گزارش پروژه سوم علوم اعصاب محاسباتی

امیرحسین انتظاری

۱۴ اردیبهشت ۱۴۰۳

فهرست مطالب

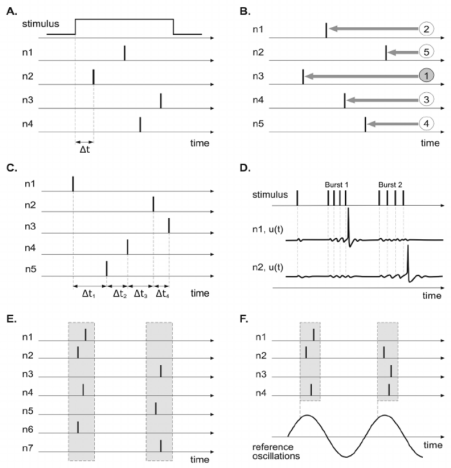
چکیده

هدف از این پروژه، پیاده سازی کدگذاری کردن ورودی شبکه، یادگیری (بدون ناظر و تقویتی) است. در این پروژه از مباحثی که در پروژه های قبلی یاد گرفتیم، (مانند مدل های نورونی، سیناپس و...)، استفاده می‌کنیم.

۱.۰ پیاده سازی کدگذاری ها

در این بخش می‌خواهیم نحوه کد کردن اطلاعات (مانند یک تصویر) در شبکه های عصبی ضربه‌ای را پیاده سازی و تحلیل کنیم. در حوزه علوم اعصاب محاسباتی، روش های کدگذاری برای ترجمه محرک های دنیای واقعی به سیگنال های عصبی که می توانند توسط مدل های محاسباتی پردازش شوند، مهم هستند. برای اینکار، روش های مختلفی وجود دارد، مانند روش time-to-first-spike، کدگذاری ترتیبی [۱]، کدگذاری براساس تاخیر [۲]، کدگذاری براساس همزمانی ضربه ها [۳]، کدگذاری اعداد، کدگذاری به روش پواسون [۴] و روش های دیگر بسیار که کاربرد های مختلفی دارند. (شکل ۱) در این پروژه، ما تمرکزمان را روی سه روش کدگذاری زیر میگذاریم و آن ها پیاده سازی و تحلیل میکنیم:

- روش کدگذاری time-to-first-spike
- روش کدگذاری مقادیر عددی
- روش کدگذاری به کمک توزیع پواسون



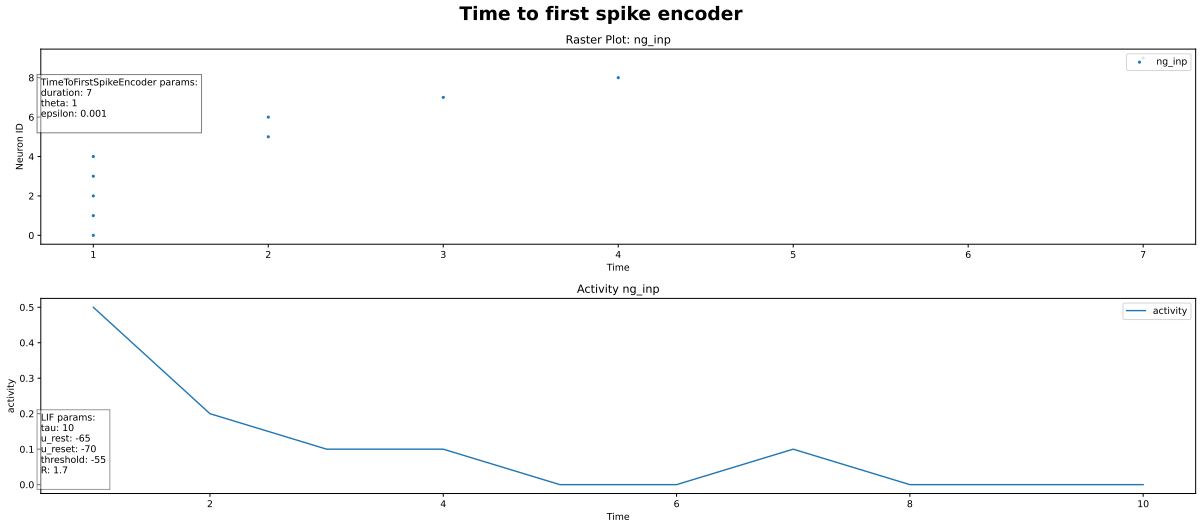
شکل ۱: روش های مختلف کد گذاری. (A) زمان برای اولین سنبله؛ (B) کدگذاری رتبه یا کدگذاری مرتبه سنبله. (C) کدگذاری تأخیر بر اساس زمان دقیق افزایش ها. (D) کدگذاری انفجاری رزونانسی. (E) کدگذاری توسط همزمانی. (F) کدگذاری فاز

۱.۱.۰ کدگذاری به روش Time-to-first-spike

کدگذاری Time-to-First-Spike (TFS) روشی است که در آن از زمان اولین اسپایک پس از شروع محرک برای انتقال شدت محرک استفاده می شود. هر چه زمان تا اولین ضربه کوتاه تر باشد، شدت محرک درک شده بیشتر می شود. این روش کدگذاری برای پردازش سریع اطلاعات بسیار مفید است.

برای کدگذاری Time-to-First-Spike (TFS) باید شدت یک محرک را به زمان‌بندی اولین ضربه عصبی تبدیل کنیم. برای اینکار، ابتدا نیاز است با نرمال‌سازی داده‌های ورود حاصل شود که همه مقادیر بین یک آستانه حداقل و حداکثر نگاشت می‌شوند و کمی نیز تنظیم می‌شوند تا از مقادیر شدید جلوگیری شود. فرآیند کدگذاری از یک آستانه استفاده می کند که به مرور توسط یک تابع نمایی کاهشی بر اساس مدت زمان کدگذاری و یک پارامتر θ کاهش می یابد. این کار به این معنی است که محرک های شدیدتر منجر به ضربه های زودتر می شود. این رمزگذار زمان اولین ضربه را برای هر محرک ثبت می‌کند و مقدار محرک پس از ضربه را برای جلوگیری از کدگذاری مجدد تنظیم می‌کند. نتیجه یک ماتریس باینری است که زمان های افزایش نسبت به شروع محرک را نشان می دهد و به طور موثر اولین پاسخ عصبی قابل توجه را به شدت های مختلف محرک ثبت می کند. به عنوان مثال، یک الگوی ورودی مانند آرایه $[100, 70, 30, 10, 0]$ را با $\theta = 1$ کدگذاری کنیم، نمودار $raster$ و نمودار فعالیت به صورت شکل ۲ خواهد شد.

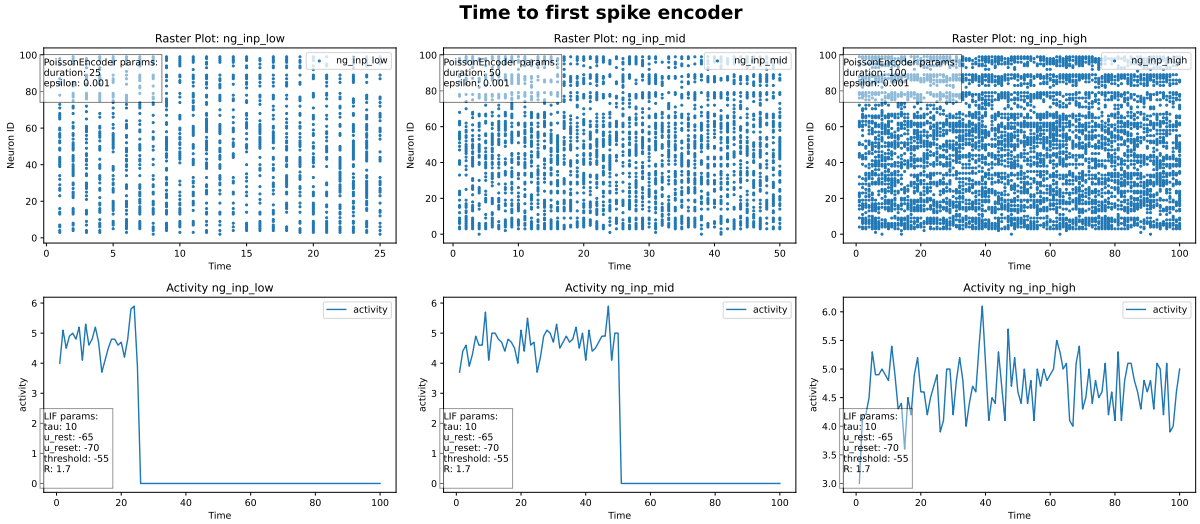
---
[۱] Rank-order encoding
[۲] Latency encoding
[۳] Coding by synchrony
[۴] Poisson



شکل ۲: کدگذاری به روش $TTFS$. همانطور که در شکل ملاحظه می شود، با انتخاب کردن $\theta = 1$ مقادیر بالای ۱۰۰ در یک لحظه ضربه زده و مقادیر کمتر در لحظه های بعدی ضربه می زنند. همچنین هر چه مقادیر کمتر می شوند، فاصله بین ضربه ها نیز کمتر می شود.

تغییر در θ

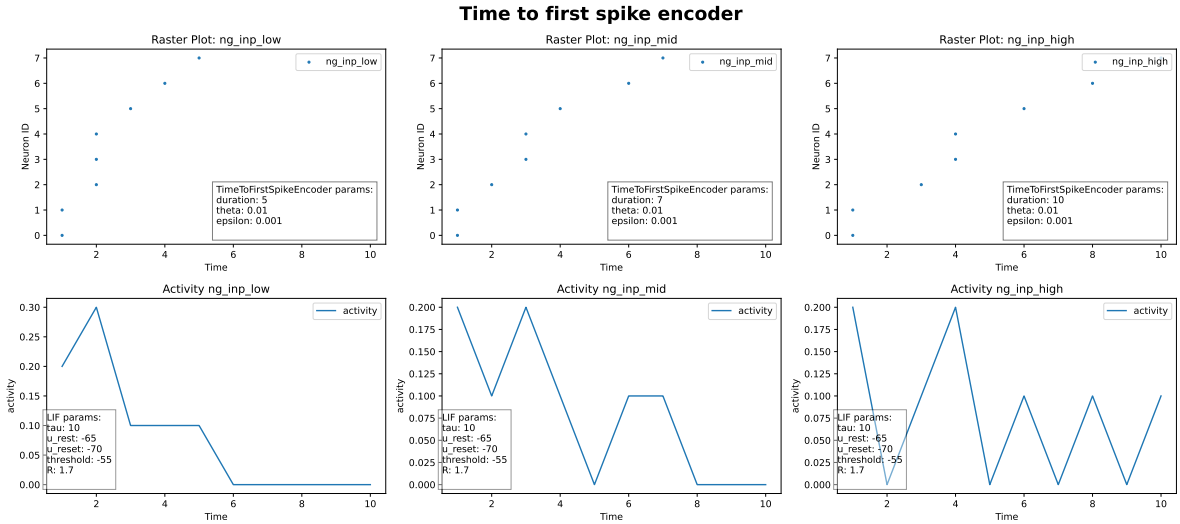
دلیل آنکه مقادیر بالا در یک زمان ضربه زده اند این است که مقدار θ نسبتا زیاد انتخاب شده است. در شکل ۳ ملاحظه میکنیم با انتخاب $\theta = 0.01$ فواصل بین ضربه ها با محرک شدیدتر نیز بیشتر می شود. به طور کلی میتوان گفت که کاهش θ میتواند باعث تفکیک بیشتر بین مقادیر ورودی شود. ۳



شکل ۳: کدگذاری به روش $TTFS$. همانطور که در شکل ملاحظه می شود، کاهش پارامتر θ میتواند باعث بیشتر شدن حساسیت جمعیت به مقادیر مختلف ورودی شود.

تغییر در مدت زمان ورودی

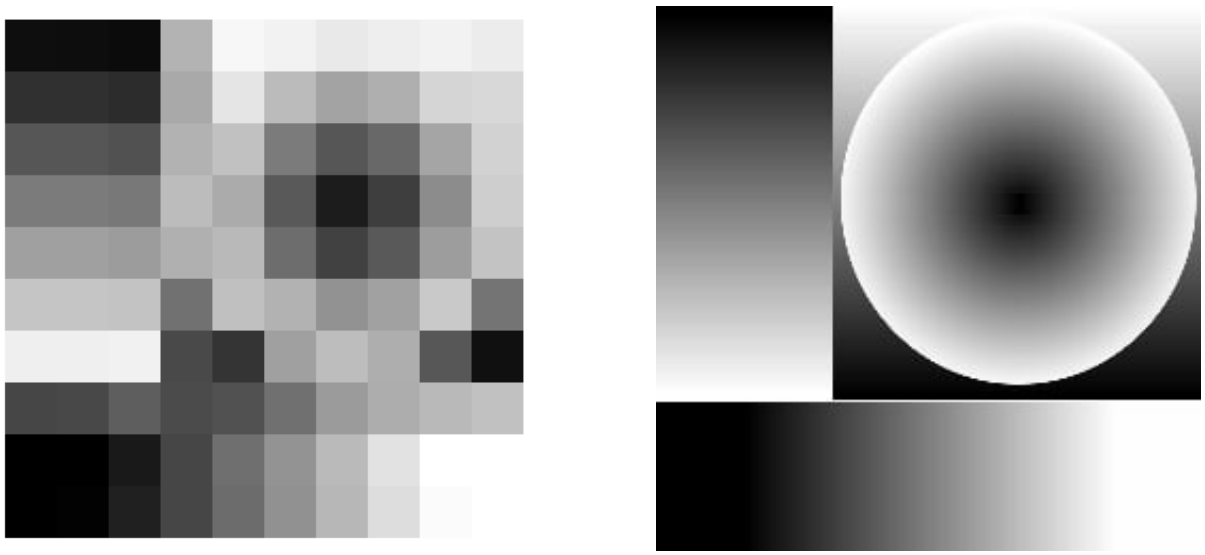
پارامتر دیگری که در خروجی کدگذاری تاثیر دارد، مدت زمانی ست که ورودی به شبکه داده می شود. این پارامتر تعیین میکند که چقدر طول میکشد تا ورود به نورون ها داده می شود. از این رو، کاهش این پارامتر باعث می شود که الگوی ضربه زدن کدگذاری شده در زمان کمتری به شبکه داده شود و از این رو فواصل بین ضربه ها می تواند کاهش یابد یا حتی در یک زمان ضربه زده شود. از آنجا که در روش کدگذاری $TTFS$ فاصله تا اولین نورونی که ضربه می زند مهم است، انتخاب نادرست این پارامتر میتواند باعث شود تا ورودی به خوبی کدگذاری نشود. افزایش این مدت زمان نیز باعث می شود تا فواصل بین ضربه ها بیشتر شود. (شکل ۴)



شکل ۴: کدگذاری به روش $TTFS$. همانطور که ملاحظه می شود، از راست به چپ، با کاهش مدت زمان ورودی، فاصله بین ضربه ها کمتر شده و زمان ضربه زدن نورون ها فشرده تر می شود. مثلا نورون شماره ۳ و۴ در یک زمان ضربه می زنند.

کدگذاری تصویر

حال یک قدم فراتر گذاشته، و سعی میکنیم یک تصویر را با روش $TTFS$ کدگذاری کنیم. برای انتخاب تصویر، از مخزن دانشگاه واترلو [۵] استفاده میکنیم. این تصاویر همگی سیاه و سفید هستند. از این رو میتوانیم کل تصویر را به صورت یک آرایه درنظر بگیریم، به طوری که سطر های تصویر را پشت سر هم ردیف میکنیم. یک تصویر مانند شکل ۵ در نظر میگیریم. حال یک آرایه به طول $m \times n$ داریم که مقادیر آن بین ۰ و ۲۵۶ هستند. از آنجا که ابعاد تصاویر مخزن، حدود 500×500 هستند، تعداد نورون های مورد نیاز برای کد کردن این ورودی بسیار زیاد خواهد بود و از هدف این پروژه خارج می شود. از این رو ابتدا ابعاد تصاویر را دستکاری کرده و به 10×10 تغییر می دهیم. اکنون کافی است مانند بخش قبل ، این آرایه را به عنوان ورودی کدگذاری کرده و به شبکه بدهیم.

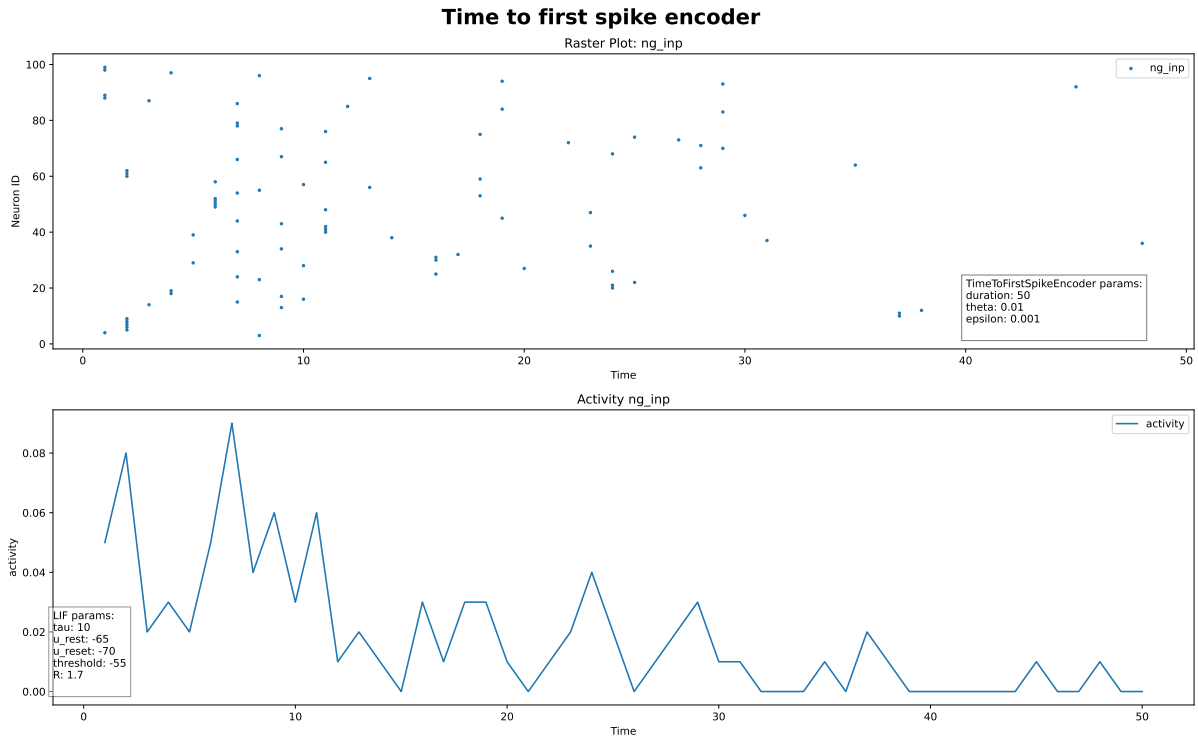


(آ) تصویر اصلی (ب) تصویر کوچک شده

شکل ۵: یک تصویر به عنوان محرک

حال اگر این تصویر را پس از به عنوان ورودی به شبکه بدهیم نمودار ضربه های کدگذاری شده این تصویر به صورت شکل ۶ خواهد بود. تغییر در مدت زمان ورودی یا θ نتیجه ای مشابه با تغییراتی که در هنگام کدگذاری آرایه اعداد داشتیم خواهد داشت.

[۵] مخزن تصاویر دانشگاه واترلو



شکل ۶: کدگذاری تصویر به روش $TTFS$.

۲.۱.۰ کدگذاری اعداد

در این بخش به بررسی یک روش برای کدگذاری اعداد می پردازیم. این روش با استفاده از تابع توزیع نرمال کدگذاری می‌کند. کد گذاری بدین صورت است که برای کد کردن یک عدد، چندین توزیع نرمال مختلف بین یک بازه از اعداد را در نظر گرفته، و مقدار عدد ورودی را به هر یک از این توزیع ها داده و مشاهده میکنیم که این عدد هر یک از این توزیع ها را در کجا قطع می کند. به طور به خصوص فرض کنید می خواهیم یک عدد بین ۱ تا ۱۰، مثلا ۷.۴ را کدگذاری کنیم. حال به ازای هر یک از اعداد صحیح در این بازه یک نورون و یک تابع توزیع نرمال با میانگین آن عدد در نظر میگیریم. حال مقدار هر یک از این توزیع ها در نقطه ۷.۴ را محاسبه میکنیم. هر توزیعی که مقدار بیشتری داشت، بدان معنا است که زودتر ضربه می زند. (شکل ۷ را نگاه کنید.)

زمان ضربه زدن نورون ها به روش های مختلفی می تواند انجام شود. مثلا تابع توزیع با میانگین ۱، ۲، ۷، ۸، ۹ یا ۱۰ برای عدد ۷.۴ مقدار بسیار کم و نزدیک به همی دارند، و در نتیجه همگی در آخرین لحظات پنجره زمانی [۶] ضربه خواهند زد. از این رو، باید انتخاب کنیم که آیا نیاز به ضربه زدن همه نورون های بازه داریم یا ضربه زدن مقادیر نزدیک تر کافی است. اینکار توسط انتخاب یک پارامتر مانند ϵ انجام میگیرد که تعیین میکند مقادیر توزیع ها در x از چه مقداری باید بیشتر باشند. همچنین برای مقادیر بزرگتر، میتوانیم یک بازه بزرگتر انتخاب کنیم، یا اعداد را به همین بازه ۰ تا ۱۰ مان نگاشت کنیم. همه این روش ها بستگی به استفاده ما دارد. نکته دیگری که در رابطه با این روش کدگذاری وجود دارد، این است که این روش برای کدگذاری داده های زیاد ممکن است مناسب نباشد، چرا که ما به ازای کد کردن هر عدد، نیاز به تعدادی نورون داریم. (به عنوان مثال، برای کد کردن تنها ۱۰۰ عدد، نیاز به $100 * 10 = 1000$ نورون داریم!)

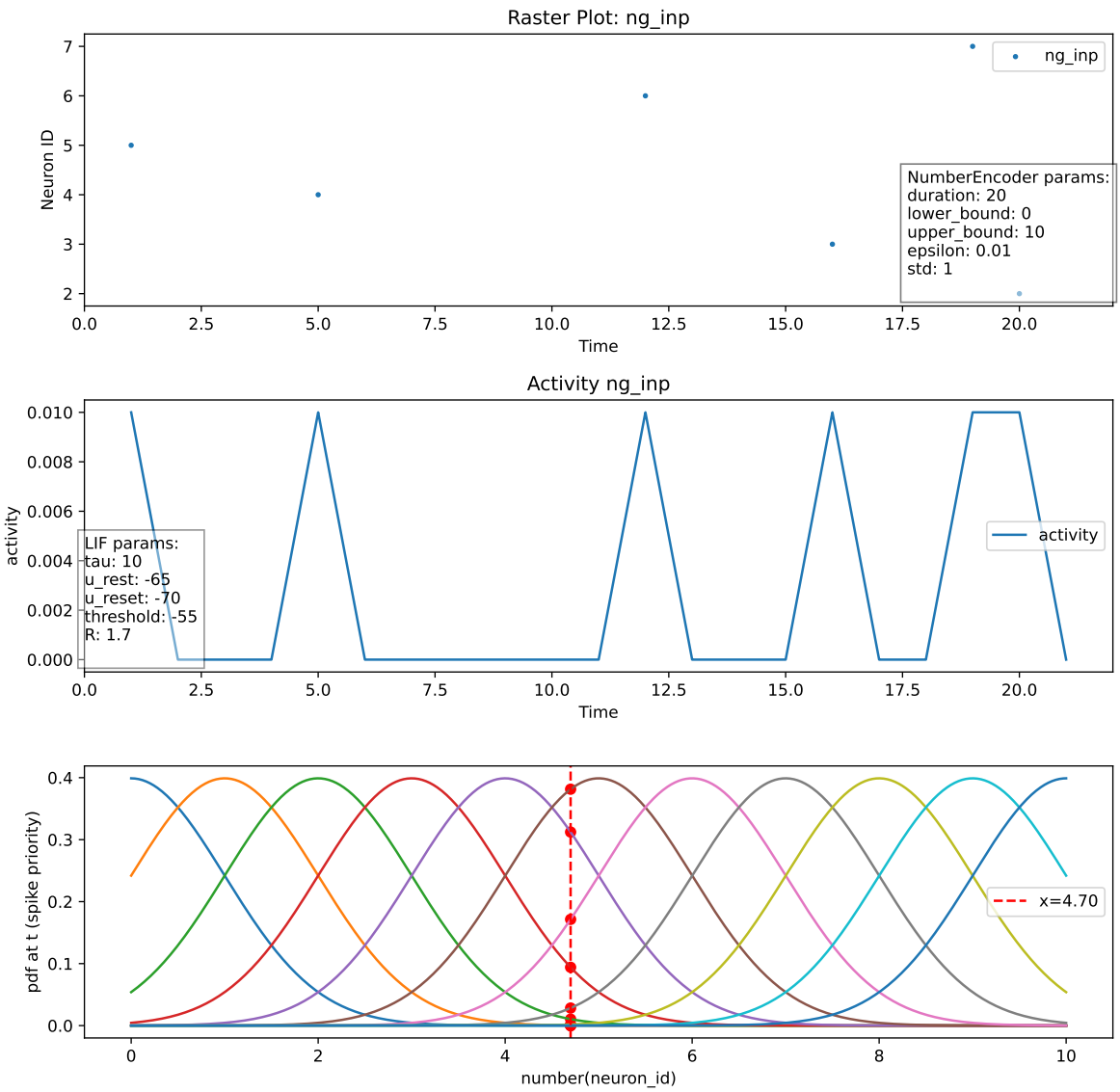
۳.۱.۰ کدگذاری به کمک توزیع پواسون

قبل از اینکه بررسی کنیم کدگذاری به روش پواسون چگونه انجام می شود، بیایید تعریف توزیع پواسون را یکبار با یکدیگر مرور کنیم: **توزیع پواسون:** در آمار و احتمال توزیع پواسون یک توزیع احتمالی گسسته است که احتمال اینکه یک حادثه به تعداد مشخصی در فاصلهٔ زمانی یا مکانی ثابتی رخ دهد را شرح می‌دهد؛ به شرط اینکه این حوادث با نرخ میانگین مشخصی و مستقل از زمان آخرین حادثه رخ دهند. (توزیع پواسون همچنین برای تعدادی از حوادث در فاصله‌های مشخص دیگری مثل مسافت، مساحت یا حجم استفاده شود)[۴]

یک متغیر تصادفی گسسته X دارای توزیع پواسون است، با پارامتر $\lambda > 0$ اگر تابع جرم احتمالی به صورت زیر باشد:

$$f(k, \lambda) = P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad (۱)$$

[۶] time window



شکل ۷: کدگذاری اعداد. همانطور که در شکل مشاهده می شود، اولین توزیعی که $x = 4.7$ را قطع میکند، توزیع مربوط به نورون ۵ است که در واقعیت نیز عدد ۵ نزدیک ترین عدد صحیح به ۴.۷ است. توزیع های بعدی که $x = 4.7$ را قطع کرده اند به ترتیب مربوط به توزیع با میانگین ۴، توزیع با میانگین ۶، توزیع با میانگین ۳، توزیع با میانگین ۷ و توزیع با میانگین ۳ هستند. بقیه توزیع ها بدلیل داشتن مقدار کم در $x = 4.7$ و نزدیک بودن به هم، با توجه به انتخاب ϵ با مقدار کم، ضربه ای نزده اند.

توزیع پواسون را می توان برای سیستم هایی با تعداد زیادی رویداد ممکن که هر کدام نادر هستند اعمال کرد. تعداد چنین رویدادهایی که در یک بازه زمانی ثابت رخ می دهند، در شرایط مناسب، یک عدد تصادفی با توزیع پواسون است.

معادله را می توان طوری بازنویسی کرد اگر به جای میانگین تعداد رویدادهای λ ، نرخ متوسط r به ما داده شود. که در آن وقایع رخ می دهد. از این رو $r = \lambda t$ و:

$$P(X = k) = \frac{e^{-rt}(rt)^k}{k!} \tag{۲}$$

ما میدانیم نورون ها ضربه هایی از خود تولید می کنند که این ضربه ها را می توان به عنوان رویدادهای گسسته مدل کرد. به عبارت دیگر زمان هر ضربه را می توان به عنوان یک رویداد تصادفی مشاهده کرد.

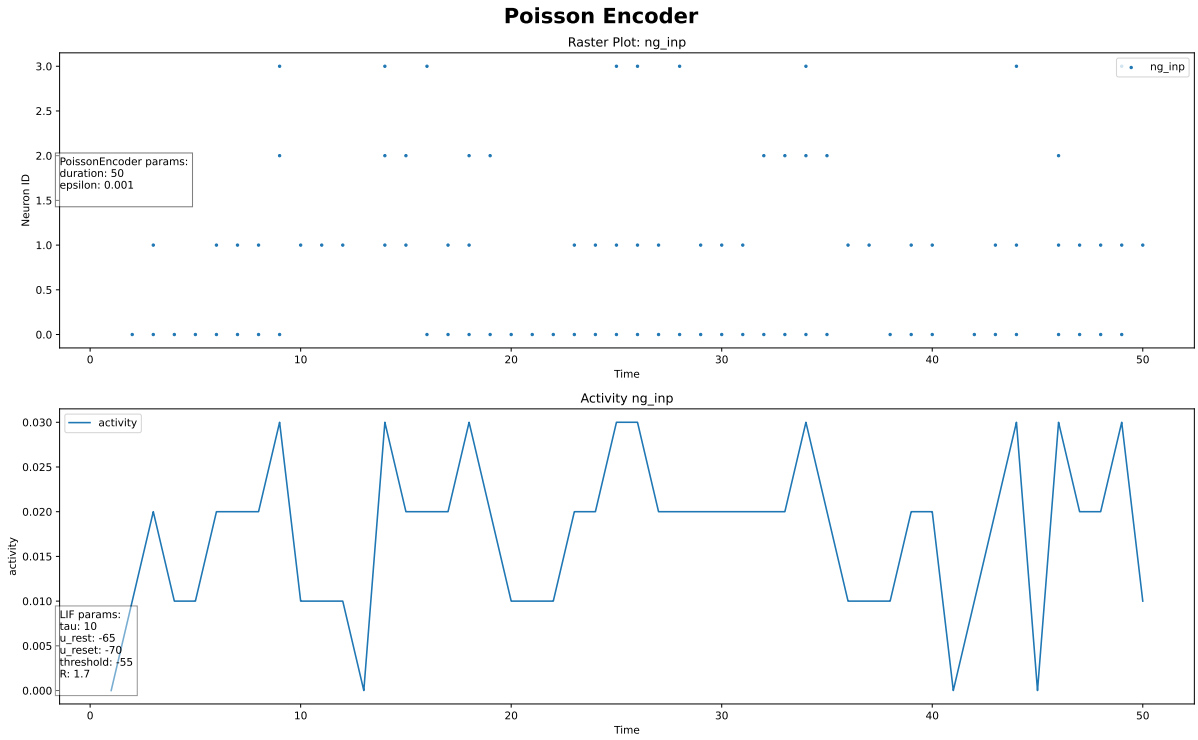
با استفاده از توزیع پواسون، می‌توانیم احتمال P مشاهده دقیق k ضربه در بازه زمانی ثابت t را به صورت زیر مدل کنیم:[۴]

$$P(X = k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!}, \tag{۳}$$

که در آن،

- X تعداد ضربه ها است.
- e عدد اویلر است
- k تعداد ضربه هایی است که ما به طور خاص در بازه t به آنها علاقه مندیم.
- λt تعداد ضربه های مورد انتظار در بازه t است.

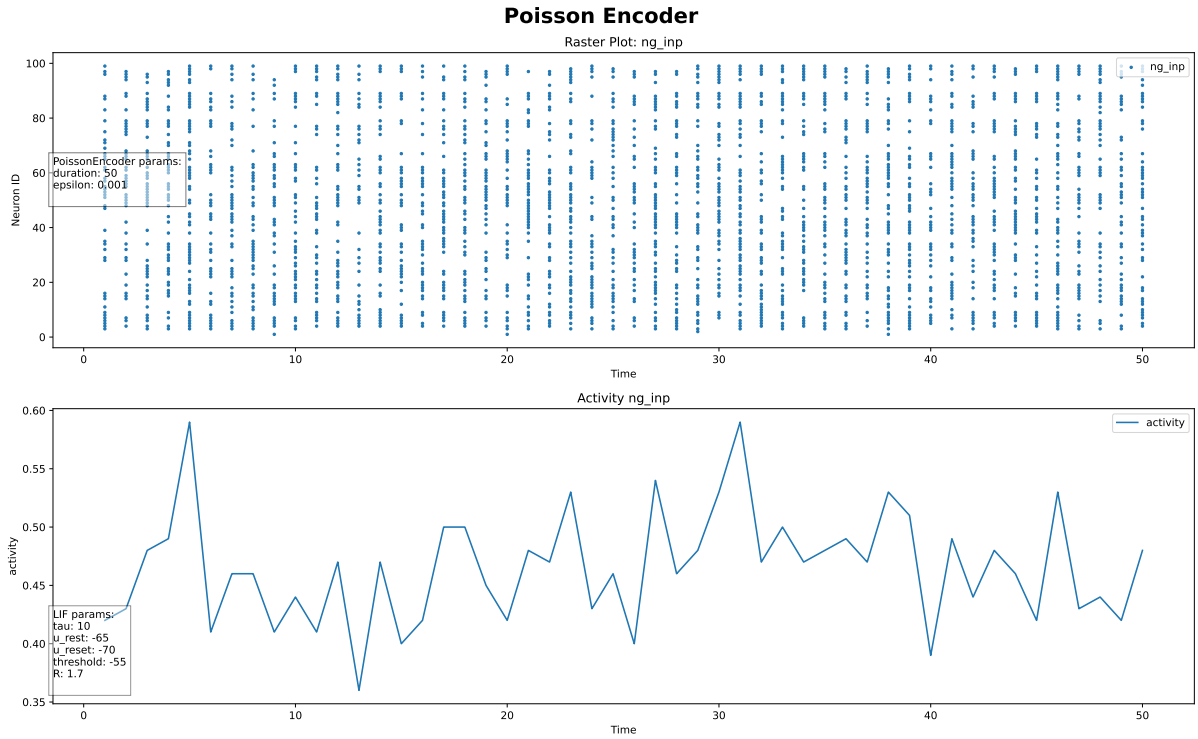
حال که نحوه عملکرد و پیاده سازی این کدگذاری را فهمیدیم، میتوانیم به سراغ آزمایش کردن کدگذاری ورودی ها برویم. به طور مثال، همانطور مشاهده می شود، به ازای ورودی $[100, 70, 30, 10, 0]$ ضربه های کدگذاری شده به صورت شکل ۸ خواهد بود.



شکل ۸: کدگذاری به کمک توزیع پواسون. همانطور که ملاحظه می شود، نورون شماره صفر که بیشترین مقدار، یعنی ۱۰۰ را دارد، بیشترین ضربه را در پنجره زمانی ورودی زده است. نورون های دیگر نیز به نسبت اندازه خود نرخ ضربه داشته اند.

کدگذاری تصویر

کدگذاری تصویر به روش پواسون یکی از مواردی است که در ادامه پروژه نیز با آن سر و کار داریم. از این رو می‌خواهیم یک دید کلی نسبت به کدگذاری تصویر به کمک توزیع پواسون داشته باشیم. برای اینکار دوباره از مخزن دانشگاه واترلو استفاده میکنیم و تصویر ۵ استفاده میکنیم. ضربه های کدگذاری شده به صورت شکل ۹ خواهد بود.



شکل ۹: کدگذاری تصویر به کمک توزیع پواسون. شکل بالا مربوط به کدگذاری یک تصویر 10×10 است. تصاویر ابتدا به صورت یک آرایه در می آیند و سپس کدگذاری می شوند. در شکل بالا هر نورون مسئول کد کردن یک پیکسل است.

۲.۰ یادگیری بدون ناظر

۳.۰ قانون یادگیری تقویتی

کتاب‌نامه

[۱] Computational Neuroscience Course, School of computer science, University of Tehran

[۲] PymoNNtorchPytorch-adapted version of PymoNNto

[۳] Neuronal Dynamics, Wulfram Gerstner, Werner M. Kistler, Richard Naud and Liam Paninski

[۴] Poisson Distribution. Wikipedia [Link]