گزارش پروژه سوم علوم اعصاب محاسباتی

امیرحسین انتظاری

۱۹ خرداد ۱۴۰۳

فهرست مطالب

چکیده

هدف از این پروژه، پیاده سازی کدگذاری کردن ورودی در شبکه های عصبی ضربه‌ای، یادگیری بدون ناظر و یادگیری تقویتی است. است. در این پروژه از مباحثی که در پروژه های قبلی یاد گرفتیم، (مانند مدل های نورونی، سیناپس و...)، استفاده می‌کنیم تا یک شبکه عصبی ضربه ای دو لایه را تشکیل دهیم. ابتدا روش های مختلف کدگذاری، از جمله کدگذاری Time-to-first-spike، کدگذاری اعداد به کمک توزیع نرمال و کدگذاری به روش پواسون رابررسی میکنیم. بعد از کد کردن محرک ها، نوبت به ورودی دادن آن ها به شبکه می‌شود. در ادامه، شبکه ای متشکل از دو لایه را پیاده سازی می‌کنیم و قانون های یادگیری مختلف، از جمله قانون یادگیری انعطاف‌پذیری وابسته به زمان ضربه ($STDP$) و مدل تقویتی آن یعنی $RSTDP$ را روی مدل اعمال میکنیم و پارامتر های آن را در شرایط مختلف آزمایش و تحلیل می‌کنیم.

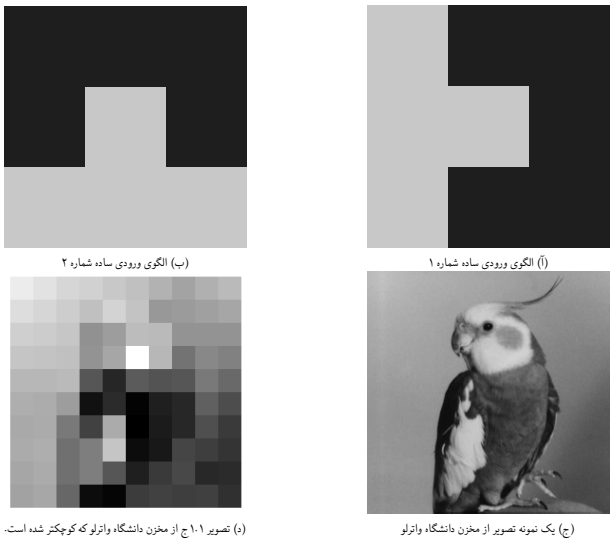
بخش ۱

ساز و کار های درون یک لایه نورونی

۱.۱ مقدمه

۱.۱.۱ مجموعه داده مورد استفاده

در این پروژه، برای الگو های ساده، از الگوی ۱.۱آ و ۱.۱ب استفاده می‌کنیم و برای الگو های پیچیده تر از از مخزن دانشگاه واترلو [۱] با نسبت اندازه های مختلف 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. کدگذاری مورد استفاده در این پروژه نیز، کد گذاری پواسون می‌باشد.



(آ) الگوی ورودی ساده شماره ۱

(ب) الگوی ورودی ساده شماره ۲

(ج) یک نمونه تصویر از مخزن دانشگاه واترلو

(د) تصویر ۱.۱ج از مخزن دانشگاه واترلو که کوچکتر شده است.

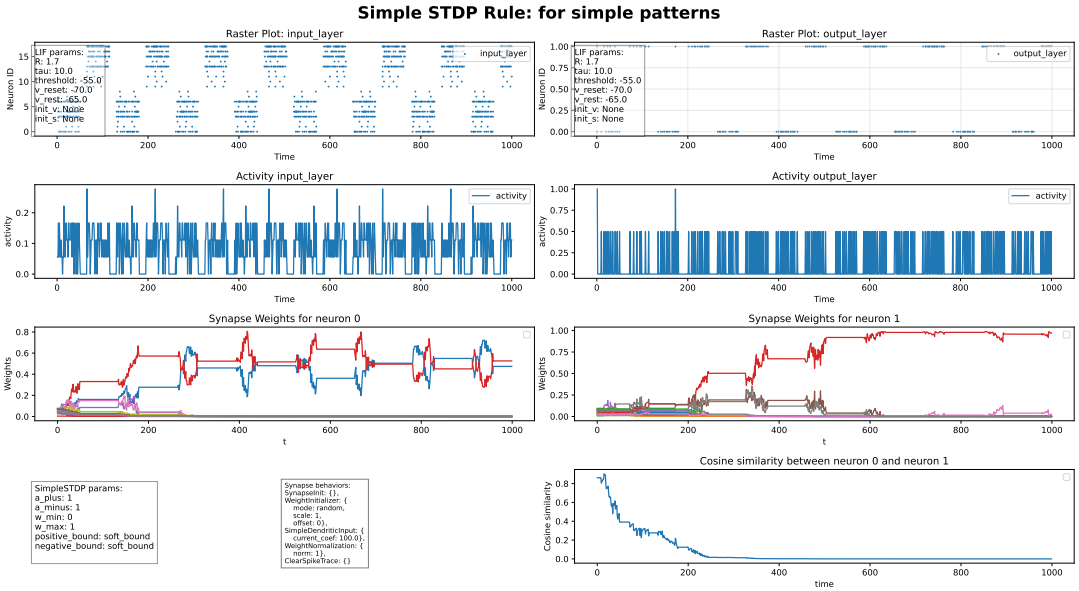
شکل ۱.۱: مجموعه داده مورد استفاده

۲.۱.۱ یک مدل ساده

در این پروژه می‌خواهیم با ساز و کار های موجود بین نورون های یک لایه آشنا شویم و تاثیر آن ها را بر فرآیند یادگیری بررسی کنیم. برای اینکار لازم است ابتدا، یک آزمایش ساده برای حالتی که هیچ یک از این ساز و کار ها درون لایه ها وجود ندارد انجام دهیم تا نتایج را بهتر باآن مقایسه کنیم. همانطور که در فایل پروژه نیز آمده است، در شبکه مورد نظر، تنها یک لایه ورودی و یک لایه خروجی در نظر میگیریم. سپس یک سیناپس با اتصال کامل و وزن های اولیه تصادفی بین این دو لایه ایجاد می‌کنیم. همچنین از قانون یادگیری انعطاف‌پذیری وابسته به زمان ضربه ($STDP$[۲]) برای آموزش شبکه استفاده می‌کنیم. از آنجا که این آزمایش به عنوان یک مرجع برای مقایسه در نظر گرفته می‌شود، رفتار دندریت ها را نیز یک رفتار ساده در نظر میگیریم. در نهایت، شبیه سازی را برای ۱۰۰۰ تکرار انجام می‌دهیم. همانطور که پروژه قبل نیز ملاحظه کردیم، قانون یادگیری $STDP$ به تنهایی نمی‌تواند در یادگیری الگو ها به خوبی $RSTDP$ عمل کند. هر چند که این قانون، مطابق شکل ۲.۱ توانسته است الگو های ورودی را به خوبی یاد بگیرد، اما این یادگیری پایدار نیست و به ازای تکرار ها مختلف ممکن است به طرق متفاوت عمل کند. (شکل ۳.۱)

[۱] مخزن تصاویر دانشگاه واترلو

[۲] Plasticity Spike-Timing-Dependent

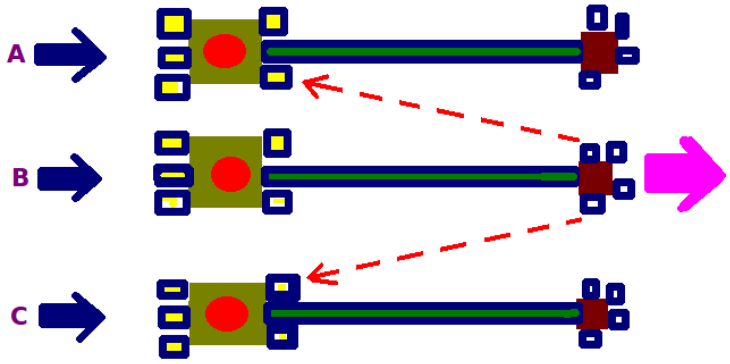


شکل ۲.۱: **یک شبکه ساده با قانون یادگیری $STDP$ ساده و بدون ساز و کار اضافه.** برای درک بهتر و تحلیل ساز و کار های درون یک لایه، بهتر است یک شبکه ساده در نظر گرفته و رفتار آن را با شبکه هایی که ساز و کار به آن ها اضافه می‌شود مقایسه کنم. همانطور که در پروژه قبل دیدیم، قانون یادگیری $STDP$ می‌تواند در یادگیری الگو های ساده، تا حد قابل قبولی عمل کند. هر چند این عملکرد ممکن است پایدار نباشد.(شکل ۳.۱)

شکل ۳.۱: **مقایسه نتایج شبیه سازی های مختلف شبکه تنها با قانون یادگیری $STDP$.** همانطور که در شکل بالا نیز مشاهده می‌کنیم، اجرای شبیه سازی یک شبکه یکسان که تنها قانون یادگیری $STDP$ را دارد، می‌تواند منجر به نتایج متفاوت شود. در شکل بالا، در شبیه سازی اول، شبکه نتوانسته است به خوبی الگو ها را یاد بگیرد، اما شبیه سازی های دوم و سوم عملکرد خوبی داشته‌اند.

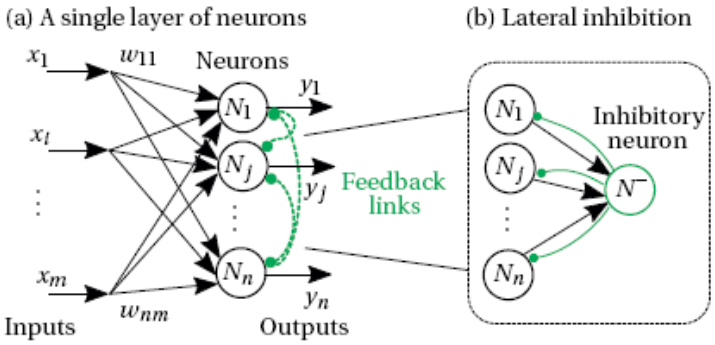
۲.۱ افزودن ساز و کار مهار جانبی (Lateral Inhibition)

در علوم اعصاب، مهار یا بازداری جانبی ظرفیت یک نورون برانگیخته برای کاهش فعالیت همسایگان خود است. مهار جانبی گسترش پتانسیل های عمل را از نورون های برانگیخته به نورون های همسایه در جهت جانبی غیرفعال می کند. این عمل یک کنتراست در تحریک ایجاد می کند که باعث افزایش ادراک حسی می شود و به آن تضاد جانبی نیز گفته می شود و عمدتاً در فرآیندهای بینایی، بلکه در پردازش لمسی، شنوایی و حتی بویایی نیز رخ می دهد. [۴]



شکل ۴.۱: محرکی که بر هر سه نورون تأثیر می‌گذارد، اما بر B قوی‌ترین یا اول تأثیر می‌گذارد، اگر B سیگنال‌های جانبی را به همسایگان A و C ارسال کند تا ضربه نزنند، در نتیجه آنها را مهار می‌کند. مهار جانبی در بینایی برای $sharp$ کردن سیگنال‌های مغز (فلش صورتی) استفاده می‌شود.

برای شبیه سازی ساز و کار مهار جانبی، ممکن است بتوان به چندین روش عمل کرد. از آنجا که می‌خواهیم یک نورون برانگیخته همسایگان خود را مهار کند، می‌توان اینکار را از طریق ایجاد یک جریان منفی از سمت این نورون به نورون های همسایه یا پایین آوردن اختلاف پتانسیل آن ها این کار را انجام داد. روشی که درون این پروژه استفاده شده است، همان روشی است که در کتابخانه $CoNeX$ نیز استفاده شده است. شمای کلی این روش، در شکل ۵.۱ که از [۵] برداشته شده، آمده است.

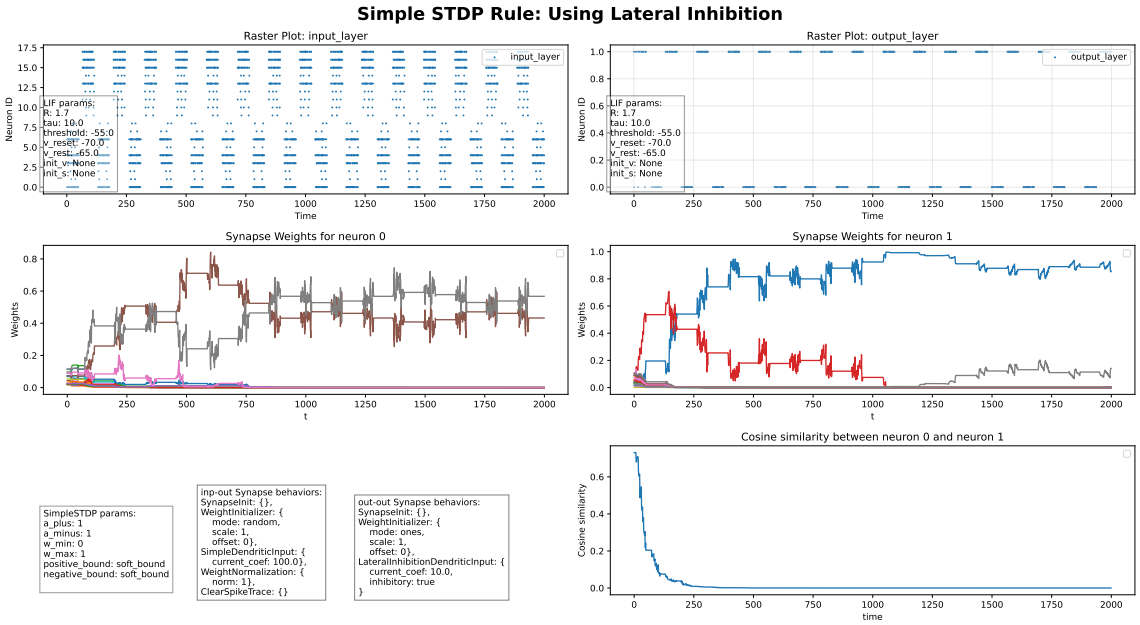


شکل ۵.۱: **یک شبکه SNN تک لایه با پیوندهای بازخورد** (الف) نورون های تحریکی $N1...Nn$ با پیوندهای بازخوردی در ارتباط هستند که فعالیت های واگرا را در نورون ها تقویت می کند. (ب) پیوندهای بازخورد اغلب ساز و کار مهار جانبی را اجرا می کنند، که در آن یک نورون بازدارنده اضافی N- نورون های تحریکی را مهار می کند. (رجوع کنید به [۵])

حال که با نحوه ساز و کار مهار جانبی آشنا شدیم، آن را به مدلمان افزوده و آزمایش می‌کنیم. برای اینکار لازم است که یک سیناپس جدید، از لایه خروجی به خودش تشکیل و دهیم، و به آن رفتار مهار جانبی را اضافه کنیم.

همانطور که در شکل ۶.۱ نیز ملاحظه می‌کنیم افزودن ساز و کار مهار جانبی به لایه خروجی باعث می‌شود فرایند یادگیری مدل بهبود یابد و نه تنها در مراحل زودتری، بتواند الگو ها را تشخیص دهد، بلکه با تکرار شبیه سازی، با احتمال بیشتری دقت پایداری را حفظ کند. همچنین

همانطور که ملاحظه می‌کنیم نمودار شباهت کسینوسی آن به سمت صفر میل می‌کند.



شکل ۶.۱: **افزودن ساز و کار مهار جانبی به لایه خروجی.** همانطور که در شکل بالا مشاهده می‌کنیم، افزودن ساز و کار مهار جانبی به لایه خروجی، باعث بهبود فرایند یادگیری مدل می‌شود. به طوری که مدل هم در مراحل زودتری الگو ها را تشخیص می‌دهد، و هم با تکرار آزمایش، با احتمال بیشتری نسبت به حالتی که این ساز و کار وجود ندارد پایداری خود را حفظ می‌کند. نکته مهم دیگری که در این شکل مشاهده می‌شود، نمودار شباهت کسینوسی بین وزن های لایه خروجی است. این نمودار در حالتی که این ساز و کار وجود نداشت، پس از اینکه تا حدی کم می‌شد، نزدیک صفر نوسان میکرد، اما با افزودن ساز و کار مهار جانبی، شباهت کسینوسی بعد از حدود ۲۵۰ تکرار به مقدار صفر میل می‌کند.

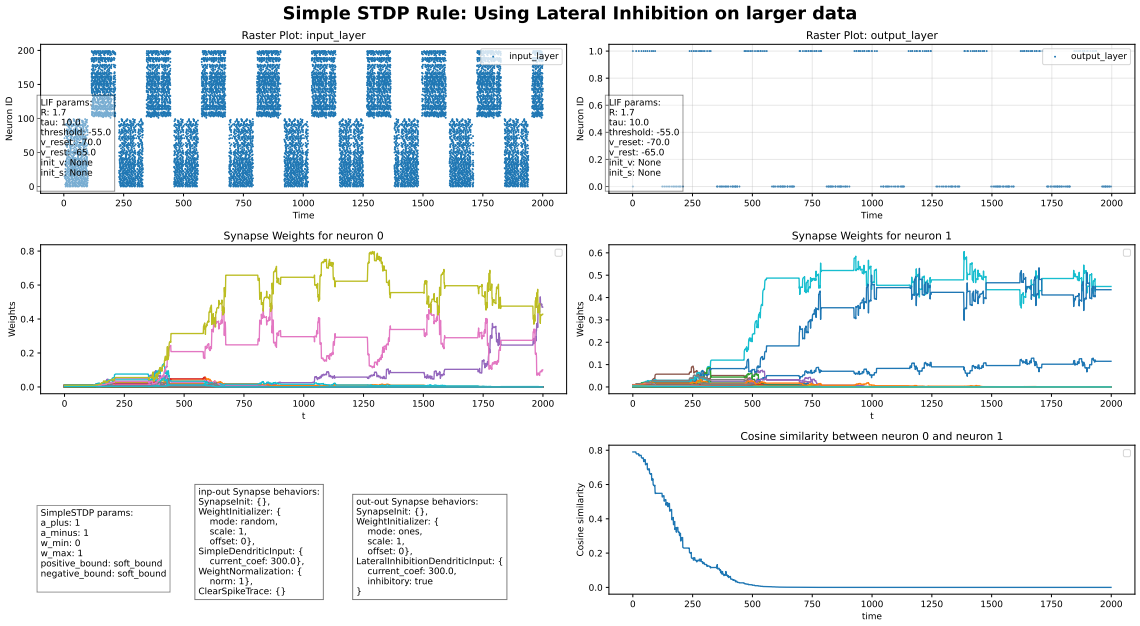
تکرار آزمایش با الگوی پیچیده تر نیز (مجموعه داده دانشگاه واترلو) نتایج مشابهی به همراه خواهد داشت و مدل توانایی تشخیص الگو ها را همچنان دارد.(شکل ۷.۱)

نتیجه گیری

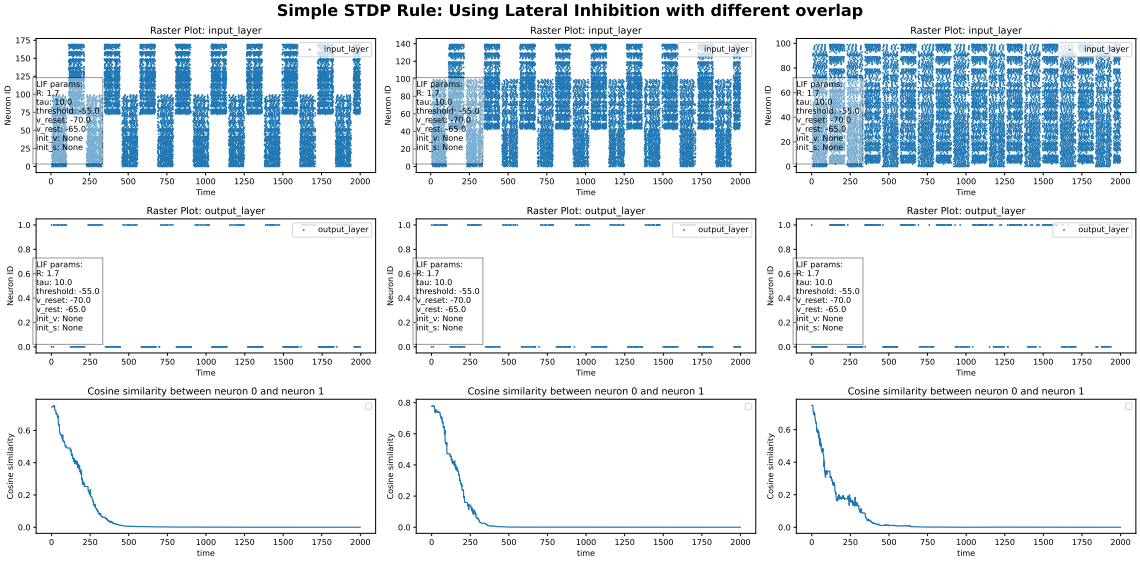
افزودن ساز و کار مهار جانبی به لایه خروجی، باعث می‌شود توانایی مدل در تشخیص الگو ها بیشتر شود و نورون ها به الگو های متفاوت حساس تر شوند. در ساز و کار مهار جانبی، برانگیخته شدن یک نورون، باعث می‌شود همسایه های خود را مهار کند. از این رو، افزودن این ساز و کار به لایه خروجی، باعث می‌شود هنگامی که یک نورون به یک الگو حساس می‌شود، نورون دیگر را مهار کرده و در نتیجه، تشخیص الگو ها بهتر شود.

۱.۲.۱ آزمایش مدل با اشتراک متفاوت الگو ها

در این قسمت، آزمایش را با میزان اشتراک متفاوت الگو ها تکرار می‌کنیم. برای اینکار، سه مقدار متفاوت 20%، 50% و 100% را بین دو الگو در نظر گرفته و برای هر کدام شبیه سازی را انجام می‌دهیم. مطابق شکل ۸.۱ مشاهده می‌کنیم که با افزودن میزان اشتراک بین الگو های ورودی نیز همچنان مدل توانایی تشخیص دادن الگو ها را دارد. هر چند در مدلی که میزان اشتراک به حداکثر خود رسیده است، مشاهده می‌کنیم که مدل دیرتر الگو ها را تشخیص می‌دهد و همچنین نمودار شباهت کسینوسی آن نیز دیر به سمت صفر میل می‌کند.



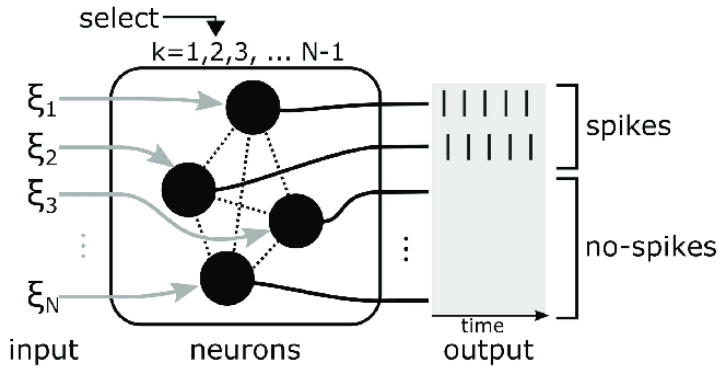
شکل ۷.۱: **افزودن ساز و کار مهار جانبی به لایه خروجی: الگو های های پیچیده تر.** مشاهده می‌کنیم حتی با شبیه سازی آزمایش با الگو های پیچیده تر، یعنی تصاویر مجموعه داده دانشگاه واترلو نیز، مدل توانایی خود در تشخیص دادن را از دست نداده است.



شکل ۸.۱: **ساز و کار مهار جانبی در لایه خروجی: آزمایش میزان اشتراک متفاوت الگوها** مشاهده می‌کنیم حتی با شبیه سازی آزمایش با میزان اشتراک متفاوت الگو هانیز مدل توانایی خود در تشخیص دادن را از دست نداده است. هرچند در شکل سمت راست که مدل حداکثر اشتراک را در الگو ها دارد، تشخیص دادن در مراحل دیرتری اتفاق می‌افتد، اما در نهایت موفق به تشخیص الگو ها می‌شود. علاوه بر آن، نمودار شباهت کسینوسی نیز دیر به سمت صفر میل می‌کند. در این مدل، بقیه پارامتر ها همانند شکل ۷.۱ می‌باشد.

۳.۱ k ـ برنده همه چیز را می‌گیرند(K-Winners-Take-All)

ساز و کار K-winners-take-all (KWTA) در شبکه‌های عصبی ضربه‌ای (یا حتی مصنوعی) شکلی از یادگیری رقابتی است که به تعیین اینکه کدام نورون‌ها در یک شبکه به شدت در پاسخ به ورودی‌های خاص فعال می‌شوند کمک می‌کند. در زمینه شبکه‌های عصبی ضربه‌ای، که در آن سعی می‌کنیم از نحوه عملکرد نورون‌های واقعی الهام بگیریم، ساز و کار KWTA با اجازه دادن به نورون‌ها برای رقابت بر اساس پتانسیل‌های غشایی یا سرعت ضربه زدن عمل می‌کند. می‌دانیم هر نورون در یک شبکه جریان را دریافت می کند و این جریان ورودی بر پتانسیل آنها تأثیر می گذارد. نورون هایی که به آستانه خاصی می رسند ضربه می‌زنند. از این رو در پیاده سازی KWTA، از بین تمام نورون‌های یک لایه یا گروه خاص، فقط k نورونی که پتانسیل‌شان بالاتر است (یا آن‌هایی که به شدت افزایش یافته اند) مجاز به فعال ماندن یا ”برنده شدن“ هستند. بقیه سرکوب می شوند یا فعالیت آنها کاهش می یابد، از این رو اصطلاح ”برنده ها همه چیز را می گیرند“ برای آن به کار برده می‌شود. ساز و کار KWTA معمولاً از طریق روش‌های مختلفی مانند بازخورد مهاری، که در آن نورون‌ها فعالیت نورون‌های دیگر را در مجاورت خود مهار می‌کنند، یا از طریق طراحی شبکه که خود معماری تضمین می‌کند که فقط k نورون برتر می‌توانند در هر نقطه فعال باقی بمانند، انجام می‌شود. زمان. این رویکرد را می توان به عنوان شکلی از رقابت بین نورون ها دید که یک ویژگی اساسی سیستم های عصبی بیولوژیکی است و برای کارهایی مانند تشخیص الگو و فرآیندهای تصمیم گیری در هر دو سیستم طبیعی و مصنوعی بسیار مهم است. خلاصه این مفهوم در شکل ۹.۱ آمده است.



شکل ۹.۱: **ساز و کار « k ـ برنده همه چیز را می‌گیرند».** در هر مرحله از شبیه سازی، فقط آن نورون‌هایی که بزرگترین ورودی را دریافت می‌کنند، ضربه می‌زنند، در حالی که ضربه های همه نورون‌های دیگر به صورت پویا سرکوب می‌شوند.

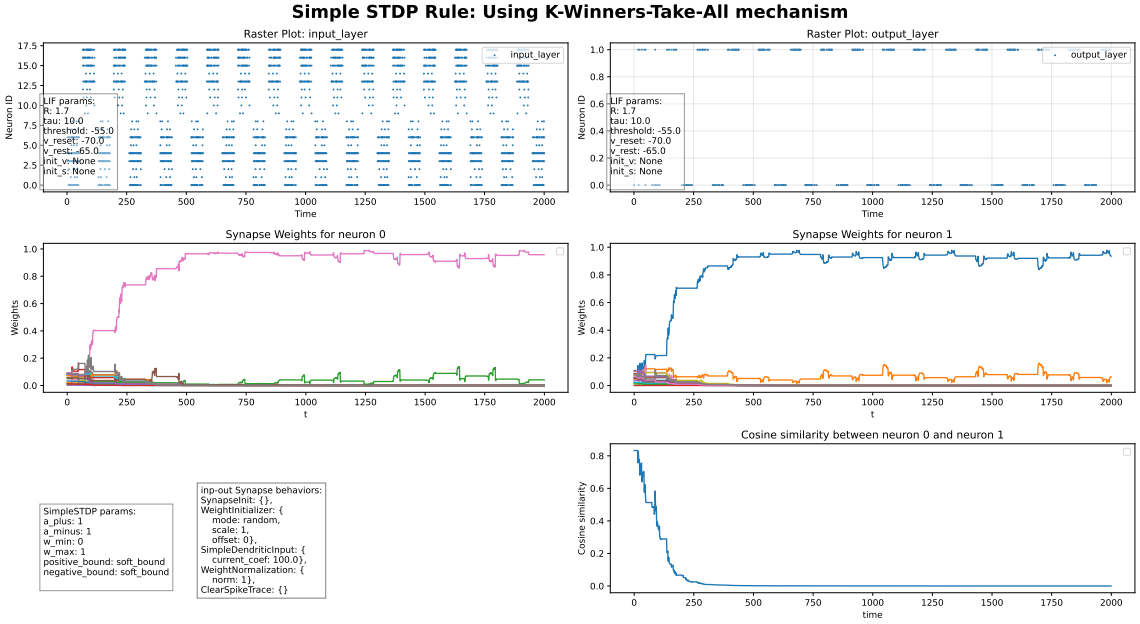
حال می‌خواهیم این ساز و کار را به شبکه‌مان اضافه کنیم. ابتدا، حالتی را در نظر می‌گیریم که در آن شبکه، از یک لایه ورودی و خروجی و یک سیناپس از لایه ورودی به لایه خروجی استفاده می‌کند. همچنین مشابه قسمت قبل، از قانون یادگیری $STDP$ برای آموزش مدل استفاده می‌کنیم. همانطورکه در شکل ۱۰.۱ نیز ملاحظه می‌کنیم، افزودن ساز و کار k-winners-take-all به لایه خروجی باعث می‌شود در همان مراحل اولیه، مدل الگو ها را یاد بگیرد. هر چند در این حالت نیز ممکن است در برخی شبیه سازی ها مدل نتواند الگو ها ها را به خوبی یاد بگیرد، اما قرار گرفتن این ساز و کار در کنار ساز و کار های دیگر که بررسی کردیم یا بررسی خواهیم کرد، توانایی مدل را افزایش خواهد داد. همانطور که از شکل برمی‌آید، اضافه کردن این ساز و کار باعث شده است تمایز وزن های بین نورون ها را شاهد باشیم و همچنین شباهت کسینوسی نسبت به حالت بدون ساز و کار یا با ساز و کار مهار جانبی، زودتر به صفر میل کند.

حال به لایه خروجی، ساز و کار مهار جانبی که در بخش قبل انجام دادیم را نیز اضافه می‌کنیم تا رفتار مدل را در حضور هر دو ساز و کار بررسی کنیم. مطابق شکل ۱۱.۱ مشاهده می‌کنیم که با داشتن هر دو ساز و کار k-winners-take-all و مهار جانبی، مدل به خوبی توانسته است دو الگو را تشخیص دهد. مطابق شکل، دریافت می‌شود که در مراحل اولیه، مدل در یادگیری الگو ها دچار مشکل بوده است با این حال، توانسته است پس از مراحلی، الگو ها را یاد بگیرد. در حالی که هنگامی که این شرایط برای مدل های دیگر پیش می‌آید، مدل حتی پس از طی کردن مراحل بیشتر شبیه سازی نیز موفق به یادگیری الگو ها نمی‌شود.

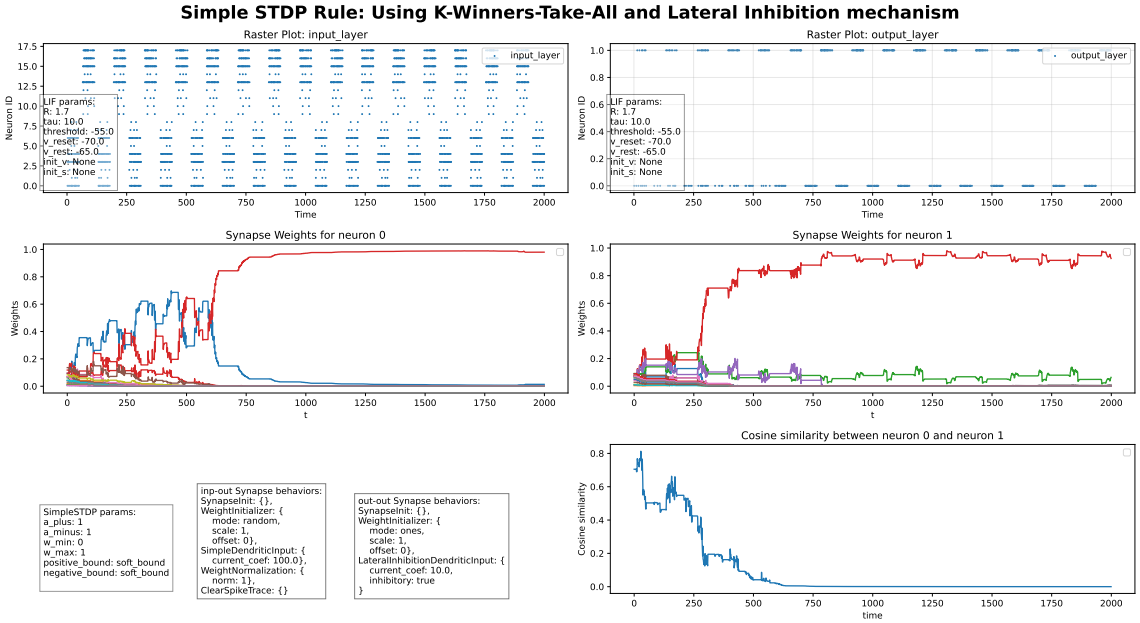
حال مدل را برای الگو های پیچیده تر نیز آزمایش می‌کنیم. مطابق شکل ۱۲.۱ مشاهده می‌کنیم که مدل از عهده تشخیص الگو های پیچیده‌تر نیز برمی‌آید. مجددا در این حالت نیز جداسازی وزن ها به خوبی انجام می‌شود و شباهت کسینوسی نیز به صفر میل می‌کند. همچنین مدل پایدار است و با تکرار آزمایش نیز می‌تواند الگو ها را تشخیص دهد.

۱.۳.۱ آزمایش مدل با اشتراک متفاوت الگو ها

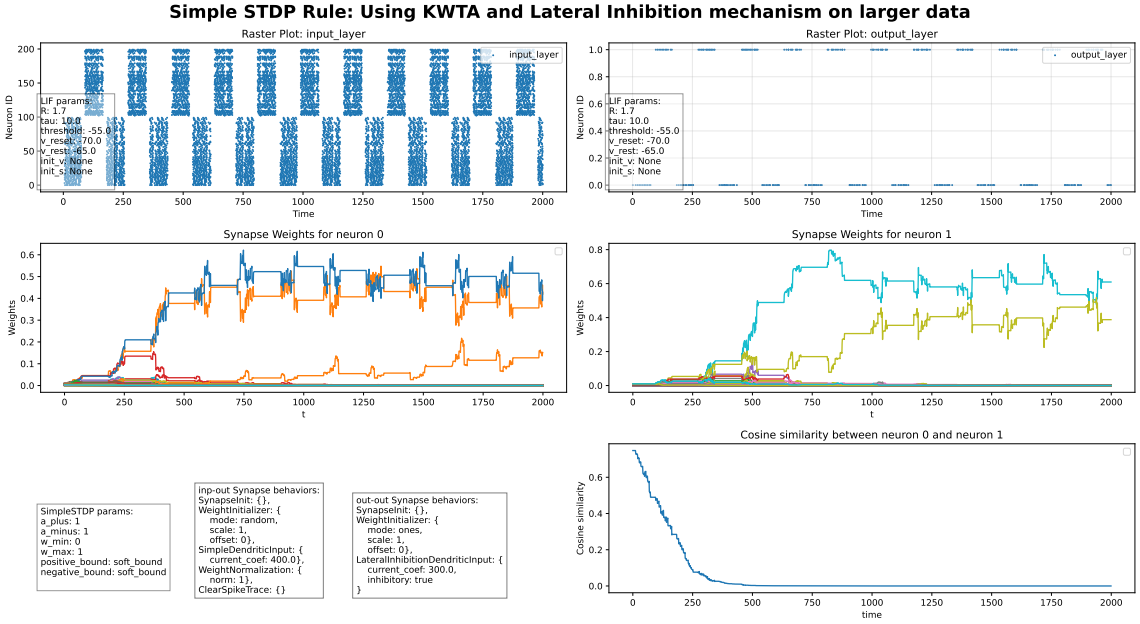
حال مشابه قسمت قبل، آزمایش های بالا را برای الگو ها با میزان اشتراک متفاوت انجام می‌دهیم. انتظار داریم که عملکرد شبکه‌ای که هر دو ساز و کار را دارد، بهترین عملکرد را داشته باشد. شکل ۱۳.۱ این موضوع را تایید می‌کند.



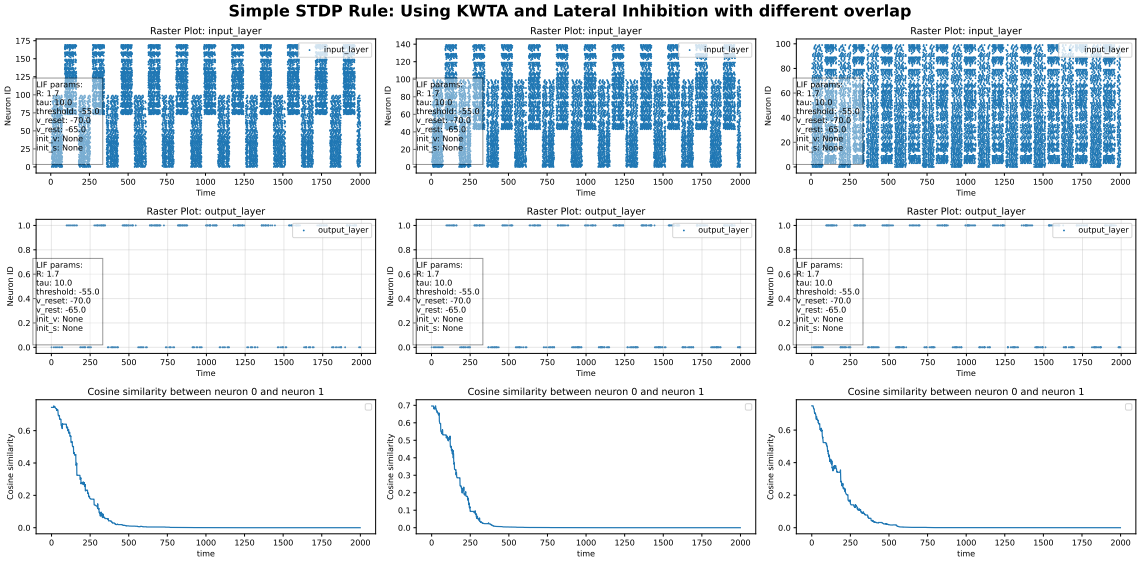
شکل ۱۰.۱: **اضافه کردن ساز و کار k-winners-take-all به تنهایی در لایه خروجی یک شبکه ساده.** مشاهده می‌کنیم که شبکه توانسته است به خوبی الگو های ورودی را یاد بگیرد. نسبت به حالتی که ساز و کاری وجود نداشت یا حالتی که ساز و کار مهار جانبی داشتیم، یاد گرفتن الگو ها زودتر اتفاق افتاده و همچنین شباهت کسینوسی زودتر به سمت صفر میل کرده است.



شکل ۱۱.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی.** مشاهده می‌کنیم که با داشتن هر دو ساز و کار k-winners-take-all و مهار جانبی، مدل به خوبی توانسته است دو الگو را تشخیص دهد. مطابق شکل، دریافت می‌شود که در مراحل اولیه، مدل در یادگیری الگو ها دچار مشکل بوده است با این حال، توانسته است پس از مراحلی، الگو ها را یاد بگیرد. در حالی که هنگامی که این شرایط برای مدل های دیگر پیش می‌آید، مدل حتی پس از طی کردن مراحل بیشتر شبیه سازی نیز موفق به یادگیری الگو ها نمی‌شود. همچنین پس از طی مراحلی، وزن ها به خوبی نسبت به الگو های ورودی تنظیم شده اند و مشاهده می‌کنیم که شباهت کسینوسی نیز به صفر میل کرده است. نکته مهمی که در این مدل وجود دارد این است که پایدار است و هر چقدر آن را شبیه سازی کنیم، در تشخیص الگو ها موفق است.



شکل ۱۲.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی: داده های پیچیده‌تر.** مطابق شکل ملاحظه می‌کنیم مدل از عهده تشخیص الگو های پیچیده‌تر نیز برمی‌آید. در این حالت نیز وزن ها به خوبی تنظیم می‌شوند و شباهت کسینوسی نیز به صفر میل می‌کند. نکته مهم اینکه مدل پایدار است و با تکرار آزمایش نیز می‌تواند الگو ها را تشخیص دهد.



شکل ۱۳.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی:میزان اشتراک متفاوت.** همانطور که انتظار داشتیم، شبکه‌ای که هر دو ساز و کار را دارد، نسبت به انواع میزان اشتراک الگو ها پایدار است و در هر سه حالت توانسته است الگو ها را تشخیص دهد. هر چند هنگامی که آزمایش می‌شد، برای اشتراک با میزان کم یا متوسط، همیشه مدل درست تشخیص می‌داد، ولی هنگامی که حداکثر اشتراک را داشتیم، در بعضی شبیه سازی ها مدل نمی‌توانست به درستی دو الگو را تشخیص دهد.

نتیجه گیری

مطابق آزمایش هایی که انجام دادیم، مشاهده کردیم که اضافه کردن ساز و کار k-winners-take-all به یک شبکه ساده، می‌تواند همانند ساز و کار مهار جانبی، قدرت مدل را در تشخیص الگو ها بهبود ببخشد. هر چند تاثیر آن اندکی بهتر از ساز و کار مهار جانبی است چرا که پایداری

مدل را نسبت به تکرار آزمایش افزایش می‌داد. اما در کل تاثیری که هر دوی این ساز و کار ها روی رفتار مدل به تنهایی می‌گذارند یکسان است. ما در مدل هایمان به این علاقه داریم که هر نورون فقط هنگامی که ورودی مخصوص خود را می‌بیند ضربه بزند و در پنجره زمانی که ورودی دیگر داده می‌شود ضربه‌ای نزند. هر دوی این ساز و کار ها به نحوی اینکار را سهولت می‌بخشند. یکی با دادن جریان منفی به نورون دیگر، و دیگری با سرکوب کردن نورون دیگر.

در نهایت مشاهده کردیم که هر دوی این ساز و کار ها به تنهایی، تا حدودی نسبت به افزایش میزان اشتراک مقاوم هستند. هر چند اشتراک حداکثری ممکن است پایداری آن ها را کاهش دهد. بهترین عملکرد را هم زمانی شاهد بودیم که هر دوی این ساز و کار ها به لایه خروجی اضافه شدند و مدل نسبت به تکرار آزمایش یا افزایش اشتراک الگو ها پایداری نشان می‌داد. در بخش بعدی، ساز و کار دیگری به شبکه‌مان اضافه می‌کنیم.

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] Lateral inhibition. Wikipedia [Link]

[۵] Unsupervised Learning of Phase-Change-Based Neuromorphic Systems. Wozniak, Stanislaw Andrzej