accidently sharing d common anchors is much higher than the chance for one sequence containing d pre-selected anchors.

Now we move to estimate the probability of discovering common anchors in motif regions of a pair of sequences. Let s_1 and s_2 be two sequences which contain motif m of width w with similarity t. Common anchors in s_1 and s_2 can be either anchors in motif m or other anchors happen to be contained in both sequences. However, anchors in motif m more likely appear as common anchors than other random anchors. We here use the probability of discovering common anchors only from the motif as a lower bound. Denote the motif region in s_1 and s_2 as $o_1 = o_{11}o_{12}\ldots o_{1w}$ and $o_2 = o_{21}o_{22}\ldots o_{2w}$, respectively. The following lemma shows that we could compute the number of common anchors in o_1 and o_2 by simply counting the number of identical characters found in the same positions of o_1 and o_2.

Lemma 3 *Given two sequences s_1 and s_2 and their corresponding motif occurrence o_1 and o_2, let $I = \{i|\delta(o_{1i}, o_{1i}) = 1\}$, where $\delta(\cdot,\cdot)$ returns 1 if its two arguments are the same and returns 0 otherwise. Then I is the set of positions at which o_1 and o_2 contain the same character. Let $k = |I|$ is the cardinality of set I, then the number of the common anchors in the motif regions of s_1 and s_2 is at least $\binom{k}{q}$.*

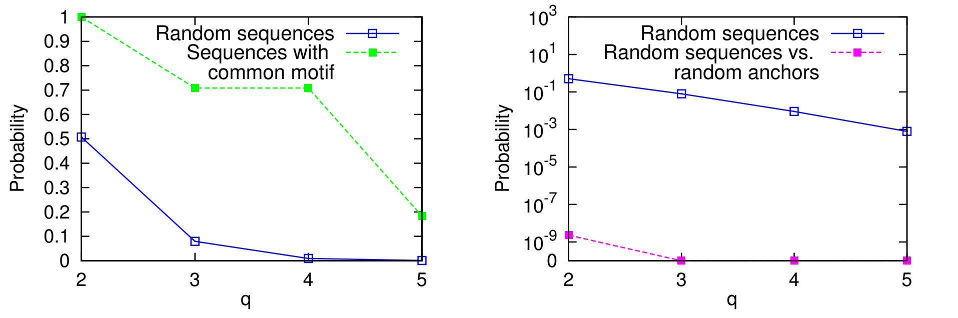
Proof: It is easy to see that characters at the ith position of o_1 and o_2 are the same if $i \in I$. Then any q-anchor composed by characters at q positions drawn from set I without replacement is a common anchor of of s_1 and s_2. Therefore the number of the common anchors in the motif regions of s_1 and s_2 is at least the number of draws, which is $\binom{k}{q}$. ■

The above theorem shows that if we want to compute the probability of discovering d common anchors in o_1 and o_2, we could (1) find the smallest integer k which satisfies $\binom{k}{q} \geq d$, and (2) compute the probability of having the same characters in k positions of o_1 and o_2. We apply this idea in the following theorem.

Theorem 2 *Let k be the smallest integer which satisfies $\binom{k}{q} \geq d$. Given two sequences s_1 and s_2 which contain motif m of length w with similarity t, the probability that s_1 and s_2 share at least d common q-anchors is at least $\frac{\sum_{i=0}^{k-1} \binom{t}{i}\binom{w-t}{t-i}}{\binom{w}{t}}$.*

Proof: We only need to compute the probability of having the same characters in k positions of o_1 and o_2. Motif occurrence o_1 share the same character with motif m at at least t positions. Without loss of generality, we assume that o_1 and m share the same characters at the first t positions. Motif occurrence o_2 also share the same character with motif m at at least t positions. Consider event $A_i = \{$When drawing t positions without replacement from a length w motif, i positions are chosen in the first t positions and $t-i$ positions are chosen from the rest $w-t$ positions$\}$. The probability of o_1 and o_2 share the same characters at no less than k positions is $\sum_{i=k}^{t} \Pr\{A_i\}$, as we assume that o_1 and m share the same characters at the first t positions. Note that the probability of event A_i is $\frac{\binom{t}{i}\binom{w-t}{t-i}}{\binom{w}{t}}$. Therefore, the probability of o_1 and o_2 sharing at least d common q-anchors is lower bounded by $\sum_{i=k}^{t} \Pr\{A_i\} = \frac{\sum_{i=k}^{t} \binom{t}{i}\binom{w-t}{t-i}}{\binom{w}{t}}$. ■

In order to investigate how different values of q will affect our anchor based similarity measure, we vary q and calculate the probabilities according to Theorem 1, Corollary 1



(a) Probabilities from Theorem 1 and Theorem 2 (b) Probabilities from Theorem 1 and Corollary 1

Figure 2.5: The probabilities that two random sequences and two sequences containing the same motif share at least d common q-anchors, and the probability that a random sequence contains d random q-anchors when $l = 15, w = 10, t = 7, d = 5$.

and Theorem 2. Figure 2.5 shows the probabilities when we fix $l = 15, w = 10, t = 7, d = 5$ and range q from 2 to 5. As we can see, even though all the probabilities drop when q is increased, for two sequences containing the same motif, their probability of sharing d common q-anchors is always much higher than sequences that are merely generated according to the background model. And given d random q-anchors, the probability that a sequence contains them is extremely low. With this property, we could use the anchor based similarity between two sequences to indicate their probability of containing the same motif and further make clustering decisions.

2.4 Anchor based clustering

Recall that there are mainly two challenges for clustering sequences based on motifs they contain: (1)Motifs are unknown beforehand, and (2) Pairwise sequence comparison shall be avoided. By adopting the concept of anchor based similarity measure, both problems can be avoided. In this section, we are going to introduce the design of anchor-based clustering.

According to our theoretical analysis in Section 3, the probability for two random sequences to have common anchors is much lower than sequences containing the same motif. Thus we derive an initial design of our algorithm as follows. A sequence is randomly picked as a cluster center, and then all the other sequences are compared with it using anchor based similarity measure. For any sequence, if its anchor based similarity with the center is equal to or greater than a threshold, it will be captured by this cluster. Repeat this for the rest of sequences until there is no sequence left. This method has two drawbacks. If we continuously choose sequences not containing any motif as cluster centers, the number of comparisons could be as large as $O(N^2)$ which is no better than pairwise comparisons. Furthermore, it might generate many small useless clusters for sequences that do not contain any motif; yet one still needs to run a motif finding algorithm on these clusters.

Running a traditional K-means algorithm on the anchor representation of sequences is also problematic. The number of comparisons could be quadratic with respect to the number of sequences in order to calculate the *mean* or *centroid* of a cluster. Hence we propose an *anchor based sequence clustering* (ASC) algorithm. ASC iteratively clusters sequences and selects a few anchors that are likely from potential motifs as cluster centers. Each sequence is first decomposed to a set of anchors and then clustered based on their corresponding anchors. Instead of using the *mean* or *centroid* of the sequences like traditional K-means, we carefully (re)select a set of anchors to represent the most distinctive features of a motif as the cluster center at each iteration. At first, K centers are initialized. In each iteration, sequences are assigned to their closest cluster and then the center of each cluster is adjusted. Not only is the center adjusted based on the new membership, but the anchor set used to represent the center is also adjusted. This process is repeated until all the clusters are stabilized.

Algorithm 1 outlines the anchor based sequence clustering algorithm. In our algorithm, a cluster center is a set of anchors, not the motif or the consensus string of sequences in the cluster.

Algorithm 1: Anchor based Sequence Clustering

```
input : A set of sequences $S$, the number of clusters $K$, the number of anchors $d$
        in a cluster center
output: $K$ clusters of sequences
for $s_i \in S$ do
    Decompose $s_i$ into a set of anchors
end
Calculate the odd score (Section 2.4.1) of all the anchors
Select top $d \times K$ anchors based on their odd scores and randomly divide them into
$K$ anchor sets; each has $d$ anchors
repeat
    Assign each sequence to a cluster
    Adjust the center of each cluster
until termination condition (Section 2.4.4)
return sequences in each cluster
```

2.4.1 Choose Initial Anchors

Lemma 4 *Given a set of sequences with length l, the maximum number of q-anchors ($q \geq 2$) that could possibly appear in it is $|\Sigma|^q \cdot \sum_{i=q}^{l} \binom{i-2}{q-2}$. It is $|\Sigma|^2 \cdot (l-1)$ when $q = 2$.*

Proof: For each possible length i, there are $\binom{i-2}{q-2}$ ways to place the $q-2$ characters other than the start and end of the q-anchor. And there are $|\Sigma|^q$ possible choices for the characters for each of q-anchor placement. Therefore the maximum number of possible q-anchors in length l sequence is $|\Sigma|^q \cdot \sum_{i=q}^{l} \binom{i-2}{q-2}$. ■

For example, if $|\Sigma| = 20$ and $l = 15$, the number of possible 2-anchors is 5,600. The number of possible q-anchors increases exponentially with q as shown in Figure 2.6. Rather than using all q-size anchors, we propose an anchor filtering method by comparing the background probability and the observed probability of an anchor.

Definition 6 (Anchor's Background Probability) *Given a q-anchor a, let t be its length including gaps. Its background probability*

$$P_{background}(a) = 1 - (1 - \prod_{\beta_i \in a} \theta_i)^{l-t+1},$$

where l is the length of sequences.

Definition 7 (Anchor's Observed Probability) *Given a q-anchor a, its observed probability*

$$P_{observed}(a) = \frac{f(a)}{N},$$

where f(a) is the number of sequences that contain the anchor a and N is the total number of sequences.

Those anchors that are over-represented are more useful for clustering sequences. So all the anchors that have $P_{observed} \lesssim P_{background}$ will be discarded.

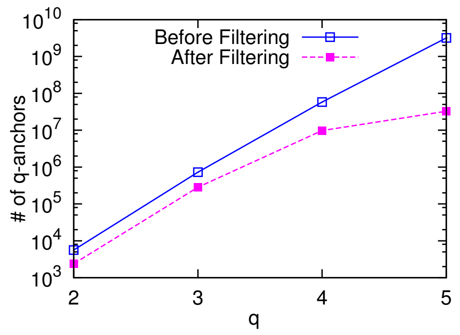


Figure 2.6: The numbers of q-anchors before and after filtering for a data set of 11,642 protein sequences

Figure 2.6 illustrates the effectiveness of our filtering method on a data set of 11,642 protein sequences. More than half of anchors are filtered out by this method. It is getting more effective with the increase of q.

Intuitively, we want to use anchors that are uniquely derived from motif regions as the **cluster center**, such that the number of anchors a sequence shares with the center could indicate how likely this sequence contains the motif. That being said, we use d anchors as the center of a cluster. These d anchors represent the most distinguishable features of a motif. Thus the clustering algorithm will cluster the sequences based on which features they contain the most.

The first step of the clustering algorithm would be initializing centers for clusters. One way of doing this is to randomly assign some anchors as the center of a cluster. But this could make our algorithm take more iterations to converge or easily get stuck in local optimal. Here, we propose an initialization method based on the *odd score* of anchors.

Recall that $P_{background}$ and $P_{observed}$ are the background probability and the observed probability of seeing a sequence containing an anchor respectively. We use the odd score to indicate how much an anchor is different from the background.

Definition 8 (Odd Score) *The odd score of anchor a is defined as the log ratio between the observed and background probabilities of a,*

$$S(a) = \log P_{observed}(a) - \log P_{background}(a).$$

For an anchor, a higher odd score means it is more distinguishable from the background, thus having a higher probability of belonging to a motif. Therefore, before initializing centers, we first calculate the odd score for anchors and then rank them accordingly. Assume we have K cluster centers (later we will remove this requirement) and each cluster center has d anchors, we select the top $K \times d$ anchors and randomly draw d anchors without replacement as each cluster's center.

2.4.2 Adjusting Anchors

Sequence Assignment. Let $\mathcal{C} = (C_1, C_2, ..., C_K)$ refer to the K centers of clusters. In each iteration. the i^{th} sequence s_i will be assigned to its closest center $C \in \mathcal{C}$. Since each sequence is represented as a set of anchors, we use A_i to denote the set of anchors of s_i. The distance between s_i and a cluster center C_k is calculated as

$$dist(s_i, C_k) = |C_k| - |A_i \cap C_k|,$$

where $|C_k|$ (i.e, d) is the number of anchors in the k^{th} cluster's center and $|A_i \cap C_k|$ is the number of common anchors between the sequence and the center. Then for s_i, the closest center C can be found by

$$C = \arg\min_{C \in \mathcal{C}} dist(s_i, C).$$

We restrict that one sequence can only belong to one cluster as in our setting one sequence contains at most one motif.

Center Adjustment. In each iteration, centers of clusters are adjusted according to the sequences assigned to each cluster. In order to select the anchors that not only belong to a motif, but also represent the motif's most distinguishable features, we propose a ranking function based on the abundance score proposed as follows.

Definition 9 (Abundance Score) *For the k^{th} cluster and an anchor a, let $f_k(a)$ be the number of sequences in this cluster containing the anchor and N_k be the total number of sequences in this cluster. The abundance score is defined as*

$$S_k(a) = \log \frac{f_k(a)}{N_k} - \log \frac{f(a)}{N}.$$

For each cluster, we select d anchors with the highest abundance scores because they are more abundant in the cluster than in the whole set of sequences. The anchors of a motif are likely the most abundant ones when most of the sequences in the cluster contain the motif.

2.4.3 Extra cluster

In addition to the K clusters, we set up an extra cluster to collect sequences that do not share any anchor with the existing cluster centers. This is essential due to two cases: (1) Not all the sequences in the data set contain motifs, and (2) If the number of sequences that contain a motif is small, the anchors of that motif might not be selected. In this case, the corresponding sequences will not belong to any cluster. If we allow the sequences in these cases to be placed into other clusters, they could potentially affect the center adjustment process.

2.4.4 Termination Condition

The goal of our clustering algorithm is to minimize the distance between sequences and their cluster center so that sequences in a cluster will have a higher probability of containing the same motif. This objective can be captured by calculating the entropy of anchors in each cluster. The entropy of the k^{th} cluster is

$$H(C_k) = -\frac{1}{|C_k|} \sum_{a \in C_k} \frac{f_k(a)}{N_k} \log \frac{f_k(a)}{N_k}.$$

We want to minimize this entropy to ensure that most of sequences in the cluster contain anchors in the center. Therefore, the entropy can be used to measure the quality of the cluster. For each cluster, we re-calculate $H(C_k)$ using this objective function after the assignment of sequences. Assume $H^i(C_k)$ and $H^{i+1}(C_k)$ are the entropy scores after the i^{th} and $(i+1)^{th}$ iteration for the k^{th} cluster. Define

$$\delta_k = H^{i+1}(C_k) - H^i(C_k).$$

We stop updating the k^{th} cluster if δ_k is smaller than a pre-defined threshold.

For each iteration, the computational complexity comes from two parts: comparing each sequence with centers and selecting top anchors according to their abundance scores. Since we have N sequences, K clusters, and d anchors per cluster, the number of sequence-anchor comparisons would be $O(d \cdot K \cdot N)$. And the time complexity of checking whether an anchor is contained by a sequence is constant since we have built inverted index with bit vectors of sequence ids for anchors. The time complexity of enumerating of all q-anchors in sequences of length l is $O(l^q \cdot N)$. Assume the total number of anchors is $|A|$, the complexity of selecting top d anchors for K clusters is $O(K \cdot |A|)$ in the worst case. The complexity of each iteration is $O(d \cdot K \cdot N + l^q \cdot N + K \cdot |A|)$ which is linear with respect to the number of sequences N, the number of clusters K, and the number of anchors d. Note that this complexity is only valid for one iteration. ASC takes multiple iterations to finish the clustering task. Though the number usually is small, it might change with respect to d.

One issue we shall pay attention to is that $|A| = |\Sigma|^q \cdot \sum_{i=q}^{l} \binom{i-2}{q-2}$ according to Lemma 4. This complexity is exponential in terms of q. Fortunately, the results are already pretty good when we set $q = 2$ according to our experiments. If not specifically mentioned, we will be using 2-anchors in our implementation.

2.4.5 Eliminating K

The cluster number K corresponds to the number of motifs within the dataset, which usually is not known beforehand. In practice, our algorithm still works when K is not equal to the true number of motifs.

1. K is smaller than the true number of motifs. In this case, one cluster may contain two or more motifs. The existing motif discovery algorithms such as MEME [2] can handle sequence datasets with multiple motifs inside.

2. K is larger than the true number of motifs. In this case, two clusters may contain the same motif. It is fine to have duplicate motifs. Section 2.5 will discuss merging motifs.

Since our goal is to find motifs in sequences, not to achieve the best clustering result, we do not have a high requirement on the quality of clusters. The above two cases show that the setting of K is not critical in our method.

To further reduce the number of parameters, we propose a recursive clustering framework to eliminate parameter K. At first the whole pool of sequences are divided into two groups using our anchor based sequence clustering algorithm. Then each group is recursively clustered until all of its children can not be further divided into more clusters. Finally, we combine all the extra clusters into one set of sequences and rerun this whole process again to generate more clusters. This recursive process runs till the size of the combined extra clusters is small enough for the traditional motif finding algorithms to handle.

Figure 2.7 gives an example where the sequences are divided into 11 clusters. As the sequences are continuously being divided into two groups, some are marked as final clusters as the size of their children is smaller than a threshold τ, which is the maximum number of sequences that existing motif finding algorithms such as MEME can handle (we set $\tau = 600$ by default). Then after 8 clusters are generated, all the extra clusters are combined together and go through the process recursively to generate 3 more clusters.

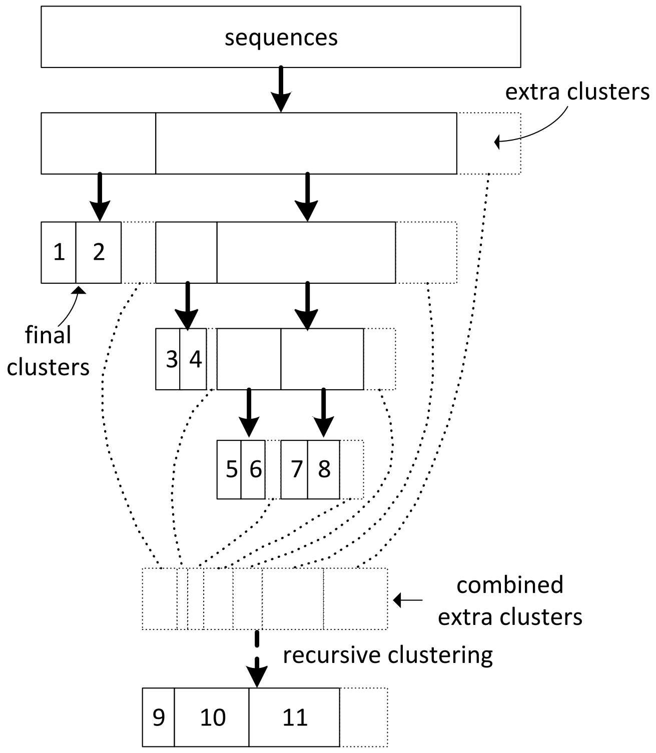


Figure 2.7: An example of the recursive clustering framework in which a set of sequences is divided into 11 groups

This framework is different from traditional hierarchical clustering as we actually have two layers: inner loop and outer loop. Inner loop carries out recursive clustering while the outer loop collects the sequences that are considered as "random" in each iteration to ensure the capture of low frequent motifs.

2.5 Motif Merging and Filtering

Similar motifs might be discovered in different clusters. Hence, we need to merge them if they are close to each other. If motifs are represented as consensus string, we could use string distance measure such as hamming distance to cluster them. If motifs are represented as position weight matrix (PWM), we could use a similarity measure based on Kullback-Leibler (KL) divergence.

Definition 10 (Position weight matrix (PWM)) *For a motif with width w, the PWM is a matrix M with size of $w \times |\Sigma|$. $M_{ij} = \frac{f(i,\beta_j)}{n}$ which is the probability of seeing the character β_j in position i of the motif.*

Since discovered motifs might have different lengths, we need to consider all the possible ways of aligning two motifs. We first segment motifs with a sliding window and compare the motif segments using the slide window. The length of the sliding window is chosen as the minimum length of motifs w_{min}.

The distance of two PWMs is defined as the minimum distance between their segments of length w_{min} and the distance of segments is defined as the average distance of their rows. Let M and M' denote the PWMs of two motifs, the distance between their i^{th} and j^{th} rows is defined as

$$D_{KL}(M_{i*}||M'_{j*}) = \sum_{t=1}^{|\Sigma|} M_{it} \log \frac{M_{it}}{M_{jt}},$$

where M_{i*} and M_{j*} are the i^{th} and j^{th} rows in the corresponding PWMs. The distance between M and M' is

$$D_{KL}(M||M') = \min_{i,j} \frac{1}{w_{min}} \sum_{t=0}^{w_{min}-1} D_{KL}(M_{(i+t)*}||M'_{(j+t)*}),$$

where $1 \leq i \leq w - w_{min} + 1$ and $1 \leq j \leq w' - w_{min} + 1$.

If $D_{KL}(M||M') \leq \theta$ (we set $\theta = 1.5$), the two motifs are considered to be similar. The sequences in their corresponding clusters will be merged together and we go through the motif discovery process again to get a unified motif.

2.6 Experiments

In this section, we perform experiments to evaluate the anchor based sequence clustering (ASC) algorithm on both synthetic and real data sets. We have built the whole pipeline of motif discovery upon MEME [10], MUSI [7], GibbsCluster [11] and ACME [12]. With ASC being applied, we refer to them as ASC+MEME, ASC+MUSI, ASC+GibbsCluster and ASC+ACME. It is worth mentioning that ASC can be adopted by any existing motif finding algorithm. (1) We first show how much ASC can actually improve existing motif finding algorithms in terms of runtime without a quality trade-off. (2) We then examine the characteristics of ASC and verify the design.

MEME is the most popular motif discovery algorithm with high discovery rate. In comparison with MEME, MUSI and GibbsCluster are the two state-of-the-art probabilistic methods that were developed to improve the runtime with an accuracy trade-off. ACME is a recently proposed parallel motif extracting algorithm based on suffix tree. It's designed to extract motifs from an extremely long sequence and reported scaling to large alphabet set. When ASC is applied to these algorithms as a pre-processing step, it can be five orders of magnitude faster than MEME and 50+ times faster than MUSI/GibbsCluster/ACME, without losing any accuracy. All the experiments are conducted on a server with 2.67GHz Intel Xeon CPU (32 cores) and 1TB RAM. The real datasets and the source code are available at `http://www.cs.ucsb.edu/~honglei/abp/download.htm`.

2.6.1 Data Sets

We compare ASC with other algorithms using both real and synthetic data sets. Only protein sequences are used in our experiments as there already exist several methods [4, 5, 6] that work very well for large scale DNA sequences. In the real dataset, we directly compare the quality of discovered motifs, while in the synthetic data, we embed some motifs and treat them as ground truth. Most of our data sets contain fairly short sequences, which is a common setting in deep sequencing phage-selected peptide datasets [13, 14] for which none of the existing tools scales well.

Real data. For the real data, we use 5 datasets of protein sequences as shown in Table 2.3. *Celiac* is from [1], consisting of 11,642 peptide sequences recognized by serum antibodies from patients with Celiac Disease. Each sequence in this data set has a length of 15. The other 4 datasets, *FXIIa*, *uPA*, *SrtA* and *PK* are from [13], containing shorter sequences ranging from 8 to 10.

Table 2.3: Real datasets

Name	# of sequences	Length of sequences
Celiac	11,642	15
FXIIa	13,945	10
uPA	5,525	9
SrtA	4,993	8
PK	2,149	8

Synthetic data. We generate synthetic data sets using the same procedure as shown in [15]. A set of N sequences with length l are generated by randomly choosing characters from the protein alphabet set Σ (total 20 amino acids). Then K sequences with length w are generated as parent motifs in the same manner. Finally, each parent motif is planted to p percent of all the sequences. And e characters of the parent motif are randomly mutated to other characters in Σ each time before being planted into a sequence. The values of w, p and e are integers randomly drawn from $[6, 12]$, $[1, 20]$ and $[0, \lceil 0.3w \rceil]$

respectively to mimic the real data set, where $\lceil 0.3w \rceil$ represents the smallest integer not less than $0.3w$. We generate different data sets by varying l, K and N. In order to test how the noise in the data set would affect our results, we also generate data sets by gradually increasing e from $\lceil 0.3w \rceil$ to $\lceil 0.9w \rceil$.

2.6.2 Motif Discovery

Real Data. Since MEME usually finds more and better motifs than MUSI and GibbsCluster, both of which trade accuracy for efficiency. We first compare the quality of motifs discovered by ASC+MEME with MEME.

For the 11,642 protein sequences in Celiac, MEME is applied to find 20 motifs from it. Then, we redo the process of motif discovery again with ASC+MEME (MEME is set to find 20 motifs in each cluster) and compare their results. We say that a motif x discovered by MEME is successfully recalled if there exists a motif y in ASC+MEME's results and $D_{KL}(M^x || M^y) \leq 1.5$ [16]. Table 2.4 shows the results of ASC+MEME with respect to different numbers of clusters. It is possible for the number of discovered motifs to be smaller than the number of recalled motifs since MEME may return duplicate motifs while we have merged those motifs. With the increasing cluster number, ASC+MEME not only recalls all the 20 motifs discovered by MEME, but also discovers new motifs. When we remove the k constraint and use the framework proposed in Section 2.4.5 to automatically generate clusters, 20 motifs can be fully recovered.

Table 2.4: Motifs returned by ASC+MEME for *Celiac*

# of clusters	# of motifs recalled	# of motifs found
10	17	16
20	18	19
40	20	22
60	20	24
w/o k	20	24

For the other 4 real datasets, we use the motifs reported in [1] as ground truth. For MEME, we set it to find 10 motifs which is already larger than the number of reported motifs. For ASC+MEME, we try $k = 10$ and the case with k removed (MEME is still set to find 10 motifs). Both MEME and ASC+MEME can recall all the motifs that are reported by [1]. Table 2.5 shows both of them run better and ASC+MEME finds more.

Table 2.5: Comparisons with the reported motifs

	FXIIa	uPA	SrtA	PK
# of reported motifs	2	2	1	1
MEME	2	4	1	2
ASC+MEME ($k = 10$)	7	4	2	2
ASC+MEME (w/o k)	7	4	2	2

Though MEME and ASC+MEME both can recall all the originally reported motifs , they differ a lot in efficiency. The only two datasets MEME can finish running within 24 hours are *SrtA* and *PK* while ASC+MEME only takes minutes to run for each of the 4 datasets as shown in Figure 2.8.

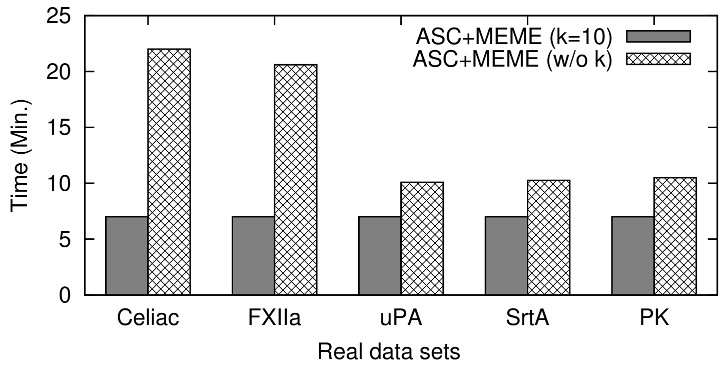


Figure 2.8: Runtime of ASC+MEME for the real data sets

Synthetic Data. In the synthetic data experiment, we count the number of planted motifs discovered by each method. For MEME and GibbsCluster, we set the number of clusters identical to the number of planted motifs. For MUSI, since we always get a segment fault whenever we set the number of motifs larger than 10, we set it to 10 through the experiments. For ACME, because it requires several additional parameters (minimum number of occurrences, maximum number of allowed mismatches and number of threads) than those probabilistic methods, we report its results separately at the end of this section.

Since the planted motifs are a kind of consensus strings, we extract the consensus strings from each probabilistic method by using the most frequent characters at each position of a motif. We say that a planted consensus string x is recalled if there exists a consensus string y in the result such that $lh(x, y, min(|x|, |y|)) \leq 2$, where $min(|x|, |y|)$ is the smaller length of x and y. We conduct experiments by fixing $d = 5$ and varying l and K, where d is the number of anchors in each cluster center, l is the length of input sequences and K is the number of planted motifs. Figure 2.9 shows the number of recalled motifs for these methods when we fix $l = 15$. We also randomly sample 10% of all the sequences in the data set and run MEME, which is referred as Sampling+MEME. Moreover, we refer to the naive method which randomly divides sequences into several clusters as random sequence clustering (RSC). RSC+MEME fails to discover most of the planted motifs. In the contrast, ASC+MEME (ASC+MUSI and ASC+GibbsCluster) outperforms the original algorithm MEME (MUSI and GibbsCluster). We then conduct experiments to test our method's robustness with respect to the sequence length. Figure 2.10 shows the number of recalled motifs when we fix $K = 20$ and vary l. Due to space constraint, we omitted the results of RSC+MEME since they follow a similar trend. The improvements of ASC over existing motif finding algorithms are consistent for different l.

Next, we check the robustness of these algorithms by gradually increasing the noise (the number of mutated characters e) when a motif is planted to sequences. We fix $l = 15$, $d = 5$, $K = 10$, $N = 1k$ and compare our results with MEME and GibbsCluster. We do not include MUSI here because it fails to run with unknown error. The reason

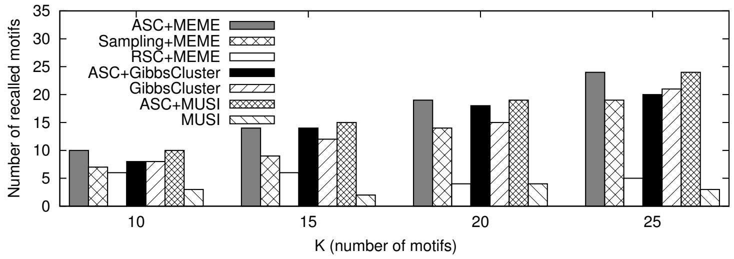


Figure 2.9: Recalled Motifs: $d = 5, N = 10k, l = 15$, varying K

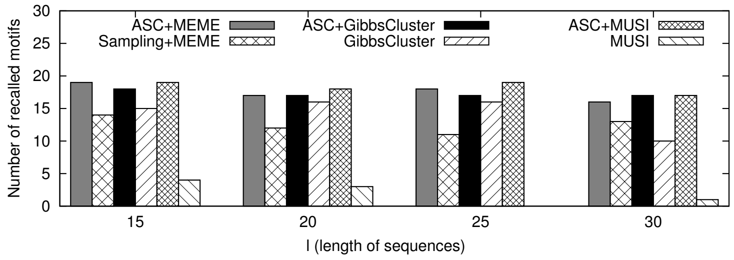


Figure 2.10: Recalled Motifs: $d = 5, N = 10k, K = 20$, varying l

we choose such a small data set is because it takes too much time for MEME to run on a larger data set. As shown in Figure 2.11, ASC+MEME's accuracy is comparable to MEME even when there are a large amount of noises. ASC+MEME even recalled more motifs than MEME when $e = 0.6w$ and $0.8w$ where w is the width of the planted motif.

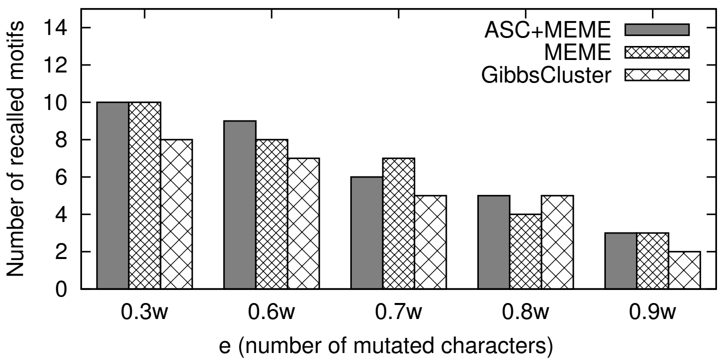
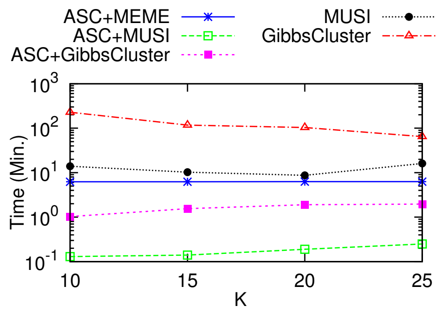
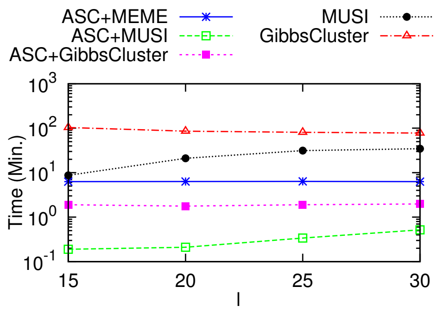


Figure 2.11: Number of recalled motifs when $l = 15, d = 5$, $K = 10$, $N = 1k$ with different e

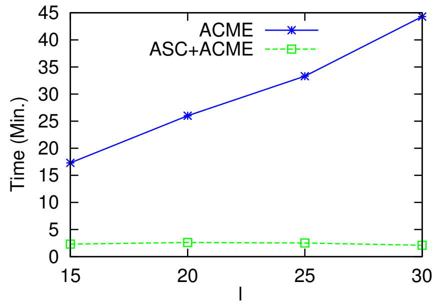
We then examine the runtime of these methods by fixing $d = 5$, $N = 10k$, and varying l and K. In Figure 2.12, we omitted the results of MEME because it takes more than two weeks to run for one data point. Figure 2.12(a) shows the overall runtime when we fix $l = 15$ and vary K. Note that the runtime of ASC+MEME (ASC+MUSI and ASC+GibbsCluster) is the cumulative running time of ASC and MEME (MUSI and GibbsCluster). ASC+MEME is much faster than MEME, MUSI and GibbsCluster, and its runtime does not change much over K. ASC+MUSI is still two orders of magnitude faster than MUSI and GibbsCluster. Figure 2.12(b) shows the runtime when we fix $K = 20$ and vary l. The increase of l does not affect ASC's consistent improvement over runtime.



(a) When $l = 15$



(b) When $K = 20$

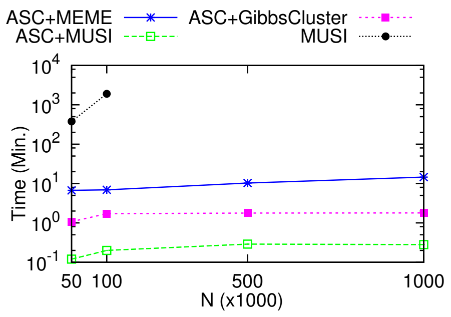


(c) When $K = 10$

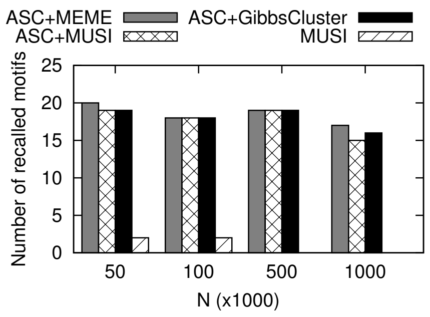
Figure 2.12: Runtime: $d = 5$,$N = 10k$

To compare ASC+ACME with ACME, we concatenate short sequences as a long sequence. We also use 3 and 100 for the maximum number of allowed mismatches and the minimum number of occurrences when $N = 10k$. Note that these parameters are set according to how we generated the synthetic data and are usually not accessible to users when working with real data. In the following experiments, we use 30 threads for ACME. When we set $d = 5, K = 10$ and vary l, both ASC+ACME and ACME can recall all the planted motifs. Moreover, as shown in Figure 2.12(c), ASC+ACME has consistent improvements over the runtime. Results with different parameters are omitted as they follow a similar trend.

Scalability. The scalability of these methods is tested by varying the number of sequences. Figure 2.13 shows the runtime and the number of recalled motifs when we vary N from 50 thousand to 1 million. Some results are omitted because we couldn't get them within a reasonable amount of time for the corresponding methods. In particular, MEME didn't finish in one month. As we can see, ASC+MEME (ASC+MUSI and ASC+GibbsCluster) takes much less time than MUSI and GibbsCluster. Moreover, ASC+MEME (ASC+MUSI and ASC+GibbsCluster) can work in some cases that MUSI and GibbsCluster can't.



(a) Runtime



(b) Recall

Figure 2.13: Scalability w.r.t. Number of Sequences

2.6.3 Properties of ASC

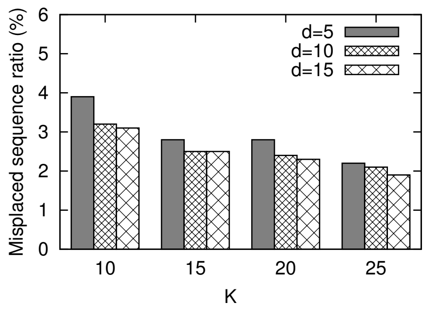
In this section, we analyze the design of ASC and check its performance using synthetic data.

Effectiveness. The goal of ASC is to group the sequences that contain the same motif into one cluster. We call a pair of sequences a misplaced pair if the two sequences contain the same motif but are placed into different clusters by ASC. We use misplaced sequence ratio (MSR) to measure the ratio of misplaced pairs. MSR is defined as

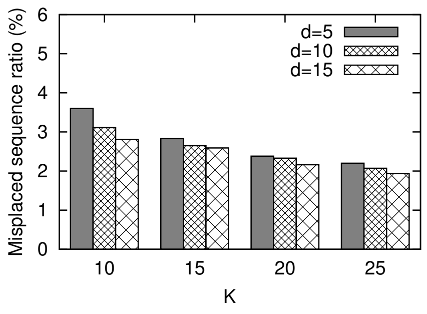
$$MSR = \frac{\#\ of\ misplaced\ pairs}{(\#\ of sequences\ containing\ motifs)^2}.$$

If a sequence contains multiple motifs, we will count it multiple times. A lower MSR indicates better clustering accuracy.

We first conduct experiments on two data sets of 1 million sequences with the input sequence length $l = 15$ and $l = 20$ respectively. The number of embedded motifs K and the number of anchors d in each cluster center are varied. Figure 2.14 demonstrates that the MSRs of ASC follow similar trends for different l. The misplaced sequence ratio is quite small.



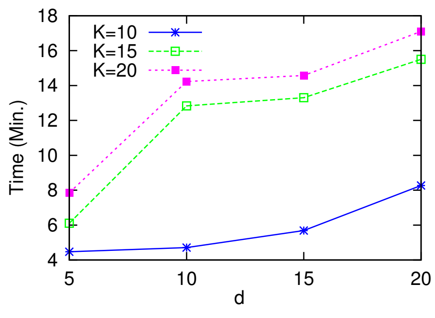
(a) When l=15



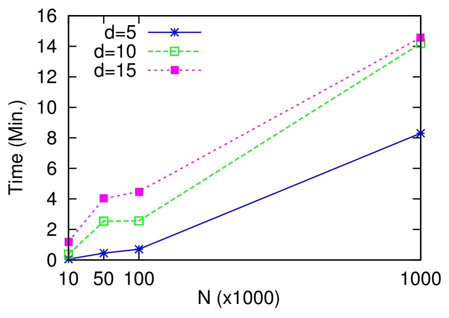
(b) When l=20

Figure 2.14: Misplaced Sequence Ratio of ASC

Efficiency. We then conduct experiments to test the clustering efficiency of ASC. Figure 2.15 shows the running time of ASC when l is fixed to 15. Due to space constraint, we omit the results for $l = 20$ since they follow a similar trend. We first fix N to 1 million and vary the values of d and K to see how ASC behaves, where d is the number of anchors in each cluster center and K is the number of planted motifs. Figure 2.15(a) shows that the runtime increases when either d or K increases. This is because larger K or d would make the clustering process take more time to terminate. This is not a problem since a small d can already give us a very good MSR and the running time of ASC is negligible compared to the time taken by the following motif discovery process. We further conduct experiments to test how ASC responds with an increasing number of sequences by fixing $K = 20$ and vary d and N. Figure 2.15(b) shows that with different d, the runtime of ASC increases almost linearly with respect to N which is consistent with our time complexity analysis in Section 2.4.4.



(a) When $N = 1M$



(b) When $K = 20$

Figure 2.15: Running Time of ASC when $l = 15$

2.7 Related Work

Motif finding is a classic problem and has been extensively studied for more than a decade. Federico *et al.* gives a very good survey of motif finding algorithms before and after next-generation sequencing era [3]. But still, with new challenges posted by the increasing size and complexity of big data, it is not completely solved. Most of the existing algorithms can be divided into two categories, combinatorial and probabilistic.

Combinatorial methods. Usually these methods use a combinatorial definition of motif, and treat the motif finding problem as an approximate pattern matching problem. For instance, in order to find motifs with length W, all possible 4^W sequences are enumerated (for DNA sequence with $\{A,G,C,T\}$ as the alphabet set). These sequences are modified with ambiguity codes, e.g., setting K=A *or* C, to allow mismatches. An exhaustive matching with the input sequences is conducted to count the frequency of sequences. The most frequent sequences will be outputted as motifs. Methods like SMILE [17] and Weeder [18] adopted this idea and also created index structure to speed up the matching process. However, since they need to do exhaustive searching, they are still very slow and can not scale w.r.t a larger alphabet set. Recently, a method named ACME [12] is reported to be able to scale to large alphabet set by utilizing parallel computing. We compared our algorithm with ACME in the experiments section.

The combinatorial methods usually can get optimal results. However, they need large search space and often require a few parameters like the length of motifs and the number of allowed mismatches, which users may not have knowledge of.

Probabilistic methods. These methods treat the data as composed of two components, the motif model and background model. They can infer the parameters of motif and background models that fit the data, thus classifying the subsequences either to motif or background. Currently, there are two approaches used to perform inference: Expectation Maximization (EM) and Gibbs Sampling.

EM. Multiple EM for Motif Elicitation (MEME) [2, 19, 10] is currently the most popular motif finding software. It was first proposed in 1994 by T. Bailey *etc.* The idea is to first break each input sequence into overlapping k-mers. Then use EM to estimate the model that fits the data best. Even though they kept improving the algorithm, there are still some problems with this method. It can not handle large data sets, e.g., larger than 600 protein sequences of length 15 in our real dataset.

STEME [6] can speed up the EM process by using suffix tree index for the sequences.

But due to the complexity of suffix tree, this method can only support DNA sequences.

Recently, T. Kim *et al* proposed MUSI [7] which is fast and can work with protein sequences. MUSI first uses MAFFT [20] to do multiple sequences alignment and then use EM to infer PWMs from the alignment results. We compared our algorithm with MUSI in the experiment section.

Gibbs sampling. Gibbs Sampling has similar mechanism with EM, but adopts a stochastic way to modify the current solution. Algorithms such as motif sampler [21] and AlignACE [22] belong to this category. The advantage of these methods is that they have a better chance to escape from local optima.

GibbsCluster [11] adopts the idea of Gibbs sampling to do multiple sequences alignment and clustering. At first, sequences are randomly divided into several clusters and aligned together. Then, in each iteration, a sequence is selected to do "shifting" and moved to another cluster with some probability. This process is repeated until the alignment score reaches a local optima. Motifs are extracted from aligned sequences in each cluster.

Gapped q-gram. Another related topic is gapped q-gram. A gapped q-gram is a subset of q characters of a fixed non-contiguous shape. For example, the 3-grams of shape $\#\#\_\#$ in the string $ACAGCT$ are AC_G, CA_C and AG_T. This concept is similar to the *anchors* we have used in our work, but they have different focuses.

Gapped q-gram is mostly used in string matching problems to filter candidates that could potentially match a target sequence. Most of these studies are focused on how to find some optimal "shapes" that could maximize the filtering effects. S. Burkhardt *et al* [23, 9] proposed to use gapped q-grams in a string matching problem and they also showed how to use experiments to choose the shape of gapped q-grams. M. Fontaine [24] introduced a method of selecting shapes of gapped q-grams for approximate string matching. A very popular local alignment tool named PatternHunter [25, 26] is also based on gapped q-gram. They used gapped q-grams as seeds to find possible aligned regions. It was shown that the problem of finding even one optimal gapped q-gram seed is NP-hard.

There is another work [27] that also represented a motif as a set of anchors like we did. It is used to search for occurrences of a motif in a set of sequences, which is different from our problem setting as we do not know the motif beforehand.

Record matching and deduplication. This problem is trying to identify records in a database that refer to the same entity. A commonly used technique in this domain is called "blocking" which divides the database into blocks and compare only the records that fall into the same block [28, 29]. This idea is similar to our pre-processing strategy, but it cannot be directly applied to motif discovery. Our design of anchor representation, initial anchor selection and iterative anchor set adjustment is essential for successfully identifying motifs.

To tolerant data heterogeneity, some methods [30, 31] use q-gram to do matching and blocking. However, none of the methods have adopted gapped q-gram in any way. So, our anchor (gapped q-grams with variable shapes) based clustering algorithm may be also interesting to this domain.

Time series motifs. A time series is a sequence of numerical and continuous data points. Time series motifs are frequent repeated patterns in time series data. Several recent papers [32, 33] targeted on this problem, but their problem settings are totally different from ours.

2.8 Conclusions

In this work, we examined the motif discovery problem in the context of big data, where massive short sequences are generated by the newest sequencing techniques. Existing motif finding algorithms usually work only in a small scale or have to trade accuracy for efficiency. Our strategy is not to develop another motif finding algorithm and make it scalable. Instead, we resort to a different methodology which clusters a sequence dataset into multiple small subsets and then reuses existing motif finding algorithms. This strategy is generic. Our experimental results are extremely appealing. The anchor-based clustering approach can reduce the runtime in more than two orders of magnitude, without losing any accuracy. Sometimes it even discovers more motifs.

Chapter 3

Prediction and Validation of Inhibitory Connections from Spike Trains

3.1 Introduction

As proposed by D. O. Hebb [34] a cell assembly describes a network of neurons that is repeatedly activated in a manner that strengthens excitatory synaptic connections. An assembly of this sort has a spatiotemporal structure inherent in the sequence of activation, and consequently strong internal synaptic relationships, which distinguish them from other groups of neurons. Although assemblies of this sort can be defined in numerous ways, an informative approach consistent with mechanisms of neural encoding is to identify statistically significant time-varying relationships among the spike trains of simultaneously recorded neurons [35, 36, 37, 38, 39]. Obtaining these neural activity measurements requires recording from many neurons in parallel that can be spatially localized and temporally resolved at sub-millisecond time scales [40]. Widely used approaches for recording from multiple neurons such as calcium imaging and voltage sensitive dyes as a proxy for electrical activity or multiple implanted micro electrodes do not satisfy all these requirements. Novel instrumentation is required to meet the challenge of drawing complete neural circuits.

Dissociated neurons can self-organize, acquire spontaneous activity, and form networks according to molecular synaptogenic drivers that can be heuristically visualized and probed with multi-electrode arrays (MEAs). The work presented here utilizes MEAs with sub-millisecond time resolution and precise signal localization. The Neural Circuit Probe (NCP) uses mobile probes for local chemical delivery to a neural circuit of cultured neurons on a commercial MEA with 120 electrodes. Local drug delivery transiently and reversibly modulates the electrical behavior of individually identified neurons to assess their contributions to the circuit behavior. The dynamics of neuronal networks require both excitatory and inhibitory signals. Excitatory cells alone cannot generate ”cell assemblies” because such interconnections would only lead to more excitation. A rivalry between excitatory and inhibitory neurons ensures the stability of global neuronal firing rates over larger spatial domains of brain while allowing for sharp increases in local excitability which is necessary for sending messages and modifying network connections [41]. In a neuronal network described in terms of correlations among statistically significant time-varying relationships among the spike trains of simultaneously recorded neurons, the detection of excitatory relationships can be inferred based upon correlations between spikes with constant latencies that approximate synaptic transmission [42, 43]. However, the identification of inhibitory relationships is more difficult: distinguishing endogenous low firing rates from active inhibition is not obvious.

In this chapter, we demonstrate that tools from statistical inference can predict functionally inhibitory synaptic connections and show how inhibition propagates in a network to affect other neurons. We did this by first fitting a Generalized Linear Model (GLM) to spike trains recorded from neurons in a hippocampal culture and inferring effective interactions between these neurons. We then used the fitted model to perform simulated *in silico* experiments in which we simulated the effect of silencing individual neurons in a network on the activity of others. We tested the predictions from these simulated silencing experiments by performing real experiments in which we applied Tetrodotoxin (TTX) to silence neurons and thereby validated our approach toward the detection of inhibitory interactions between pairs of neurons.

3.2 Methods

3.2.1 Spike sorting

For each MEA recording, we first removed redundancy propagation signals [44] and then did spike sorting [45]. Extracellular signals were band pass filtered using a digital 2nd order Butterworth filter with cutoff frequencies of 0.2 and 4 kHz. Spikes were then detected and sorted using a threshold of 6 times the standard deviation of the median noise level.

The data in Fig 3.3a were gathered in one recording session and each "unit" corresponds to one spike train after the spike sorting algorithms were applied on the raw data. However, the data in Fig 3.3d and Fig 3.3g were gathered in several recording sessions. So, the labels of units could be inconsistent in different recording sessions after the spike sorting algorithms were applied. Hence, to make the data consistent across different recording sessions, for these two datasets, we merged the spike trains from the same electrode as one unit.

3.2.2 A pipeline to identify and validate putative inhibitory connections

We used a novel pipeline to first identify putative inhibitory connections from spike trains and then validate them with a Neural Circuit Probe (NCP) that we built. Mouse hippocampal neurons were dissociated and plated on a multi-electrode array (MEA). To begin with, as shown in Fig 3.1a, their spontaneous spiking activity was modeled using a Generalized Linear Model (GLM) in which the outcome is a zero or one (spike or no spike) random variable and single neurons generate spikes according to a Poisson process. The rate of this process was determined by the spikes from other neurons. Parameters of the GLM were fit using a gradient descent algorithm to minimize the negative log likelihood of the recorded spike trains.

We next conducted *in silico* interventional experiments to identity inhibitory connections as shown in Fig 3.1b. Single neurons were silenced or activated *in silico* and then these data were used to infer predicted effects on connectivity using the fixed parameters from the GLM as determined above. The procedure for running the *in silico* interventional experiment was as follows. First, we selected one neuron as our interventional target. Throughout the simulation experiment, the state of this neuron was fixed to either 0 (silenced) or 1 (activated). Then, for all the other neurons, we ran the GLM with the inferred parameters to get the probabilities of seeing a spike at the next time point. Each probability represented how likely it was for a neuron to generate a spike at the next time point. Given the probability, we sampled a binary value (0 or 1) from a Bernoulli distribution as the state of the neuron for the next time point, where 0 refers to no spike and 1 means spike. We continued doing this to generate simulated recordings one time point at a time until a desired length T had been reached, where T is the number of time points in our *in silico* interventional recording. To find inhibitory connections, we investigated the generated simulated data to find those neurons that were negatively correlated (Pearson correlation coefficient) with the intervention taken on the target neuron. These neurons were considered as potentially inhibited by the interventional target.

Finally, we conducted real TTX delivery experiments to validate the putative inhibitory connections predicted from the *in silico* interventional experiments as shown in Fig 3.1c. In these experiments, TTX was delivered using the NCP as a delivery system. The NCP delivered TTX in a manner highly localized to a single electrode and in sufficiently low concentration that its potency dropped below threshold once it diffused beyond a single electrode. The NCP can detect increased impedance as the probe approached the cell and therefore allowed us to deliver TTX as close as possible to the cell without directly contacting the cell. Each TTX delivery resulted in the rapid onset of complete silencing of the neuron to which it was applied. As a result, putative inhibitory connections were validated when we observed activation of an inhibited neuron for a duration that approximated the time of TTX-induced silencing.

3.2.3 Generalized Linear Model

We used GLM to model the spiking of neurons. Let m denote the number of neurons being recorded and $x_{i,t}$ be the number of spikes of neuron i at time t. We assume $x_{i,t}$ is drawn from a Poisson distribution with rate $\lambda_{i,t}$ which is written as

$$\lambda_{i,t} = \exp\,(b_i + \sum_{j=1}^{m} \sum_{l=minlag}^{maxlag} \theta_{i,j,l} x_{j,t-l}). \tag{3.1}$$

where b_i is a parameter controlling the spontaneous firing rate of neurons i and $\theta_{i,j,t}$ denotes the effective interaction from neuron j to neuron i at time lag l. We assume that the firing rate of neurons i depends on the activities of all neurons in a history window

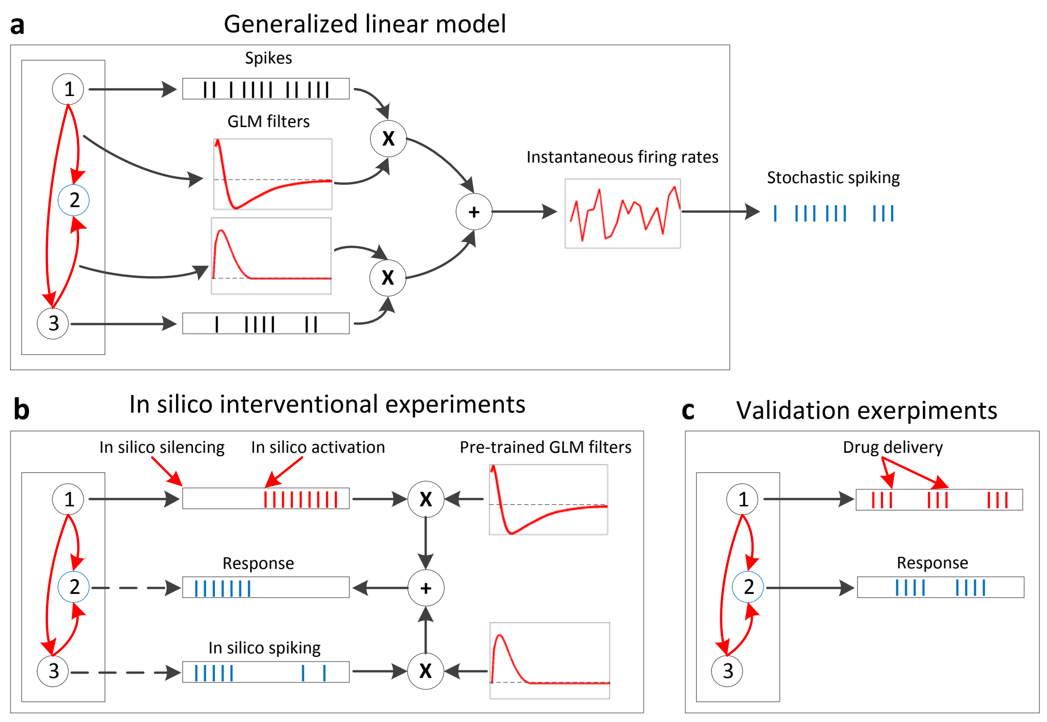


Figure 3.1: An overview of the procedure that we used to identify and validate direct and indirect inhibitory connections. (a) A Generalized Linear Model (GLM), in which the firing of a neuron is modeled as determined by the spikes from other neurons, was used. Filters of the GLM were inferred from a training recording of spontaneous firings. (b) In silico experiments were conducted by performing simulated interventions on a neuron and generating simulated responses with pre-trained GLM filters. Putative inhibitory connections were then identified by comparing the simulated interventions and responses. (c) Real drug delivery experiments were conducted to validate the putative inhibitory connections.

that spans from time $t - maxlag$ to time $t - minlag$, where $minlag$ and $maxlag$ are the minimum and maximum time lags we consider.

Given Eq. 3.1 for the firing rate of individual neurons, the likelihood for the observation of neuron i at time t, $L_{i,t}$ is

$$L_{i,t} = p(x_{i,t}|\lambda_{i,t}) = \frac{{\lambda_{i,t}}^{x_{i,t}} e^{-\lambda_{i,t}}}{x_{i,t}!}. \tag{3.2}$$

In spike train data with one millisecond time bin, there are at most one spike at any time point and therefore $x_{i,t}$ takes the value of 0 or 1. Hence, the log-likelihood is

$$\log L_{i,t} = x_{i,t} \log \lambda_{i,t} - \lambda_{i,t}. \tag{3.3}$$

The log-likelihood for all the observations in a recording with length T is

$$\log L = \sum_{i=1}^{m} \sum_{t=maxlag}^{T} \log L_{i,t}. \tag{3.4}$$

The model described above includes too many parameters and there is nothing in the model that ensures the inferred parameters to vary smoothly with time, something that isas expected from interactions between pairs of neurons. Furthermore, the model has too many parameters and this might cause problems for robustly inferring them. To ensure the smoothness of the filters, instead of directly using a history window of spikes in the model, following [46], we use their filtered versions that are created by convolving with several cosine bumps. To minimize the number of fitting parameters and prevent overfitting, we add an $L-1$ regularizer to the likelihood. These remedies are described further below.

We first design p cosine basis functions where the l^{th} cosine basis function can be written as:

$$f_l(t) = \frac{1}{2}\{1 + cos[a \ln{(t+b)} - \Theta_l]\} \tag{3.5}$$

for all times t such that satisfy

$$-\pi \leq a \ln{(t+b)} - \Theta_l \leq \pi$$

and

$$f_l(t) = 0$$

outside the interval defined above. The values of a, b and Θ_l are manually chosen. One of the factors to be considered when choosing their values is the locations where the peaks of the bumps occur. During experiments, we used pairwise cross-correlations to determine the locations of the peaks.

In the *naive* GLM without the basis functions, for neuron j, we used a history window of spikes to model its influence on other neurons. Now the raw spikes are convolved with p cosine basis functions to get the filtered versions, of which the l^{th} value is calculated as follows:

$$\tilde{x}_{j,l,t} = \sum_{\Delta=1}^{\tau} f_l(\Delta) x_{j,t-\Delta},$$

where τ is the length of the history window that is covered by the cosine basis functions. Eq. (3.1) is rewritten as:

$$\lambda_{i,t} = \exp\left(b_i + \sum_{j=1}^{m} \sum_{l=1}^{p} \beta_{i,j,l} \tilde{x}_{j,l,t}\right), \tag{3.6}$$

where $\beta_{i,j,l}$ is the weight of the l^{th} basis function for the influence from the neuron j to neuron i.

As mentioned above, to prevent overfitting, we added an $L1$ regularization term to penalize non-zero filter parameters. The loss function we want to minimize is rewritten as

$$J = -\sum_{i=1}^{m} \sum_{t=maxlag}^{T} \log L_{i,t} + r \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{l=1}^{p} |\beta_{i,j,l}|,$$

where r is the regularization constant. The value of r is decided by doing 10-fold cross validation on a spontaneous firing recording of 60 seconds. We used the Area Under the Receiver Operating Characteristic curve (AUC-ROC) as our metric to evaluate the performance of the fitted model to do predictions on future spikes given previous spiking histories.

3.2.4 *in silico* interventional experiments

To identify inhibitory connections from an ensemble of neurons, one straightforward way is to investigate the GLM filters obtained by fitting the spike trains, as these filters represent the relations of neurons captured by GLM. However, the inhibitory effects among neurons can be rather complex than obvious, and simply using the GLM is usually not enoughsufficient. For example, two of the inhibitory connections we identified in this study were not observable from the their corresponding GLM filters, but became obvious once interventions were applied. Therefore, in this study, we have proposed a method to conduct *in silico* interventional experiments which could discover hidden inhibitory connections by running simulated experiments.

To cold start the simulated experiment, we used a history window of length τ with none spiking states (i.e., all the neurons take the value 0 in a time window of τ). The instantaneous firing rate of neuron i at time t was calculated according to E.q. (3.6) in Methods section. Therefore, the probabilities of seeing and not seeing a spike are

$$p(x_{i,t} = 1|\lambda_{i,t}) = \lambda_{i,t}e^{-\lambda_{i,t}}$$

and

$$p(x_{i,t} = 0|\lambda_{i,t}) = e^{-\lambda_{i,t}}$$

Because in our setting, there is at most one spike in the one millisecond time bin, $x_{i,t}$ can only take the value of 0 or 1. However, if we run simulated experiments by directly sampling from a Poisson distribution, the value $x_{i,t}$ takes could be arbitrary instead of binary. Hence, we normalize the probability of getting a spike at time point t as

$$\begin{aligned} p(x_{i,t} = 1|\lambda_{i,t}) &= \frac{p(x_{i,t} = 1|\lambda_{i,t})}{p(x_{i,t} = 1|\lambda_{i,t}) + p(x_{i,t} = 0|\lambda_{i,t})} \\ &= \frac{\lambda_{i,t}}{1 + \lambda_{i,t}}, \end{aligned} \tag{3.7}$$

For neuron i at time point t, we generate the simulated sate by sampling a value from a Bernoulli distribution with the probability of Eq. (3.7).

During the *in silico* interventional experiments, we selected one neuron as our interventional target and fixed its state to be either 0 (silenced) or 1 (activated). Then, the responses from other neurons were gathered and compared with the intervention taken on the target neuron by calculating their Pearson correlation coefficients. Those neurons that were negatively correlated with the intervention were considered as potentially inhibited by the interventional target. The algorithm is shown in Algorithm 2.

Algorithm 2: Identifying Putative Inhibitory Connections

```
input  : A recording $X$ of spontaneous firing events and a target neuron $t$
output: Top $k$ neurons that are potentially inhibited by $t$
Train a GLM model using $X$
Conduct in silico experiments where neuron $t$ is intervened
for each neuron $i$ do
    Calculate the Pearson correlation coefficient between simulated recordings of
    neuron $i$ and neuron $t$
end
Select top $k$ neurons that have the largest negative Pearson correlation coefficient
with neuron $t$
return top $k$ neuron ids
```

3.2.5 Instrumentation for validating putative inhibitory connections

Identification of single cell contributions to a neuronal circuit requires precise access to and control over functionally identified cells. To accomplish this goal we built a neural circuit probe (NCP) consisting of (1) a head unit that accepts various probes, (2) an integrated perfusion chamber plus light ring illumination system, (3) a probe control system with computer interface which implements a simple feedback system for an automated approach, and (4) a commercial MEA (MEA2100, Multi Channel Systems) mounted to a custom X-Y translation stage (Fig 3.2).

The NCP controller uses proportional and integral feedback control to position the various probes, and can accept a variety of input signals, such as ion current used here. An amplifier is located on the head unit that amplifies the current signal before going to the controller. The NCP software allows the operator to engage and disengage the probe using feedback. It is also used to control the location of the MEA stage beneath the probe, allowing the operator to position the probe above neurons of interest. A pneumatic control system attached to the probe regulates a pressure line for chemical delivery (Fig 3.2a, Fig 3.2b). An integrated pressure sensor, connected to the MEA data acquisition system, measures the duration and magnitude of pressure for temporal alignment with the MEA signal.

Local targeted drug delivery with the NCP can be used to modify their electrical behavior. This was done with small pipettes typically with inner diameters of 1-2 microns. In this example (Fig 3.2c, Fig 3.2d) we applied the Na+ channel blocker tetrodotoxin (TTX, 500 nM) to induce a temporary and reversible cessation of activity from that cell. Thus with the NCP we can do targeted drug delivery with high spatial resolution.

3.3 Results

3.3.1 Identifying Putative Inhibitory Connections

Following the aforementioned procedure, a recording with spontaneous activity from 17 units over a duration of 20 seconds divided into one millisecond time bins was used to fit the GLM model (Fig 3.3a). Each unit corresponded to a spike train after spike sorting and removal of the redundancy inherent in propagation signals[44]. Then unit 10 was chosen as the in silico interventional target, i.e. it was fixed in a silent state (no spikes at all times) for 10 seconds and then fixed for 10 seconds in an active state (continuous spiking). Simulations with the fitted GLM identified five units with the highest probability to be inhibited by unit 10 (Fig 3.3c). The strongest candidate for inhibition by unit 10 was unit 8. Note that the filters from the fitted GLM also suggested that the connection from unit 10 to unit 8 was predominantly inhibitory (Fig 3.3b).

Additional *in silico* experiments on another cell culture were also conducted to identify putative inhibitory connections by following the aforementioned procedure. For these experiments, we used a 60 second recording of spontaneous firing events (Fig 3.3d) to fit a GLM. The GLM parameters for the connections from unit 12 to five units are shown in Fig 3.3e. We calculated the Pearson correlation coefficients between the in silico intervention on unit 12 and simulated responses of every other neuron. The top five negatively correlated units were chosen and investigated (Fig 3.3f). In another example, we chose unit 23 as the in silico interventional target. Similarly, Fig 3.3h shows the GLM parameters for the connections from unit 23 to five other units and Fig 3.3i shows the top five units that were identified as candidates for inhibition by unit 23.

3.3.2 Validation of Inhibitory Connections

Given putative inhibitory connections identified in the first example (Fig 3.3c), to validate experimentally that unit 8 was an inhibitory target of unit 10, TTX was delivered four times on unit 10 (Fig 3.4a) using the neural circuit probe as a delivery tool. The instrument delivered TTX in a manner highly localized to a single electrode and in sufficiently low concentration that its potency dropped below threshold once it diffused beyond a single electrode. Each TTX delivery resulted in the rapid onset of complete silencing of the neuron to which it was applied. Delivery of TTX to the electrode corresponding to unit 10 resulted in the activation of unit 8 and activation of the target neuron for a duration that approximated the time of TTX-induced silencing. These experimental data clearly demonstrated that the top inhibitory connection (from 10 to 8) predicted by our simulated experiment was validated by the actual TTX delivery experiment.

In the second example, 92 was a strong candidate for inhibition by unit 12. To validate this inferred connection experimentally, we delivered TTX to unit 12 and, as predicted, observed an inhibitory effect from unit 12 to unit 92 (Fig 3.4b). Its also worth mentioning that even though unit 92 is not the top 1 candidate predicted by our *in silico* interventional experiments, its within the top 5 predictions out of 120 possible units. This shows that the in silico interventional experiments could give accurate predictions of putative inhibitory connections. In the final example, we also delivered TTX to unit 23 and observed rebound of firing on unit 92 (Fig 3.4c) which was predicted by the *in silico* interventional experiments (Fig 3.3i).

3.3.3 Indirect connections

The inhibitory connections identified in this study may not be direct. A unit could be causing inhibitory effects on another unit through a third unit. To study the possibilities of inhibitory connections, we have revisited the three examples of inhibitory connections validated in Fig 3.4. For each example, we introduced a third unit and convolved the GLM filters of the two connections in a potential inhibitory connection. Fig 3.5 shows the convolution outputs that exhibited inhibitory effects. To understand the inhibitory effects from unit 10 to unit 8, we show three possible cases where unit 10 could cause an inhibitory effect on unit 8 through a third unit (Fig 3.6a).

The second example shown in Fig 3.3 and Fig 3.4 illustrates an important feature that the in silico experiments offers in describing how signals propagate in the network. In this example the inhibitory effects from 12 to 92 is not obviously manifested in the filters shown in Fig 3.3e, i.e. the magnitude of the curve representing the connection from 12 to 92 is not as significant as others. However, this inhibitory effect is ranked high according to the negative Pearson correlation score given the simulated experimental results. One explanation for this is the indirect connections among units. It may be the case that unit 12 is not directly inhibiting unit 92, but it could cause an inhibitory effect through other units.

To explore this possibility further, we show three possible indirect inhibitory connections from unit 12 to unit 92 (Fig 3.6b). Each indirect connection consists of a predominantly excitatory connection and a predominantly inhibitory connection, which could cause a net inhibitory effect. Therefore, it supports the idea that the inhibitory effect from unit 12 to unit 92 were caused by indirect inhibitory connections.

As a final example, Fig 3.4c shows another inhibitory effect between pairs of neurons, in this case unit 23 to unit 92, as discovered from the *in silico* experiments on the fitted GLM and then validated by experiments. Similarly, we show three indirect inhibitory connections from 23 to 92 (Fig 3.6c).

3.4 Discussion

Understanding how neuronal signals propagate in local network is an important step in understanding information processing. The standard way to predict how the activity of one neuron influences another is through intracellular paired recordings along with pharmacologic probes. Using such intracellular recordings, one can establish the presence or absence of direct or indirect connections between pairs of neurons and thus to some degree predict how activity in one neuron affects the others. Inspired by the successes of this technology, we show here how it can be extended to larger networks of neurons using advanced mechatronic positioning of a probe over an array of electrodes with the Neural Circuit Probe. As a demonstration of the potential power of this device, we demonstrated its utility in testing the predictions of *in silico* modeling.

We first fitted a GLM model to spikes recorded from a culture using MEAs, then performed *in silico* experiments in which we silenced one of the units, and identified what other units will change their activity upon this inactivation. We then went back to the culture and silenced the same unit using TTX and observed that the inhibitory effects predicted by the *in silico* experiments showed up when TTX was applied.

The results presented here thus opened the door to using statistical models not only to characterize the statistics of neural spike trains or functional connectivity between neurons, but to make predictions about the response of the network to changes. Although using GLM to study the circuitry of a neuronal network is never going to be as accurate as intracellular recordings, the simplicity of fitting the model to data and performing in silico experiments with it are great advantages that support the idea of using this approach to make educated guesses about the likely outcomes of manipulations to the network, i.e. offering a virtual culture, similar to a previous attempt to use GLMs to build a virtual retina. [47].

In using the GLM in neural data analysis, one typically assumes that a single neuron generates spikes via e.g. a Poisson process. The rate of this process is determined by the spikes from other neurons filtered by interactions that are inferred from data using convex optimization. The inferred model is then used for a variety of purposes that include evaluating the role of correlations in shaping population activity, for example, in the retina [46], the motor cortex [48, 49], the functional connectivity between grid cells [50], or the relative influence of task related covariates on shaping neural responses in the parietal cortex [51]. Despite the widespread use of the GLM in neural data analyses, a potentially very powerful aspect of this class of models has been left unexplored: the ability of the GLM to make predictions about how a neuronal network responds to interventions. At the microcircuit level, this amounts to identifying meaningful interactions between pairs of neurons and using them to make predictions about how external manipulations of one or more neurons can affect the others. The main reason for the fact that GLMs have not been used for this purposes so far is that, in general, the ground truth about connectivity is not known and, therefore, it is not possible to compare the interactions inferred by GLM with the real ones. The results presented in this chapter add a new dimension to how these statistical models can be used in neuroscience by showing that, although the relationship between individual synaptic interactions and those inferred by the GLM may not be known, the inferred connections can still be employed to make specific predictions about the functional connectivity of a neuronal network. Our results thus demonstrate how statistical models can be used to decipher neuronal microcircuitry at a detailed level without using more complicated experimental techniques such as multi-unit intracellular recordings.

To acquire the traces for analysis by the GLM we used Multi-electrode arrays (MEAs) capable or recording extracellular action potentials (eAP) from hundreds of neurons simultaneously. Planer MEAs serve as a substrate for glial and neuronal growth in which neural ensembles self organize over multiple days [52, 53]. The close apposition of neurons with each recording electrode produce recordings with high signal-to-noise properties while the ease of making manipulations in the culture permits access not otherwise possible in vivo. Because cultured neurons self-organize into spontaneously active synaptic circuits and because neurons fire action potentials primarily in response to synaptic input, recording extracellular action potentials (also referred to as spikes) can sample the connectivity phenotypes in large groups of neurons. Here we introduce how to use GLM based simulated experiments to identify putative inhibitory connections and then validate our approach to these inhibitory connections by conducting in vivo experiments.

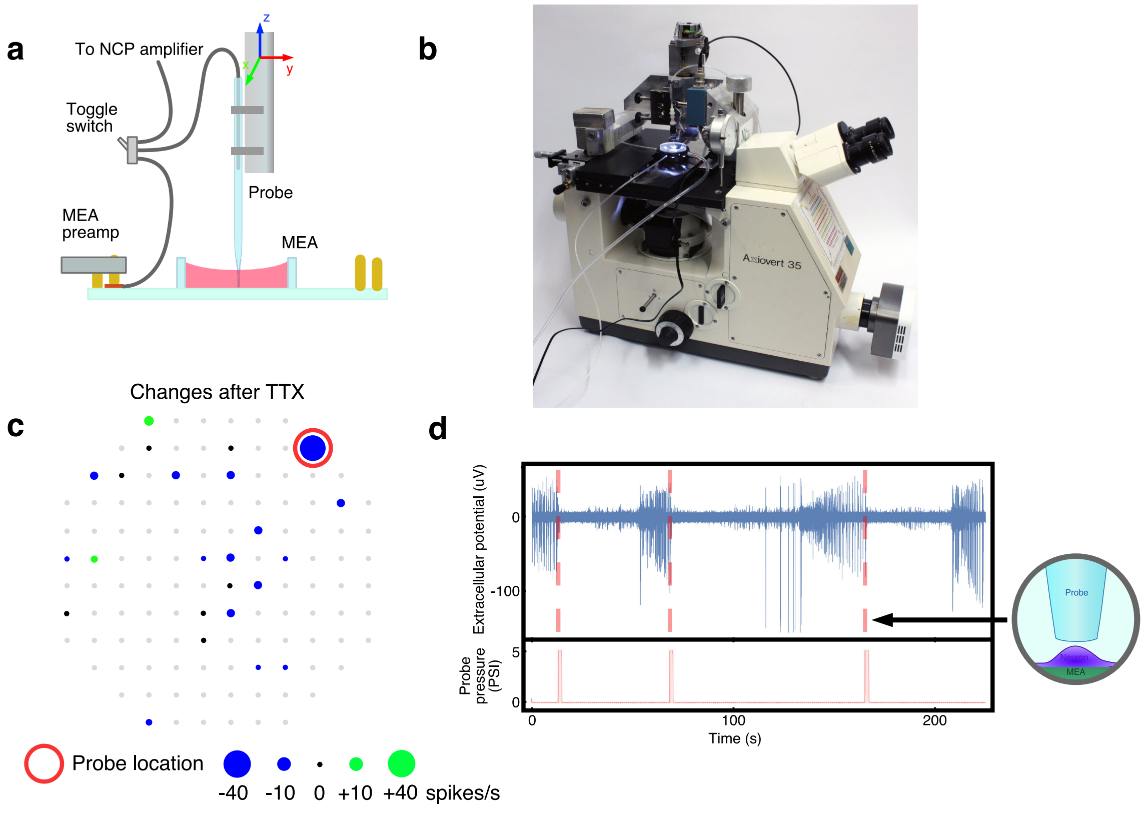


Figure 3.2: Illustration of the Neural Circuit Probe (NCP) and real drug delivery experiments to validate putative inhibitory connections. (a) Schematic drawing of the key components. The probe is positioned in x and y to center it in the field of view of the microscope. Then the MEA is translated in x and y to bring a target neuron directly under the probe. Finally the probe is automatically lowered, with ion conductance feedback, to just above, but not touching, the neuron. (b) Overview of the NCP situated on an inverted microscope. (c) The changes of firing rates at all electrodes before and after TTX application. Gray dots are electrodes with no spiking activities recorded. Black dots are electrodes with no spiking rate changes. When we blocked spiking at the specific electrode (red circle) it had widespread secondary effects on the firing rates observed at other MEA electrodes. Though the firing rate decreased for many electrodes (blue dots), for two electrodes it increased (green dots). (d) A transient increase of probe pressure delivered TTX (500 nM), which reversibly blocked spiking activity, with high spatial resolution. This process was repeated 3 times.

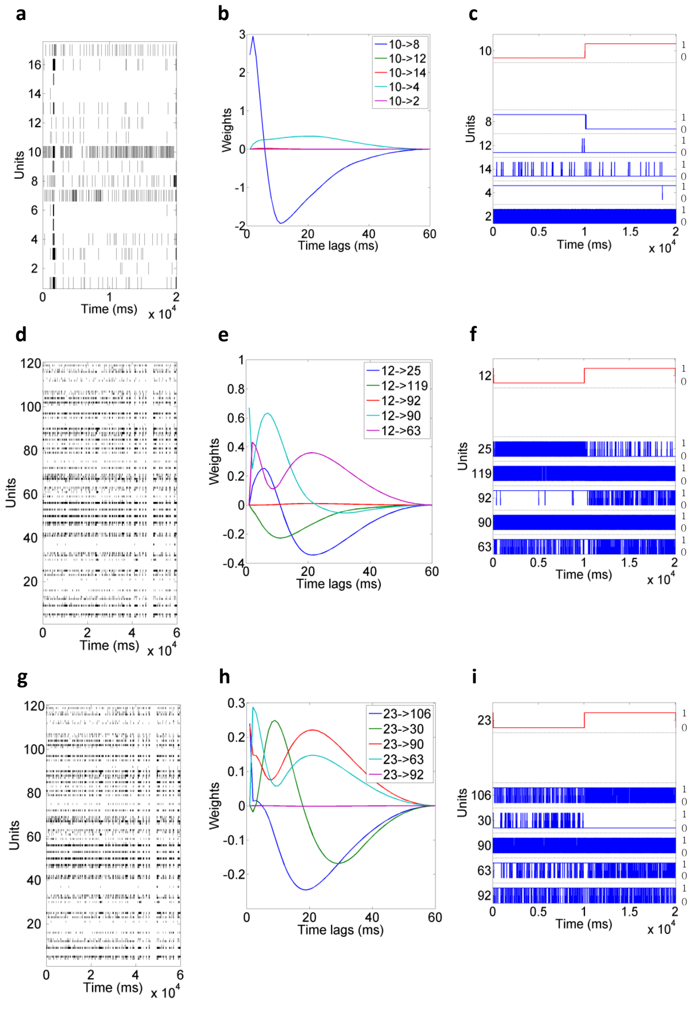


Figure 3.3: Real data examples of the procedure that we used to identify putative inhibitory connections. (a) A training recording of 17 units for a duration of 20 seconds which were divided into 1 millisecond time bins. The black bars represent spikes. (b) Filters of the GLM inferred from the training recording. Note that at different time lags, the strength of the connection between two units is also different. (c) Simulated data for top 5 units that were negatively correlated with the intervened unit 10. The red and blue lines represent the instantaneous firing rates for the simulated recordings. The labels on the left of the y-axis represent the unit numbers and the labels on the right represent the range of the instantaneous firing rates (0 to 1). Note that when unit 10 was changed from silent state to active state, conversely, unit 8 changed to silent state from active state, which implied a putative inhibitory connection. (d) A training recording of 120 electrodes for a duration of 60 seconds which were divided into 1 millisecond time bins. (e) Filters of the GLM inferred from the training recording. (f) Simulated data for top 5 units that were negatively correlated with the intervened unit 12. (g) A training recording of 120 electrodes for a duration of 60 seconds which were divided into 1 millisecond time bins. (h) Filters of the GLM inferred from the training recording. (i) Simulated data for top 5 units that were negatively correlated with the intervened unit 23.

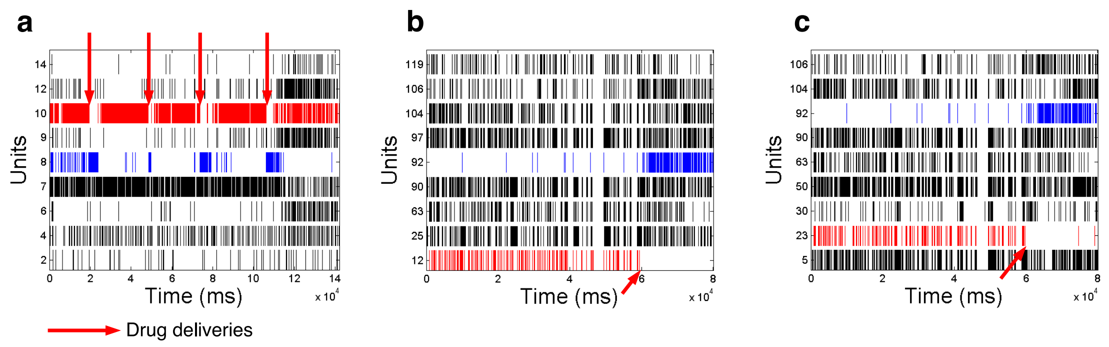


Figure 3.4: Real TTX experiments to validate putative inhibitory connections. (a) Real TTX experimental recording where unit 10 was silenced 4 times by delivering TTX. Unit 8 rebounded every time unit 10 was silenced, which indicated an inhibitory connection from 10 to 8. (b) Real TTX experimental recording where unit 12 was intervened. (c) Real TTX experimental recording where unit 23 was intervened.

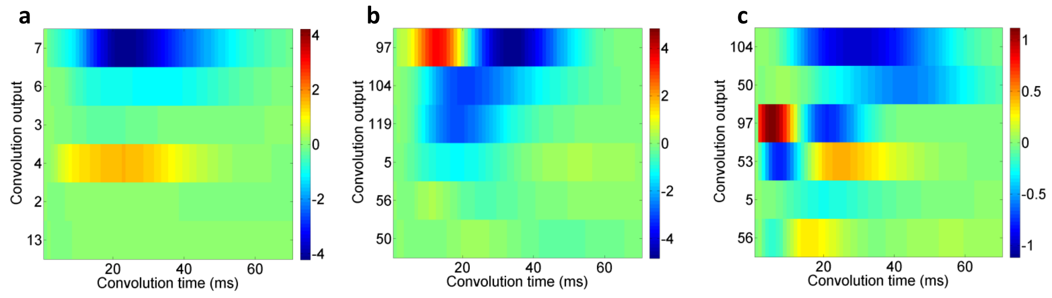


Figure 3.5: Convolutions of the GLM filters from indirect connections. (a) Convolution of the GLM filters from the connection $10 \to m$ and $m \to 8$, where m (y-axis) is an intermediate unit. The convolutions when m is 10 or 8, which indicates a direct connection, are omitted. (b) Convolution of the GLM filters from the connection $12 \to m$ and $m \to 92$, where m (y-axis) is an intermediate unit. (c) Convolution of the GLM filters from the connection $23 \to m$ and $m \to 92$, where m (y-axis) is an intermediate unit.

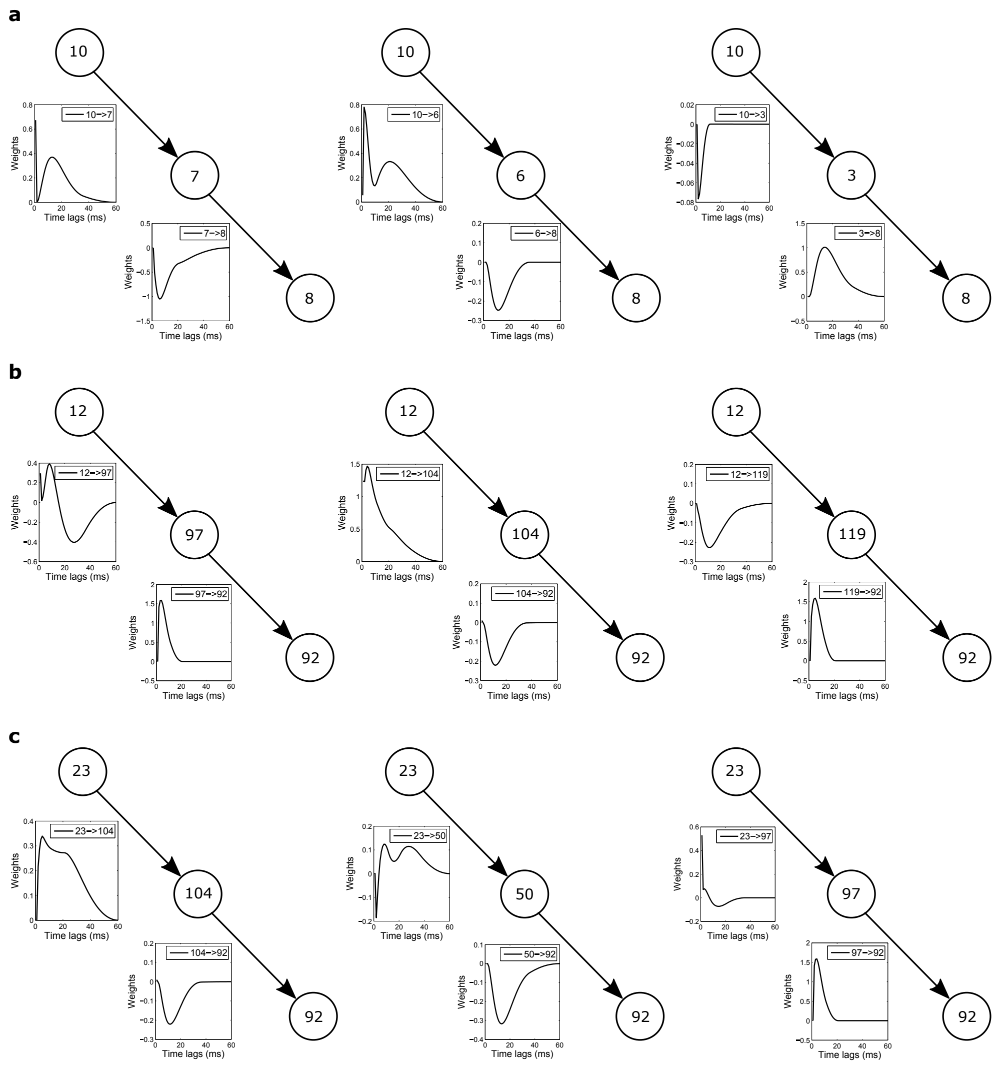


Figure 3.6: Indirect inhibitory connections. (a) Three possible cases where unit 10 has an inhibitory influence on unit 8 through a third unit. The first case consists of an excitatory connection ($10 \rightarrow 7$) and an inhibitory connection ($7 \rightarrow 8$). The second case consists of an excitatory connection ($10 \rightarrow 6$) and an inhibitory connection ($6 \rightarrow 8$). The third case consists of an inhibitory connection ($10 \rightarrow 3$) and an excitatory connection ($3 \rightarrow 8$). (b) Three possible cases where unit 12 has an inhibitory influence on unit 92 through a third unit. The first case consists of a predominantly inhibitory connection ($12 \rightarrow 97$) and an excitatory connection ($97 \rightarrow 92$). The second case consists of an excitatory connection ($12 \rightarrow 104$) and an inhibitory connection ($104 \rightarrow 92$). The third case consists of an inhibitory connection ($12 \rightarrow 119$) and an excitatory connection ($119 \rightarrow 92$). (c) Three possible cases where unit 23 has an inhibitory influence on unit 92 through a third unit. The first case consists of an excitatory connection ($23 \rightarrow 104$) and an inhibitory connection ($104 \rightarrow 92$). The second case consists of an excitatory connection ($23 \rightarrow 50$) and an inhibitory connection ($50 \rightarrow 92$). The third case consists of a predominantly inhibitory connection ($23 \rightarrow 97$) and an excitatory connection ($97 \rightarrow 92$).

Chapter 4

Active Learning of Functional Networks from Spike Trains

4.1 Introduction

Spike trains are series of neural firing events, which are considered as the language neurons use to encode the external world and communicate with each other. Learning functional networks from spike trains is a fundamental problem with many critical applications in neuroscience. For example, a functional network that describes the temporal dependence relations among neurons is not only the first step to understand the function of neural circuits [54], but also has practical applications such as diagnosing neurodegenerative diseases [55].

Since *Generalized Linear Model* (GLM) is commonly used as a temporal generative model for spike trains [56, 46, 57], the routine [38, 54] of inferring functional networks from spike trains is shown in Figure 4.1 with an example. A spike train recording of 5 neurons is used to infer the GLM, from which a functional network is derived. The spike train dataset is a set of binary arrays, where "1" represents a firing event (spike) and "0" describes quiet state (no spike). Meanwhile, in the functional network, the edge between node 1 and node 4 with label "+1" represents an excitatory connection with time lag 1 (the firing of neuron 1 at time $t-1$ stimulates the firing of neuron 4 at time t). Similarly, the directed edge from node 4 to node 3 with label "-[1,20]" represents an inhibitory connection with time lags from 1 to 20 (the firings of neuron 4 at time $t-20$ through $t-1$ suppress the firing of neuron 3 at time t).

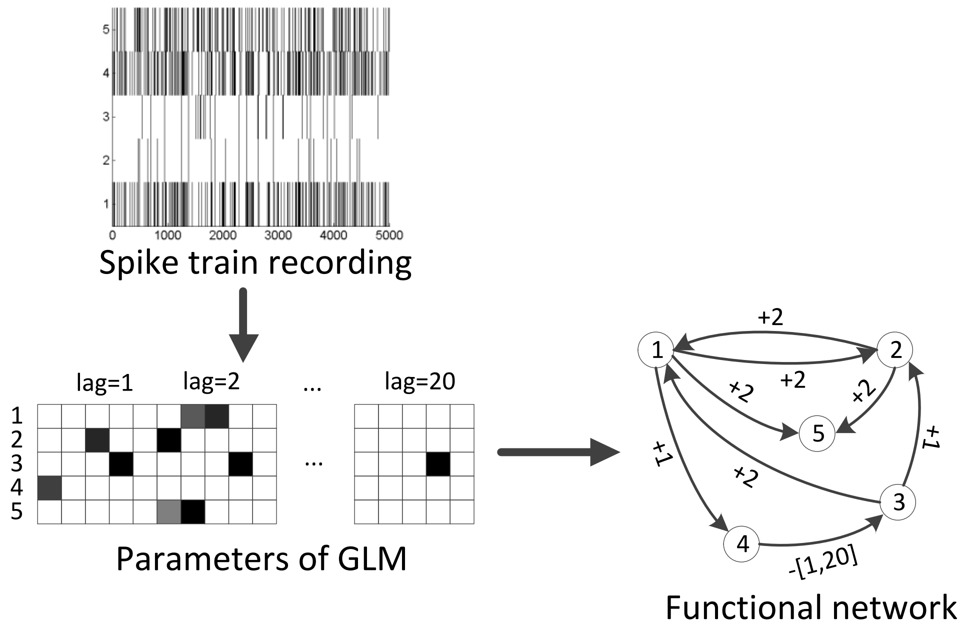
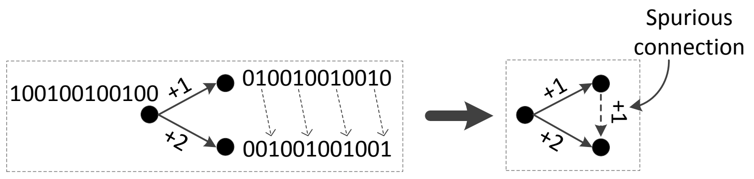
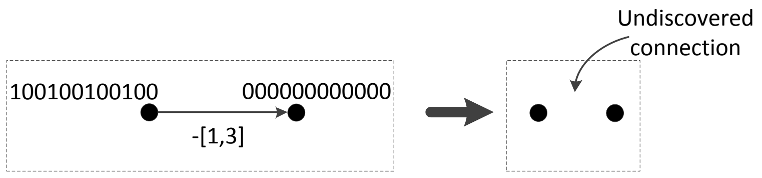


Figure 4.1: An example of inferring a functional network from spike trains.

Despite the popularity of this approach, we can not rely on it to get accurate functional networks. To illustrate, we give two examples. Figure 4.2(a) shows spike trains from three neurons where the firings of two of them are being driven by another neuron with different time lags. When a functional network is inferred, the aforementioned algorithm could easily get confused and a spurious excitatory connection will be drawn in the resulted network. In another example shown in Figure 4.2(b), the activities of a neuron are suppressed by an inhibitory connection and thus, there are not enough evidence to infer the inhibitory connection. Unfortunately, most of existing works [38, 46, 58] suffer from this problem because they are learning functional networks from purely observational data. As we will demonstrate in Section 4.5, by adopting interventional data, in which we could selectively fix the states of some neurons, the accuracy of the inferred functional network could be significantly improved. However, conducting interventional experiments is often very expensive in terms of time and money, so the interventions must be chosen with care. Hence, in this chapter, we focus on the problem of how to design an active learning framework that could utilize as few intervential experiments as possible to get the maximum accuracy gain when inferring a functional network.



(a) Example of an excitatory network



(b) Example of an inhibitory network

Figure 4.2: Examples of inferred networks with only observational data.

There are previous works [56, 59] that focused on the problem of selecting external stimuli for a better estimation of GLM. But their approach can not be directly used in our problem, because they only consider the case where there is just one neuron while we are interested in inferring functional connectivities among multiple neurons. Meanwhile, learning functional networks should not be confused with learning the structure of causal networks [60] (static or dynamic Bayesian networks). In structure learning of causal networks, possible topological structures are searched and evaluated based on a statistical score function such as Minimum Description Length (MDL) [61] and Bayesian Dirichlet equivalent (BDe) score [62]. In contrast, the structure and parameters of a functional network are jointly learned by inferring a temporal generative model. Several works [63, 64, 65] focused on the problem of active learning for structures of causal networks which is different from our functional network learning problem.

To the best of our knowledge, we are the first to propose active learning models for inferring functional networks from spike trains. Our active learning framework is shown in Figure 4.3. The functional network is iteratively updated by conducting interventional experiments. In each iteration, the next intervention is chosen based upon the results seen so far towards a full identification of the functional network. In particular, we introduce two models, the variance model and the validation model, to choose interventions that are most beneficial for learning the functional network.

The variance model (Section 4.3) uses a Gaussian distribution to approximate the posterior distribution of GLM parameters given the data. And then the intervention that can maximally reduce the expected entropy of the posterior distribution is chosen. In addition, we also propose an initialization method that takes higher order interactions into consideration, which could significantly improve the performance of the variance model. Meanwhile, the validation model (Section 4.4) has the objective to validate the most of our existing connections. It picks interventions by maximizing the expected probability of our current knowledge about the GLM parameters.

These two models represent two different strategies of choosing interventions. The variance model works best to discover hidden inhibitory connection, while the validation model focuses on eliminating spurious excitatory connections. Experimental results with both synthetic and real datasets show that when these two models are applied, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

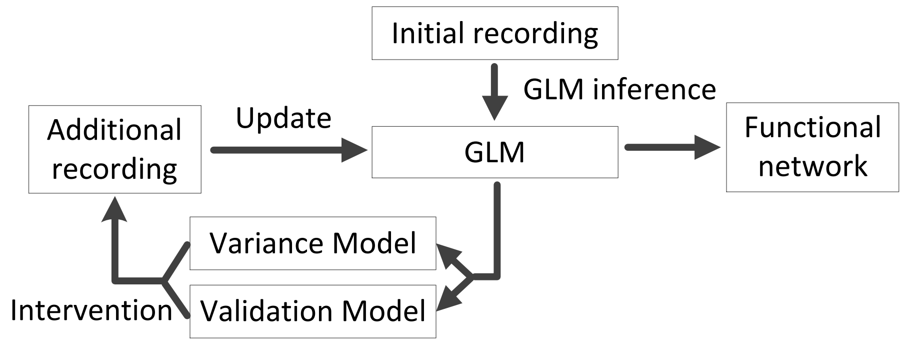


Figure 4.3: Pipeline of the active learning framework.

4.2 Preliminaries

In this section, we first briefly introduce the Generalized Linear Model (GLM) and then show the framework to infer GLM when both observational and interventional data are used. Table 4.1 summarizes some common notations that we are going to use in this chapter.

4.2.1 Generalized Linear Model

Let m denote the number of neurons being recorded and $x_{i,t}$ be the number of spikes of neuron i at time t. Usually in spike train data, there are at most one spike at any time point, so $x_{i,t}$ takes the value of 0 or 1. We assume $x_{i,t}$ depends on all the neurons' activities in a history window that spans from time $t-maxlag$ to time $t-minlag$, where $minlag$ and $maxlag$ are the minimum and maximum time lags we consider. Let $\theta_{i,j,t}$ be the parameter that models the effect from neuron j to neuron i at time lag l. For any neuron i, it also has a spontaneous firing rate which is controlled by a bias term b_i. We first model the instantaneous firing rate of of neuron i at time t, $\lambda_{i,t}$, as follows,

$$\lambda_{i,t} = e^{(b_i + \sum_{j=1}^{m} \sum_{l=minlag}^{maxlag} \theta_{i,j,l} x_{j,t-l})}. \tag{4.1}$$

Table 4.1: Notations

Notations	Description
m	Number of neurons
$minlag$	Minimum time lag to consider
$maxlag$	Maximum time lag to consider
h	$maxlag - minlag + 1$
n	$m \times h + 1$
T	Length of recordings
$x_{i,t}$	The state of neuron i at time point t
$\theta_{i,j,l}$	Parameters for the effect from neuron j to neuron i with time lag l
b_i	The bias term for neuron i which controls the spontaneous firing rate
$\boldsymbol{s_t}$	Input vector at time t with dimensions $n \times 1$
$\boldsymbol{r_t}$	Response vector at time t with dimensions $m \times 1$
$\boldsymbol{s_{1:t}}$	A matrix of input vectors from time point 1 to time point t, with dimensions $n \times t$
$\boldsymbol{r_{1:t}}$	A matrix of response vectors from time point 1 to time point t, with dimensions $m \times t$
$\boldsymbol{W}$	Parameters of GLM as a matrix with dimensions $m \times n$
$\boldsymbol{W}(i, \cdot)$	The i^{th} row of matrix W
$\boldsymbol{w}$	Flattened copy of matrix W

We then assume that $x_{i,t}$ is drawn from a Poisson distribution with mean $\lambda_{i,t}$. In other words, we assume that the firing of neurons follows a Poisson process which is a common assumption [57, 38, 46]. Hence, the log-likelihood for the observation of neuron i at time t, $\log L_{i,t}$, is calculated as

$$\log L_{i,t} = \log p(x_{i,t}|\lambda_{i,t}) = x_{i,t} \log \lambda_{i,t} - \lambda_{i,t}. \tag{4.2}$$

The log-likelihood for all the observations in a recording with length T is

$$\log L = \sum_{i=1}^{m} \sum_{t=maxlag}^{T} \log L_{i,t}. \tag{4.3}$$

To simplify our analysis later, we rewrite the log-likelihood function in matrix format. First, for a recording with T time points, we reconstruct it into an input matrix $\boldsymbol{s_{1:t}}$ and a response matrix $\boldsymbol{r_{1:t}}$, where $t = T - maxlag$. $\boldsymbol{s_{1:t}}$ is a $n \times t$ matrix with each column $\boldsymbol{s_k}$ representing the input vector at time point k, where $n = m \times h + 1$ and $h = maxlag - minlag + 1$. Similarly, $\boldsymbol{r_{1:t}}$ is an $m \times t$ matrix with each column $\boldsymbol{r_k}$ representing the response vector at time point k. $\boldsymbol{s_k}$ and $\boldsymbol{r_k}$ are constructed as follows.

$$\boldsymbol{s_k} = \begin{pmatrix} 1 \\ x_{1,k-minlag} \\ \vdots \\ x_{m,k-minlag} \\ \vdots \\ x_{1,k-maxlag} \\ \vdots \\ x_{m,k-maxlag} \end{pmatrix}, \boldsymbol{r_k} = \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{m,k} \end{pmatrix}$$

We also rewrite the parameters of GLM as a matrix $\boldsymbol{W}$ with dimensions $m \times n$. Each row in W contains the parameters to predict responses of one neuron. For example, $\boldsymbol{W}(i, \cdot)$ contains the parameters responsible for the response of neuron i, where $\boldsymbol{W}(i, \cdot)$ denotes the i^{th} row of $\boldsymbol{W}$. $\boldsymbol{W}$ is constructed as follows.

$$\boldsymbol{W} = \begin{pmatrix} b_1 & \dots & b_m \\ \theta_{1,1,minlag} & & \theta_{m,1,minlag} \\ \vdots & & \vdots \\ \theta_{1,m,minlag} & & \theta_{m,m,minlag} \\ \vdots & & \vdots \\ \theta_{1,1,maxlag} & & \theta_{m,1,maxlag} \\ \vdots & & \vdots \\ \theta_{1,m,maxlag} & \dots & \theta_{m,m,maxlag} \end{pmatrix}^T$$

Following Eq. (4.1), (4.2) and (4.3), the log-likelihood function in matrix format is

$$\log L(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}}) = sum(\boldsymbol{r_{1:t}} \circ (\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}) - e^{\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}}),$$

where *sum* is a function that sums over all the elements in a matrix and $\circ$ represents Hadamard product which is essentially element-wise multiplication.

4.2.2 Active Learning of GLM

Given a recording with input matrix $\boldsymbol{s_{1:t}}$ and response matrix $\boldsymbol{r_{1:t}}$, to learn the GLM, we use batch gradient ascent to infer the parameters that maximize the log-likelihood function. The gradients with respect to $\boldsymbol{W}$ are calculated as

$$\begin{aligned} D(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}}) &= \frac{\partial \log L(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})}{\partial \boldsymbol{W}} \\ &= \boldsymbol{r_{1:t}} \cdot \boldsymbol{s_{1:t}^T} - e^{\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}} \cdot \boldsymbol{s_{1:t}^T}, \end{aligned} \tag{4.4}$$

where $D(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$ is a $m \times n$ matrix.

In the active learning framework, the interventional experiments are conducted iteratively to update the GLM. Let $I = \{\iota_1, \iota_2, ..., \iota_c\}$ be the set of interventions we can choose from. In this work, we focus on deterministic interventions which means ι_i defines an action of forcing one or several neurons to take a fixed state. For example, ι_i could represent silencing one neuron i. Let Q be the set of neurons that are intervened. Q is empty when only observational data is used. Intuitively, for any neuron q in Q, its state will no longer depend on its parents in the functional network. So when $\boldsymbol{W}$ is being updated, the parameters that are responsible for the response of this neuron will not be changed.

Assuming n recordings has been collected and Q_i is the set of neurons that are intervened in the i^{th} recording. We first calculate the gradients $\boldsymbol{D_i}$ for the i^{th} recording with Eq. (4.4), and then for all the $q \in Q_i$, we set $\boldsymbol{D}_{q,\cdot} = 0$, where $\boldsymbol{D}_{q,\cdot}$ is the q^{th} row in $\boldsymbol{D}$. Eventually, the gradients for all the n recordings are calculated as follows,

$$\boldsymbol{D} = \sum_{i=1}^{n} \boldsymbol{D_i}. \tag{4.5}$$

In summary, the pipeline of the active learning framework is as follows: Given n recordings we have seen so far, infer the GLM using batch gradient ascent (Eq. (4.5)); Then choose an intervention from I and conduct the intervention experiment to collect the $(n+1)^{th}$ recording; Repeat this procedure until the budget for doing experiments has run out. In the following sections, we introduce two models to intelligently choose the next intervention.

4.3 Variance Model

By conducting interventional experiments, previously undiscovered connections could be revealed. However, how to choose the most informative intervention is still a hard problem to be solved. In this section, we propose the *variance model* to choose interventions based on the following intuitions: (1) Inhibitory connections tend to be undiscovered due to lack of evidence; (2) Lacking of evidence means high uncertainty about our knowledge of the inferred functional network; (3) The uncertainty about our knowledge of the inferred functional network could be quantified as the entropy of the posterior probability distribution of the parameters given the data. Moreover, we also introduce an initialization method that takes higher order interactions into consideration, which proves to be very effective for further improving the performance of the variance model.

4.3.1 Choose Interventions

Assuming we have a recording of t time points which is formalized as an input-output pair $(\boldsymbol{s}_{\mathbf{1}:t}, \boldsymbol{r}_{\mathbf{1}:t})$. Let $\boldsymbol{w}$ be the flattened copy of the GLM parameter matrix $\boldsymbol{W}$. Our knowledge about $\boldsymbol{w}$ can be summarized by the posterior probability distribution $p(\boldsymbol{w}|\boldsymbol{s}_{\mathbf{1}:t}, \boldsymbol{r}_{\mathbf{1}:t})$ and the entropy of $p(\boldsymbol{w}|\boldsymbol{s}_{\mathbf{1}:t}, \boldsymbol{r}_{\mathbf{1}:t})$, $H(p(\boldsymbol{w}|\boldsymbol{s}_{\mathbf{1}:t}, \boldsymbol{r}_{\mathbf{1}:t}))$, quantifies the uncertainty of our knowledge. Our goal is to choose the intervention that can maximally reduce the uncertainty.

In this work, we focus on deterministic interventions which gives us the ability to assume the next input vector with intervention, $\boldsymbol{s}_{t+\mathbf{1}}$, is uniquely defined by the intervention type chosen from I. Now the problem can be formalized as choosing $\boldsymbol{s}_{t+\mathbf{1}}$ such that the entropy of $p(\boldsymbol{w}|\boldsymbol{s}_{\mathbf{1}:t+\mathbf{1}}, \boldsymbol{r}_{\mathbf{1}:t+\mathbf{1}})$ can be maximally reduced. Since the response vector $\boldsymbol{r}_{t+\mathbf{1}}$ is unknown, we use the expected entropy instead and the objective of the variance model is

$$\arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})), \tag{4.6}$$

where $\boldsymbol{r_{t+1}}$ is the response vector for $\boldsymbol{s_{t+1}}$.

However, it's difficult to directly compute and optimize the expected entropy exactly. Since the likelihood function of $\boldsymbol{w}$ also belongs to the exponential family, we approximate $p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$ as a Gaussian distribution,

$$\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}} \sim \mathcal{N}(\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}}),$$

where $\boldsymbol{u_{t+1}}$ and $\boldsymbol{C_{t+1}}$ denote the mean and covariance of $\boldsymbol{w}$ given $(\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$. Accordingly, we have the following theorem.

Theorem 3 *When $p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$ is approximated as a Gaussian distribution, we could solve the objective function (Eq.* (4.6)*) as*

$$\arg\max_{\boldsymbol{s_{t+1}}} (e^{\boldsymbol{W}\cdot\boldsymbol{s_{t+1}}})^T \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}), \tag{4.7}$$

Proof: First, we have

$$H(\mathcal{N}(\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})) = \frac{1}{2}\log|\boldsymbol{C_{t+1}}| + const,$$

where $|\boldsymbol{C_{t+1}}|$ represents the determinant of $\boldsymbol{C_{t+1}}$.

In order to calculate $\boldsymbol{C_{t+1}}$, we have

$$\boldsymbol{C}_{\boldsymbol{t+1}}^{-1} = -\frac{\partial^2 \log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})}{\partial \boldsymbol{w}^2},$$

because the inverse covariance matrix equals to the second partial derivative of the log-Gaussian density function w.r.t. $\boldsymbol{w}$.

By expending $\log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})$ (see supplementary materials for more details), we can get

$$\begin{aligned} \boldsymbol{C_{t+1}^{-1}} &= \boldsymbol{C_t^{-1}} + F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) \\ &= (\boldsymbol{s_{t+1}} \otimes \boldsymbol{I}) \cdot diag(e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}}) \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}), \end{aligned} \tag{4.8}$$

where $\otimes$ represents Kronecker product, $diag$ is a function that takes all the elements of a matrix and reconstruct them into a diagonal matrix, $\boldsymbol{I}$ is a $m \times m$ identity matrix, and

$$F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) = -\frac{\partial^2 \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}})}{\partial \boldsymbol{w}^2},$$

which is the Fisher information (the negative of the second derivative of the log likelihood with respect to $\boldsymbol{w}$). It's interesting to see that the Fisher information does not depend on the response vector $\boldsymbol{r_{t+1}}$.

Finally, Eq. (4.6) can be solved as

$$\begin{aligned} &\arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})) \\ &= \arg\max_{\boldsymbol{s_{t+1}}} \log |\boldsymbol{C_t^{-1}} + F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})| \\ &= \arg\max_{\boldsymbol{s_{t+1}}} tr(\log (\boldsymbol{I} + \boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}))) \\ &= \arg\max_{\boldsymbol{s_{t+1}}} (e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}})^T \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}), \end{aligned}$$

where tr is the function to calculate the trace of a matrix and $\boldsymbol{J}$ is a $m \times 1$ vector with ones in all its entries. ■

As we can see from Eq. (4.7), the expected entropy relies on the value of $\boldsymbol{W}$. We can use the expectation of $\boldsymbol{W}$ to eliminate this unknown variable. To simplify the calculation, we assume $\boldsymbol{W}(i,\cdot)$, the i^{th} row of $\boldsymbol{W}$, which contains the parameters to predict the responses of neuron i, also follows a Gaussian distribution $\mathcal{N}(\boldsymbol{u_t^i}, \boldsymbol{C_t^i})$. $\boldsymbol{u_t^i}$ and $\boldsymbol{C_t^i}$ are subsets of $\boldsymbol{u_t}$ and $\boldsymbol{C_t}$ that correspond to the parameters in $\boldsymbol{W}(i,\cdot)$.

Now Eq. (4.6) becomes

$$
\begin{aligned}
&\arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{W}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})) \\
&\approx \arg\max_{\boldsymbol{s_{t+1}}} \\
&\begin{pmatrix} E_{\boldsymbol{W}(i,\cdot)\sim\mathcal{N}(\boldsymbol{u_t^1},\boldsymbol{C_t^1})} e^{\boldsymbol{s_{t+1}}^T\cdot\boldsymbol{W}(i,\cdot)^T} \\ \vdots \\ E_{\boldsymbol{W}(i,\cdot)\sim\mathcal{N}(\boldsymbol{u_t^m},\boldsymbol{C_t^m})} e^{\boldsymbol{s_{t+1}}^T\cdot\boldsymbol{W}(i,\cdot)^T} \end{pmatrix}^T \\
&\cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I_{m\times m}}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J_{m\times 1}}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \\
&\begin{pmatrix} e^{\boldsymbol{u_t^1}\cdot\boldsymbol{s_{t+1}}+\frac{1}{2}\boldsymbol{s_{t+1}^T}\cdot\boldsymbol{C_t^1}\cdot\boldsymbol{s_{t+1}}} \\ \vdots \\ e^{\boldsymbol{u_t^m}\cdot\boldsymbol{s_{t+1}}+\frac{1}{2}\boldsymbol{s_{t+1}^T}\cdot\boldsymbol{C_t^m}\cdot\boldsymbol{s_{t+1}}} \end{pmatrix}^T \\
&\cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}),
\end{aligned}
\tag{4.9}
$$

Eq. (4.9) consists of two terms. The first term is a $1 \times m$ vector and the second term is a $m \times 1$ vector.

From Eq. (4.9), we can get some intuitions about the variance model. The term $e^{\boldsymbol{u_t^i}\cdot\boldsymbol{s_{t+1}}}$ indicates that the model is trying to find the interventions that can increase the activities of the neurons so that previously undiscovered connection would have a higher chance of get revealed. The term $(\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J})$ indicates that the model will give larger weights to the interventions that have influences on the connections with higher variance.

4.3.2 Update $\boldsymbol{u}$ and $\boldsymbol{C}$

Without losing generality, we assume i recordings has been seen so far and $\boldsymbol{C_i}$ corresponds to the most updated covariance matrix. When the $(i+1)^{th}$ recording comes, we show how to calculate $\boldsymbol{u_{i+1}}$ and $\boldsymbol{C_{i+1}}$. When $i = 0$, we use $\boldsymbol{C_0}$ to denote the initial covariance matrix. In the next section, we will show how to initialize $\boldsymbol{C_0}$ to take higher order interactions into consideration.

Since the log-likelihood function of GLM and the log-Gaussian density function are both concave, every time a new recording comes, we just redo the inference with the method introduced in Section 4.2.2 and use the inferred $\boldsymbol{w}$ to approximate $\boldsymbol{u_{i+1}}$. Given $\boldsymbol{u_{i+1}}$ and $\boldsymbol{C_i}$, we use Eq. (A.1) to update $\boldsymbol{C_{i+1}}$.

4.3.3 Initialization

When calculating the covariance matrix with the initial recording, we could just set $\boldsymbol{C_0}$ to be an identity matrix. We refer to this method as the *basic variance model.* However, we demonstrate that the performance of the variance model can be further improved by proposing a heuristic initialization method that considers higher order connections.

A deeper analysis about how we update $\boldsymbol{C}$ gives us the following theorem.

Theorem 4 *When $\boldsymbol{C}$ is initialized as an identity matrix and being updated according to equations* (A.1), $\forall\ i \neq j$, $i \in [1, m]$, $j \in [1, m]$, $k \in [1, n]$ *and* $c \in [1, n]$, *the covariance between $\boldsymbol{W}(i, k)$ and $\boldsymbol{W}(j, c)$ will always equal to 0, where $\boldsymbol{W}(i, k)$ is the GLM parameter in i^{th} row and k^{th} column of $\boldsymbol{W}$ (similarly for $\boldsymbol{W}(j, c)$).*

Proof:

According to Eq. (A.1), we have

$$\boldsymbol{C} = (\boldsymbol{C_0}^{-1} + (\boldsymbol{s} \otimes \boldsymbol{I}) \cdot diag(e^{\boldsymbol{W} \cdot \boldsymbol{s}}) \cdot (\boldsymbol{s}^T \otimes \boldsymbol{I}))^{-1}$$

By applying the Sherman-Morrison-Woodbury formula, we get

$$\begin{aligned}&\boldsymbol{C} = \boldsymbol{C_0} - \boldsymbol{C_0} \cdot (\boldsymbol{s} \otimes \boldsymbol{I}) \cdot \\ &(diag(e^{\boldsymbol{W}\boldsymbol{s}})^{-1} + (\boldsymbol{s}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_0} \cdot (\boldsymbol{s} \otimes \boldsymbol{I}))^{-1} \cdot (\boldsymbol{s}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_0}\end{aligned}$$

When $\boldsymbol{C_0}$ is initialized as an identity matrix, we can prove Theorem 4 by carrying out matrix operations. ■

The intuition behind Theorem 4 is that the parameters responsible for different neurons (different rows in $\boldsymbol{W}$) are independent with each other. As an example shown in Figure 4.4, two connections form a chain and the covariance between their corresponding parameters will not be updated. However, this chain represents higher order interactions in the functional network. Taking them into consideration is beneficial when choosing interventions. Accordingly, we propose a heuristic initialization method that proves to be working very well.

We first calculate the average firing rates, $(a_1, a_2 \ldots a_m)$, for all the neurons using the initial recording. Then an input vector $\boldsymbol{s}$ is constructed by using the average firing rates as the values for each neuron in all time lags. For any two parameters $\boldsymbol{W}(i,k)$ and $\boldsymbol{W}(j,c)$ where $i \neq j$, let $C_{\boldsymbol{W}(i,k)-\boldsymbol{W}(j,c)}$ denote their covariance. We initialize this value as follows,

$$C_{\boldsymbol{W}(i,k)-\boldsymbol{W}(j,c)} = \frac{1}{a_k a_c e^{\boldsymbol{W}(i,\cdot)\cdot \boldsymbol{s}} e^{\boldsymbol{W}(j,\cdot)\cdot \boldsymbol{s}}}$$

where we use the most updated $\boldsymbol{u}$ to approximate $\boldsymbol{W}$. This initialization method is designed to follow the intuition that more information indicates smaller (co)variance.

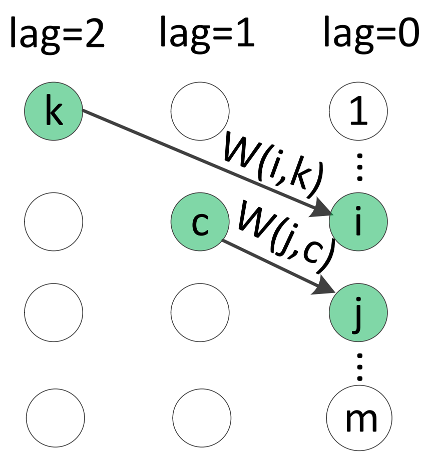


Figure 4.4: An example of higher order interactions.

Here, the amount of information is quantified by the average or predicted firing rate.

4.4 Validation Model

The variance model works pretty well for many cases. However, it still has some weaknesses. For example, in Figure 4.5, we have three neurons connected in a chain with spontaneous firing rates $(0.05, 0.0001, 0.0001)$ and the firings of neuron 2 and 3 are mainly driven by neuron 1. When a functional network is inferred, a spurious connection is likely to appear. Assuming we have the ability to silence one of the neurons, and our goal is to use the interventional data to maximally decrease the strength of the spurious connection. Using the variance model, neuron 3 will be picked to be silenced. Clearly, it's not the best option as when the state of neuron 3 is fixed, all the parameters for the incoming connections will not be updated. The variance model picks neuron 3 because it's trying to reduce the uncertainty about the parameters by increasing the neuronal activities of the whole network. Picking other neurons would reduce more activities than neuron 3. However, if we are able to pick neuron 2 as the target, the spurious connection will be filtered because the incoming connection that is driving the firing of neuron 3 is

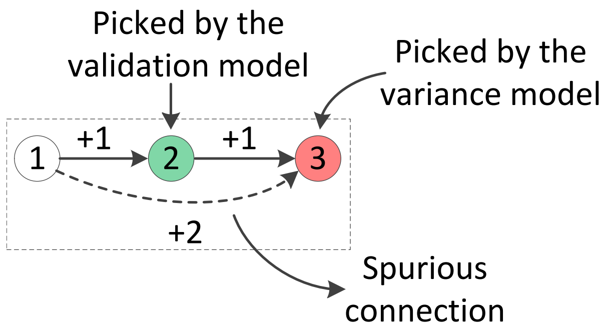


Figure 4.5: A chain network where the variance model picks neuron 3 and the validation model picks neuron 2.

blocked and we will know whether there is a connection from neuron 1 to neuron 3 or not. So in some cases, the variance model is not picking the best interventions.

Hence, we propose another model, called validation model, in accompany with the variance model. Instead of trying to increase activities of the system so that we can discover previously missed connections, the goal of the validation model is to maximally validate our existing knowledge about the functional network. Our current knowledge can be represented as the most updated GLM parameters, $\boldsymbol{u_t}$. For a new inverventional input vector $\boldsymbol{s_{t+1}}$, the objective is to maximally increase our confidence about $\boldsymbol{u_t}$, which is measured by $p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$. The objective function is formalized as

$$\underset{\boldsymbol{s_{t+1}}}{\arg\max} \log p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}}). \tag{4.10}$$

To have more intuitions about the validation model, consider a procedure of making decisions about the connections in a functional network given the GLM parameters. The significance of the parameters is measured by their posterior probabilities. Any parameter that has a posterior probability higher than a threshold will result in a connection in the functional network. By pursuing the objective function, we can increase our confidence about connections in the functional network or filter out spurious connections.

Since $\boldsymbol{r_{t+1}}$ is unknown, we use its expectation and rewrite the objective functions as follows,

$$\begin{aligned}
&\arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} (\log \boldsymbol{u_t} + \log p(\boldsymbol{r_{1:t}}|\boldsymbol{s_{1:t}}, \boldsymbol{u_t}) + \\
&\log p(\boldsymbol{r_{t+1}}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) + const) \qquad (4.11) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log p(\boldsymbol{r_{t+1}}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} E_{\boldsymbol{r_{t+1}(i)}} \log p(\boldsymbol{r_{t+1}(i)}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}),
\end{aligned}$$

where $\boldsymbol{r_{t+1}(i)}$ represents the i^{th} element in the response vector $\boldsymbol{r_{t+1}}$. When spike train data is considered, $\boldsymbol{r_{t+1}(i)}$ can only take the value of 0 or 1. So, we have

$$\begin{aligned}
&\arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} \sum_{\boldsymbol{r_{t+1}(i)}=0,1} \log p(\boldsymbol{r_{t+1}(i)}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} (-e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}} + \\
&(\boldsymbol{u_t^i s_{t+1}} - e^{\boldsymbol{u_t^i s_{t+1}}}) \cdot e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}}) \qquad (4.12) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} (e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}} (\boldsymbol{u_t^i s_{t+1}} - e^{\boldsymbol{u_t^i s_{t+1}}} - 1)) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} \lambda_i \cdot e^{-\lambda_i} \cdot (\log \lambda_i - \lambda_i - 1),
\end{aligned}$$

where $\lambda_i = e^{\boldsymbol{u_t^i s_{t+1}}}$ and $\boldsymbol{u_t^i}$ represents the parameters in $\boldsymbol{u_t}$ that are responsible for the response of neuron i in a row vector.

4.5 Experiments

In this experimental study, we use both synthetic and real spike train data sets to test the effectiveness of our active learning models. All the computations are conducted on a server with 2.67GHz Intel Xeon CPU (32 cores) and 1TB RAM.

4.5.1 Data Sets

Interventions. A very recent equipment called Neuronal Circuit Probe (NCP) was developed to do interventional experiments when recording spike trains from neurons. NCP could locate a single neuron and deliver drugs locally to this neuron. In our experiments, a drug that could silence neurons is used. In other words, assuming we have m neurons being recorded, there are m types of interventions we can do with each one corresponding to fixing the sate of a neuron to 0.

Synthetic data. We use three steps to generate simulated spike train data. First, the structure of the functional network is proposed. Then, a GLM parameter matrix is created according to the functional network. Finally, simulated spike train data is generated by running the GLM. If a neuron is intervened in the simulated experiment, its value will be always set to 0. We use 1 millisecond as the time bin in the recordings and each recording has a length of 20 seconds which are 20,000 data points. All the parameters in the simulation process are chosen to mimic real neurons. Due to space constraints, more details about the synthetic data could be found in the supplementary materials.

Real data. We use a Multielectrode Array (MEA) with 120 channels to record signals from neurons on a culture. Each channel corresponds to a node in the functional network we want to learn. We use 1 millisecond as the time bin to discretize neuronal signals to ensure there is at most 1 spike at each time bin.

The spike train recordings can be divided into two categories: observational recording and interventional recording. For the observational recording, the neurons are recorded without any drug deliveries. For the interventional recording, the neurons are recorded while drugs that can silence neurons are delivered at channels selected by different methods. Each interventional experiment is conducted after the neurons have fully recovered from the previous experiment. We use an observational recording with 60 seconds as the initial recording and each additional recording has a length of 20 seconds. Finally, another 60 seconds observational recording is reserved as the test set.

4.5.2 Evaluation

Methods. To illustrate the effectiveness of our active learning models, we compare our approaches with several baselines. The models we have proposed could be organized as four approaches: (1) ***Basic variance model***. The variance model using identity matrix as initialization; (2) ***Variance model***. The variance model using our initialization method; (3) ***Validation model***; (4) ***Mixture***. Alternately using *variance model* and *validation model* to choose interventions. We use two baselines to compare with: (1) ***Extend***. Simply adding more observational recordings without any interventions. (2) ***Firing rate***. Choosing the neuron that has the highest firing rate as the intervention target.

Metrics. For the synthetic datasets, since we have the ground truth which is the GLM parameter matrix $\boldsymbol{W}$, the inferred $\bar{\boldsymbol{W}}$ is directly compared with $\boldsymbol{W}$. The Frobenius norm of their difference is used to characterize the error of the inferred model,

$$e = \|\bar{\boldsymbol{W}} - \boldsymbol{W}\|_F.$$

Since we want to repeat our tests with different experimental settings (structure of the functional networks and parameters of the GLM) and report the average, we need to normalize the errors. Let $E = \{e_1, e_2, ..., e_c\}$ denote the set of errors when different number of recordings and different models are used. We normalize e_i as

$$\frac{e_i - u}{\sigma},$$

where u is the mean of E and σ is the standard deviation of E.

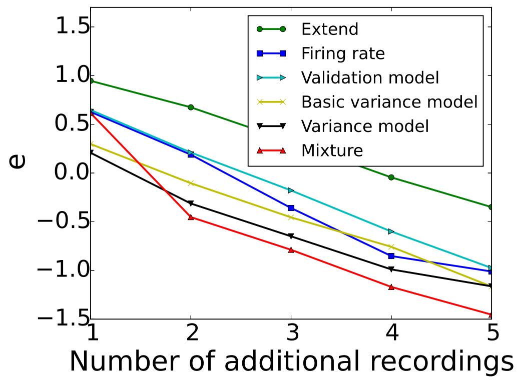
For real datasets, since we don't have the ground truth, we reserve some observational recordings as the test set, and use the predictive ability of the inferred model to measure its accuracy. So, the negative log-likelihood on the test dataset is used as the evaluation metric. A lower negative log-likelihood means the inferred model is more accurate.

4.5.3 Random Networks

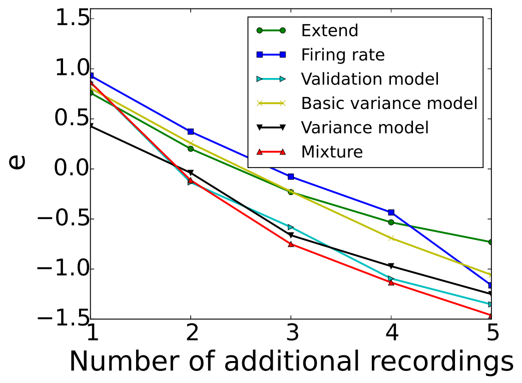
To test the effectiveness of our models, we conduct simulated experiments with random networks of different sizes. Given the size of the network, we randomly generate 10 networks and report the average of the normalized errors. All the simulated experiments are done interactively which means every time a new additional recording is added, the intervention models are re-calculated to pick the next intervention.

We first test the case when the functional network contains 10 nodes. As shown in Figure 4.6(a), for all the methods, when more additional recordings are added, the inferred model is getting more accurate. However, when *Mixture* is used to guide the intervention experiments, we can achieve the most accuracy gains. Another observation is that the variance model works better than the basic variance model because of our initialization method. It's worth mentioning that the validation model is not working very well because the size of the network is too small such that there are not a lot of spurious connections when the network is inferred.

We then increase the size of the random networks to 20 nodes and redo the experiments. As shown in Figure 4.6(b), the variance model, the validation model and the mixture method achieves the best results. The variance model shows consistent advantages over other models. The validation model shows a huge performance improvement compared to the previous experiment for the reason that the size of the random networks



(a) Random networks with 10 nodes.



(b) Random networks with 20 nodes.

Figure 4.6: Averages of normalized errors with random networks.

is larger and more spurious connections will be eliminated by the validation model. Interestingly, when the interventions are chosen by firing rate, it performs even worse than simply adding observational recordings.

4.5.4 Real Data

For biological reasons the neurons can not be recorded for too long. So, in real experiments, instead of choosing interventions interactively, we use batch experimental design. An initial recording of 60 seconds is collected to train the GLM and intervention models. Then a ranking of interventions is generated by each intervention model. We use this ranking without updating it to guide following experiments. We also use the negative log-likelihood on a test set with a recording of 60 seconds to measure the accuracy of the inferred model.

As shown in Figure 4.7, the variance model, the validation model and the mixture method could achieve lower negative log-likelihood (higher accuracy) than simply extending observational recording or picking interventions according to firing rate. The performance gain of our models over the *Firing rate* method is not as obvious in the second intervention experiment as in the first one. The reason may be because we are not able to update our intervention models by using the new recording. When the experiments can be done interactively, more accuracy gain will be achieved.

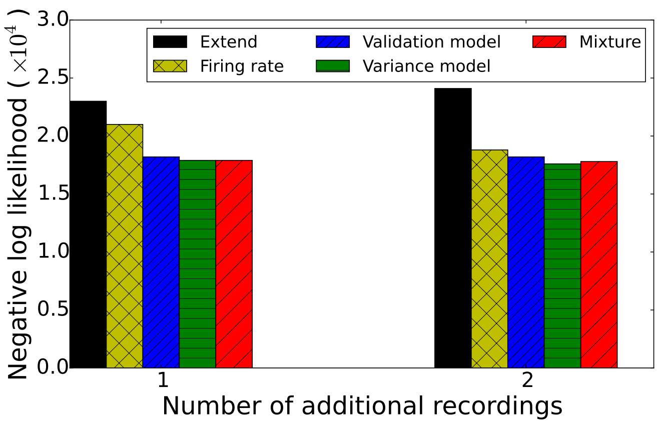


Figure 4.7: Evaluation by real data.

4.6 Related Work

The problem of learning functional networks should not be confused with the problem of learning the structure of a causal network [60] (static or dynamic Bayesian networks). In structure learning of static or dynamic causal Bayesian networks, numerous works [66, 67, 68, 62, 61] have been proposed. However, these works focus on how to efficiently search the structure space or how to evaluate the the proposed structure.

When learning a static causal Bayesian network, it can be proved that given only observational data, we can not differentiate networks in a Markov equivalence class, in which the networks have the same skeleton but may have different directions for some edges [69]. So some researchers [64, 65] try to tackle this problem with an active learning framework. In these methods, they will choose interventions that can orientate most edges. Another work [63] based on active learning framework keeps a distribution of possible structures and choose interventions that can maximally reduce the entropy of this distribution, but the ordering of nodes needs to be given.

We study the problem of learning functional networks. The structure and parameters are jointly learned by inferring a Generalized Linear Model. GLM is widely used in spike train analysis, but most works [38, 46] focus on learning GLM from observational data. J. Lewi *et al* proposes methods [59, 70, 56] to select external stimuli for a better estimation of GLM when there is only one neuron. However, we are interested in modeling interactions among multiple neurons, which is a different problem.

4.7 Conclusions

In this work, we study the problem of learning functional networks from spike trains in an active learning setting. In particular, we propose two models, the variance model and the validation model, to choose the most informative intervention so that we can get the maximum accuracy gain for the inferred network. Our experimental results with both synthetic and real data show that by applying our approaches, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

Appendix A

Supplementary Materials for Active Learning of Functional Networks from Spike Trains

A.1 Proof of Theorem 3

Here we give the extended version of the proof.

First, we have

$$H(\mathcal{N}(\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})) = \frac{1}{2}\log|\boldsymbol{C_{t+1}}| + const,$$

where $|\boldsymbol{C_{t+1}}|$ represents the determinant of $\boldsymbol{C_{t+1}}$.

In order to calculate $\boldsymbol{C_{t+1}}$, we have

$$\boldsymbol{C}_{\boldsymbol{t+1}}^{-1} = -\frac{\partial^2 \log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})}{\partial \boldsymbol{w}^2},$$

because the inverse covariance matrix equals to the second partial derivative of the log-Gaussian density function w.r.t. $\boldsymbol{w}$. Furthermore, we can expand $\log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})$ as follows,

$$\begin{aligned}
&\log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}}) \\
&\approx \log p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}}) \\
&= \log p(\boldsymbol{w}) + \log p(\boldsymbol{r_{1:t}}|\boldsymbol{w}, \boldsymbol{s_{1:t}}) + \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}}) + \\
&const \\
&\approx \log p(\boldsymbol{w}|\boldsymbol{u_t}, \boldsymbol{C_t}) + \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}}) + const,
\end{aligned}$$

where $\boldsymbol{u_t}$ and $\boldsymbol{C_t}$ are derived from previous observations $(\boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$ and treated as known parameters.

Now, we can get

$$\boldsymbol{C}_{\boldsymbol{t+1}}^{-1} = \boldsymbol{C}_{\boldsymbol{t}}^{-1} - \frac{\partial^2 \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}})}{\partial \boldsymbol{w}^2}. \tag{A.1}$$

The second term of Eq. (A.1) is the Fisher information (the negative of the second derivative of the log likelihood with respect to $\boldsymbol{w}$) and can be calculated as

$$\begin{aligned}
&F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) \\
&= -\frac{\partial^2 \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}})}{\partial \boldsymbol{w}^2} \\
&= -\frac{\partial^2 \log p(\boldsymbol{r_{t+1}}|\boldsymbol{W}, \boldsymbol{s_{t+1}})}{\partial \boldsymbol{W}^2} \\
&= -\frac{\partial D(\boldsymbol{W}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})}{\partial \boldsymbol{W}} \\
&= -\frac{\partial D(\boldsymbol{W}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})}{\partial e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}}} \cdot \frac{\partial e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}}}{\partial \boldsymbol{W} \cdot \boldsymbol{s_{t+1}}} \cdot \frac{\partial \boldsymbol{W} \cdot \boldsymbol{s_{t+1}}}{\partial \boldsymbol{W}}
\end{aligned}$$

Given $\frac{\partial \boldsymbol{AXB}}{\partial \boldsymbol{X}} = \boldsymbol{B}^T \otimes \boldsymbol{A}$, we have

$$\begin{aligned}
&F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) \\
&= (\boldsymbol{s_{t+1}} \otimes \boldsymbol{I}) \cdot diag(e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}}) \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}).
\end{aligned}$$

where $\otimes$ represents Kronecker product, *diag* is a function that takes all the elements of a matrix and reconstruct them into a diagonal matrix and $\boldsymbol{I}$ is a $m \times m$ identity matrix. It's interesting to see that the Fisher information does not depend on the response vector $\boldsymbol{r_{t+1}}$.

Given the above equations, the objective function can be solved as

$$\begin{aligned}
&\arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})) \\
&= \arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log |\boldsymbol{C_{t+1}}| \\
&= \arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log |\boldsymbol{C_{t+1}}^{-1}| \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \log |\boldsymbol{C_t}^{-1} + F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})| \\
&= \arg\max_{\boldsymbol{s_{t+1}}} (\log |\boldsymbol{C_t}^{-1}| + \log |\boldsymbol{I} + \boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})|) \qquad \text{(A.2)} \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \log |\boldsymbol{I} + \boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})| \\
&= \arg\max_{\boldsymbol{s_{t+1}}} tr(\log (\boldsymbol{I} + \boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}))) \\
&\approx \arg\max_{\boldsymbol{s_{t+1}}} tr(\boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} (e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}})^T \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}),
\end{aligned}$$

where *tr* is the function to calculate the trace of a matrix, $\boldsymbol{I}$ is an $m \times m$ identity matrix and $\boldsymbol{J}$ is a $m \times 1$ vector with ones in all its entries.

A.2 Synthetic Data

Here we give the detailed procedure to generate synthetic data.

We use three steps to generate simulated spike train data. First, the structure of the functional network is proposed. Then, a GLM parameter matrix is created according to the functional network. Finally, simulated spike train data is generated by running the GLM. All the parameters in the simulation process are chosen to mimic real neurons.

Propose structure of the functional network. In this step, we use two methods to build the functional network: 1) manually creating some small networks to show the effectiveness and intuition of our models; 2) generating random networks to evaluate the generalizability of our models. The random networks are generated as follows. First the number of nodes m is chosen. Then for each pair of nodes in the network, they have a probability of 0.3 of getting connected and for each connection, it has a probability of 0.2 of being an inhibitory connection. We use $[1, 20]$ as time lags for an inhibitory connection and a time lag drawn from $[1, 2]$ for an excitatory connection. Each node also has an inhibitory connection directed to itself to mimic the refractory period of neurons.

Create GLM parameter matrix from the functional network. First, for any neuron i, the value of b_i is randomly drawn from $[-9, -3]$, which means a spontaneous firing rate ranging from 0.0001 to 0.05. Then other values in the parameter matrix are picked according to the connections in this network. For example, if there is an edge from neuron i to neuron j with time lag l, the corresponding value in the parameter matrix $\boldsymbol{W}$ will be set to a none-zero value to represent the strength of this connection. If the connection is excitatory, the value will be randomly drawn from $[0, -b_i]$. If the connection is inhibitory, the value will be set to b_i to ensure the inhibition of all excitatory inputs.

Generate simulated spike train data. Given a GLM parameter matrix, we run the GLM with initial states of neurons set to be 1. At each time step, the probabilities of getting a spike are computed before Gaussian noises with mean 0 and variance 0.0005 are added. Then Bernoulli distributions with these probabilities are used to draw values of 0 or 1. If a neuron is intervened in the simulated experiment, its value will be set to 0 no matter what the probability is. We continue this process until the desired length of recording is reached.

A.3 Small Networks

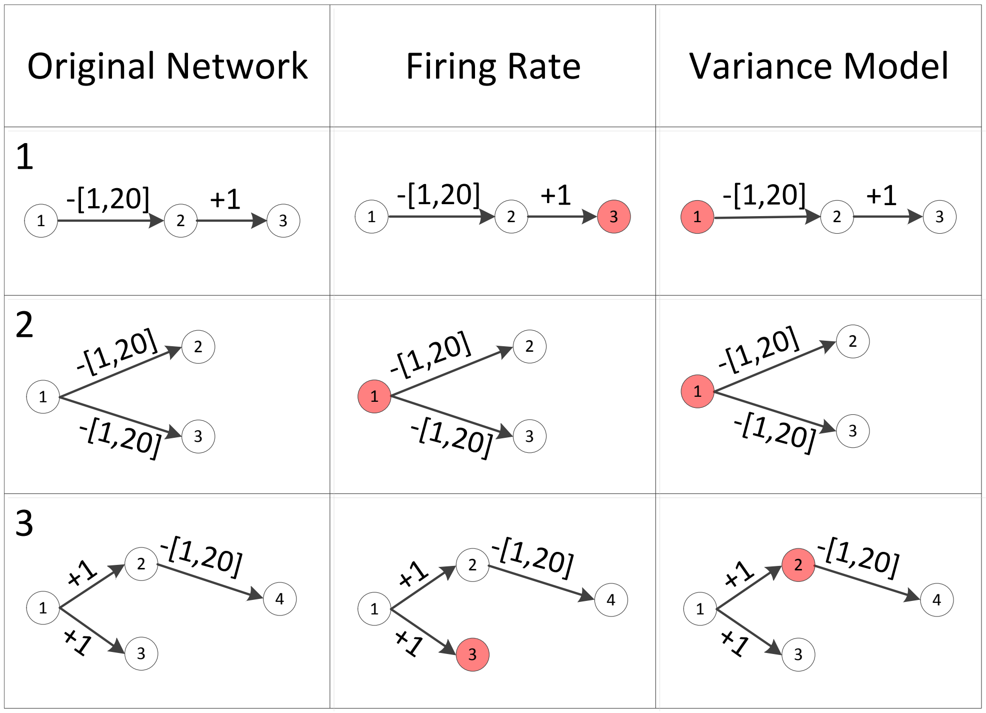
Here we report experiments that could validate the intuitions behind our active learning models.

Since the variance and validation model are using different strategies to choose interventions, we use some small networks to illustrate the effectiveness of their intuitions. In the following experiments, in addition to the aforementioned two baseline methods, we add another one, *First recording*, which infers the GLM only with the initial recording.

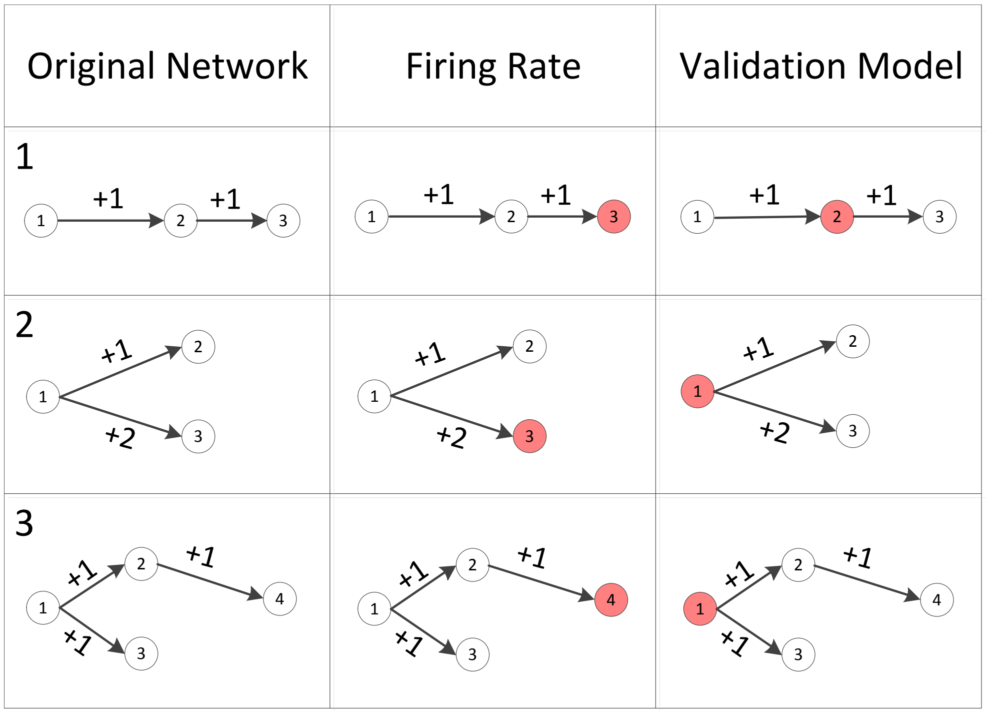
Variance model. Figure A.1(a) shows three types of functional networks with inhibitory connections. All the neurons in these networks have a spontaneous firing rate of 0.05. The first column in Figure A.1(a) shows the structures of the proposed networks. The second column shows the neuron that will be picked as the intervention target if it's selected by firing rate. The third column shows the neuron that the variance model will choose. For instance, in network 1, node 3 is chosen according to firing rate and node 1 is chosen by the variance model. It's clear that when network 1 is inferred by using only observational data, the connection from node 1 to node 2 will likely be missed. By silencing node 1, which is chosen by the variance model, we can discover this connection. One the other hand, choosing interventions by firing rate will not help.

To illustrate the effectiveness of the variance model, we show the error e of the inferred model when an additional recording is add under the guidance of different methods. As shown in Figure A.2(a), if the simulated intervention experiments are conducted under the guidance of the variance model, we can always achieve the highest accuracy.

Validation model. Similarly for the validation model, we manually create three functional networks shown in Figure A.1(b). The networks are constructed with only excitatory connections and each node has a firing rate of 0.0001 except node 1 has a firing rate of 0.05, which means the activities of the networks are mainly driven by node 1. By comparing the intervention targets chosen by the validataion model and according to the firing rate, we can see that the validation model could always choose the node that can maximally filter spurious connections. As shown in Figure A.2(b), the validation model could achieve the highest accuracy gain in all three cases.

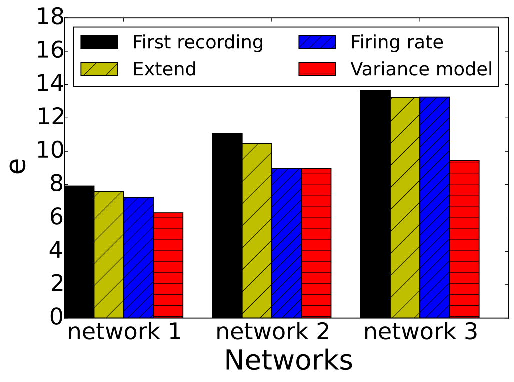


(a) Examples for the variance model.

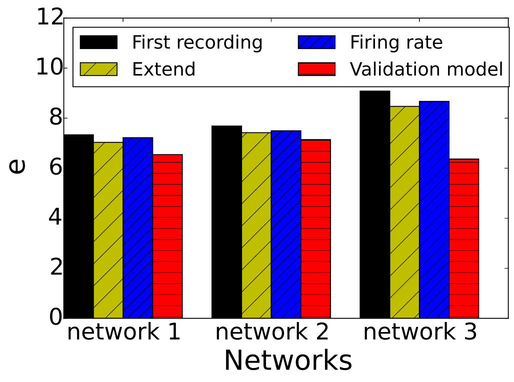


(b) Examples for the validation model.

Figure A.1: The neurons that will be chosen by different methods for small networks.



(a) When examples for the variance model are used.



(b) When examples for the validation model are used.

Figure A.2: The errors of the inferred functional networks by different methods for small networks.

Bibliography

[1] J. T. Ballew, J. A. Murray, P. Collin, M. Mäki, M. F. Kagnoff, K. Kaukinen, and P. S. Daugherty, *Antibody biomarker discovery through in vitro directed evolution of consensus recognition epitopes*, *Proceedings of the National Academy of Sciences* **110** (2013), no. 48 19330–19335.

[2] T. L. Bailey and C. Elkan, *Fitting a mixture model by expectation maximization to discover motifs in bipolymers*, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology* (1994) 28–36.

[3] F. Zambelli, G. Pesole, and G. Pavesi, *Motif discovery and transcription factor binding sites before and after the next-generation sequencing era*, *Briefings in bioinformatics* **14** (2013), no. 2 225–237.

[4] T. L. Bailey, *Dreme: motif discovery in transcription factor chip-seq data*, *Bioinformatics* **27** (2011), no. 12 1653–1659.

[5] P. Machanick and T. L. Bailey, *Meme-chip: motif analysis of large dna datasets*, *Bioinformatics* **27** (2011), no. 12 1696–1697.

[6] J. E. Reid and L. Wernisch, *Steme: efficient em to find motifs in large data sets*, *Nucleic acids research* **39** (2011), no. 18 e126.

[7] T. Kim, M. S. Tyndel, H. Huang, S. S. Sidhu, G. D. Bader, D. Gfeller, and P. M. Kim, *Musi: an integrated system for identifying multiple specificity from very large peptide or nucleic acid data sets*, *Nucleic acids research* (2011).

[8] C. E. Grant, T. L. Bailey, and W. S. Noble, *Fimo: Scanning for occurrences of a given motif*, *Bioinformatics* **27** (2011), no. 7 1017–1018.

[9] S. Burkhardt and J. Kärkkäinen, *One-gapped q-gram filters for levenshtein distance*, in *Combinatorial pattern matching*, pp. 225–234, Springer, 2002.

[10] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble., *Meme suite: tools for motif discovery and searching*, *Nucleic acids research* **37** (2009), no. 2 202–208.

[11] M. Andreatta, O. Lund, and M. Nielsen, *Simultaneous alignment and clustering of peptide data using a gibbs sampling approach*, *Bioinformatics* **29** (2013), no. 1 8–14.

[12] M. Sahli, E. Mansour, and P. Kalnis, *Acme: A scalable parallel system for extracting frequent patterns from a very long sequence*, *The VLDB Journal* **23** (2014), no. 6 871–893.

[13] I. R. Rebollo, M. Sabisz, V. Baeriswyl, and C. Heinis, *Identification of target-binding peptide motifs by high-throughput sequencing of phage-selected peptides*, *Nucleic acids research* **42** (2014), no. 22 e169–e169.

[14] W. L. Matochko, K. Chu, B. Jin, S. W. Lee, G. M. Whitesides, and R. Derda, *Deep sequencing analysis of phage libraries using illumina platform*, *Methods* **58** (2012), no. 1 47–55.

[15] C. Jia, M. B. Carson, and J. Yu, *A fast weak motif-finding algorithm based on community detection in graphs*, *BMC bioinformatics* **14** (2013), no. 1 1471–2105.

[16] D. E. Schones, P. Sumazin, and M. Q. Zhang, *Similarity of position frequency matrices for transcription factor binding sites*, *Bioinformatics* **21** (2005), no. 3 307–313.

[17] L. Marsan and M. Sagot, *Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification*, *Journal of Computational Biology* **7** (2000), no. 3-4 345–362.

[18] G. Pavesi, G. Mauri, and G. Pesole, *An algorithm for finding signals of unknown length in dna sequences*, *Bioinformatics* **17** (2001), no. 1 S207–S214.

[19] T. L. Bailey, N. Williams, C. Misleh, and W. W. Li., *Meme: discovering and analyzing dna and protein sequence motifs*, *Nucleic acids research* **34** (2006), no. 2 369–373.

[20] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, *Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform*, *Nucleic acids research* **30** (2002), no. 14 3059–3066.

[21] A. F. Neuwald, J. S. Liu, and C. E. Lawrence, *Gibbs motif sampling: detection of bacterial outer membrane protein repeats*, *Protein science* **4** (1995), no. 8 1618–1632.

[22] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church, *Computational identification of cis-regulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae*, *Journal of molecular biology* **296** (2000), no. 5 1205–1214.

[23] S. Burkhardt and J. Kärkkäinen, *Better filtering with gapped q-grams*, *Fundamenta informaticae* **56** (2003), no. 1 51–70.

[24] M. Fontaine, S. Burkhardt, and J. Kärkkäinen, *Bdd-based analysis of gapped q-gram filters*, *International Journal of Foundations of Computer Science* **16** (2005), no. 06 1121–1134.

[25] B. Ma, J. Tromp, and M. Li, *Patternhunter: faster and more sensitive homology search*, *Bioinformatics* **18** (2002), no. 3 440–445.

[26] M. Li, B. Ma, D. Kisman, and J. Tromp, *Patternhunter ii: Highly sensitive and fast homology search*, *Journal of Bioinformatics and Computational Biology* **2** (2004), no. 03 417–439.

[27] E. Giaquinta, S. Grabowski, and E. Ukkonen, *Fast matching of transcription factor motifs using generalized position weight matrix models*, *Journal of Computational Biology* **20** (2013), no. 9 621–630.

[28] R. Baxter, P. Christen, and T. Churches, *A comparison of fast blocking methods for record linkage*, in *ACM SIGKDD*, vol. 3, pp. 25–27, Citeseer, 2003.

[29] U. Draisbach and F. Naumann, *A generalization of blocking and windowing algorithms for duplicate detection*, in *Data and Knowledge Engineering (ICDKE), 2011 International Conference on*, pp. 18–24, IEEE, 2011.

[30] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, *Robust and efficient fuzzy match for online data cleaning*, in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 313–324, ACM, 2003.

[31] C. Xiao, W. Wang, and X. Lin, *Ed-join: an efficient algorithm for similarity joins with edit distance constraints*, *Proceedings of the VLDB Endowment* **1** (2008), no. 1 933–944.

[32] N. Begum and E. Keogh, *Rare time series motif discovery from unbounded streams*, *Proceedings of the VLDB Endowment* **8** (2014), no. 2 149–160.

[33] Y. Li, U. Leong Hou, M. L. Yiu, and Z. Gong, *Quick-motif: An efficient and scalable framework for exact motif discovery*, ICDE, 2015.

[34] D. O. Hebb, *The organization of behavior: A neuropsychological approach.* John Wiley & Sons, 1949.

[35] J. Keat, P. Reinagel, R. C. Reid, and M. Meister, *Predicting every spike: a model for the responses of visual neurons*, *Neuron* **30** (2001), no. 3 803–817.

[36] J. W. Pillow, L. Paninski, V. J. Uzzell, E. P. Simoncelli, and E. Chichilnisky, *Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model*, *Journal of Neuroscience* **25** (2005), no. 47 11003–11013.

[37] I. H. Stevenson, J. M. Rebesco, L. E. Miller, and K. P. Körding, *Inferring functional connections between neurons*, *Current opinion in neurobiology* **18** (2008), no. 6 582–588.

[38] I. H. Stevenson, J. M. Rebesco, N. G. Hatsopoulos, Z. Haga, L. E. Miller, and K. P. Kording, *Bayesian inference of functional connectivity and network structure from spikes*, *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **17** (2009), no. 3 203–213.

[39] J. W. Pillow, Y. Ahmadian, and L. Paninski, *Model-based decoding, information estimation, and change-point detection techniques for multineuron spike trains*, *Neural computation* **23** (2011), no. 1 1–45.

[40] G. L. Gerstein and K. L. Kirkland, *Neural assemblies: technical issues, analysis, and modeling*, *Neural Networks* **14** (2001), no. 6 589–598.

[41] P. Jonas and G. Buzsaki, *Neural inhibition*, *Scholarpedia* **2** (2007), no. 9 3286.

[42] M. R. Cohen and A. Kohn, *Measuring and interpreting neuronal correlations*, *Nature neuroscience* **14** (2011), no. 7 811–819.

[43] S. Ostojic, N. Brunel, and V. Hakim, *How connectivity, background activity, and synaptic properties shape the cross-correlation between spike trains*, *Journal of Neuroscience* **29** (2009), no. 33 10234–10253.

[44] K. R. Tovar, D. C. Bridges, B. Wu, C. Randall, M. Audouard, J. Jang, P. K. Hansma, and K. S. Kosik, *Recording action potential propagation in single axons using multi-electrode arrays*, *bioRxiv* (2017) 126425.

[45] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, *Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering*, *Neural computation* **16** (2004), no. 8 1661–1687.

[46] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. Chichilnisky, and E. P. Simoncelli, *Spatio-temporal correlations and visual signalling in a complete neuronal population*, *Nature* **454** (2008), no. 7207 995.

[47] I. Bomash, Y. Roudi, and S. Nirenberg, *A virtual retina for studying population coding*, *PloS one* **8** (2013), no. 1 e53363.

[48] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, *A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects*, *Journal of neurophysiology* **93** (2005), no. 2 1074–1089.

[49] J. M. Rebesco, I. H. Stevenson, K. P. Körding, S. A. Solla, and L. E. Miller, *Rewiring neural interactions by micro-stimulation*, *Frontiers in systems neuroscience* **4** (2010).

[50] B. Dunn, M. Mørreaunet, and Y. Roudi, *Correlations and functional connections in a population of grid cells*, *PLoS computational biology* **11** (2015), no. 2 e1004052.

[51] I. M. Park, M. L. Meister, A. C. Huk, and J. W. Pillow, *Encoding and decoding in parietal cortex during sensorimotor decision-making*, *Nature neuroscience* **17** (2014), no. 10 1395–1403.

[52] D. de Santos-Sierra, I. Sendiña-Nadal, I. Leyva, J. A. Almendral, S. Anava, A. Ayali, D. Papo, and S. Boccaletti, *Emergence of small-world anatomical networks in self-organizing clustered neuronal cultures*, *PloS one* **9** (2014), no. 1 e85828.

[53] M. Ivenshitz and M. Segal, *Neuronal density determines network connectivity and spontaneous activity in cultured hippocampus*, *Journal of neurophysiology* **104** (2010), no. 2 1052–1060.

[54] S. Linderman, C. H. Stock, and R. P. Adams, *A framework for studying synaptic plasticity with neural spike train data*, in *Advances in Neural Information Processing Systems*, pp. 2330–2338, 2014.

[55] P. D. Maia and J. N. Kutz, *Compromised axonal functionality after neurodegeneration, concussion and/or traumatic brain injury*, *Journal of computational neuroscience* **37** (2014), no. 2 317–332.

[56] J. Lewi, R. J. Butera, and L. Paninski, *Efficient active learning with generalized linear models*, in *International Conference on Artificial Intelligence and Statistics*, pp. 267–274, 2007.

[57] Z. Chen, *An overview of bayesian methods for neural spike train analysis*, *Computational intelligence and neuroscience* **2013** (2013) 1.

[58] D. Patnaik, S. Laxman, and N. Ramakrishnan, *Discovering excitatory networks from discrete event streams with applications to neuronal spike train analysis*, in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pp. 407–416, IEEE, 2009.

[59] J. Lewi, D. M. Schneider, S. M. Woolley, and L. Paninski, *Automating the design of informative sequences of sensory stimuli*, *Journal of computational neuroscience* **30** (2011), no. 1 181–200.

[60] J. Pearl, *Causality: models, reasoning and inference*, *Economet. Theor* **19** (2003) 675–685.

[61] W. Lam and F. Bacchus, *Learning bayesian belief networks: An approach based on the mdl principle*, *Computational intelligence* **10** (1994), no. 3 269–293.

[62] D. Heckerman, D. Geiger, and D. M. Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, *Machine learning* **20** (1995), no. 3 197–243.

[63] S. Tong and D. Koller, *Active learning for structure in bayesian networks*, in *International joint conference on artificial intelligence*, vol. 17, pp. 863–869, LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[64] Y.-B. He and Z. Geng, *Active learning of causal networks with intervention experiments and optimal designs*, *Journal of Machine Learning Research* **9** (2008), no. 11.

[65] M. Steyvers, J. B. Tenenbaum, E.-J. Wagenmakers, and B. Blum, *Inferring causal networks from observations and interventions*, *Cognitive science* **27** (2003), no. 3 453–489.

[66] N. Dojer, *Learning bayesian networks does not have to be np-hard*, in *Mathematical Foundations of Computer Science 2006*, pp. 305–314. Springer, 2006.

[67] Z. Wang and L. Chan, *Using bayesian network learning algorithm to discover causal relations in multivariate time series*, in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pp. 814–823, IEEE, 2011.

[68] N. X. Vinh, M. Chetty, R. Coppel, and P. P. Wangikar, *Globalmit: learning globally optimal dynamic bayesian network with the mutual information test criterion*, *Bioinformatics* **27** (2011), no. 19 2765–2766.

[69] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search.* MIT press, 2000.

[70] J. Lewi, R. Butera, and L. Paninski, *Sequential optimal design of neurophysiology experiments*, *Neural Computation* **21** (2009), no. 3 619–687.