Understanding the Computational Power of Spiking Neural Network

Chi-Ning Chou[1], Kai-Min Chung[1], and Chi-Jen Lu[1]

[1]Institute of Information Science, Academia Sinica

January 5, 2017

Abstract

Biologists have proposed dozens of neural models to explain human brains which have been utilized by computer scientists to design *artificial neural network (ANN)* and yielded amazing results in many practical fields such as machine learning and signal processing etc. However, it remains a mystery why neural networks have nontrivial computational power. In the literature, theory towards an explanation of the power of neural networks can be divided into two categories: the structural aspect and algorithmic aspect. The former, e.g. [Maa99], [Maa97], [PMB12], considered how can information being propagated in neural network, and how can neural network being constructed into gadgets to provide digital computation. The later, e.g. [ABGM14] [AGMM15], focused on how to design neural algorithm and provide provable guarantee.

In this work, we view neural networks through the lens of computations. Concretely, we focused on *spiking neural network (SNN)* without any artificial construction. We asked the two fundamental questions: "What is the role of spikes in neural network," and "What intrinsic computational problem does SNN actually solve?" Motivated by an empirical finding suggested that the firing rate of SNN is relevant to quadratic programming, we first successfully derived a geometric rate convergence result that confirmed this heuristics. Furthermore, we found out that SNN is actually implicitly solving a much more sophisticated problem, the ℓ_1 minimization problem. By connecting SNN and ℓ_1 minimization problem in their *dual space*, we answered the above two questions by viewing spikes as approximating the projection operator in a gradient descent algorithm for the dual program of ℓ_1 minimization problem.

1 Introduction

Understanding the reason why our brains are so powerful has been one of the major problems in science since the beginning of last century. Biologists have proposed dozens of mathematical models for explaining the underlying mechanisms of this complicated nervous system. The *integrate-and-fire* model proposed by Lapicque in 1907 [Lap07] has been one of the most widely-used model among many competitors. A number of variations such as the Hodgkin-Huxley neuron model [HH52], leaky integrate-and-fire (LIF) model and Stein's model [Ste65] were also established for

more sophisticated modeling. This family of neural models are known as the *electrical voltage model*. Most of the works in neuroscience care about either whether the statistical properties of neural models such as firing rate, spiking time fit well to or how well the model can predict the behaviors of experimental observation.

In this work, we focused on the spiking neural networks with integrate-and-fire setting [Lap07, Adr26]. The model consists of neurons associated with potential value, a.k.a. voltage. When the potential exceeds the given threshold, a neuron will *fire a spike* and drop its potential by one unit. For other neurons that receive the spike, their potentials will either increase or decrease according to the *connectivities* among these neurons and the spiking one. Apart from the spikes among neurons, a fixed input charging will also increase or decrease the potential of each neuron with a constant rate. It turns out that this simple dynamics and its variation can nicely characterize the natural behavior of neurons [I$^+$03]. Concretely, the experimental data from neural biology recording the synapses is statistically similar to the empirical performance of these mathematical models.

When putting several neurons into a network, these mathematical models become even more interesting and people would refer to this kind of neural network as *spike neural network* (SNN). Here, we briefly introduce a simple version of SNN's dynamics. In a network consisting of n neurons indexed by $1, 2, \ldots, n$, use vector $\mathbf{u}(t)$ to record the potentials of them at time t. For each pair of neuron (i, j), record the connectivity between neuron i and j into the (i, j) entry of matrix A. If neuron i fires a spike at time t, then we put a delta function in $\mathbf{s}_i(t)$ otherwise putting a zero. Finally, denote the input charge to each neuron in vector $\mathbf{I}$. Now, we can write down a simple SNN dynamics as follows[1]:

$$\frac{d\mathbf{u}(t)}{dt} = -A\mathbf{s}(t) + \mathbf{I} \tag{1}$$

Studying the joint behavior in SNN is a natural and important issues for neural scientists. Specifically, the firing rate of SNN is arguably one of the most important features in neural study. As its name reveals, the firing rate of an SNN with dynamics in (1) is defined as the average of the total amount, *i.e.,* $\int_0^{\mathsf{T}} \mathbf{s}(t)dt/T$. Started by Adrian and Zotterman [Adr26] in 1926, firing rate has been treated as an important role in neural coding and information processing. Understand, estimating and reproducing the firing rate have been an important task in computational neuroscience. In 2013, Barrett et al. [BDM13] found out that the firing rate from a network of leaky integrate-and-fire neurons can be estimated from a *quadratic programing problem* with parameters related to the setting of the network. They performed simulation in a two-neuron example and resulted in a nice empirical evidence. They further remarked that this heuristics can be interpreted as using neural model to solve a least square problem, which is definitely a non-trivial task for such a simple dynamics.

As we are theoretical computer scientists, we formulated the above observations into the following questions:

- What is the role of spikes in neural network?
- What intrinsic problem does SNN actually solve?

The answer we gave to the first problem is: SNN is actually solving ℓ_1 minimization problem which is far more beyond the previous belief in only solving linear system problem. As to the second

[1]This is a simple integrate-and-fire model which might not be realistic in many situations. But for simplicity, we consider this model here.

problem, our result is much more astonishing: SNN is performing a primal-dual algorithm in which simultaneously select the correct variables and approximate the solution.

1.1 Related works

While biologists have constructed a series of nice mathematical models for the nature neural activities, computer scientists found them to be good computation components for designing brilliant algorithms. In 1943, McCulloch and Pitts [MP43] initiated the study of using neural network as a computation primitive. Rumelhart and McClelland [RMG$^+$88] proposed the backpropagation algorithm in 1988. Hochreiter and Schmidhuber [HS97] designed the model of *recurrent neural network* (RNN) in 1997 to characterize the long term and short term memory. Convolution neural network was introduced in 1980 by Fukushima and Kunihiko [Fuk80] and has great applications in lots of machine learning problems [KSH12]. Through this line of exciting works, computer scientists have found great heuristics in utilizing the nice non-linear and non-convex structure of neural network to design sophisticated algorithms to solve practical problems that cannot be solved with traditional algorithms.

However, we were not satisfied with these explanations. Demonstrating the computability of neural network did not answer the fundamental question of what does neural network is exactly doing. Just a the basic computation of circuit is doing simple logical operation, we want to understand what kind of basic computation problem does neural network is solving and how.

However, these works focused on the *digital* aspect of SNN, *i.e.,* using SNN to construct small digital gadgets and simulate modern computation model, and overlook the fundamental reason why SNN can provide such computational power. How can these seemingly simple and symmetric structure can carry computation? Apart from using SNN to compose small computation gadgets, can the structure of SNN being more powerful?

As we are *theoretical* computer scientists, we are not satisfied with the current neural algorithms which are extremely lacked of theoretical proof for the reason they work well. Thus, a natural and simple questions we would ask is:

What kind of computation problems do neural network actually solves?

Furthermore, we are curious about

How do neural network carry computation?

So far, most of the theoretical analyses of neural network were focusing on using neural network as a computation component for general purpose computation model. For instance, Siegelmann and Sontag [SS91] showed that neural network is *Turing-complete*. Following works even showed that neural network is *super-Turing-complete* [BGS97, Sie03]. See [Sie12] for an overview. Maass [Maa97] studied the computational complexity of spike neural network (SNN) and constructed [Maa96] SNN for threshold circuits, Turing machines, and RAM. See [PMB12] for a nice overview.

1.2 Paper structure

The rest of the paper is organized as follows. In Section 2 we introduced the notations, formulated the models and stated our results. In Section 3 we proved the SNN convergence for the linear system problem. The geometric interpretation of SNN is demonstrated in Section 4 and we proved that SNN is performing ℓ_1 minimization.

2 Preliminaries

In this section, we will first introduce the spiking neural network (SNN) model we are going to investigate in Section 2.1. Variants of SNN models in the literature will be discussed in Appendix C. In Section 2.2, we introduced the three problems that are closely related to our SNN model.

2.1 The simple SNN model

A SNN model is a dynamic system that consists of n neurons numbered $1, 2, \ldots, n$. For $i \in [n] := \{1, 2, \ldots, n\}$, neuron i is associated with a time-varying *potential*[2] $\mathbf{u}_i(t) \in \mathbb{R}$ for time $t \geq 0$. Assume $\mathbf{u}_i(0) = 0$. Neuron i also receives a steady rate of *external charging*[3] specified by constant $\mathbf{I}_i \in \mathbb{R}$, *i.e.,* the potential of neuron i is added by $\mathbf{I}_i$ in after an unit of time. There is also a *threshold* $\eta > 0$ such that a neuron will fire a spike when its potential exceeds η. Specifically, here we consider a *two-sided* SNN model[4], *i.e.,* neuron i will fire a positive (resp. negative) spike at time t if $\mathbf{u}_i(t) > \eta$ (resp. $\mathbf{u}_i(t) < -\eta$). Spikes will cause either inhibition or exhibition to the neighboring neurons according to the connectivity between them. Concretely, suppose the connectivity between neuron i and neuron j is C_{ij}. If neuron i fires a positive (resp. negative) spike, then the potential of neuron j will decrease (resp. increase) by C_{ij}. In particular, the potential of neuron i itself is decreased (resp. increased) by C_{ii}.

Graphically, an SNN can be thought of as a weighted graph where nodes are neurons and the weight of an edge records the *connectivity*[5] between two neurons. Namely, the adjacency matrix C of the graph records the connectivities between any pairs of neurons. We call it the *connectivity matrix*. Here, we does not set any constraint to C, however, in most of the situation, we would assume C is symmetric. Particularly, an interesting example [BD11, BDM13, SD14] to study is given by $C = A^\intercal A$ and $\mathbf{I} = A^\intercal \mathbf{b}$ for some matrix A and vector $\mathbf{b}$ where $A^\intercal$ is the transpose of A. Figure 1 illustrates an example of a three-neuron SNN.

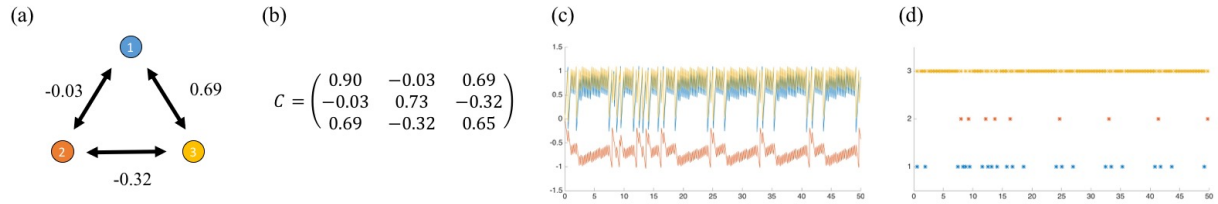

Figure 1: A three-neuron SNN. The connectivities among the three neurons colored in blue, orange, and yellow, respectively, are shown in (a) with connectivity matrix given in (b). The external charging rate is $\mathbf{I} = (2.175, -0.95, 2.025)$. We ran a simulation and plotted the potentials in the first 50 seconds in (c). The razor plot, (d), recorded the timing when neurons fired.

We called the SNN model defined above a *simple SNN* and its dynamic is described by the following differential equation

[2]Sometimes being referred to *membrane potential*, *voltage* or *state variable*.

[3]Sometimes begin referred to *input signal* or *input current*.

[4]We remark that typically, the SNN models consider only positive spike. Here we consider two-sided model for notationally convenience since there is a simple reduction to one-sided model. Details will be discussed in Section C.

[5]Sometimes being refereed to *connection strength* or *postsynaptic potential*.

$$\frac{d\mathbf{u}(t)}{dt} = -A^\intercal A\mathbf{s}(t) + A^\intercal \mathbf{b}, \tag{2}$$

where $\mathbf{u}_i(0) = 0$ for all $i \in [n]$ and $\mathbf{s}_i(t)$ records the firing time of neurons i at time t. Concretely,

$$\mathbf{s}_i(t) = \sum_k \delta(t - t_i^+) - \sum \delta(t - t_i^-), \tag{3}$$

where $\{t_i^+\}$ (or $\{t_i^-\}$) records the positive (or negative) firing times of neuron i. As a result, for any $T > 0$, the total numbers of spikes before T are recorded in vector

$$\mathbf{S}(T) = \int_0^T \mathbf{s}(t)dt, \tag{4}$$

and the firing rate at time T is defined as $\hat{\mathbf{x}}(T) := \mathbf{S}(T)/T$. Formally, we define a continuous simple SNN as follows.

Definition 1 (continuous simple SNN) *Given an $m \times n$ matrix A, vector $\mathbf{b}$, and $\eta > 0$, the dynamics (2) defines a simple continuous SNN with potential $\mathbf{u}(t) \in \mathbb{R}^n$, spike vector $\mathbf{s}(t) \in \{0, \pm 1\}^n$, and firing rate $\hat{\mathbf{x}}(t) \in \mathbb{R}^n$ for any $t \geq 0$.*

Moreover, we can *discretize* the dynamic of a simple SNN defined in (2) by setting $\mathbf{u}[\tau] := \mathbf{u}(\tau \cdot \Delta t)$ for some time step $\Delta t > 0$. The dynamics of a simple discrete SNN is then

$$\mathbf{u}[\tau + 1] = \mathbf{u}[\tau] - A^\intercal A\mathbf{s}[\tau] + A^\intercal \mathbf{b} \cdot \Delta t, \ \forall \tau \in \mathbb{N} \cup \{0\}, \tag{5}$$

where $\mathbf{u}_i[0] = 0$ for all $i \in [n]$. Similarly, the spike vector $\mathbf{s}[t]$ is defined as $\mathbf{s}_i[\tau] = \mathbf{1}_{\mathbf{u}_i[\tau] > \eta} - \mathbf{1}_{\mathbf{u}_i[\tau] < -\eta}$ for all $i \in [n]$. Accordingly, the total number of spikes after $L \in \mathbb{N}$ time step is now given by $\mathbf{S}[L] = \sum_{t=0}^{T-1} \mathbf{s}[\tau]$ and the firing rate is $\hat{\mathbf{x}}[L] = \mathbf{S}[L]/(L \cdot \Delta t)$. Formally, we define the discrete simple SNN as follows.

Definition 2 (discrete simple SNN) *Given an $m \times n$ matrix A, vector $\mathbf{b}$, and $\eta > 0$, the dynamics in (5) defines a simple discrete SNN with potential $\mathbf{u}[\tau] \in \mathbb{R}^n$, spike vector $\mathbf{s}[\tau] \in \{0, \pm 1\}^n$, and firing rate $\hat{\mathbf{x}}[\tau] \in \mathbb{R}^n$ for any $\tau \in \mathbb{N} \cup \{0\}$.*

Note that in continuous simple SNN, we use T to denote the time and in discrete simple SNN we use L to denote the number of time step. To translate from discrete to continuous, just simply think of $T = L \cdot \Delta t$.

Remark 1 The notion of **time** in the discrete simple SNN and the continuous simple SNN are different. For discrete simple SNN, since we have to pay for discretization, the there would be an additional $1/\Delta t$ term in the time complexity. As to the continuous simple SNN, there's no such term and we consider the continuous time measure. Basically, this two time complexity notions are different since the implementations are different.

Remark 2 Note that when we let $\Delta t \to 0$ in the discrete simple SNN, it becomes continuous SNN. If that is the case, $L/\Delta t$ in the discrete simple SNN will be the continuous time in the continuous simple SNN.

2.2 Computational problems

In this subsection, we are going to introduce three computational problems: *linear system*, *LASSO*, and ℓ_1 *minimization*. We will formally define each problems and its goal. Current status of each problem and important results will also be discussed.

Let us begin with some notation. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be two vectors. Let $|\mathbf{x}| \in \mathbb{R}^n$ denote the entry-wise absolute value of $\mathbf{x}$. The notation $\mathbf{x} \preceq \mathbf{y}$ means that, $\mathbf{x}_i \leq \mathbf{y}_i \ \forall i \in [n]$. Let A be an $m \times n$ real matrix. For any $i \in [n]$, denote the i-th column of A as A_i and its negation to be A_{-i}, *i.e.,* $A_{-i} = -A_i$. When A is a square matrix, we define the A-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ to be $\|\mathbf{x}\|_A := \sqrt{\mathbf{x}^\intercal A \mathbf{x}}$. Let $A^\dagger$ to be the pseudo-inverse of A. Define the maximum eigenvalue of A as $\lambda_{\max}(A) := \max_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_A / \|\mathbf{x}\|_2$, the minimum non-zero eigenvalue of A to be $\lambda_{\min}(A) := 1/(\max_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{x}\|_{A^\dagger} / \|\mathbf{x}\|_2)$, and the condition number of A to be $\kappa(A) := \lambda_{\max}(A)/\lambda_{\min}(A)$. If we do not specified, the following $\lambda_{\max}, \lambda_{\min}$, and κ are the eigenvalues and condition number of $A^\intercal A$.

2.2.1 Linear system

Barrett et al. proposed that the firing rate of an SNN can be analyzed by a quadratic programming problem [BDM13]. Concretely, this quadratic programming problem is actually a linear system problem with error measure in a quadratic form. Here, we define the linear system problem as follows.

Problem 1 (Linear system) *Given an $m \times n$ real matrix A and a vector $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} = \mathbf{b}$.*

For simplicity, we assume the linear system under consideration has at least one solution denoted as $\mathbf{x}^*$[6]. A linear system can be solved in $O(n^{2.3\cdots})$ by Gaussian elimination-based method. Once we add more constraints on A, some faster iterative algorithms can be applied. For example, if A is *positive semidefinite* (PSD), then the linear system problem can be solved in $O(n^2\sqrt{\kappa(A)}\log 1/\epsilon)$ by the preconditioned conjugate gradient method [GV61, GO88], where ϵ is a multiplicative approximation error. If A is *symmetric and diagonally dominant* (SDD), using some sophisticated preconditioning techniques [CKP$^+$14, CMP$^+$14], it can be solved in almost linear time [KS16, KOSZ13, KMP10]. See [Vis12] for a comprehensive overview.

2.2.2 LASSO

Rozell *et al.* [RJBO08] proposed the Locally Competitive Algorithm (LCA), which is a neural algorithm that aims to solve sparse approximation problems. Concretely, LCA was designed to solve the *least absolute shrinkage and selection operator (LASSO)* problem [Tib96] and a line of works [RJBO08, BRR12, Tan16] have devoted to prove the convergence. Furthermore, Shapero et al. proposed a spike version of LCA, which also aims for LASSO. Although the dynamics in both LCA and spike LCA are different from our simple SNN, we still introduced the LASSO for a comprehensive study. For a comparison of these SNN models, see Appendix C for details.

Problem 2 (LASSO) *Given an $m \times n$ real matrix A and a vector $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{x}^* := \min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \cdot \|\mathbf{x}\|_1$, where $\lambda > 0$ to be the regularization parameter.*

[6] If not specified, $\mathbf{x}^*$ is an arbitrary solution of the linear system.

Introduced by Tibshirani [Tib96] in 1996, LASSO is an optimization problem whose solutions are, in lots of applications, *sparse*. As a result, it has been widely used in variable selection, statistics and data mining, engineering [Fuc01, T$^+$97] etc. There have been variants and generalizations of LASSO [YL06, Zou06, ZH05, TSR$^+$05] and several algorithms were proposed [EHJ$^+$04, Fu98]. There were also asymptotic analysis for LASSO types algorithms [KF00].

Remark 3 LASSO is equivalent to the *Basis Pursuit De-Noising* (BPDN) problem defined as follows.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \|A\mathbf{x}-\mathbf{b}\|_2 \\ \text{subject to} \quad & \|\mathbf{x}\|_1 \leq t, \end{aligned} \tag{6}$$

where one can compute the relation among t and λ interchangeably. The major difference is that LASSO has no constraint so that algorithms for LASSO don't need to worry about being infeasible.

2.2.3 ℓ_1 minimization

In this work, we further observed and proved that the simple SNN is actually solving the ℓ_1 minimization problem, which is more sophisticated and defined as follows.

Problem 3 (ℓ_1 minimization) *Given an $m \times n$ matrix A and vector $\mathbf{b} \in \mathbb{R}^m$, the goal of ℓ_1 minimization is to solve the following optimization problem.*

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\textit{minimize}} \quad & \|\mathbf{x}\|_1 \\ \textit{subject to} \quad & A\mathbf{x}=\mathbf{b} \end{aligned} \tag{7}$$

Similarly, we assume the ℓ_1 minimization problem is feasible, *i.e.,* there exists optimal solution $\mathbf{x}^*$. ℓ_1 minimization problem, a.k.a. the *basis pursuit* (BP) problem proposed by Chen et al. [CDS01], is widely used in compressive sensing, signal processing, face recognition etc, for recovering sparse solution. Many algorithms had been proposed [YSGM10] but less theoretical analysis have appeared. To our knowledge, for $n \times n$ square matrix cases, algorithm based on linear programming can solve the problem in $O(n^3)$ [DT08]. Under some sparsity condition, the problem can be solved in $O(mnd)$ [DT08] where d is the sparsity of the solution. Nevertheless, most of the algorithms for ℓ_1 minimization problem don't have asymptotic analysis.

2.3 Our results

In this work, we analyzed the simple SNN models in Definition 1 and Definition 2. We first show that their firing rate will converge to the solution of the linear system defined by the connectivity matrix $C = A^\intercal A$ and external charging $\mathbf{I} = A^\intercal \mathbf{b}$ in a geometric converging rate.

Theorem 2.1 (informal) *Let $\epsilon > 0$ and choose the parameters properly. The discrete and continuous simple SNN converge to the solution of Problem 1 in $A^\intercal A$-norm with multiplicative error ϵ in parallel discrete time $O(\frac{\kappa(A^\intercal A)\cdot n}{\epsilon})$ and parallel continuous time $O(\frac{\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}}{\epsilon\cdot\|\mathbf{x}^*\|_{A^\intercal A}})$ respectively.*

When the linear system has more than one unique solution, it is nature to ask the question about which solution the simple SNN converge to? The first guess could be the *least square solution*, *i.e.,* the solution that has the smallest ℓ_2 norm. The corresponding least square problem [Sti81, Leo80]

has many applications in data fitting. The simplest way to find the least square solution is multiply $\mathbf{b}$ with the pseudo-inverse of A.

However, with simulation, we soon found out that this is not the case. The solution produced by our simple SNN does not converge to the least square solution. Nevertheless, we observed that the simple SNN is actually even more powerful in the way that its firing rate is even converging to the solution that has the smallest ℓ_1 norm, *i.e.,* the optimal solution of the ℓ_1 minimization problem. As we introduced before, ℓ_1 minimization is a difficult problem in general with many practical applications. We proved the following convergence theorem for the simple SNN to the ℓ_1 minimization problem.

Theorem 2.2 (informal) *When properly choosing the parameters, continuous SNN converge to the solution of Problem 3 with a primal-dual searching fashion in time* $O(\frac{\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}+\mathbf{OPT}}{\epsilon})$, *where* **OPT** *is the optimal value of* ℓ_1 *minimization problem.*

The proof of Theorem 2.1 utilizes a simple conservation observation and will be discussed in Section 3. The proof of Theorem 2.2 is much more sophisticated. We will first introduce the notion of *dual space* of the simple SNN and prove the convergence of an *ideal* algorithm in the dual space in Section 4.2. Then, we will couple the discrete simple SNN with the ideal algorithm in Section 4 and prove its convergence.

3 Warm-up: Convergence to linear system

In this section, we are going to formally present the convergence theorem of the simple SNN to the linear system problem. Let's start with the discrete simple SNN.

Theorem 3.1 *For any* $\epsilon > 0$, *let* $\hat{\mathbf{x}}(t)$ *be the firing of a discrete simple SNN defined by matrix* A *and vector* $\mathbf{b}$. *When the threshold* $\eta = \lambda_{\max}$, *the discretized step* $\Delta t < \frac{\sqrt{\lambda_{\min}}}{24\cdot\sqrt{n}\cdot\|\mathbf{x}^*\|_{A^\intercal A}}$, *and the discrete parallel time* $T \geq \frac{2\kappa(A^\intercal A)\cdot n}{\epsilon}$, *we have* $\|\hat{\mathbf{x}}(T) - \mathbf{x}^*\|_{A^\intercal A} \leq \epsilon \cdot \|\mathbf{x}^*\|_{A^\intercal A}$.

From Remark 2, we immediately get the following corollary for the continuous simple SNN.

Corollary 3.1.1 *For any* $\epsilon > 0$, *let* $\hat{\mathbf{x}}(t)$ *be the firing of a continuous simple SNN defined by matrix* A *and vector* $\mathbf{b}$. *When the threshold* $\eta \geq \lambda_{\max}$, *the continuous parallel time* $t \geq \frac{2\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}}{\epsilon\cdot\|\mathbf{x}\|_{A^\intercal A}}$, *we have* $\|\hat{\mathbf{x}}(t) - \mathbf{x}^*\|_{A^\intercal A} \leq \epsilon \cdot \|\mathbf{x}^*\|_{A^\intercal A}$.

The convergence results from Theorem 3.1 and Corollary 3.1.1 supported the observations of [BDM13] that the firing rate of SNN is solving certain quadratic programming problem with problem instance being encoded in the network connectivity and external charging.

In Section 3.1, we sketch a proof of Theorem 3.1 and the details are provided in Appendix A.

3.1 Proof of the convergence to linear system

The main technique in the proof of Theorem 3.1 utilize the *conservation argument.* Intuitively, first observe that the only two input/output of a simple SNN are the external charging $A^\intercal\mathbf{b}$ and the spikes potential variation $A^\intercal A\mathbf{s}(t)$ from spikes. Next, if one can prove the boundedness of potential

vector $\mathbf{u}(t)$, it means that the input/output of a simple SNN must be *conservative*. Technically, we identified the correct measure for the potential vector $\mathbf{u}(t)$ such that when the norm of $\mathbf{u}(t)$ is too large, then there must be some neurons fire. Moreover, as long as there are some neurons fire, the norm of $\mathbf{u}(t)$ will decrease. As a result, the boundedness of $\mathbf{u}(t)$ is derived.

First, the conservation argument is based on the following telescoping over the dynamics of the discrete simple SNN in (5).

$$\sum_{t=0}^{T-1} \mathbf{u}(t+1) = \sum_{t=0}^{T-1} \mathbf{u}(t) - A^\intercal A\mathbf{s}(t) + A^\intercal \mathbf{b}\Delta t \tag{8}$$

$$= \Big(\sum_{t=0}^{T-1} \mathbf{u}(t)\Big) - T \cdot \Delta t \cdot \Big(A^\intercal A\hat{\mathbf{x}}(T) - A^\intercal \mathbf{b}\Big). \tag{9}$$

Subtract $\sum_{t=0}^{T-1} \mathbf{u}(t)$ on both sides and divide with $T \cdot \Delta t$, we have

$$\frac{\mathbf{u}(T) - \mathbf{u}(0)}{T \cdot \Delta t} = -A^\intercal A\hat{\mathbf{x}}(T) + A^\intercal \mathbf{b}. \tag{10}$$

Next, our analysis utilized the transformation between different matrix norm, here we listed some useful facts in linear algebra for the later analysis.

Lemma 1 *Let $M^\dagger$ be the pseudoinverse of positive semidefinite matrix M.*

1. *For any $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_M = \|M\mathbf{x}\|_{M^\dagger}$.*
2. *$MM^\dagger = M^\dagger M$ is the orthogonal projection matrix of the range space of $M^\intercal M$.*

Proof: See Appendix A.1 ■

Now, take $(A^\intercal A)^\dagger$-norm on the both sides of (10), by the first part of Lemma 1 and the fact that $A^\intercal A\mathbf{x}^* = A^\intercal \mathbf{b}$, we have

$$\frac{\|\mathbf{u}(0) - \mathbf{u}(T)\|_{(A^\intercal A)^\dagger}}{T \cdot \Delta t} = \| - A^\intercal A\hat{\mathbf{x}}(T) + \mathbf{b}\|_{(A^\intercal A)^\dagger} \tag{11}$$

$$= \|\hat{\mathbf{x}}(T) - \mathbf{x}^*\|_{A^\intercal A}. \tag{12}$$

As a result, to upper bound the error $\|\hat{\mathbf{x}}(T) - \mathbf{x}^*\|_{A^\intercal A}$, it suffices to upper bound $\|\mathbf{u}(0) - \mathbf{u}(T)\|_{(A^\intercal A)^\dagger}$. Moreover, as we take $\mathbf{u}(0) = \mathbf{0}$, now we only need to show $\|\mathbf{u}(T)\|_{(A^\intercal A)^\dagger}$ is bounded by some ϵ multiplicative factor of $\|\mathbf{x}^*\|_{A^\intercal A}$ when T and Δt are chosen appropriately. The following lemma provides an upper bound for $\|\mathbf{u}(T)\|_{(A^\intercal A)^\dagger}$.

Lemma 2 *For any time $t \geq 0$, let $\mathbf{u}(t)$ be the potential of a discrete simple SNN. When the threshold $\eta = \lambda_{\max}$ and the discretized step $\Delta t < \frac{\sqrt{\lambda_{\min}}}{24 \cdot \sqrt{n} \cdot \|\mathbf{x}^*\|_{A^\intercal A}}$, we have $\|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger} \leq 2\sqrt{\kappa(A^\intercal A) \cdot \lambda_{\max} \cdot n}$.*

Proof: The lemma can be proved with induction and some properties of the dynamics of SNN. See Appendix A.2 for details. ■

We now prove Theorem 3.1 using Lemma 2.

Proof of Theorem 3.1: By (12) and Lemma 2, when $\Delta t < \frac{\lambda_{\min}}{24\sqrt{n}\cdot\|\mathbf{x}^*\|_{A^\intercal A}}$ and $T \geq \frac{2\kappa(A^\intercal A)\cdot n}{\epsilon}$, we have

$$\|\hat{\mathbf{x}}(T) - \mathbf{x}^*\|_{A^\intercal A} \leq \frac{\|\mathbf{u}(T)\|_{(A^\intercal A)^\dagger}}{T\cdot\Delta t} \tag{13}$$

$$\leq \frac{2\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}\cdot\epsilon\cdot\|\mathbf{x}^*\|_{A^\intercal A}}{2\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}} = \epsilon\cdot\|\mathbf{x}^*\|_{A^\intercal A}. \tag{14}$$

■

4 ℓ_1 minimization and the dual world

In the previous section we saw that SNN is solving a linear system problem $A\mathbf{x} = \mathbf{b}$ where $A^\intercal A$ is the connectivity matrix and $A^\intercal\mathbf{b}$ is the input charge of SNN. Note that since the convergence in Theorem 3.1 only said that the $(A^\intercal A)$-norm of the residual error will converge to 0, when the solution of $A\mathbf{x} = \mathbf{b}$ is not unique, it is natural to ask: Which solution will the firing rate converge to? Interestingly, we observed and proved that the firing rate will converge to the solution with the least ℓ_1 norm, *i.e.,* the the firing rate is the solution of the ℓ_1 minimization problem defined in Problem 3 with parameter A and $\mathbf{b}$.

It turned out that the connection between SNN and ℓ_1 minimization is much deeper than previous simple connection with linear system. There are three steps to prove the connection. First, define a *dual world* for SNN. Next, define an *ideal* algorithm in that dual world and prove that it converge to the optimal solution of the dual program of ℓ_1 minimization problem. Finally, couple SNN with the ideal algorithm and show that SNN will eventually being stuck in the neighborhood of the optimal solution of the dual of ℓ_1 minimization problem.

There are several new notions needed to be introduced. In the rest of this section, we first defined the *primal world* and the *dual world* of SNN in Section 4.1 and Section 4.2 respectively and discuss its connection with the primal and dual program of ℓ_1 minimization. Then, in Section 5, we will prove this connection in the continuous SNN.

4.1 The primal world of SNN

First, let's revisit the dynamics of the potential vector $\mathbf{u}(t)$, *i.e.,* the primal world. The primal world that $\mathbf{u}(t)$ lives in is the $\mathbb{R}^n$ space where each coordinate is corresponded to a neuron. Because of the spiking rule, $\mathbf{u}(t)$ will roughly be stuck in a n-dimensional cube in $\mathbb{R}^n$. Concretely, $-\eta\cdot\mathbf{1} \preceq \mathbf{u}(t) \preceq \eta\cdot\mathbf{1}$. Intuitively, for all $i, j \in [n]$, neuron i is holding the i-th column vector of matrix A, we call it the *characteristic vector* of neuron i. The connectivity among neuron i and neuron j is $(A^\intercal A)_{i,j} = \langle A_i, A_j\rangle$, which is the correlation among the characteristic vectors of neuron i and j. As a result, by the dynamics of SNN in (2) and (5), when neuron i fires a spike, it will suppress the neurons that are positively correlated with it and lift the neurons that are negatively correlated with it.

4.2 The dual world of SNN

With the concepts of primal world and characteristic vector at hand, let's define the dual vector $\mathbf{v}(t)$ as follows.

$$\text{(Continuous SNN)} \qquad \frac{d\mathbf{v}(t)}{dt} = -\alpha \cdot A\mathbf{s}(t) + \mathbf{b} \qquad , \ t \in [0, \infty), \tag{15}$$

$$\text{(Discrete SNN)} \qquad \mathbf{v}(t+1) = \mathbf{v}(t) - \alpha \cdot A\mathbf{s}(t) + \mathbf{b} \cdot \Delta t \qquad , \ t \in \mathbb{N} \cup \{0\}, \tag{16}$$

where $\mathbf{v}(0) = \mathbf{0}$. Here, we use the same notations for both discrete and continuous SNN. In the following, it will be easy to know which model we are using from the context.

By comparing the primal dynamics defined in (2) and (5) with the dual dynamics defined in (15) and (16), one can see that there's actually a simple transformation between the the dual vector to the primal vector of SNN.

$$\mathbf{u}(t) = A^\intercal \mathbf{v}(t). \tag{17}$$

Since the primal vector lies in the column space of A, there's a unique transformation from primal vector back to dual vector utilizing the pseudo-inverse of A.

$$\mathbf{v}(t) = (A^\intercal)^\dagger \mathbf{u}(t), \tag{18}$$

where $(A^\intercal)^\dagger$ is the pseudo-inverse of $A^\intercal$.

As we discussed previously in Section 4.1, the primal vector $\mathbf{u}(t)$ is working on the primal space $\mathbb{R}^n$ and to some degree informally approximating the primal program (19). Now, let's take a look at the dual vector $\mathbf{v}(t)$, we are going to see that it is actually *walking* in the dual world $\mathbb{R}^m$ and trying to approximate the dual program (20).

First, the spiking rule of SNN turns out to limit $\mathbf{v}(t)$ in the polytope defined by $\|A^\intercal \mathbf{v}\|_\infty \leq \eta$, which is, with proper scaling, exactly the feasible region of the dual program. Concretely, the boundaries of the polytope play the role as *walls* preventing dual vector $\mathbf{v}(t)$ from going out. We can formally define this notion as follows.

Definition 3 (spiking wall) *Define the spiking wall of* $S \subseteq [\pm n]$ *as* $W_S = \{\mathbf{v} \mid A_i^\intercal \mathbf{v} = \eta, \ \forall i \in S\}$.

Namely, when the activation set at time t is $\Gamma(t)$, the dual vector $\mathbf{v}(t)$ hits the wall $W_{\Gamma(t)}$.

Intuitively, the task of dual vector $\mathbf{v}(t)$ seems to maximize the correlation with $\mathbf{b}$ in this polytope. The input charge $\mathbf{b} \cdot \Delta t$ (or $\mathbf{b} \cdot dt$) can now be regarded as moving $\mathbf{v}(t)$ toward the direction of $\mathbf{b}$, which in fact, increase the correlation between $\mathbf{b}$ and $\mathbf{v}(t)$. Finally, the role of spike in this dual world is like pushing the dual vector back to the interior of the feasible region as long as it hits the boundary. Intuitively, the spikes help the dual vector to get a fresh start so that it can search the maximum correlation with $\mathbf{b}$ again. See Table 1 for a summary of the comparison between the primal and the dual world.

	Primal space $\mathbf{u}(t)$	Dual space $\mathbf{v}(t)$
Region	Stay in the high-dimensional cube	Stay in the high-dimensional polytope
Update	Charged the correlation with $\mathbf{b}$	Walk in the direction of $\mathbf{b}$
Spike	Fit vector $\mathbf{b}$ and affect others	Jump to the interior of polytope for a fresh start
Goal	Fire as less spikes as possible	Maximize the correlation with $\mathbf{b}$

Table 1: Comparison between the primal and the dual world.

4.3 Revisit ℓ_1 minimization problem: The dual program

Before seeing the formulation of the dual world of SNN, let's revisit the ℓ_1 minimization and its dual program. First, with some change of variables, we can rewrite the ℓ_1 minimization program defined in (7) to the following standard linear program.

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \sum_{i\in[n]} \mathbf{w}_i \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & -\mathbf{w} \preceq \mathbf{x} \preceq \mathbf{w} \\ & \mathbf{w} \succeq 0 \end{aligned} \tag{19}$$

The dual program can then be beautifully written down as follows.

$$\begin{aligned} \underset{\mathbf{v}}{\text{maximize}} \quad & \mathbf{b}^\intercal \mathbf{v} \\ \text{subject to} \quad & \|A^\intercal \mathbf{v}\|_\infty \le 1 \end{aligned} \tag{20}$$

There's a trivial continuous gradient ascent algorithm for the dual of ℓ_1 minimization problem as follow. We call it the *ideal algorithm*.

Algorithm 1 Ideal algorithm for (20)

1: $\mathbf{v}^{\text{ideal}}(0) = \mathbf{0}$
2: **for** $t \ge 0$ **do**
3: $\quad \mathbf{v}^{\text{ideal}}(t+dt) \leftarrow \Pi_{\|A^\intercal \mathbf{v}^{\text{ideal}}\|_\infty \le 1}(\mathbf{v}^{\text{ideal}}(t) + \mathbf{b}dt)$
4: **end for**

Intuitively, in this algorithm, a particle in the dual space $\mathbb{R}^m$ is assigned with a gravity in the direction of $\mathbf{b}$ so that it will walk as far as possible. When the particle hits a *wall*, *i.e.,* the boundary of the feasible polytope of (20), it will then move along the wall while in the meantime, still being dragged by the gravity. Canceled out by the normal force of the wall, the particle will end up moving in the direction of $\mathbf{b}$'s non-negative projection on that wall. Denote the position of this particle at time t to be $\mathbf{v}^{\text{ideal}}(t)$, the ideal algorithm wants to make $\mathbf{b}^\intercal \mathbf{v}^{\text{ideal}}(t)$ as large as possible. In the end, when the particle can no longer walk further in the direction of $\mathbf{b}$, the particle will stop in a corner of the polytope[7]. By the convexity of the polytope, we can argue that the position of the article will end up maximizing the dual program of ℓ_1 minimization problem. Nevertheless, one can recover the primal optimal solution with the help of dual optimal solution. Details of the proof will be provided in Section 4 and Appendix B.

It turns out that we can define a *dual vector* in the dual space $\mathbb{R}^m$ of SNN and prove that SNN is actually approximating Algorithm 1 in the dual space.

4.4 Ideal algorithm for ℓ_1 dual program

In this subsection, we are going to show that the ideal algorithm for ℓ_1 dual program will converge to the optimal solution in the dual space and produce a primal solution converges to the primal optimal.

[7]Or, a boundary that is orthogonal to $\mathbf{b}$.

4.4.1 Close form of the dynamics of ideal algorithm

Let's first write down the close form of ideal algorithm. We can actually use the conic projection to define the dynamics of ideal algorithm.

Definition 4 (conical projection) *At time t, denote the active set of ideal algorithm as $\Gamma^{ideal}(t) := \{i \in \pm[n] \mid A_i^\intercal \mathbf{v}(t) = 1\}$. Let $S \subseteq [\pm n]$, the conical projection of vector $\mathbf{b}$ on a set of vectors A_S is defined as*

$$p_{A_S^\dagger}(\mathbf{b}) := \sum_{i \in S} a_i^* A_i \tag{21}$$

where

$$(a_i^*)_{i \in S} = \arg \min_{a_i \geq 0, i \in S} \|\mathbf{b} - \sum_{i \in S} a_i A_i\|_2. \tag{22}$$

Furthermore, at time t, denote

$$\mathbf{b}^{ideal}(t) := p_{A^\dagger_{\Gamma^{ideal}(t)}}(\mathbf{b}). \tag{23}$$

As a result, we can rewrite the dynamics of ideal algorithm as

$$\frac{d\mathbf{v}(t)}{dt} = \mathbf{b} - \mathbf{b}^{\text{ideal}}(t). \tag{24}$$

The following corollary follows directly from (24).

Corollary 4.0.1 *The objective value of ideal algorithm is nondecreasing, i.e., $\mathbf{b}^\intercal d\mathbf{v}(t) = [\|\mathbf{b}\|_2^2 - \|\mathbf{b}^{ideal}(t)\|_2^2]dt$ and thus $\mathbf{b}^\intercal \mathbf{v}(t)$ is non-decreasing.*

Now we understand the basic dynamics of the ideal algorithm, there's one important and difficult thing to deal with: when the particle changes the wall. That is, when $\Gamma(t) \neq \Gamma(t + dt)$, what's the dynamics of ideal algorithm? First, the following lemma tell us how $\mathbf{b}(t)$ change in such situation.

Lemma 3 *For ideal algorithm, at any time t, $proj_{A^\dagger_{\Gamma^{ideal}(t+dt)}}(\mathbf{b}) = proj_{A^\dagger_{\Gamma^{ideal}(t) \cup \Gamma^{ideal}(t+dt)}}(\mathbf{b})$.*

Proof: See Appendix B.1 ■

Intuitively, Lemma 3 told us that the conical projection of $\mathbf{b}$ on the new wall will be the same as the conical projection of $\mathbf{b}$ on the union of the old and new wall. With this lemma, we directly have the following corollary.

Corollary 4.0.2 *In ideal algorithm, $\|\mathbf{b}^{ideal}(t)\|_2$ is non-decreasing.*

That is to say, the conic approximation for $\mathbf{b}$ will become better and better.

4.4.2 Convergence of ideal algorithm

Before we introduce the convergence theorem of the ideal algorithm, let's first define the primal output from ideal algorithm at time t.

Definition 5 (primal output) *For ideal algorithm at time t, the primal output $\hat{\mathbf{x}}(t)$ is defined as follows.*

$$\hat{\mathbf{x}}_i(t) = \begin{cases} a_i^*(t) & , \text{ if } i \in \Gamma(t) \\ 0 & , \text{ else,} \end{cases}$$

where $(a_i^*(t))_{i\in\Gamma(t)}$ is the conical coefficients of $\mathbf{b}^{ideal}(t)$ over $\Gamma^{ideal}(t)$ in Definition 4.

The following theorem shows that the primal output of the ideal algorithm will converge to the optimal solution of ℓ_1 minimization problem.

Theorem 4.1 (convergence of ideal algorithm) *For any $\epsilon > 0$ and $\delta > 0$, take $T \geq \max\{\frac{\mathbf{OPT}}{\epsilon\cdot\|\mathbf{b}\|_2}, \frac{n}{\delta\cdot\sqrt{\lambda_{\min}}}\}$, then*

(1) $\|A\hat{\mathbf{x}}(T) - \mathbf{b}\| \leq \epsilon \cdot \|\mathbf{b}\|_2$

(2) $|\|\hat{\mathbf{x}}(T)\|_1 - \mathbf{OPT}| \leq \delta \cdot \mathbf{OPT}$

Proof: First, the following lemma gives us upper and lower bounds for the performance of $\hat{\mathbf{x}}(T)$.

Lemma 4 *For ideal algorithm and any time T,*

$$|\|\hat{\mathbf{x}}(T)\|_1 - \mathbf{OPT}| \leq \frac{n}{\sqrt{\lambda_{\min}}} \cdot \|\mathbf{b} - \mathbf{b}(T)\|_2 \tag{25}$$

Proof: See Appendix B.2 ■

Assume the statement is wrong, *i.e.,* $\|\mathbf{b} - \mathbf{b}(T)\|_2 > \frac{\sqrt{\lambda_{\min}}\cdot\delta\cdot\mathbf{OPT}}{n}$. Then by Corollary 4.0.1 and Corollary 4.0.2, for any $0 \leq t \leq T$, $\|\mathbf{b} - \mathbf{b}(t)\|_2 > \frac{\sqrt{\lambda_{\min}}\cdot\delta\cdot\mathbf{OPT}}{n}$. Since $T \geq \frac{n}{\sqrt{\lambda_{\min}}\cdot\delta}$, by the dynamics of ideal algorithm and the linearity of inner product,

$$\mathbf{b}^\intercal\mathbf{v}(T) = \int_0^T \mathbf{b}^\intercal d\mathbf{v}(t) > T \cdot \frac{\sqrt{\lambda_{\min}} \cdot \delta \cdot \mathbf{OPT}}{n} \geq \mathbf{OPT}, \tag{26}$$

which is a contradiction. As a result, by Lemma 4, $|\|\hat{\mathbf{x}}(T)\|_1 - \mathbf{OPT}| \leq \delta \cdot \mathbf{OPT}$.

Next, with the same argument, if $\|\mathbf{b} - \mathbf{b}(T)\|_2 > \epsilon \cdot \|\mathbf{b}\|_2$, it will lead to a contradiction. Thus, $\|A\hat{\mathbf{x}}(T) - \mathbf{b}\| \leq \epsilon \cdot \|\mathbf{b}\|_2$. Recall that $\mathbf{b}(T) = A\hat{\mathbf{x}}(T)$. ■

4.4.3 d-ideal algorithm

Let $0 < d < \eta$, define the d-ideal algorithm as follows. First, define the d-ideal active set

$$\Gamma^{ideal(d)}(t) := \{i \in \pm[n] | A_i^\intercal \mathbf{v}^{ideal(d)}(t) \geq \eta - d\}, \tag{27}$$

and the d-ideal conic projection

$$\mathbf{b}^{ideal(d)}(t) := p_{A^\dagger_{\Gamma_d^{ideal(d)}(t)}}(\mathbf{b}), \tag{28}$$

and the d-ideal dynamics

$$\frac{d\mathbf{v}^{ideal(d)}(t)}{dt} = \mathbf{b} - \mathbf{b}^{ideal(d)}(t). \tag{29}$$

Similarly, we have the following lemma to show the monotonicity of $\mathbf{v}^{ideal(d)}(t)$.

Lemma 5 *For d-relaxed ideal algorithm, at any time t, $proj_{A^{\dagger}_{\Gamma^{ideal(d)}(t+dt)}}(\mathbf{b}) = proj_{A^{\dagger}_{\Gamma^{ideal(d)}(t)\cup\Gamma^{ideal(d)}(t+dt)}}(\mathbf{b})$.*

As long as the distance between optimal corner and any its neighbor is smaller than d, then the convergence theorem in Theorem 4.1 will hold since $\Gamma^{ideal(d)}(t) = \Gamma(t)$.

Intuitively, this is like replacing the particle in the ideal algorithm to a n dimensional ball with radius d or replacing the dual polytope from radius η to radius $\eta - d$.

5 Continuous SNN in the dual world

In Section 4.2, we defined the dual vectors of both discrete SNN and continuous SNN and their relations with the primal vectors. In Section 4.4, we defined ideal algorithm for both the primal and dual program of ℓ_1 minimization problem and the convergence. Specifically, we relaxed the concept of ideal algorithm in Section 4.4.3 and showed that the convergence also holds. Now, it's finally the time to prove the convergence of continuous SNN with the help of d-ideal algorithm.

The high level idea of the connection of d-ideal algorithm and continuous SNN is as follows.

- For any time $t \geq 0$, $\mathbf{v}^{CSNN}(t)$ and $\mathbf{v}^{ideal(d)}(t)$ are close under certain distance measure. As a result, by the convergence of d-ideal algorithm, when t large enough, $\mathbf{v}^{CSNN}(t)$ will stay in a neighborhood around the optimal point of dual program.
- With some perturbed primal-dual argument, we can prove that when $\mathbf{v}^{CSNN}(t)$ stays close enough to the optimal point of dual program, $\hat{\mathbf{x}}^{CSNN}(t)$ will converge to the optimal solution of the primal program and it will also become more and more feasible.

Note that the parameter d, α, and η here haven't been fixed yet. In the following discussion, we will see how to properly choose them so that the convergence can work.

References

[ABGM14] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *ICML*, pages 584–592, 2014.

[Adr26] Edgar D Adrian. The impulses produced by sensory nerve endings. *The Journal of physiology*, 61(1):49–72, 1926.

[AGMM15] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. *arXiv preprint arXiv:1503.00778*, 2015.

[BD11] Martin Boerlin and Sophie Denève. Spike-based population coding and working memory. *PLoS Comput Biol*, 7(2):e1001080, 2011.

[BDM13] David G Barrett, Sophie Denève, and Christian K Machens. Firing rate predictions in optimal balanced networks. In *Advances in Neural Information Processing Systems*, pages 1538–1546, 2013.

[BGS97] José L Balcázar, Ricard Gavalda, and Hava T Siegelmann. Computational power of neural networks: a characterization in terms of kolmogorov complexity. *IEEE Transactions on Information Theory*, 43(4):1175–1183, 1997.

[BRR12] Aurèle Balavoine, Justin Romberg, and Christopher J Rozell. Convergence and rate analysis of neural networks for sparse approximation. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(9):1377–1389, 2012.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CDS01] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[CKP+14] Michael B Cohen, Rasmus Kyng, Jakub W Pachocki, Richard Peng, and Anup Rao. Preconditioning in expectation. *arXiv preprint arXiv:1401.6236*, 2014.

[CMP+14] Michael B Cohen, Gary L Miller, Jakub W Pachocki, Richard Peng, and Shen Chen Xu. Stretching stretch. *arXiv preprint arXiv:1401.2454*, 2014.

[DT08] David L Donoho and Yaakov Tsaig. Fast solution of-norm minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54(11):4789–4812, 2008.

[EHJ+04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[Fu98] Wenjiang J Fu. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3):397–416, 1998.

[Fuc01] J-J Fuchs. On the application of the global matched filter to doa estimation with uniform circular arrays. *IEEE Transactions on Signal Processing*, 49(4):702–709, 2001.

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[GO88] Gene H Golub and Michael L Overton. The convergence of inexact chebyshev and richardson iterative methods for solving linear systems. *Numerische Mathematik*, 53(5):571–593, 1988.

[GV61] Gene H Golub and Richard S Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):157–168, 1961.

[HH52] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[I+03] Eugene M Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[KF00] Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *Annals of statistics*, pages 1356–1378, 2000.

[KMP10] Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 235–244. IEEE, 2010.

[KOSZ13] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 911–920. ACM, 2013.

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians: Fast, sparse, and simple. *arXiv preprint arXiv:1605.02353*, 2016.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Lap07] Louis Lapicque. Recherches quantitatives sur lexcitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen*, 9(1):620–635, 1907.

[Leo80] Steven J Leon. *Linear algebra with applications*. Macmillan New York, 1980.

[Maa96] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural computation*, 8(1):1–40, 1996.

[Maa97] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[Maa99] Wolfgang Maass. 2 computing with spiking neurons. 1999.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[PMB12] Hélene Paugam-Moisy and Sander Bohte. Computing with spiking neuron networks. In *Handbook of natural computing*, pages 335–376. Springer, 2012.

[RJBO08] Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

[RMG+88] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.

[SD14] Cristina Savin and Sophie Deneve. Spatio-temporal representations of uncertainty in spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 2024–2032, 2014.

[Sie03] Hava T Siegelmann. Neural and super-turing computing. *Minds and Machines*, 13(1):103–114, 2003.

[Sie12] Hava Siegelmann. *Neural networks and analog computation: beyond the Turing limit*. Springer Science & Business Media, 2012.

[SRH13] Samuel Shapero, Christopher Rozell, and Paul Hasler. Configurable hardware integrate and fire neurons for sparse approximation. *Neural Networks*, 45:134–143, 2013.

[SS91] Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

[Ste65] Richard B Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173, 1965.

[Sti81] Stephen M Stigler. Gauss and the invention of least squares. *The Annals of Statistics*, pages 465–474, 1981.

[SZHR14] Samuel Shapero, Mengchen Zhu, Jennifer Hasler, and Christopher Rozell. Optimal sparse approximation with integrate and fire neurons. *International journal of neural systems*, 24(05):1440001, 2014.

[T+97] Robert Tibshirani et al. The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395, 1997.

[Tan16] Ping Tak Peter Tang. Convergence of lca flows to (c) lasso solutions. *arXiv preprint arXiv:1603.01644*, 2016.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[TSR+05] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[TUKM15] Dominik Thalmeier, Marvin Uhlmann, Hilbert J Kappen, and Raoul-Martin Memmesheimer. Learning universal computations with spikes. *arXiv preprint arXiv:1505.07866*, 2015.

[Vis12] Nisheeth K Vishnoi. Laplacian solvers and their algorithmic applications. *Theoretical Computer Science*, 8(1-2):1–141, 2012.

[YL06] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[YSGM10] Allen Y Yang, S Shankar Sastry, Arvind Ganesh, and Yi Ma. Fast 1-minimization algorithms and an application in robust face recognition: A review. In *2010 IEEE International Conference on Image Processing*, pages 1849–1852. IEEE, 2010.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[Zou06] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

A Lemmas for Theorem 3.1

A.1 Proof of Lemma 1

Consider the spectrum decomposition of M such that $M = U\Sigma U^\intercal$, where U is an orthogonal matrix and Σ is a diagonal matrix containing the eigenvalues of M. The pseudoinverse of M is $M^\dagger = U\Sigma^\dagger U^\intercal$, where $\Sigma^\dagger$ is defined as follow: for any $i \in [n]$

$$(\Sigma^\dagger)_{ii} = \begin{cases} 1/\Sigma_{ii} & \text{, if } \Sigma_{ii} \neq 0 \\ 0 & \text{, if } \Sigma_{ii} = 0. \end{cases} \tag{30}$$

Thus, for any $\mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{x}\|_M = \sqrt{\mathbf{x}^\intercal M \mathbf{x}} = \sqrt{(U^\intercal \mathbf{x})^\intercal \Sigma (U^\intercal \mathbf{x})} \tag{31}$$

$$(\because \Sigma = \Sigma\Sigma^\dagger\Sigma) = \sqrt{(U^\intercal \mathbf{x})^\intercal \Sigma\Sigma^\dagger\Sigma (U^\intercal \mathbf{x})} \tag{32}$$

$$(\because UU^\intercal = U^\intercal U = I) = \sqrt{(U^\intercal \mathbf{x})^\intercal \Sigma U^\intercal U \Sigma^\dagger U^\intercal U \Sigma (U^\intercal \mathbf{x})} = \sqrt{(M\mathbf{x})^\intercal M^\dagger (M\mathbf{x})} \tag{33}$$

$$= \|M\mathbf{x}\|_{M^\dagger}. \tag{34}$$

For the second part of Lemma 1, we have

$$M^\dagger M = U\Sigma^\dagger U^\intercal U \Sigma U^\intercal = U\Sigma^\dagger \Sigma U^\intercal = UDU^\intercal, \tag{35}$$

where D is the diagonal matrix such that D_{ii} is 1 iff the corresponding eigenvector is in the range space of M.

A.2 Proof of Lemma 2

At the t-th iteration, denote the set of neurons that are exceeding threshold as $\Gamma(t) := \{i \mid |\mathbf{u}_i(t)| > \lambda\}$. We have the following observation.

Lemma 6 *For any $t \geq 0$, $\mathbf{u}(t)^\intercal \mathbf{s}(t) \geq \eta \cdot |\Gamma(t)|$.*

Proof: Observe that

$$\mathbf{u}(t)^\intercal \mathbf{s}(t) = \sum_{i \in [n]} \mathbf{u}_i(t) \cdot \mathbf{s}(t) \tag{36}$$

$$= \sum_{i\in\Gamma(t)} \mathbf{u}_i(t)\cdot \mathrm{sign}(\mathbf{u}_i(t)) = \sum_{i\in\Gamma(t)} |\mathbf{u}_i(t)| \tag{37}$$

$$\geq \sum_{i\in\Gamma(t)} \eta = \eta\cdot|\Gamma(t)|. \tag{38}$$

■

Lemma 7 *For any $t>0$, $\mathbf{u}(t)^\intercal (A^\intercal A)^\dagger A^\intercal A\mathbf{s}(t) = \mathbf{u}(t)^\intercal \mathbf{s}(t)$.*

Proof: First, note that $(A^\intercal A)^\dagger A^\intercal A$ is the orthogonal projection matrix of the range space of $A^\intercal A$. Next, by the dynamics of SNN and $\mathbf{u}(0)=\mathbf{0}$, we know that $\mathbf{u}(t)$ is in the range space of $A^\intercal A$. As a result, $\mathbf{u}(t)^\intercal (A^\intercal A)^\dagger A^\intercal A\mathbf{s}(t) = \Big((A^\intercal A)^\dagger A^\intercal A\mathbf{u}(t)\Big)^\intercal \mathbf{s}(t) = \mathbf{u}(t)^\intercal\mathbf{s}(t)$. ■

We now prove Lemma 2 using Lemma 6.

Proof of Lemma 2: The proof is based on an induction over time $t\geq 0$. Let $B = 2\sqrt{\kappa(A^\intercal A)\cdot\lambda_{\max}\cdot n}$, our goal is to show that $\|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger}\leq B$ for any t when $\Delta t \leq \frac{\sqrt{\lambda_{\min}}}{24\cdot\sqrt{n}\cdot\|\mathbf{x}^*\|_{A^\intercal A}}$. Clearly that $\|\mathbf{u}(0)\|_{(A^\intercal A)^\dagger}\leq B$, assume $\|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger}\leq B$, consider

$$\|\mathbf{u}(t+1)\|^2_{(A^\intercal A)^\dagger} = \|\mathbf{u}(t) - A^\intercal A\mathbf{s}(t) + A^\intercal\mathbf{b}\cdot\Delta t\|^2_{(A^\intercal A)^\dagger} \tag{39}$$

$$= \|\mathbf{u}(t)\|^2_{(A^\intercal A)^\dagger} + \|A^\intercal A\mathbf{s}(t) - A^\intercal A\mathbf{x}^*\cdot\Delta t\|^2_{(A^\intercal A)^\dagger}$$

$$- 2\mathbf{u}(t)^\intercal (A^\intercal A)^\dagger\Big(A^\intercal A\mathbf{s}(t) - A^\intercal A\mathbf{x}^*\cdot\Delta t\Big). \tag{40}$$

Expand the $\|A^\intercal A\mathbf{s}(t) - A^\intercal A\mathbf{x}^*\cdot\Delta t\|^2_{(A^\intercal A)^\dagger}$ term, by Lemma 1, we have

$$\|A^\intercal A\mathbf{s}(t) - A^\intercal A\mathbf{x}^*\cdot\Delta t\|^2_{(A^\intercal A)^\dagger} = \|\mathbf{s}(t) - \mathbf{x}^*\cdot\Delta t\|^2_{A^\intercal A} \tag{41}$$

$$\leq \|\mathbf{s}(t)\|^2_{A^\intercal A} + \|\mathbf{x}^*\cdot\Delta t\|^2_{A^\intercal A} - 2\mathbf{s}(t)^\intercal A^\intercal A\mathbf{x}^*\cdot\Delta t \tag{42}$$

$$\leq \lambda_{\max}\cdot|\Gamma(t)| + \|\mathbf{x}^*\|^2_{A^\intercal A}\cdot\Delta t^2 + 2\sqrt{\lambda_{\max}\cdot n}\cdot\|\mathbf{x}^*\|_{A^\intercal A}\cdot\Delta t. \tag{43}$$

The last inequality is from the observations that $\|\mathbf{s}(t)\|^2 = |\Gamma(t)|$ and $\|\mathbf{s}(t)\|^2\leq n$. Next, by Lemma 6, Lemma 7, Cauchy-Schwartz inequality, and the induction hypothesis, we have

$$-2\mathbf{u}(t)^\intercal (A^\intercal A)^\dagger\Big(A^\intercal A\mathbf{s}(t) - A^\intercal A\mathbf{x}^*\cdot\Delta t\Big) = -2\mathbf{u}(t)^\intercal\mathbf{s}(t) + 2\|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger}\cdot\|\mathbf{x}^*\|_{A^\intercal A}\cdot\Delta t. \tag{44}$$

$$\leq -2\eta\cdot|\Gamma(t)| + 2B\cdot\|\mathbf{x}^*\|_{A^\intercal A}\cdot\Delta t. \tag{45}$$

Combine (40), (43), (44), and induction hypothesis, we have

$$\|\mathbf{u}(t+1)\|^2_{(A^\intercal A)^\dagger} = \|\mathbf{u}(t)\|^2_{(A^\intercal A)^\dagger} + (\lambda_{\max} - 2\eta)\cdot|\Gamma(t)|$$

$$+ (2\sqrt{\lambda_{\max}\cdot n}\cdot\|\mathbf{x}^*\|_{A^\intercal A} + 2B\cdot\|\mathbf{x}^*\|_{A^\intercal A})\cdot\Delta t + \|\mathbf{x}^*\|^2_{A^\intercal A}\cdot\Delta t^2. \tag{46}$$

By the choice of $\eta, \Delta t$, and B, we have

$$\|\mathbf{u}(t+1)\|^2_{(A^\intercal A)^\dagger} = \|\mathbf{u}(t)\|^2_{(A^\intercal A)^\dagger} - \lambda_{\max}\cdot|\Gamma(t)| + \frac{\sqrt{\lambda_{\max}\cdot\lambda_{\min}}}{6} + \frac{\lambda_{\max}}{6} + \frac{\lambda_{\min}}{144n} \tag{47}$$

$$\leq \|\mathbf{u}(t)\|^2_{A^\intercal A^\dagger} - \lambda_{\max} \cdot |\Gamma(t)| + \frac{\lambda_{\max}}{2}. \tag{48}$$

Now, from (48) we can see that if $\Gamma(t)$ is nonempty, then by the induction hypothesis, $\|\mathbf{u}(t+1)\|_{(A^\intercal A)^\dagger} \leq \|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger} \leq B$. If $\Gamma(t)$ is empty, then for any $i \in [n]$, $|\mathbf{u}_i(t)| \leq \eta$. Thus, we have

$$\|\mathbf{u}(t+1)\|_{(A^\intercal A)^\dagger} \leq \|\mathbf{u}(t)\|_{(A^\intercal A)^\dagger} + \frac{\lambda_{\max}}{2} \tag{49}$$

$$\leq \frac{\|\mathbf{u}(t)\|_2}{\sqrt{\lambda_{\min}}} + \frac{B}{2} \leq \frac{\eta\sqrt{n}}{\sqrt{\lambda_{\min}}} + \frac{B}{2} \tag{50}$$

$$= \sqrt{\kappa(A^\intercal A) \cdot \lambda_{\max} \cdot n} + \frac{B}{2} \leq B. \tag{51}$$

As a result, the induction holds. ■

B Proof of ideal algorithm

B.1 Proof of Lemma 3

Proof: Without loss of generality, we can choose dt small enough such that only the following two situations can happen.

- $\Gamma(t) \subseteq \Gamma(t+dt)$
- $\Gamma(t+dt) \subsetneq \Gamma(t)$

When $\Gamma(t) \subseteq \Gamma(t+dt)$, the statement is clearly correct. Consider the case where $\Gamma(t+dt) \subsetneq \Gamma(t)$, let $i \in \Gamma(t) \backslash \Gamma(t+dt)$, we have $A_i^\intercal[\mathbf{v}(t+dt) - \mathbf{v}(t)] < 0$ since $A_i^\intercal \mathbf{v}(t) = \eta$ but $A_i^\intercal \mathbf{v}(t+dt) < \eta$. By the definition of ideal algorithm, $\mathbf{v}(t+dt) - \mathbf{v}(t) = \mathbf{b} - \mathbf{b}(t)$, by the convexity of $\|\mathbf{b} - \sum_{j\in\Gamma(t)} a_j A_j\|_2^2$, we can see that A_i provides no contribution to the projection of $\mathbf{b}$ on $\Gamma(t) \cup \Gamma(t+dt)$. As a result, $proj_{A_{\Gamma(t+dt)}}(\mathbf{b}) = proj_{A_{\Gamma(t)\cup A_{\Gamma(t+dt)}}}(\mathbf{b})$. ■

B.2 Proof of Lemma 4

Define the following perturbed program of (19).

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & A\mathbf{x} - \mathbf{b}(T) = 0 \end{aligned} \tag{52}$$

Note that $\mathbf{b}(T)$ are treated as given constant to the program. It turns out that the ideal algorithm optimizes these primal-dual perturbed program with the following parameters.

Lemma 8 *Let* $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*) = (\hat{\mathbf{x}}(T), |\hat{\mathbf{x}}(T)|, \mathbf{v}(T))$*, then* $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*)$ *is the optimal solutions of (52).*

Proof: We simply check the KKT condition. First, the primal and dual feasibility can be verified by the dynamics of ideal algorithm. That is, $A\hat{\mathbf{x}}(T) = \mathbf{b}(T)$ and the inequalities clearly holds. Next, consider the Lagrangian of (55)

$$\mathcal{L}(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \|\mathbf{x}\|_1 - \mathbf{v}^\intercal(A\mathbf{x} - \mathbf{b}(T))$$

$$\partial_{\mathbf{x}}\mathcal{L}(\mathbf{x},\mathbf{w},\mathbf{v}) = -A^{\intercal}\mathbf{v}$$
$$\partial_{\mathbf{w}}\mathcal{L}(\mathbf{x},\mathbf{w},\mathbf{v}) = \partial\|\mathbf{x}\|_1$$

Now, let's verify that both partial derivative will vanish on $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*)$.

$$\partial_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\mathbf{w}^*,\mathbf{v}^*,\boldsymbol{\lambda}_+^*,\boldsymbol{\lambda}_-^*) = -A^{\intercal}\mathbf{v}(T) + (A^{\intercal}\mathbf{v}(T))_+ - (A^{\intercal}\mathbf{v}(T))_- = \mathbf{0},$$
$$\partial_{\mathbf{w}}\mathcal{L}(\mathbf{x}^*,\mathbf{w}^*,\mathbf{v}^*,\boldsymbol{\lambda}_+^*,\boldsymbol{\lambda}_-^*) = \partial\|\hat{\mathbf{x}}(T)\|_1 - (A^{\intercal}\mathbf{v}(T))_+ - (A^{\intercal}\mathbf{v}(T))_- \ni \mathbf{0}.$$

As a result, we conclude that $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*)$ is the optimal solution of (52). ■

Now, by the perturbation theorem in Chapter 5.6 of [BV04], we have

$$\mathbf{OPT} \geq \|\hat{\mathbf{x}}(T)\|_1 + \mathbf{v}(T)^{\intercal}[\mathbf{b} - \mathbf{b}(T)].$$

Thus, we have the upper bound

$$\|\hat{\mathbf{x}}(T)\|_1 \leq \mathbf{OPT} + \|\mathbf{v}(T)\|_2 \cdot \|\mathbf{b} - \mathbf{b}(T)\|_2. \tag{53}$$

To obtain the lower bound, use the other side of the perturbation as follows.

$$\|\hat{\mathbf{x}}(T)\|_1 \geq \mathbf{OPT} + (\mathbf{v}^*)^{\intercal}[\mathbf{b}(T) - \mathbf{b}] = (\mathbf{v}^*)^{\intercal}\mathbf{b}(T).$$

Observe that,

$$\begin{aligned}\mathbf{v}(T)^{\intercal}\mathbf{b}(T) &= \mathbf{v}(T)^{\intercal}A_{\Gamma(T)}\hat{\mathbf{x}}_{\Gamma(T)}(T) \\ &= \mathrm{sign}(\hat{\mathbf{x}}_{\Gamma}(T))^{\intercal}\hat{\mathbf{x}}_{\Gamma(T)}(T) = \|\hat{\mathbf{x}}(T)\|_1,\end{aligned}$$

and

$$\begin{aligned}(\mathbf{v}^*)^{\intercal}\mathbf{b}(T) &= (\mathbf{v}^*)^{\intercal}\mathbf{b} + (\mathbf{v}^*)^{\intercal}[\mathbf{b}(T) - \mathbf{b}] \\ &\geq \mathbf{OPT} - \|\mathbf{v}^*\|_2 \cdot \|\mathbf{b}(T) - \mathbf{b}\|_2.\end{aligned}$$

Finally, as both $\mathbf{v}^*$ and $\mathbf{v}(T)$ are in the feasible region $\{\mathbf{v} | \|A^{\intercal}\mathbf{v}\|_\infty \leq 1\}$, we can upper bound their 2-norm as follows.

Lemma 9 *For any* $\mathbf{v}$ *such that* $\|A^{\intercal}\mathbf{v}\|_\infty \leq 1$, $\|\mathbf{v}\|_2 \leq \frac{n}{\sqrt{\lambda_{\min}}}$.

Proof: Observe that

$$\begin{aligned}\min_{\mathbf{v}:\|\mathbf{v}\|_2=1} \|A^{\intercal}\mathbf{v}\|_\infty &\geq \min_{\mathbf{v}:\|\mathbf{v}\|_2=1} \frac{1}{n}\|A^{\intercal}\mathbf{v}\|_1 \\ &\geq \min_{\mathbf{v}:\|\mathbf{v}\|_2=1} \frac{1}{n}\|A^{\intercal}\mathbf{v}\|_2 \\ &\geq \frac{\sqrt{\lambda_{\min}}}{n}\end{aligned}$$

■

Thus, we get the desired bounds.

B.3 Proof of perturbation theorem (unused yet)

Define the following perturbed program of (19).

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \sum_{i\in[n]} |\left(A^\intercal \mathbf{v}(T)\right)_i| \cdot \mathbf{w}_i \\ \text{subject to} \quad & A\mathbf{x} - \mathbf{b} = \frac{-\mathbf{v}(T) + \mathbf{v}(T_0)}{T - T_0} \\ & \mathbf{x}_i - \mathbf{w}_i \leq -|\frac{\hat{\mathbf{x}}_i(T)}{\left(A^\intercal \mathbf{v}(T)\right)_i}| + |\hat{\mathbf{x}}_i(T)| \\ & -\mathbf{x}_i - \mathbf{w}_i \leq -|\frac{\hat{\mathbf{x}}_i(T)}{\left(A^\intercal \mathbf{v}(T)\right)_i}| + |\hat{\mathbf{x}}_i(T)| \end{aligned} \tag{54}$$

Note that $\mathbf{v}(T)$, $\mathbf{v}(T_0)$, and $\hat{\mathbf{x}}(T)$ are treated as given constant to the program. And the dual program is

$$\begin{aligned} \underset{\mathbf{v},\boldsymbol{\lambda}^\dagger,\boldsymbol{\lambda}^-}{\text{maximize}} \quad & (\mathbf{b} - \frac{\mathbf{v}(T)}{T\cdot\Delta t})^\intercal \mathbf{v} + (\boldsymbol{\lambda}_+ + \boldsymbol{\lambda}_-)^\intercal (-|\frac{\hat{\mathbf{x}}(T)}{A^\intercal \mathbf{v}(T)}| + |\hat{\mathbf{x}}(T)|) \\ \text{subject to} \quad & -|A^\intercal \mathbf{v}(T)| \preceq A^\intercal \mathbf{v} \preceq |A^\intercal \mathbf{v}(T)| \end{aligned} \tag{55}$$

It turns out that SNN optimizes these primal-dual perturbed program with the following parameters.

Lemma 10 *Let* $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*) = (\hat{\mathbf{x}}(T), |\frac{\hat{\mathbf{x}}(T)}{A^\intercal \mathbf{v}(T)}|, \mathbf{v}(T), \left(A^\intercal \mathbf{v}(T)\right)_+, \left(A^\intercal \mathbf{v}(T)\right)_-)$*, then* $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*)$ *is the optimal solutions of (54) and (55).*

Proof: We simply check the KKT condition. First, the primal and dual feasibility can be verified by the dynamics of SNN. That is. $\mathbf{v}(T) - \mathbf{v}(T_0) = (T - T_0)\cdot[-A\hat{\mathbf{x}}(T) + \mathbf{b}]$ and the inequalities clearly holds. Next, consider the Lagrangian of (55)

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{w}, \mathbf{v}, \boldsymbol{\lambda}^\dagger, \boldsymbol{\lambda}^-) = & \sum_{i\in[n]} |\left(A^\intercal \mathbf{v}(T)\right)_i| \cdot \mathbf{w}_i - \mathbf{v}^\intercal (A\mathbf{x} - \mathbf{b} + \frac{\mathbf{v}(T)}{T\cdot\Delta t}) \\ & + (\boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}^-)^\intercal (\mathbf{x} + |\frac{\hat{\mathbf{x}}(T)}{A^\intercal \mathbf{v}(T)}| - |\hat{\mathbf{x}}(T)|) - (\boldsymbol{\lambda}^\dagger + \boldsymbol{\lambda}^-)^\intercal \mathbf{w} \\ \partial_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{w}, \mathbf{v}, \boldsymbol{\lambda}^\dagger, \boldsymbol{\lambda}^-) = & -A^\intercal \mathbf{v} + \boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}^- \\ \partial_{\mathbf{w}} \mathcal{L}(\mathbf{x}, \mathbf{w}, \mathbf{v}, \boldsymbol{\lambda}^\dagger, \boldsymbol{\lambda}^-) = & |A^\intercal \mathbf{v}| - \boldsymbol{\lambda}^\dagger - \boldsymbol{\lambda}^- \end{aligned}$$

Now, let's verify that both partial derivative will vanish on $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*)$.

$$\begin{aligned} \partial_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*) = & -A^\intercal \mathbf{v}(T) + \left(A^\intercal \mathbf{v}(T)\right)_+ - \left(A^\intercal \mathbf{v}(T)\right)_- = \mathbf{0} \\ \partial_{\mathbf{w}} \mathcal{L}(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*) = & |A^\intercal \mathbf{v}(T)| - \left(A^\intercal \mathbf{v}(T)\right)_+ - \left(A^\intercal \mathbf{v}(T)\right)_- = \mathbf{0} \end{aligned}$$

Finally, let's verify the complementary slackness. For any $i \in [n]$, if $\hat{\mathbf{x}}(T) > 0$, then the first inequality holds and the second one does not hold. Nevertheless, as $(A^\intercal \mathbf{v}(t))_i$ would not change sign, we know that $\left(A^\intercal \mathbf{v}(T)\right)_i > 0$. Thus, $\left(\boldsymbol{\lambda}_-^*\right)_i = 0$. Similar argument can be applied to the case where $\hat{\mathbf{x}}(T) < 0$ and when $\hat{\mathbf{x}}(T) = 0$, both inequalities hold. As a result, the complementary slackness is proven. We conclude that $(\mathbf{x}^*, \mathbf{w}^*, \mathbf{v}^*, \boldsymbol{\lambda}_+^*, \boldsymbol{\lambda}_-^*)$ is the optimal solution of (54) and (55). ■

Now, consider a middle program as follows.

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \sum_{i\in[n]} |\left(A^\intercal \mathbf{v}(T)\right)_i| \cdot \mathbf{w}_i \\ \text{subject to} \quad & A\mathbf{x} - \mathbf{b} = 0 \\ & \mathbf{x}_i - \mathbf{w}_i \leq 0 \\ & -\mathbf{x}_i - \mathbf{w}_i \leq 0 \end{aligned} \tag{56}$$

By the perturbation theorem in Chapter 5-6 of [BV04], let $\mathbf{OPT}^{\text{middle}}$ be the optimal value of (56), we have

$$\begin{aligned} \mathbf{OPT}^{\text{middle}} &\geq \sum_{i\in[n]} |\left(A^\intercal \mathbf{v}(T)\right)_i| \cdot |\hat{\mathbf{x}}_i(T)| + \frac{\mathbf{v}(T)^\intercal[\mathbf{v}(T) - \mathbf{v}(T_0)]}{T - T_0} \\ &\quad + [\left(A^\intercal \mathbf{v}(T)\right)_+ - \left(A^\intercal \mathbf{v}(T)\right)_-]^\intercal [|\frac{\hat{\mathbf{x}}(T)}{A^\intercal \mathbf{v}(T)}| - |\hat{\mathbf{x}}(T)|] \\ &= \sum_{i\in[n]} |\hat{\mathbf{x}}_i(T)| + \frac{\mathbf{v}(T)^\intercal[\mathbf{v}(T) - \mathbf{v}(T_0)]}{T - T_0} \\ &\geq \|\hat{\mathbf{x}}(T)\|_1 - \frac{\sqrt{\epsilon} \cdot \mathbf{OPT}}{T - T_0} \end{aligned}$$

As for all $i \in [n]$, $|\left(A^\intercal \mathbf{v}(T)\right)_i| \leq 1$, we have $\mathbf{OPT} \geq \mathbf{OPT}^{\text{middle}}$. Thus,

$$\|\hat{\mathbf{x}}(T)\|_1 \leq (1 + \frac{\sqrt{\epsilon}}{T - T_0}) \cdot \mathbf{OPT} \tag{57}$$

C Other SNN models

The simple SNN dynamics defined above can be viewed as a simplified variant of existing SNN model in literatures [BDM13, SZHR14, SRH13, TUKM15].