On the Algorithmic Power of Spiking Neural Networks

Chi-Ning Chou
Harvard University

Kai-Min Chung
Academia Sinica

Chi-Jen Lu
Academia Sinica

ITCS 2019
What is Spiking Neural Networks (SNNs)?
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.
What is Spiking Neural Networks (SNNs)?

- Mathematical models for “biological neural networks”.

* By Pennstatenews https://www.flickr.com/photos/pennstatelive/37247502805
What is Spiking Neural Networks (SNNs)?

- Mathematical models for “biological neural networks”.

* Neurons: Nerve cells

* By Pennstatenews https://www.flickr.com/photos/pennstatelive/37247502805
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.

Neurons: Nerve cells

Synapses: Connections between neurons

* By Pennstatenews https://www.flickr.com/photos/pennstatelive/37247502805
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.

Neurons: Nerve cells

Synapses: Connections between neurons

Spikes: Instantaneous signals

* By Pennstatenews https://www.flickr.com/photos/pennstatelive/37247502805
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.

Chi-Ning Chou (Harvard University)
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.
• Various models since the 1900s.
  • Integrate-and-fire [Lap07], Hodgkin-Huxley [HH52], their variants [Fit61, Ste65, ML81, HR84, Ger95, KGH97, BL03, FTHVVB03, I+03, TMS14].
What is Spiking Neural Networks (SNNs)?

- Mathematical models for “biological neural networks”.
- Various models since the 1900s.
  - Integrate-and-fire \cite{Lap07}, Hodgkin-Huxley \cite{HH52}, their variants \cite{Fit61, Ste65, ML81, HR84, Ger95, KGH97, BL03, FTHVVB03, I+03, TMS14}.
- Study the behaviors/statistics of SNNs, e.g., firing rate.
What is Spiking Neural Networks (SNNs)?

• Mathematical models for “biological neural networks”.
• Various models since the 1900s.
  • Integrate-and-fire [Lap07], Hodgkin-Huxley [HH52], their variants [Fit61, Ste65, ML81, HR84, Ger95, KGH97, BL03, FTHVVB03, I+03, TMS14].
• Study the behaviors/statistics of SNNs, e.g., firing rate.
• [Barret-Denève-Machens 2013] empirically showed a connection between the firing rate of integrate-and-fire SNNs and an optimization problem.
What is Spiking Neural Networks (SNNs)?

- Mathematical models for “biological neural networks”.
- Various models since the 1900s.
  - Integrate-and-fire [Lap07], Hodgkin-Huxley [HH52], their variants [Fit61, Ste65, ML81, HR84, Ger95, KGH97, BL03, FTHVVB03, I+03, TMS14].
- Study the behaviors/statistics of SNNs, e.g., firing rate.
- [Barret-Denève-Machens 2013] empirically showed a connection between the firing rate of integrate-and-fire SNNs and an optimization problem.

SNNs seem to have non-trivial computational power. Can we understand them better through the lens of algorithms?
Integrate-and-Fire (IAF) Model [Lapicque 1907]
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: $[n] = \{1, 2, \ldots, n\}$
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: $[n] = \{1, 2, \ldots, n\}$
- Potential: $\mathbf{u}(t) \in \mathbb{R}^n$
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: $[n] = \{1, 2, ..., n\}$
- Potential: $u(t) \in \mathbb{R}^n$
- Dynamics: $u(t+1) = u(t) - Cs(t) + I$
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \([n] = \{1, 2, ..., n\}\)
- Potential: \(u(t) \in \mathbb{R}^n\)
- Dynamics: \(u(t + 1) = u(t) - Cs(t) + I\)
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \( n = \{1, 2, \ldots, n\} \)
- Potential: \( u(t) \in \mathbb{R}^n \)
- Dynamics: \( u(t + 1) = u(t) - Cs(t) + I \)

Spiking effects
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: $[n] = \{1, 2, \ldots, n\}$
- Potential: $u(t) \in \mathbb{R}^n$
- Dynamics: $u(t + 1) = u(t) - Cs(t) + I$
- External charging: $I \in \mathbb{R}^n$
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \( n = \{1, 2, ..., n\} \)
- Potential: \( u(t) \in \mathbb{R}^n \)
- Dynamics: \( u(t + 1) = u(t) - Cs(t) + I \)
- External charging: \( I \in \mathbb{R}^n \)
- Spikes: \( s(t) \in \{0, 1\}^n \)

Firing Rule

\[ s_i(t) = 1 \iff u_i(t) \geq \eta \]
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: $[n] = \{1, 2, \ldots, n\}$
- Potential: $u(t) \in \mathbb{R}^n$
- Dynamics: $u(t + 1) = u(t) - Cs(t) + I$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0, 1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$

Firing Rule $s_i(t) = 1 \iff u_i(t) \geq \eta$
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \( \{n\} = \{1, 2, ..., n\} \)
- Potential: \( u(t) \in \mathbb{R}^n \)
- Dynamics: \( u(t + 1) = u(t) - Cs(t) + I \)
- External charging: \( I \in \mathbb{R}^n \)
- Spikes: \( s(t) \in \{0, 1\}^n \)
- Connectivity: \( C \in \mathbb{R}^{n \times n} \)

Firing Rule: \( s_i(t) = 1 \iff u_i(t) \geq \eta \)
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \( n = \{1, 2, \ldots, n\} \)
- Potential: \( u(t) \in \mathbb{R}^n \)
- Dynamics: \( u(t + 1) = u(t) - Cs(t) + I \)
- External charging: \( I \in \mathbb{R}^n \)
- Spikes: \( s(t) \in \{0, 1\}^n \)
- Connectivity: \( C \in \mathbb{R}^{n \times n} \)

Firing Rule:
\[ s_i(t) = 1 \iff u_i(t) \geq \eta \]
Integrate-and-Fire (IAF) Model [Lapicque 1907]

- Neurons: \([n] = \{1,2,\ldots,n\}\)
- Potential: \(u(t) \in \mathbb{R}^n\)
- Dynamics: \(u(t + 1) = u(t) - Cs(t) + I\)
- External charging: \(I \in \mathbb{R}^n\)
- Spikes: \(s(t) \in \{0,1\}^n\)
- Connectivity: \(C \in \mathbb{R}^{n \times n}\)
- Firing rate: \(x(t) = \frac{\# \text{spikes before time } t}{t}\)

Firing Rule: \(s_i(t) = 1 \iff u_i(t) \geq \eta\)

Chi-Ning Chou (Harvard University)
Example

• Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

#spikes = 0; $x_1(t) = 0$; $u_1(t) = 0$

#spikes = 0; $x_2(t) = 0$; $u_2(t) = 0$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 1$

- #spikes = 0; $x_1(t) = 0; u_1(t) = 0.2$

- #spikes = 0; $x_2(t) = 0; u_2(t) = 0$
Example

- Setup: \( C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, \; I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}, \; u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1. \)

\begin{align*}
\# \text{spikes} &= 0; \; x_1(t) = 0; \; u_1(t) = 0.4 \\
\# \text{spikes} &= 0; \; x_2(t) = 0; \; u_2(t) = 0
\end{align*}
**Example**

- **Setup:** \( C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, \quad I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}, \quad u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1. \)

- **Potential:** \( u(t) \in \mathbb{R}^n \)
- **External charging:** \( I \in \mathbb{R}^n \)
- **Spikes:** \( s(t) \in \{0,1\}^n \)
- **Connectivity:** \( C \in \mathbb{R}^{n \times n} \)
- **Firing rate:** \( x(t) \)
- \( u(t + 1) = u(t) - Cs(t) + I \)

\[ \#\text{spikes} = 0; \quad x_1(t) = 0; \quad u_1(t) = 0.6 \]

\[ \#\text{spikes} = 0; \quad x_2(t) = 0; \quad u_2(t) = 0 \]
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

$t = 4$

- #spikes = 0; $x_1(t) = 0; u_1(t) = 0.8$

- #spikes = 0; $x_2(t) = 0; u_2(t) = 0$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1$.

\[
\#\text{spikes} = 0; \quad x_1(t) = 0; \quad u_1(t) = 1
\]

\[
\#\text{spikes} = 0; \quad x_2(t) = 0; \quad u_2(t) = 0
\]
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

#spikes = 1; $x_1(t) = 0.2; u_1(t) = 1$

#spikes = 0; $x_2(t) = 0; u_2(t) = 0$

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 5$

#spikes = 1; $x_1(t) = 0.2; u_1(t) = 1$

#spikes = 0; $x_2(t) = 0; u_2(t) = 0$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

\#spikes = 1; $x_1(t) = 0.167; u_1(t) = 0.2$

\#spikes = 0; $x_2(t) = 0; u_2(t) = 0.5$
Example

• Setup: \( C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, \quad I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}, \quad u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1. \)

• Potential: \( u(t) \in \mathbb{R}^n \)
• External charging: \( I \in \mathbb{R}^n \)
• Spikes: \( s(t) \in \{0,1\}^n \)
• Connectivity: \( C \in \mathbb{R}^{n \times n} \)
• Firing rate: \( x(t) \)
• \( u(t + 1) = u(t) - Cs(t) + I \)

\( t = 7 \)

\#spikes = 1; \quad x_1(t) = 0.143; \quad u_1(t) = 0.4

\( t = 7 \)

\#spikes = 0; \quad x_2(t) = 0; \quad u_2(t) = 0.5
Example

- Setup: $C = (\begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, I = (0.2) , u(0) = (0) , \eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 8$

$\#\text{spikes} = 1; x_1(t) = 0.125; u_1(t) = 0.6$

$\#\text{spikes} = 0; x_2(t) = 0; u_2(t) = 0.5$
Example

• Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

\begin{itemize}
  \item Potential: $u(t) \in \mathbb{R}^n$
  \item External charging: $I \in \mathbb{R}^n$
  \item Spikes: $s(t) \in \{0,1\}^n$
  \item Connectivity: $C \in \mathbb{R}^{n \times n}$
  \item Firing rate: $x(t)$
  \item $u(t + 1) = u(t) - Cs(t) + I$
\end{itemize}

#spikes = 1; $x_1(t) = 0.111; u_1(t) = 0.8$

#spikes = 0; $x_2(t) = 0; u_2(t) = 0.5$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 10$

- #spikes = 1; $x_1(t) = 0.1$; $u_1(t) = 1$

- #spikes = 0; $x_2(t) = 0$; $u_2(t) = 0.5$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

\[ \#\text{spikes} = 2; \ x_1(t) = 0.2; \ u_1(t) = 1 \]

\[ \#\text{spikes} = 0; \ x_2(t) = 0; \ u_2(t) = 0.5 \]
Example

- Setup: \( C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, \quad I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}, \quad u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1. \)

\begin{align*}
\# \text{spikes} &= 2; \quad x_1(t) = 0.2; \quad u_1(t) = 1 \\
\# \text{spikes} &= 0; \quad x_2(t) = 0; \quad u_2(t) = 0.5
\end{align*}

- Potential: \( u(t) \in \mathbb{R}^n \)
- External charging: \( I \in \mathbb{R}^n \)
- Spikes: \( s(t) \in \{0,1\}^n \)
- Connectivity: \( C \in \mathbb{R}^{n \times n} \)
- Firing rate: \( x(t) \)
- \( u(t + 1) = u(t) - Cs(t) + I \)
Example

• Setup: \( C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}, \ I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}, \ u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \eta = 1. \)

\( t = 11 \)

\#spikes = 2; \ x_1(t) = 0.182; \ u_1(t) = 0.2

\#spikes = 0; \ x_2(t) = 0; \ u_2(t) = 1
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $\mathbf{u}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

#spikes = 2; $x_1(t) = 0.182$; $\mathbf{u}_1(t) = 0.2$

$\mathbf{u}(t + 1) = \mathbf{u}(t) - Cs(t) + I$

- Potential: $\mathbf{u}(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0, 1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$

#spikes = 1; $x_2(t) = 0.09$; $\mathbf{u}_2(t) = 1$
Example

- **Setup:** $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 11$

$\#\text{spikes} = 2; \ x_1(t) = 0.182; \ u_1(t) = 0.2$

$\#\text{spikes} = 1; \ x_2(t) = 0.09; \ u_2(t) = 1$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0,1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$

$t = 12$

#spikes = 2; $x_1(t) = 0.167; u_1(t) = 0.4$

#spikes = 1; $x_2(t) = 0.83; u_2(t) = 0$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

- #spikes = 200; $x_1(t) = 0.200$; $u_1(t) = 1$

- #spikes = 99; $x_2(t) = 0.099$; $u_2(t) = 0.5$

- Potential: $u(t) \in \mathbb{R}^n$
- External charging: $I \in \mathbb{R}^n$
- Spikes: $s(t) \in \{0, 1\}^n$
- Connectivity: $C \in \mathbb{R}^{n \times n}$
- Firing rate: $x(t)$
- $u(t + 1) = u(t) - Cs(t) + I$
Example

- Setup: $C = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$, $I = \begin{pmatrix} 0.2 \\ 0 \end{pmatrix}$, $u(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\eta = 1$.

$\#\text{spikes} = 200; x_1(t) = 0.200; u_1(t) = 1$

$\#\text{spikes} = 99; x_2(t) = 0.099; u_2(t) = 0.5$
The Algorithmic Power of Integrate-and-fire SNN?
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]
Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]
Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

\((C,I)\)

firing rate
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]

Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

\[ \min_{x \in \mathbb{R}^n} \|Cx - I\|_2^2 \quad \text{s.t. } x_i \geq 0, \forall i \in [n] \]

\( (C, I) \)

firing rate \( \approx \) solution
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]

Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

Using an optimization problem to estimate the firing rate of an integrate-and-fire SNN.

\[
\min_{x \in \mathbb{R}^n} \|Cx - I\|_2^2 \\
\text{s.t. } x_i \geq 0, \forall i \in [n]
\]
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]
Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

\[
\min_{x \in \mathbb{R}^n} \| Cx - I \|_2^2 \\
\text{s.t. } x_i \geq 0, \forall i \in [n]
\]

Using solution to estimate firing rate
The Algorithmic Power of Integrate-and-fire SNN?

[Barrett-Denève-Machens, NIPS 2013]

Using an “optimization problem” to analyze the firing rate of an integrate-and-fire SNN.

Using firing rate to estimate solution

Using solution to estimate firing rate

Non-negative Least Squares

\[
\min_{x \in \mathbb{R}^n} \|Cx - I\|_2^2
\]

s.t. \( x_i \geq 0, \forall i \in [n] \)

(firing rate) \( \approx \) solution

No provable analysis!
Our Contributions
Our Contributions

The **first** proof for the **firing rate** of integrate-and-fire SNNs **efficiently** solving the **non-negative least squares problem**.

• Confirm the empirical discovery of [Barret-Denève-Machens 2013].
Our Contributions

The **first** proof for the **firing rate** of integrate-and-fire SNNs **efficiently** solving the **non-negative least squares problem**.

- Confirm the empirical discovery of [Barret-Denève-Machens 2013].

What if there are **infinitely many solutions**?
Our Contributions

The **first** proof for the **firing rate** of integrate-and-fire SNNs **efficiently** solving the **non-negative least squares problem**.

- Confirm the empirical discovery of [Barret-Denève-Machens 2013].

What if there are **infinitely many solutions**?

- Further show that the **firing rate** of integrate-and-fire SNN **efficiently** finds the **sparse solution** (in the $\ell_1$ sense)
Our Contributions

The first proof for the firing rate of integrate-and-fire SNNs efficiently solving the non-negative least squares problem.

- Confirm the empirical discovery of [Barret-Denève-Machens 2013].

What if there are infinitely many solutions?

- Further show that the firing rate of integrate-and-fire SNN efficiently finds the sparse solution (in the $\ell_1$ sense), by implementing a primal-dual + projected gradient descent algorithm.
Theorem (ℓ₁ minimization problem)
Theorem (\(\ell_1\) minimization problem)

Given \(A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m\), and \(\epsilon > 0\). Suppose \(A\) satisfies some regular conditions. Set \(C = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}, I = \begin{pmatrix} A^T b \\ -A^T b \end{pmatrix}\), and properly set the integrate-and-fire SNN.
Theorem ($\ell_1$ minimization problem)

Given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, and $\epsilon > 0$. Suppose $A$ satisfies some regular conditions. Set $C = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}$, $I = \begin{pmatrix} A^T b \\ -A^T b \end{pmatrix}$, and properly set the integrate-and-fire SNN. Let $x^*$ be the optimal solution to the $\ell_1$ minimization problem.

$$\min_{x \in \mathbb{R}^n} \|x\|_1 \quad \text{s.t.} \quad Ax = b$$
Theorem ($\ell_1$ minimization problem)

Given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, and $\epsilon > 0$. Suppose $A$ satisfies some regular conditions. Set $C = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}$, $I = \begin{pmatrix} A^T b \\ -A^T b \end{pmatrix}$, and properly set the integrate-and-fire SNN. Let $x^*$ be the optimal solution to the $\ell_1$ minimization problem.

$$\min_{x \in \mathbb{R}^n} \|x\|_1 \quad \text{s.t.} \quad Ax = b$$

When $t \geq \Omega\left(\frac{n^3}{\epsilon^2}\right)$, we have (i) $\|b - Ax(t)\|_2 \leq \epsilon \cdot \|b\|_2$ and (ii) $\|x(t)\|_1 - \|x^*\|_1 \leq \epsilon \cdot \|x^*\|_1$. 

Chi-Ning Chou (Harvard University)
Theorem ($\ell_1$ minimization problem)

Given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, and $\epsilon > 0$. Suppose $A$ satisfies some regular conditions. Set $C = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}$, $I = \begin{pmatrix} A^T b \\ -A^T b \end{pmatrix}$, and properly set the integrate-and-fire SNN. Let $x^*$ be the optimal solution to the $\ell_1$ minimization problem.

$$\min_{x \in \mathbb{R}^n} \|x\|_1 \quad \text{s.t.} \quad Ax = b$$

When $t \geq \Omega(\frac{n^3}{\epsilon^2})$, we have (i) $\|b - Ax(t)\|_2 \leq \epsilon \cdot \|b\|_2$ and (ii) $\|x(t)\|_1 - \|x^*\|_1 \leq \epsilon \cdot \|x^*\|_1$. 

Chi-Ning Chou (Harvard University)
Key Technique – A **Dual View** of SNN
Key Technique – A **Dual View** of SNN

<table>
<thead>
<tr>
<th></th>
<th><strong>Primal SNN</strong></th>
<th><strong>Dual SNN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td>$u(t + 1) = u(t) - \begin{pmatrix} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{pmatrix} s(t) + \begin{pmatrix} A^T b \ -A^T b \end{pmatrix}$</td>
<td></td>
</tr>
<tr>
<td><strong>Optimization problem</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Key Technique – A **Dual View** of SNN

<table>
<thead>
<tr>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td>$u(t + 1) = u(t) - \begin{pmatrix} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{pmatrix} s(t) + \begin{pmatrix} A^T b \ -A^T b \end{pmatrix}$</td>
</tr>
<tr>
<td><strong>Optimization problem</strong></td>
<td>$\min_{x \in \mathbb{R}^n} |x|_1$ ((\ell_1) minimization)</td>
</tr>
</tbody>
</table>

Chi-Ning Chou (Harvard University)
# Key Technique – A Dual View of SNN

<table>
<thead>
<tr>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td>$u(t + 1) = u(t) - (A^TA \quad -A^TA)s(t) + (A^Tb)$</td>
</tr>
<tr>
<td><strong>Optimization problem</strong></td>
<td>$\min_{x \in \mathbb{R}^n} |x|_1 \quad (\ell_1 \text{ minimization})$</td>
</tr>
</tbody>
</table>

Chi-Ning Chou (Harvard University)

On the Algorithmic Power of Spiking Neural Networks
### Key Technique – A Dual View of SNN

<table>
<thead>
<tr>
<th></th>
<th><strong>Primal SNN</strong></th>
<th><strong>Dual SNN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td>$u(t + 1) = u(t) - \begin{pmatrix} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{pmatrix} s(t) + \begin{pmatrix} A^T b \ -A^T b \end{pmatrix}$</td>
<td>$v(t + 1) = v(t) - (A - A)s(t) + b$</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>$\min_{x \in \mathbb{R}^n} | x |_1$ (\text{$\ell_1$} minimization)</td>
<td>$\max_{v \in \mathbb{R}^m} b^T v$ (The dual of \text{$\ell_1$} minimization)</td>
</tr>
<tr>
<td><strong>problem</strong></td>
<td>s.t. $Ax = b$</td>
<td>s.t. $| A^T v |_\infty \leq 1$</td>
</tr>
</tbody>
</table>

Chi-Ning Chou (Harvard University)
# Key Technique – A Dual View of SNN

## Primal SNN

<table>
<thead>
<tr>
<th>Dynamics</th>
<th>[ u(t + 1) = u(t) - \begin{pmatrix} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{pmatrix} s(t) + \begin{pmatrix} A^T b \ -A^T b \end{pmatrix} ]</th>
</tr>
</thead>
</table>

| Optimization problem | \[
\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad \|x\|_1 \\
\text{s.t.} & \quad Ax = b 
\end{align*}
\] (\(\ell_1\) minimization) |

## Dual SNN

<table>
<thead>
<tr>
<th></th>
<th>[ v(t + 1) = v(t) - (A - A)s(t) + b ]</th>
</tr>
</thead>
</table>

| Optimization problem | \[
\begin{align*}
\max_{v \in \mathbb{R}^m} & \quad b^T v \\
\text{s.t.} & \quad \|A^T v\|_\infty \leq 1 
\end{align*}
\] (The dual of \(\ell_1\) minimization) |

\(\nu(t)\) is a projected gradient descent algorithm for dual program with non-standard projection.
<table>
<thead>
<tr>
<th>Key Technique – A Dual View of SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primal SNN</strong></td>
</tr>
<tr>
<td><strong>Dual SNN</strong></td>
</tr>
<tr>
<td>Dynamics</td>
</tr>
<tr>
<td>$u(t + 1) = u(t) - \left( \begin{array}{cc} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{array} \right) s(t) + \left( \begin{array}{c} A^T b \ -A^T b \end{array} \right)$</td>
</tr>
<tr>
<td>$v(t + 1) = v(t) - (A - A)s(t) + b$</td>
</tr>
<tr>
<td>Optimization problem</td>
</tr>
<tr>
<td>$\min_{x \in \mathbb{R}^n} |x|_1$ (\ell_1 minimization)</td>
</tr>
<tr>
<td>s.t. $Ax = b$</td>
</tr>
<tr>
<td>$\max_{v \in \mathbb{R}^m} b^T v$ (The dual of \ell_1 minimization)</td>
</tr>
<tr>
<td>s.t. $|A^T v|_\infty \leq 1$</td>
</tr>
</tbody>
</table>

$\nu(t)$ is a projected gradient descent algorithm for dual program with non-standard projection.
Key Technique – A **Dual View** of SNN

<table>
<thead>
<tr>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td><strong>Projection</strong></td>
</tr>
<tr>
<td>$u(t + 1) = u(t) - \left( \begin{array}{cc} A^T A &amp; -A^T A \ -A^T A &amp; A^T A \end{array} \right) s(t) + \left( \begin{array}{c} A^T b \ -A^T b \end{array} \right)$</td>
<td>$v(t + 1) = v(t) - (A - A)s(t) + b$</td>
</tr>
<tr>
<td><strong>Optimization problem</strong></td>
<td><strong>Optimization problem</strong></td>
</tr>
<tr>
<td>$\min_{x \in \mathbb{R}^n} |x|_1$ ((\ell_1) minimization)</td>
<td>$\max_{v \in \mathbb{R}^m} b^T v$ (The <strong>dual</strong> of (\ell_1) minimization)</td>
</tr>
<tr>
<td>s.t. $Ax = b$</td>
<td>s.t. $|A^T v|_{\infty} \leq 1$</td>
</tr>
</tbody>
</table>

$v(t)$ is a **projected gradient descent algorithm** for **dual program** with non-standard projection.
# Key Technique – A Dual View of SNN

<table>
<thead>
<tr>
<th>Dynamics</th>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(t+1) = u(t) - (A^T ! A ! - A^T A) s(t) + (A^T ! b)$</td>
<td>$v(t+1) = v(t) - (A ! - A) s(t) + b$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization problem</th>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\min_{x \in \mathbb{R}^n} |x|_1$ \ ($\ell_1$ minimization) \ s.t. $Ax = b$</td>
<td>$\max_{v \in \mathbb{R}^m} b^T v$ \ (The dual of $\ell_1$ minimization) \ s.t. $|A^T v|_\infty \leq 1$</td>
<td></td>
</tr>
</tbody>
</table>

The **firing rate** in primal SNN solves the **primal program**.

- **KKT conditions**
- **Perturbation theory**

$v(t)$ is a **projected gradient descent algorithm** for dual program with **non-standard projection**.
## Key Technique — A **Dual View** of SNN

<table>
<thead>
<tr>
<th>Primal SNN</th>
<th>Dual SNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamics</strong></td>
<td>$u(t + 1) = u(t) - (A^TA - A^TA) s(t) + (A^T b)$</td>
</tr>
<tr>
<td><strong>Optimization problem</strong></td>
<td>$\min_{x \in \mathbb{R}^n} |x|_1 \quad (\ell_1 \text{ minimization})$</td>
</tr>
<tr>
<td><strong>s.t.</strong> $Ax = b$</td>
<td><strong>s.t.</strong> $|A^T v|_\infty \leq 1$</td>
</tr>
</tbody>
</table>

The **firing rate** in **primal SNN** solves the **primal program**.

**KKT conditions**

**Perturbation theory**

$\boldsymbol{v(t)}$ is a **projected gradient descent algorithm** for **dual program** with **non-standard projection**.

Spikes are **non-monotone** and difficult to analyze!
Perspectives – Natural Algorithms
Perspectives – Natural Algorithms

“How algorithmic ideas can enrich our understanding of nature?”
- Bernard Chazelle
Perspectives – Natural Algorithms

“How algorithmic ideas can enrich our understanding of nature?”
- Bernard Chazelle

Through the lens of natural algorithms, can we understand SNNs more via its algorithmic power and even discover new algorithmic ideas?
“How algorithmic ideas can enrich our understanding of nature?”
- Bernard Chazelle

Through the lens of natural algorithms, can we understand SNNs more via its algorithmic power and even discover new algorithmic ideas?

In this work, we show that integrate-and-fire SNN uses its firing rate to efficiently solve some optimization problems in a primal-dual way!
Conclusions
Conclusions

• In this work, we give the first proof for the firing rate of integrate-and-fire SNNs efficiently solving non-negative least squares problem and $\ell_1$ minimization problem.
Conclusions

• In this work, we give the first proof for the firing rate of integrate-and-fire SNNs efficiently solving non-negative least squares problem and $\ell_1$ minimization problem.

• Related works.
  • Universality and computational complexity.
    • SNN is able to simulate Turing machines, random access machines (RAM), and threshold circuits etc. [Maa96, Maa97b, Maa99, MB01].
  • Using SNNs to solve computational/optimization problems.
    • Sparse coding [ZMD11, Tan16, TLD17], dictionary learning [LT18], pattern recognition [DC15, KGM16, BMF+17], and non-negative least squares [BDM13].
    • Implementing MCMC to solve traveling salesman problem (TSP) and constraint satisfaction problem (CSP) [BBNM11, JHM14, Maa15, JHM16].
    • Assemblies of neurons and random projection [ADMPSV18, LPVM18, PV19].
  • The efficiency of SNNs in solving computational problems.
    • Solving Winner-Take-All (WTA) problem, similarity testing, and neural coding [LMP17a, LMP17b, LMP17c, LM18].
Conclusions

• In this work, we give the first proof for the firing rate of integrate-and-fire SNNs efficiently solving non-negative least squares problem and $\ell_1$ minimization problem.

• Related works.
  • Universality and computational complexity.
    • SNN is able to simulate Turing machines, random access machines (RAM), and threshold circuits etc. [Maa96, Maa97b, Maa99, MB01].
  • Using SNNs to solve computational/optimization problems.
    • Sparse coding [ZMD11, Tan16, TLD17], dictionary learning [LT18], pattern recognition [DC15, KGM16, BMF+17], and non-negative least squares [BDM13].
    • Implementing MCMC to solve traveling salesman problem (TSP) and constraint satisfaction problem (CSP) [BBNM11, JHM14, Maa15, JHM16].
    • Assemblies of neurons and random projection [ADMPSV18, LPVM18, PV19].
  • The efficiency of SNNs in solving computational problems.
    • Solving Winner-Take-All (WTA) problem, similarity testing, and neural coding [LMP17a, LMP17b, LMP17c, LM18].

Very few works provide provable analysis on the efficiency of SNN algorithms!
Conclusions

• In this work, we give the first proof for the firing rate of integrate-and-fire SNNs efficiently solving non-negative least squares problem and $\ell_1$ minimization problem.

• Related works.
  • Universality and computational complexity.
    • SNN is able to simulate Turing machines, random access machines (RAM), and threshold circuits etc. [Maa96, Maa97b, Maa99, MB01].
  • Using SNNs to solve computational/optimization problems.
    • Sparse coding [ZMD11, Tan16, TLD17], dictionary learning [LT18], pattern recognition [DC15, KGM16, BMF+17], and non-negative least squares [BDM13].
    • Implementing MCMC to solve traveling salesman problem (TSP) and constraint satisfaction problem (CSP) [BBNM11, JHM14, Maa15, JHM16].
    • Assemblies of neurons and random projection [ADMPSV18, LPVM18, PV19].
  • The efficiency of SNNs in solving computational problems.
    • Solving Winner-Take-All (WTA) problem, similarity testing, and neural coding [LMP17a, LMP17b, LMP17c, LM18].

• Next step?
  • Giving more rigorous analysis for the efficiency of other SNN algorithms!

Very few works provide provable analysis on the efficiency of SNN algorithms!
Conclusions

• In this work, we give the first proof for the firing rate of integrate-and-fire SNNs efficiently solving non-negative least squares problem and $\ell_1$ minimization problem.

• Related works.
  • Universality and computational complexity.
    • SNN is able to simulate Turing machines, random access machines (RAM), and threshold circuits etc. [Maa96, Maa97b, Maa99, MB01].
  • Using SNNs to solve computational/optimization problems.
    • Sparse coding [ZMD11, Tan16, TLD17], dictionary learning [LT18], pattern recognition [DC15, KGM16, BMF+17], and non-negative least squares [BDM13].
    • Implementing MCMC to solve traveling salesman problem (TSP) and constraint satisfaction problem (CSP) [BBNM11, JHM14, Maa15, JHM16].
    • Assemblies of neurons and random projection [ADMPSV18, LPVM18, PV19].
  • The efficiency of SNNs in solving computational problems.
    • Solving Winner-Take-All (WTA) problem, similarity testing, and neural coding [LMP17a, LMP17b, LMP17c, LM18].

• Next step?
  • Giving more rigorous analysis for the efficiency of other SNN algorithms!

Thanks for your attention!

Very few works provide provable analysis on the efficiency of SNN algorithms!