**«به نام خدا»**

**گزارش پروژه پایانی درس شبکه های عصبی**

**اعضای گروه: زهرا ایوبی - 40114134**

**مرضیه حسینی – 40104904**

این گزارش شامل بخش های زیر است که با کلیک روی هر قسمت می توان بخش مربوطه را مشاهده نمود:

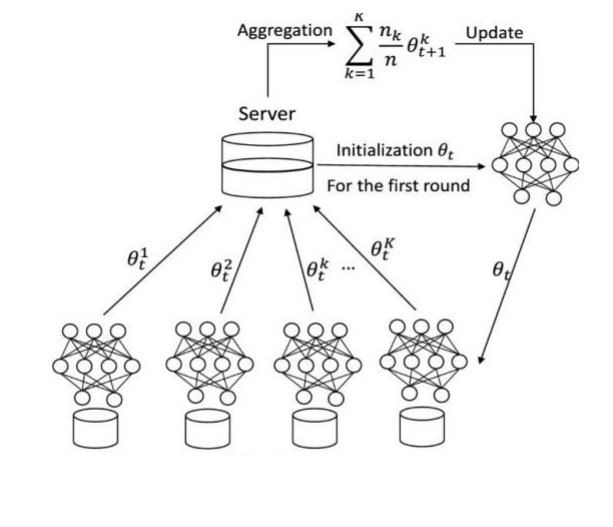
* [گزارش کد پیاده سازی شده](#_گزارش_کد_پیاده)
* [دیتاست](#_دیتاست)
  + [پیاده سازی کد سرور](#_پیاده_سازی_کد)
  + [پیاده سازی کد کلاینت](#_پیاده_سازی_کد_1)
* [گزارش نتایج آزمایش](#_گزارش_نتايج_آزمايش)
* [نتیجه گیری](#_نتيجه_گيري)

# [گزارش کد پیاده سازی شده](#_top)

در این بخش با توجه به توضیحات بیان شده در مقاله قصد داریم کد پیاده سازی شده را بررسی و شرح دهیم.

برای شبیه سازی فرآیند یادگیری فدرالی و نزدیك سازی آن به دنیای واقعی، یك process برای سرور و 10process مجزا نیز برای 10 کلاینت (تعداد ذکر شده در شرایط آزمایش مقاله برای کلاینتها) روی پورتهای جداگانه بالا می آوریم و ارتباطات بین سرور و کلاینتها بر بستر network انجام می گیرد. لازم به ذکر است که تمامی کلاینتها دارای ساختار مدل یکسان بیان شده در مقاله بوده و دیتای محلی هریك نیز با توجه به دو حالت مختلف آزمایش بیان شده در مقاله از دیتاست fashion-mnist در دو شرایط مختلف تولید و تست شد. در ادامه جزییات موارد بیان شده را مفصل بررسی می کنیم.

فرآیند کلی یادگیری فدرالی به صورت خلاصه در شکل زیر به صورت شماتیک نمایش داده شده است:



**فرآیند کلی یادگیری فدرالی به شرح زیر است:**

ابتدا سرورمقادیر اولیه پارامترهای شبکه(وزنها) را به صورت رندوم تولید کرده و برای تمام کلاینتها ارسال می کند. سپس هر کدام از این کلاینتها به صورت موازی فرآیند آموزش خود را بر روی دیتای محلی خود آغاز کرده و پس از آموزش مدلهای محلی، پارامترهای آموزش دیده را به سرور ارسال می کنند. با اینکار بدون نیاز به متمرکز سازی و جابجایی دیتا به صورت توزیع شده آموزش شبکه عصبی را انجام می دهیم. یادگیری فدرالی در مواردی که امنیت و حریم خصوصی داده ها اهمیت دارد و همچنین متمرکز سازی دیتا کار دشوار و پرچالشی باشد بسیار کاربرد دارد. پس از اینکه سرور پارامترهای آپدیت شده را از تمام کلاینتها دریافت کرد، به صورت وزندار و به نسبت تعداد نمونه های هرکلاینت به تعداد کل دیتای موجود در تمامی کلاینتها، پارامترهای آپدیت شده را تجمیع کرده و بعنوان پارامترهای اولیه راند بعدی آموزش به کلاینتها مجدد ارسال می کند. کلاینتها بعد از دریافت پارامترهای مدل تجمیع شده یا اصطلاحا مدل گلوبال سرور، آن را با دیتای تستی خود ارزیابی می کنند. این فرآیند به تعداد راندهای ماکسمیمی که در کد سرور تعیین شده ادامه می یابد. در ادامه ی گزارش، به تفصیل بخش های مختلف پروژه پیاده سازی شده یعنی اسکریپت تولید دیتاستهای محلی کلاینتها و کد پیاده سازی شده برای کلاینت و سرور را مورد بررسی قرار می دهیم.

## **[دیتاست](#_top)**

ابتدا یك توضیح اجمالی در رابطه با دیتاست استفاده شده در مقاله و منبعی که از آن استفاده کرده ایم می دهیم و سپس کد پیاده سازی شده برای تولید دو حالت مختلف درنظر گرفته شده در مقاله برای داده های محلی کلاینت ها را بررسی می کنیم.



Fashion-MNIST یک مجموعه داده از تصاویر محصولات Zalando است که شامل مجموعه آموزش با 60،000 نمونه و مجموعه تست با 10،000 نمونه است. هر نمونه یک تصویر خاکستری با ابعاد 28\*28 است که با یک برچسب از 10 کلاس ارتباط دارد. Zalando قصد دارد که Fashion-MNIST را به عنوان یک جایگزین مستقیم برای مجموعه داده اصلی MNIST برای ارزیابی الگوریتم های یادگیری ماشین استفاده کند.

مجموعه داده اصلی MNIST شامل نمونه هایی از ارقام دست‌نویس است. اعضای جامعه هوش مصنوعی/یادگیری ماشین/علوم داده این مجموعه داده را به عنوان یک معیار برای اعتبارسنجی الگوریتم‌های خود استفاده می‌کنند. در واقع، MNIST اغلب اولین مجموعه داده‌ای است که پژوهشگران امتحان می‌کنند. "اگر روی MNIST کار نکند، در کل کار نخواهد کرد"، آن‌ها می‌گویند. "حالا اگر روی MNIST کار کند، ممکن است هنوز هم بر روی دیگران شکست بخورد."

هر تصویر ارتفاع 28 پیکسل و عرض 28 پیکسل دارد، که مجموعاً 784 پیکسل است پس 784 فیچر ورودی داریم. مقدار هر پیکسل یک عدد صحیح بین 0 و 255 است. ستون نهایی دیتاست نیز شامل برچسب‌های کلاس است و نمایانگر نوع پوشاک است.

شرح برچسبها به شکل زیر است:

0 تی شرت/تاپ

1 شلوار

2 پولیور

3 لباس

4 کت

5 صندل

6 پیراهن

7 کفش ورزشی

8 کیسه

9 چکمه

این مجموعه داده را از kaggle به آدرس <https://www.kaggle.com/datasets/zalando-research/fashionmnist/download?datasetVersionNumber=4> دانلود کردیم.

مجموعه داده شامل دو فایل است:

fashion-mnist\_train.csv شامل 60000 نمونه آموزشی

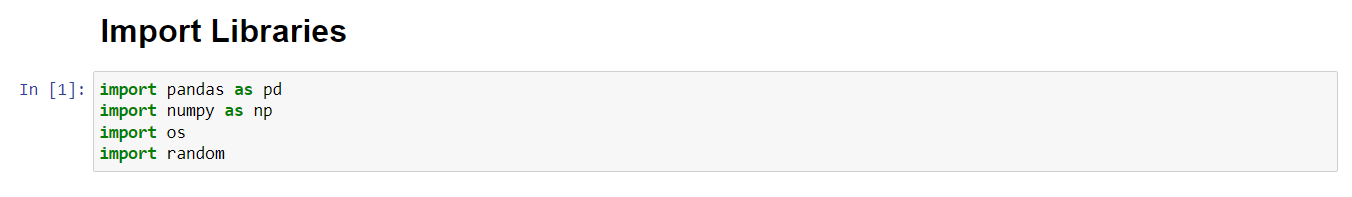
fashion-mnist\_test.csv شامل 10000 نمونه تست

ما این دو دیتاست را تجمیع کرده و سپس در کد تقسیم بندی دیتای تستی و آموزش را به نسبت 30/70 استفاده کرده ایم.

نکته: با توجه به اینکه حجم دیتا برای آموزش شبکه با 5 روش بهینه سازی بیان شده در مقاله برای حالات مختلف هایپرپارامترهای روش بهینه سازی بسیار زیاد و زمانبر بود ما برای آموزش و تست مقاله حجم دیتا را نصف کرده ایم.

متوسط زمان آموزش شبکه حتی با نصف کردن حجم دیتاست(35000 نمونه) تقریبا حدود 5 الی 6 ساعت برای هر روش برای هر مود دیتا بوده است!

با توجه به اینکه مقاله آزمایشات را برای دو حالت توزیع داده full-non-iid و 99%-non-iid انجام داده است برای تولید و شبیه سازی این دو توزیع داده برای هر کلاینت اسکریپتی پیاده سازی شده است که در ادامه به تفصیل مورد بررسی قرار می دهیم.

در این بخش کتابخانه های مورد نیاز را ایمپورت می کنیم:

در این بخش دیتاست خام اولیه برای نمونه های آموزشی و تست را تلفیق کرده و در فایل اکسلی ذخیره می نماییم:



قبل از توضیح کد بخش توزیع داده های محلی ابتدا در رابطه با دو نوع توزیع آزمایش شده در این مقاله توضیح مختصری می دهیم:

در این مقاله، تمرکز بر روی خصوصیات طبیعت non-iid داده‌ها در سناریوی یادگیری فدرالی بوده. به عبارت دیگر، داده‌های محلی به شکلی بسیار ناهمگن ایجاد شده‌اند.

دو توزیع Full-non-IID (Fully Non-Independent and Identically Distributed) و 99%-non-IID توزیع‌های داده‌های محلی که در آزمایشات امتحان شدند هستند.

1. Full-non-IID

در این حالت، داده‌ها به طور کاملاً ناهمگن (Non-Identically Distributed) و غیرمستقل (Non-Independent) بین کلاینتها توزیع شده‌اند. به عبارت دیگر، توزیع داده‌ها در هر نود کاملاً متفاوت است و این داده‌ها از یکدیگر مستقل نیستند. این موضوع می‌تواند ویژگی‌های چالش‌برانگیزی را برای یادگیری فدرالی ایجاد کند، زیرا مدل‌های آموزش دیده بر روی یک کلاینت ممکن است به سرعت در تطابق با داده‌های دیگر کلاینتها دچار مشکل شوند. این موضوع برای شبیه سازی دنیای واقعی درنظر گرفته میشود. به عبارت دیگر هر یک از این 10 کلاینت نمونه های یک کلاس خاص را در این حالت خواهد داشت.

2. 99%-non-IID

در این حالت 99% از داده‌ها به عنوان IID-non در نظر گرفته می‌شوند،. اما 1% باقی‌مانده از داده‌ها به صورت IID توزیع شده‌اند. داده‌های آموزش به نسبت 99%-1% به دو بخش تقسیم شده‌اند، که این بخش‌ها به صورت مستقل و به بهترین شکل ممکن توزیع شده‌اند. بخش 99% بر اساس کلاس‌ها مرتب شده و سپس هر کلاس به یکی از نودها اختصاص داده شده است. بخش 1% به 10 بخش مساوی تقسیم شده و سپس به داده‌های مختص نودهای خاص اضافه شده است.

پس از بیان نکات مهم در ادامه بخش تولید دیتاستهای توزیع Full-non-IID را مشاهده می کنیم:

نکات مهم به صورت کامنت در کد قرار گرفته شده است:

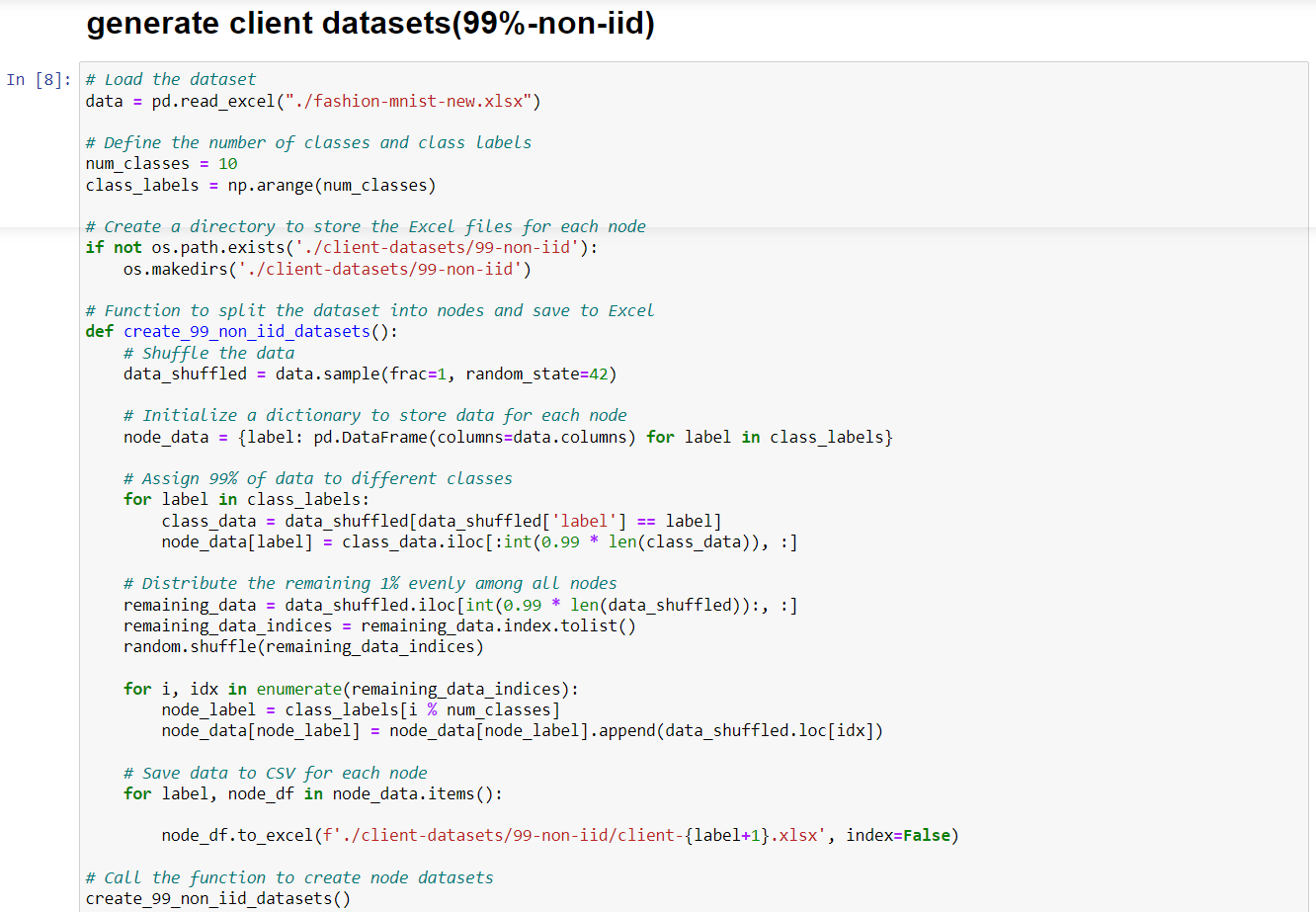


این تابع ابتدا داده‌های مربوط به کلاس مشخص شده را از دیتاست استخراج می‌کند. سپس نیمی از داده‌ها (بعلت محدودیت سخت افزاری) را انتخاب می‌کند و داده‌ها را به فایل اکسل مربوط به نود مربوطه ذخیره می‌کند.

سپس داده‌های همه نودها در یک دیتاست جدید ترکیب کرده و در یک فایل اکسل جدید ذخیره می‌شوند تا برای توزیع 99٪ استفاده شوند.

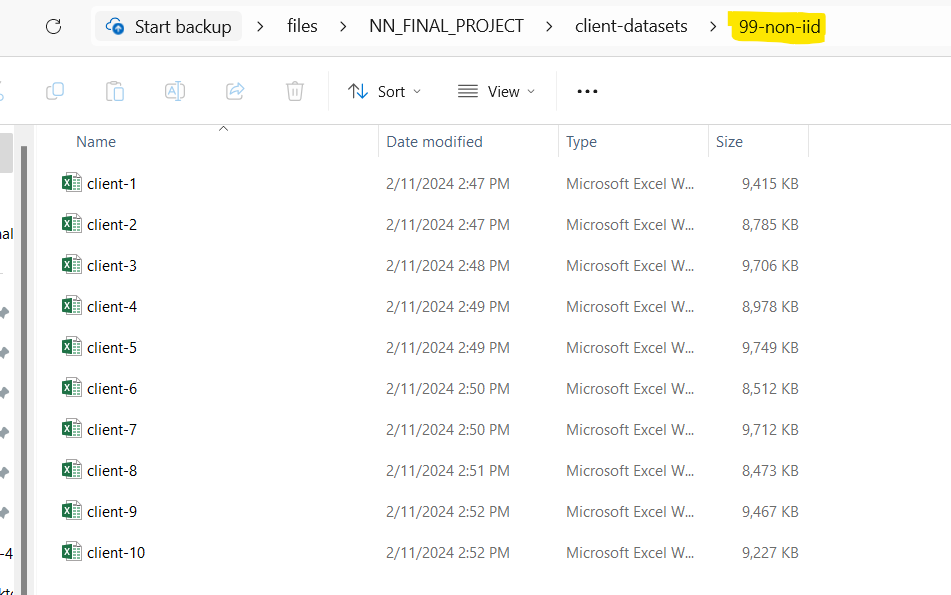
در بخش بعدی تولید داده های محلی کلاینتها با توزیع 99%-non-iid را مشاهده می کنیم:

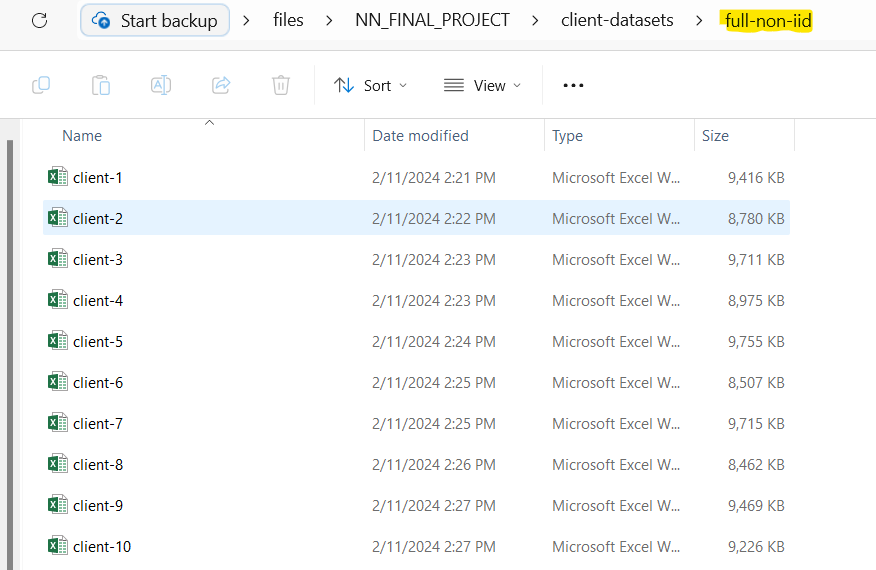
در این بخش ابتدا دیتاستی که با سایز 35000(نصف کل نمونه های داده های خام)در استپ قبلی ساخته شده بود را می‌خوانیم. سپس در تابعی داده‌ها را به صورت تصادفی شافل کرده و 99٪ از داده‌ها به طور مجزا به هر کلاس اختصاص می‌یابد و نهایتا داده‌های باقی‌مانده (1%) به صورت یکنواخت بین همه نودها توزیع می‌شود. پس در این گام دیتاست‌های مربوط به هر نود با توزیع 99%-non-iid ایجاد شده و در فایل‌های اکسل جداگانه ذخیره می شوند.



با اجرای این اسکریپت خروجی به صورت زیر خواهد بود:



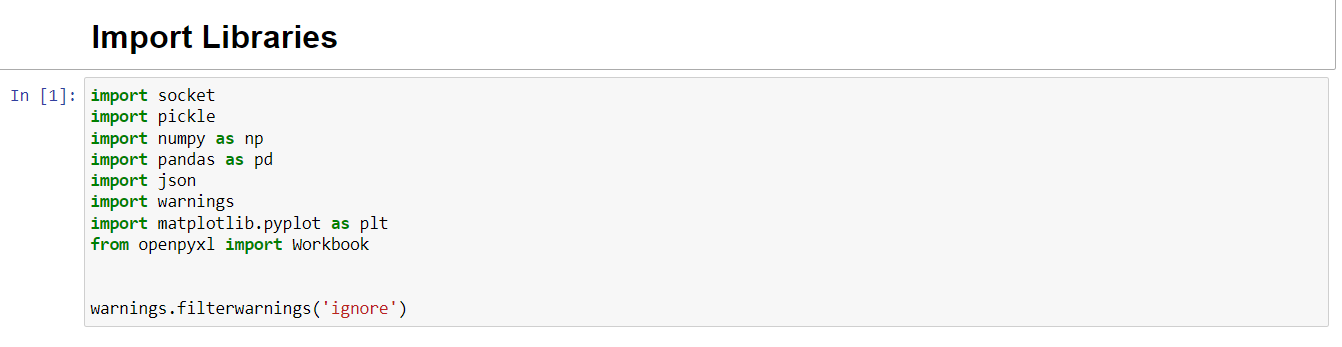




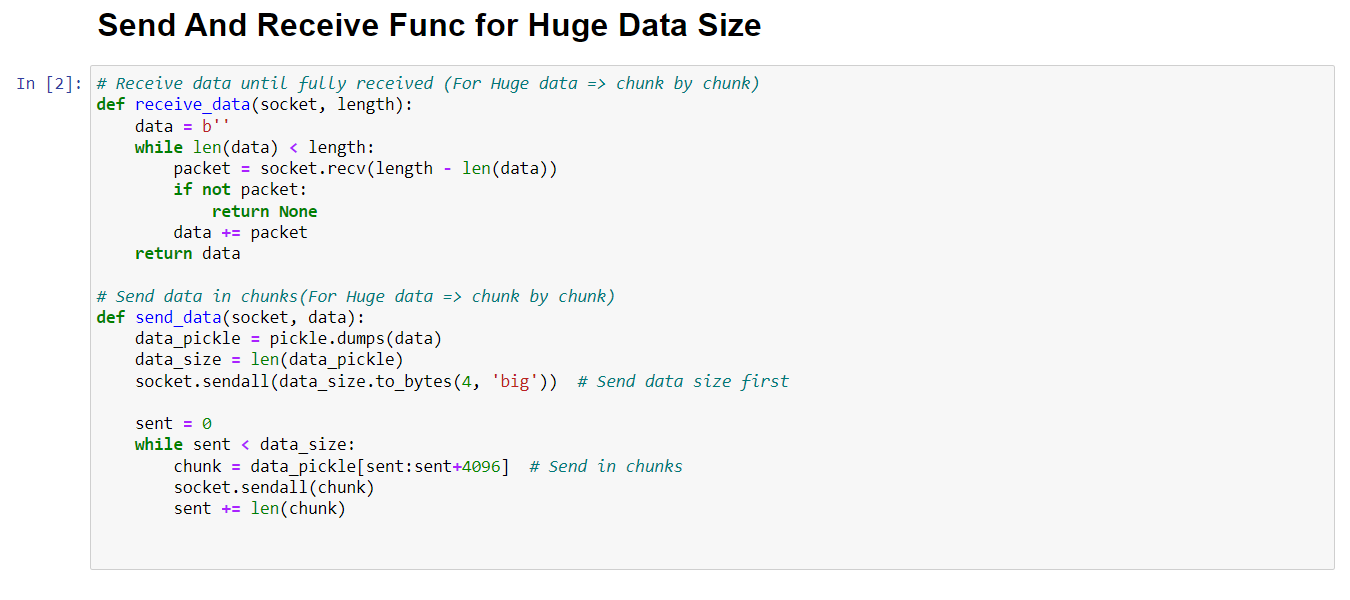
پس از تولید دیتاستهای محلی کلاینت ها برای دو توزیع حالا سراغ بررسی کد پیاده سازی شده سمت کلاینت ها و سرور می رویم.

## **[پیاده سازی کد سرور](#_top)**

در این قسمت کتابخانه های لازم را ایمپورت کرده ایم:



در قسمت بعدی، برای هندل کردن ارسال و دریافت داده های حجیم (مانند پارامترهای مدل) بین سرور و کلاینت فانکشن زیر پیاده سازی شده اند:

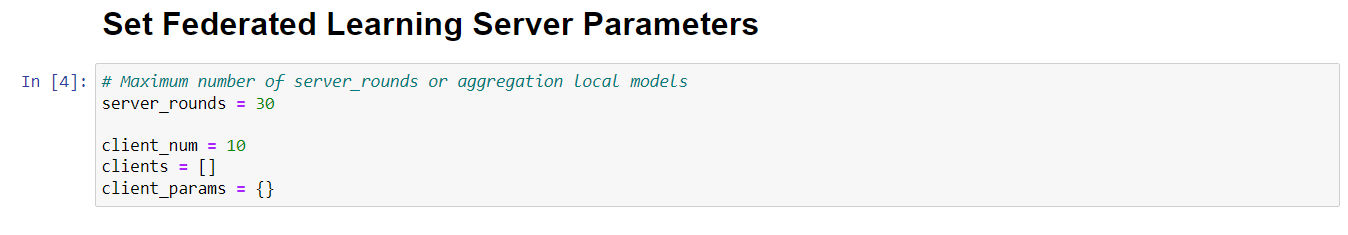


در قسمت زیر، سوکت سرور را برای ارتباطات بین سرور و کلاینت ها کانفیگ می کنیم:



در بخش زیر پارامترهای اولیه مانند تعداد راندهای فرآیند یادگیری فدرالی، به عبارت دیگر تعداد دفعات aggregation ئه

پارامترهای محلی آموزش دیده در کلاینت ها و تعداد کلاینتهای شرکت کننده را تنظیم می کنیم:

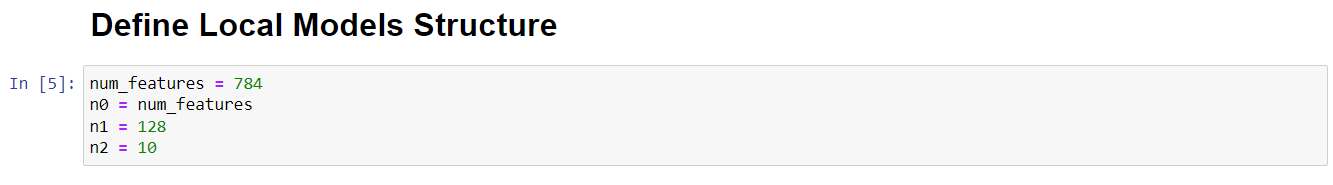


برای تعیین سایز پارامترهای اولیه مدلهای محلی، به ساختار مدل کلاینت ها نیاز داریم.

ساختار کلاینتها، تعداد لایه ها و نورونهای هرلایه به شرح زیر است:

با توجه به توضیحات مقاله شبکه استفاده شده برای آموزش یک شبکه MLP ساده با یک لایه مخفی میانی شامل 128 نورون می باشد:

تعداد نورونهای لایه ورودی هم به تعداد فیچرهای ورودی یعنی 784 تا و تعداد نورونهای لایه خروجی نیز به تعداد کلاسهای دیتاست یعنی 10 است.



در بخش بعدی پارامترهای سمت کلاینت توسط سرور تعیین می شوند. پارامترهای تعیین شده به شرح زیر هستند:

* پارامترهای وزن برای ارسال به کلاینتها به صورت تصادفی مقدار دهی اولیه می شوند.
* تعداد اپوک های آموزشی در هر راند برای هر کلاینت
* الگوریتم بهینه سازی در فرآیند آموزش کلاینت
* کانفیگ و حالات مختلف هایپرپارامترهای تست شده برای هر روش بهینه سازی در مقاله (این کانفیگ به صورت یک فایل جیسون در root پروژه با نام optimization\_params.json قرار گرفته شده است)
* کانفیگ لازم برای رسم پلات دقت بدست آمده برای حالات مختلف هر روش که در مقاله گزارش شده است(برای مشابه سازی ظاهر نمودار ها مانند عناوین و رنگ هر نمودار این کانفیگ تهیه شده است)
* سایر پارامترها مانند نوع توزیع دیتاهای محلی (full یا 99٪)- سایز mini-batch ها و مقدار ترم رگولاریزیشن(برای جلوگیری از overfit)



در کد بالا مقادیر ست شده برای این پارامترها را مشاهده می کنیم.

همچنین یک آرایه دوبعدی نیز برای ذخیره سازی دقت هر یک از حالات مختلف هایپرپارامترهای پیشنهادی برای هر روش بهینه سازی در مقاله تعریف می کنیم که دقت آن ها را در هر راند ذخیره کنیم.

بخشی از فایل کانفیگ optimization\_params.json:

پس از ست کردن کانفیگ ها و پارامترهای اولیه از اینجا به بعد فرآیند یادگیری فدرالی شامل آموزش کلاینتها بر روی داده های محلی خود و ارسال نتایج به سرور و تجمیع آن در سرور را مشاهده خواهیم کرد.

در بخش زیر، به محض متصل شدن هر یك از کلاینتها، پارامترهای اولیه به کلاینت متصل شده ارسال می شود.



در بخش زیر، فرآیند تکراری آموزش یادگیری فدرالی آغاز شده و این فرآیند به تعداد 30 راند(عدد گزارش شده در مقاله) تکرار می شود.

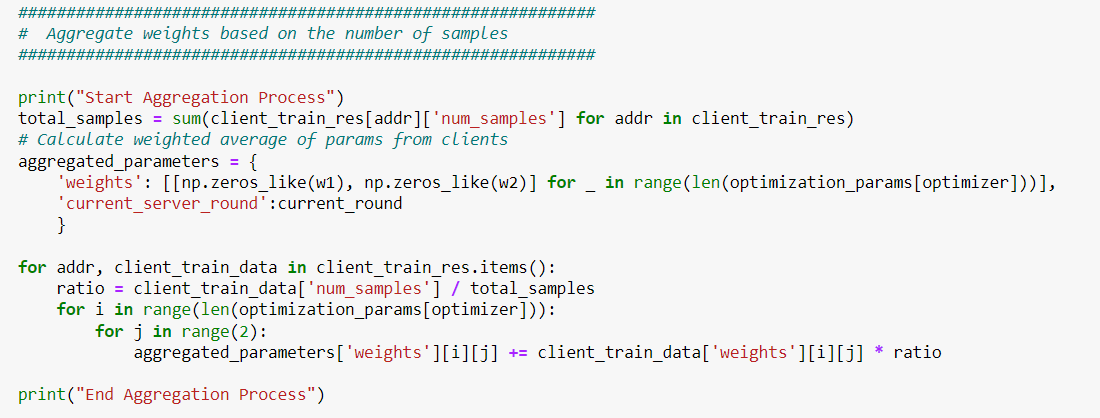
ابتدا با ارسال سیگنال ready به کلاینتها از آماده بودن آنها برای شروع راند جدید آموزش مطمئن شده و دستور آغاز فرآیند

آموزش را به آنها می دهیم. آموزش محلی روی هر کلاینت به صورت موازی انجام شده و منتظر دریافت بروزرسانی های تمام

کلاینت ها می مانیم.



پس از دریافت بروزرسانی های محلی هر یك از کلاینتها، نوبت به فاز تجمیع وزندار پارامترها به صورت زیر است:



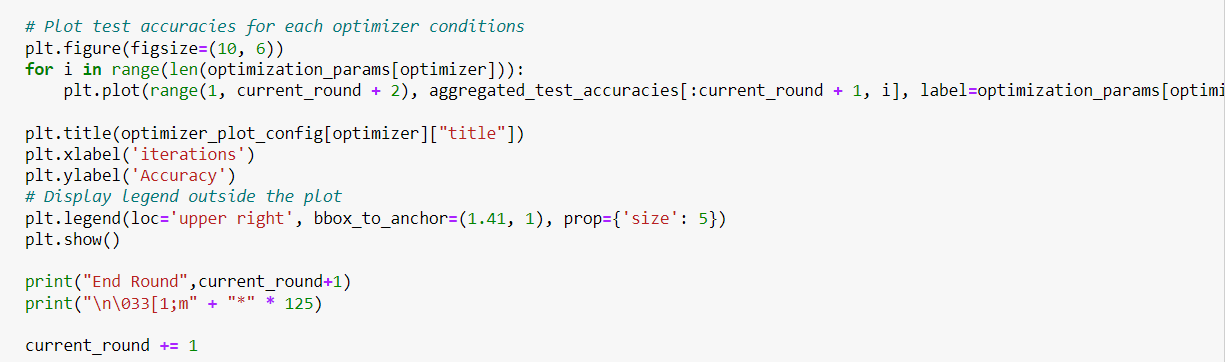
نکته: پارامترهای آغازین راند بعدی آموزش در کلاینتها همین مدل تجمیع شده جدید است.

پس از تجمیع بروزرسانی های پارامترها، مجدد مدل تجمیع شده و اصطلاحا مدل گلوبال را به کلاینتها ارسال کرده و آن را با داده های تستی کلاینتها ارزیابی می کنیم.

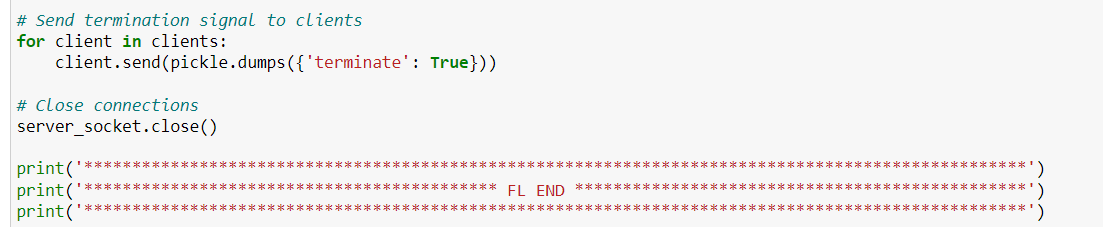
نتایج و دقتی که هر یک از کلاینت ها برای هر یک از حالات مختلف(مقادیر مختلف برای هایپرپارامترهای موجود در روش بهینه سازی) یک روش بهینه سازی خاص بدست آورده را نیز به صورت وزندار باهم میانگین گرفته و بعنوان دقت آن حالت برای روش بهینه سازی در راندی که در جریان است در آرایه ای که در بخش های قبل معرفی کردیم ذخیره می کنیم.



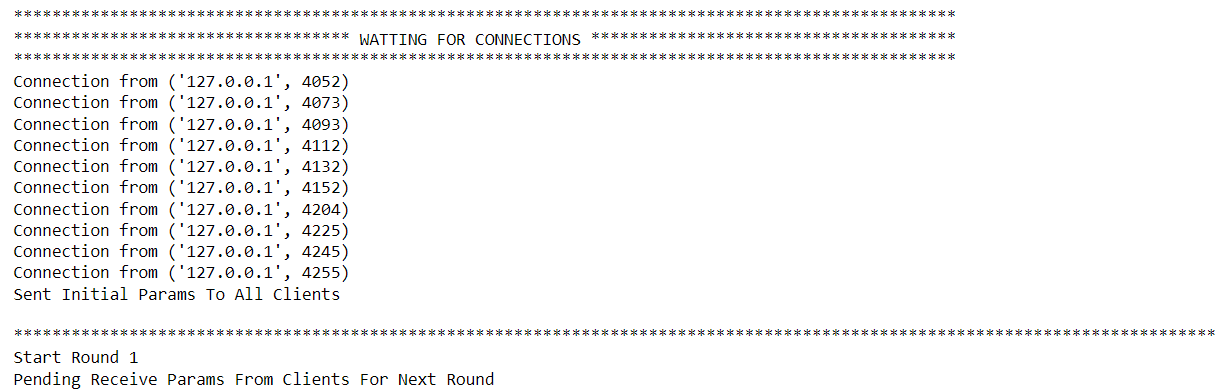
نهایتا مشابه گزارشی که در مقاله آورده شده در هر راند نیز دقت ارزیابی هر یک از حالات مختلف روش بهینه سازی انتخاب شده plot می شود:



پس از اتمام راندهای آموزشی سرور با ارسال سیگنال Terminate به کلاینت ها process آنها را متوقف خواهد کرد:



نمونه خروجی از اجرای یك راند در سرور به صورت زیر است:



پس از اتمام فرآیند یادگیری نیز نهایتا دقت های ارزیابی بدست آمده برای هر یک از حالات مختلف روش بهینه سازی در هر راند را در یک فایل اکسل گزارش می کنیم:



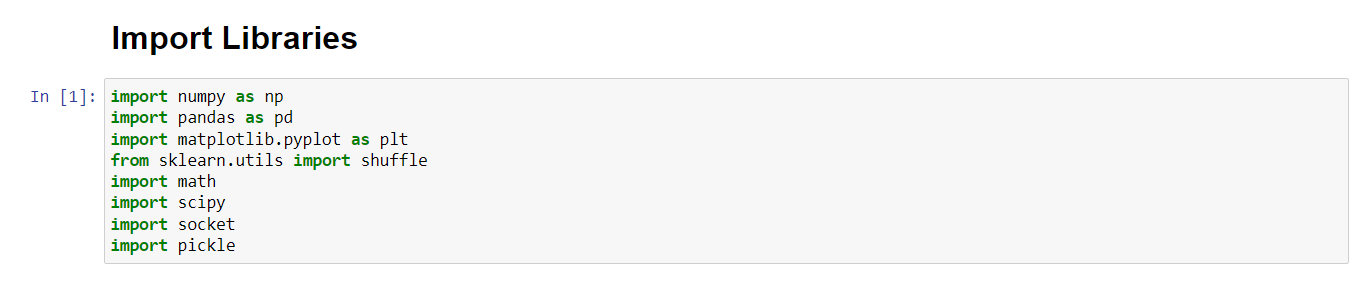
تا اینجا کد پیاده سازی شده برای سرور را بررسی کردیم. حال در ادامه کد پیاده سازی شده کلاینت را مورد بررسی قرار می دهیم.

## **[پیاده سازی کد کلاینت](#_top)**

در این بخش به بررسی کد پیاده سازی شده و فرآیند کلاینت در یادگیری فدرالی می پردازیم:

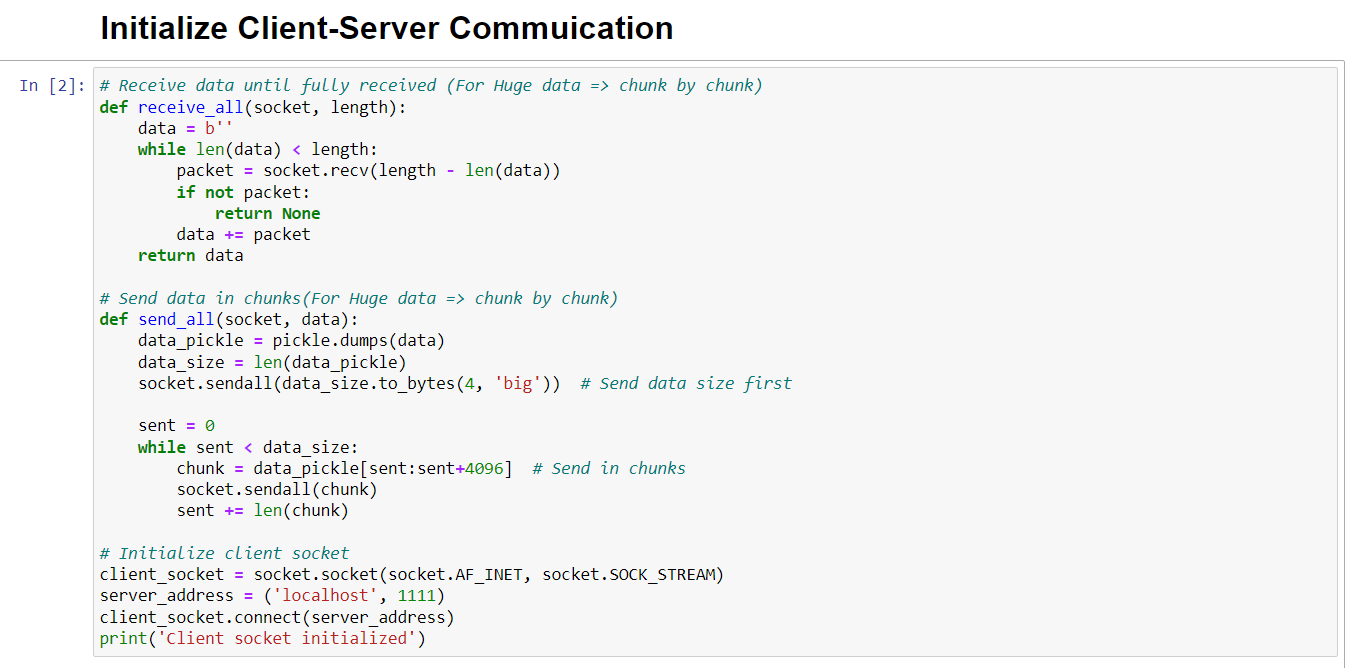
لازم به ذکر است که توضیحات ضروری به صورت کامنت در کد داده شده است.

در قسمت زیر، ابتدا کتابخانه های موردنیاز را ایمپورت می کنیم:

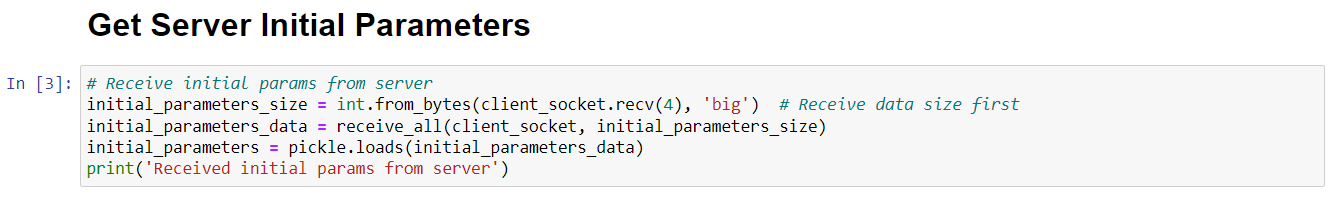


در بخش زیر توابع لازم برای ارسال و دریافت داده های حجیم به صورت چانك چانك بین کلاینت و سرور پیاده سازی شده است.

همچنین سوکت ارتباطی برای ارتباط با سرور نیز در این بخش initialize می شود.

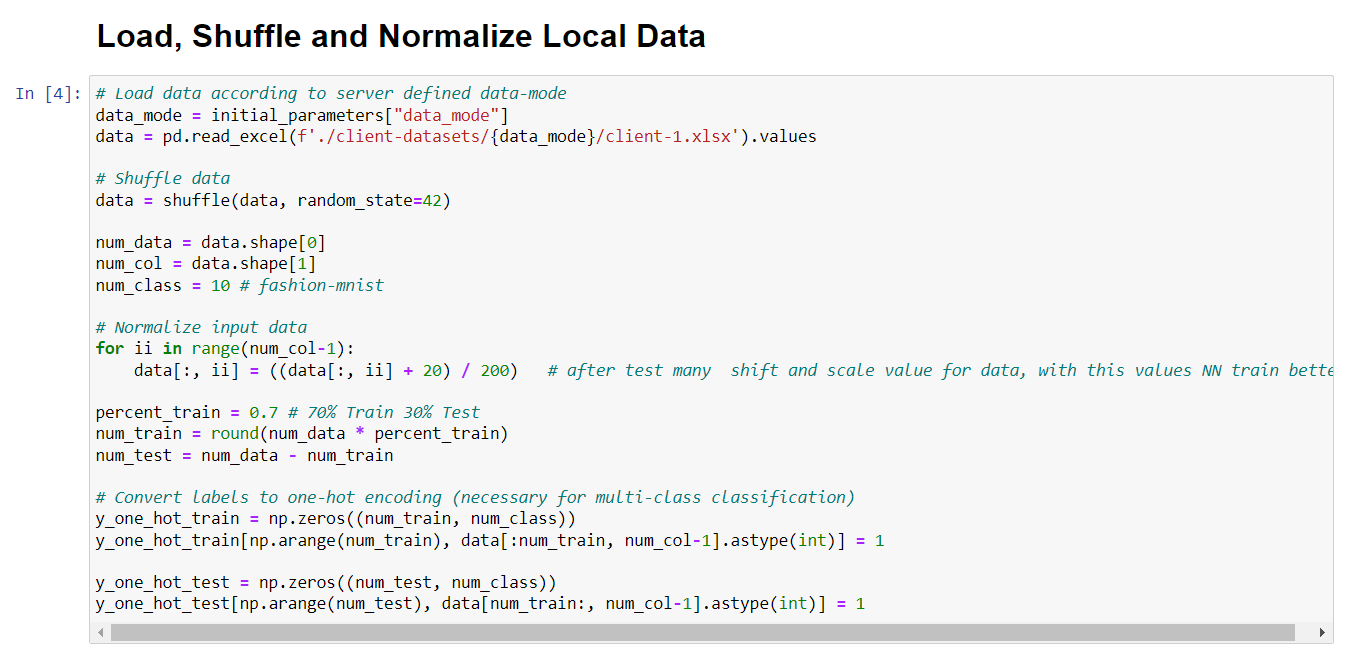


پس از برقراری اتصال بخش دریافت مقادیر اولیه پارامترهای ست شده از سمت سرور را در ادامه داریم:



در این قسمت با توجه به نوع توزیع داده تعیین شده از سمت سرور (full یا 99٪) دیتای محلی کلاینت را لود کرده و سپس پیش پردازش و نرمالسازی داده را انجام می دهیم. لازم به ذکر است که این مرحله در آموزش و همگرا شدن مدل بسیار حایزاهمیت بوده و پس از تست حالات مختلف دیدیم که با شیفت دادن 20 واحد دیتا و تقسیم آنها به عدد 200 دقت مدل خوبی بدست می آید.

پس از نرمالسازی one-hot ها را با توجه به برچسب هر نمونه برای داده های آموزشی و تستی می سازیم.



در بخش بعدی ساختار شبکه را مشاهده می کنیم:

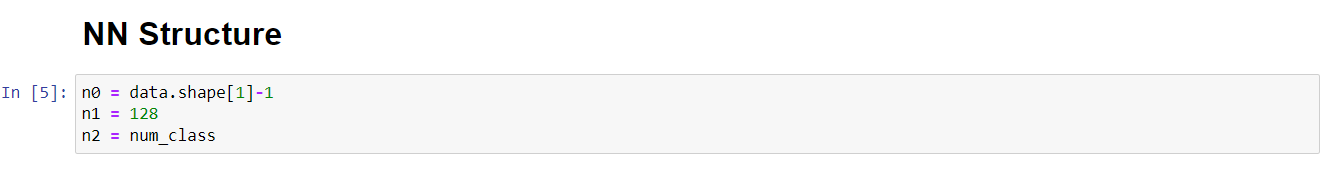
طبق توضیحات مقاله مدلهای محلی، شبکه عصبی MLP با یک لایه میانی مخفی می باشد. با تعداد نورونهای زیر:

لایه ورودی: 784

لایه فعالساز: 128

الیه خروجی: 10

همچنین طبق بیان مقاله برای نورونهای لایه مخفی، از تابع فعالسازی relu استفاده شده و برای لایه خروجی هم از تابع softmax



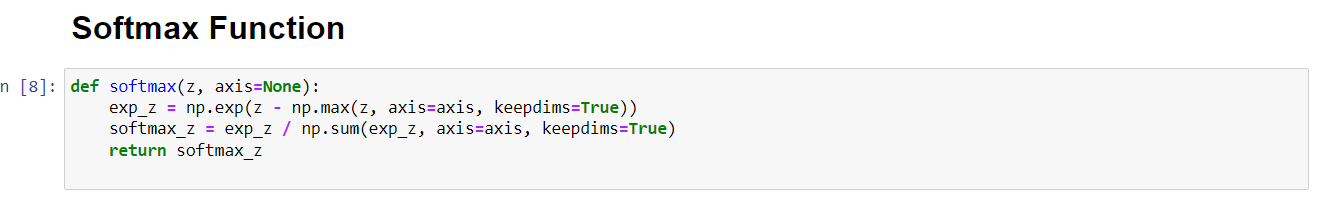
در این قسمت تنظیمات و کانفیگ های ارسالی از سمت سرور را در متغیرهای مربوطه ست می کنیم:



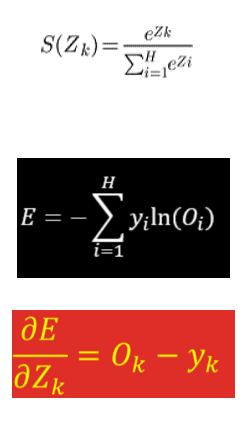
در این بخش 4 تابع فعالسازی مختلف برای آموزش پیاده سازی شده است که چون مقاله ذکر کرده که از relu استفاده نموده ماهم از این تابع استفاده می کنیم:



در این پیاده سازی، با توجه به اینکه تسک classification داریم برای لایه خروجی از softmax همراه با خطای entropy-cross استفاده کرده ایم.



در این [مقاله](https://www.mldawn.com/back-propagation-with-cross-entropy-and-softmax/) در رابطه با فواید استفاده از softmax همراه با خطای entropy-cross به تفصیل توضیح داده شده است. روابط

مربوط به مشتق نیز در این داکیومنت اثبات شده و ما از تکرار آن خودداری کرده ایم.

استفاده از entropy-cross به عنوان تابع خطا: در ترکیب با softmax ، استفاده از خطای entropy-cross معمولا

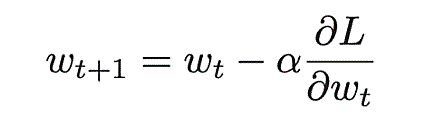
منجر به یادگیری بهتر مدل میشود. این تابع خطا به عنوان یك معیار مناسب و انعطاف پذیر برای اندازه گیری اختلاف بین

توزیع احتمالات مدل و توزیع واقعی کلاسها استفاده می شود.

**با توجه به اینکه هدف این مقاله بررسی و مقایسه دقت روش های مختلف بهینه سازی با تست حالات مختلف برای هایپرپارامترهای روش بکار گرفته می باشد، بنابر اين ما در ادامه توابع مختلفي براي هر روش بهينه سازي پياده سازي كرده ايم كه بسته به اينكه روش بهينه سازي كدام يك از روشها باشد، از يكي از اين توابع براي آموزش مدل هاي محلي استفاده خواهد شد.**

**نكته:** لازم به ذكر است كه نكات مربوط به روابط پيشرو و پس انتشار خطا به صورت كامنت در كد قرار گرفته شده است.

**آموزش با استفاده از روش بهينه سازي FedSGD - Simple Minibatch Gradient Descent**



این تابع به منظور آموزش یک شبکه عصبی با استفاده از الگوریتم Stochastic Gradient Descent (SGD) نوشته شده است. مهمترین قسمت‌های این تابع عبارتند از:

Feed-forward

- ابتدا داده‌های آموزش و برچسب‌ها به صورت دسته‌های کوچک (mini-batch) برداشته می‌شوند.

- از این داده‌ها برای اجرای مراحل Feed-forward شبکه استفاده می‌شود، که شامل محاسبه خروجی‌های لایه‌های مختلف شبکه است.

Backpropagation

- با استفاده از خروجی شبکه و برچسب‌های واقعی، خطای Cross-Entropy محاسبه می‌شود.

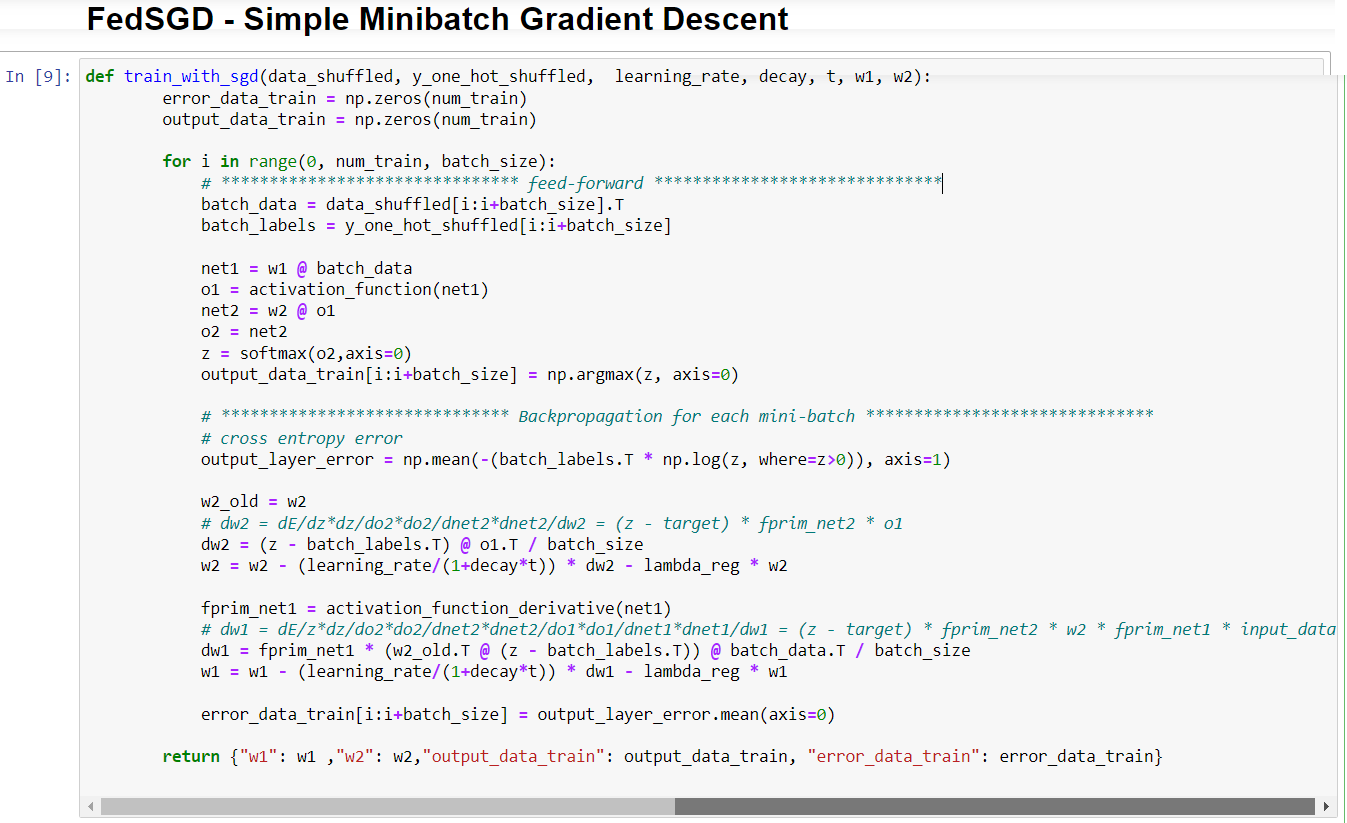
- سپس از انتهای شبکه به سمت عقب می‌رویم و گرادیان‌های خطا نسبت به وزن‌ها محاسبه می‌شود.

- از این گرادیان‌ها برای به‌روزرسانی وزن‌ها با استفاده از الگوریتم SGD و یک نرخ یادگیری متغیر كه آرگومان ورودي تابع است استفاده می‌شود.

**نكته:** نرخ یادگیری متغیر(Variable Learning Rate) كه در اين مقاله براي بعضي از حالات آزمايش در نظر گرفته شده نيز در روابط آپديت وزنها مشاهده مي كنيم. نرخ یادگیری با گذشت زمان با استفاده از یک نرخ کاهش (decay) کند می‌شود تا بهترین وزن‌ها به دست آید.

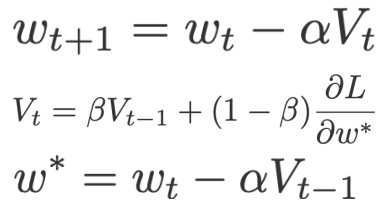
برای جلوگیری از بیش‌برازش (overfitting)، یک جمله تنظیم وزن (regularization term) به توابع به‌روزرسانی وزن‌ها اضافه شده است.

این تابع به صورت چند مرحله‌ای داده‌های آموزش را پردازش کرده و وزن‌های شبکه را به‌روزرسانی می‌کند تا به دسته‌هایی که مدل بر روی آن‌ها برازش بهتری داشته باشد، برسد.



**آموزش با استفاده از روش بهينه سازي FedSGD + Nesterov momentum**

روابط مربوط به این روش به صورت زیر تعریف می شود:



این به‌روزرسانی از Vt با مقادیر اولیه صفر استفاده می‌کند، میانگین متحرک نمایی آنچه که به آن projected gradients می گویند.

آخرین جمله در معادله دوم یعنیw\* ∂l/∂ یک projected gradients است. این مقدار را می توان با استفاده از v قبلی به دست آورد. این بدان معنی است که برای این مرحله زمانی t، باید یک انتشار رو به جلو دیگر انجام دهیم تا در نهایت بتوانیم پس انتشار یا backpropagation را اجرا کنیم. به این صورت است:

1. وزن فعلی wt را با استفاده از v قبلی به وزن پیش‌بینی‌شده w\* به‌روزرسانی می کنیم.
2. forward propagation را با استفاده از این وزن پیش بینی شده انجام می دهیم.
3. projected gradient یاw\* ∂l/∂ را به دست می آوریم.
4. بر این اساس Vt و wt+1 را محاسبه می کنیم.

در این تابع، الگوریتم Nesterov Momentum برای به‌روزرسانی وزن‌ها در شبکه عصبی استفاده شده است. این الگوریتم به‌صورت یک بهبود بر نسخه اصلی Momentum عمل می‌کند. در Momentum سنتی، یک وکتور به نام "vw" (Velocity Weight) تعریف می‌شود که گرادیان وزن‌ها را نگه می‌دارد. در هر مرحله، این وکتور به گرادیان جاری ضرب‌شده و به وزن‌ها افزوده می‌شود.

در حالت Nesterov Momentum، ابتدا یک تخمین موقعی برای گرادیان جدید ایجاد می‌شود. برای این منظور، یک مقدار بتا (beta) انتخاب می‌شود و وزن‌ها با ضرب این مقدار در وکتور Momentum (vw) به جلو حرکت می‌کنند. سپس گرادیان در موقعیت جدید محاسبه شده و از آن برای به‌روزرسانی وزن‌ها استفاده می‌شود.

**در کد پياده سازي شده، مراحل اصلی عبارتند از:**

مقدار w2\_tilde محاسبه می‌شود که این مقدار نمایانگر یک تخمین موقعی از وزن‌های جدید است.

گرادیان dw2 با استفاده از این تخمین محاسبه می‌شود و به‌روزرسانی Momentum برای w2 انجام می‌شود.

وزن‌ها با استفاده از این تخمین، Momentum و گرادیان جدید به‌روزرسانی می‌شوند.

برای w1 نیز مراحل مشابهی انجام می‌شود با توجه به تخمین موقعی جدید برای گرادیان و وزن‌ها.

از تخمین موقعی Nesterov Momentum برای محاسبه گرادیان و به‌روزرسانی وزن‌ها در مراحل Backpropagation نيز استفاده می‌شود.



**آموزش با استفاده از روش بهينه سازي FedSGD + AdaGrad**

روابط مربوط به این روش به صورت زیر تعریف می شود:

در رابطه اول، ε یک مقدار ممیز شناور کوچک است و برای اطمینان از اینکه هرگز با تقسیم بر صفر مواجه نخواهیم شد به مخرج اضافه می شود.

روش AdaGrad یک روش بهینه‌سازی است که برای مدیریت نرخ یادگیری (learning rate) در دوره‌های طولانی مدت و تطبیق آن با هر پارامتر مختص متغیرهای وزن استفاده می‌شود. در این روش، نرخ یادگیری برای هر وزن به صورت جداگانه تطبیق می‌یابد. دلیل این نام AdaGrad از "Adaptive Gradient" آمده است.

**حالا نگاهی به کد مربوط به روش AdaGrad داریم:**

تطبیق نرخ یادگیری:

- برای هر وزن (وابسته به لایه)، یک متغیر تجمعی (cumulative sum) با نام sw1 و sw2 در نظر گرفته می‌شود.

- در هر مرحله، مقدار جدید گرادیان برای وزن‌ها محاسبه می‌شود و به متغیر تجمعی اضافه می‌شود.

- نرخ یادگیری برای هر وزن با تقسیم نرخ یادگیری اصلی بر جذر گرادیان‌های تجمعی و یک عدد کوچک (epsilon) تطبیق می‌شود.

به‌روزرسانی وزن‌ها:

- وزن‌ها با استفاده از نرخ یادگیری تطبیق‌یافته به‌روزرسانی می‌شوند.

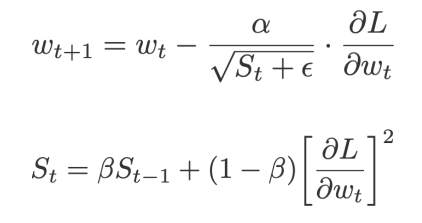
- البته، مانند سایر روش‌ها، از یک نرخ کاهش decay نیز برای کنترل یادگیری استفاده شده است.

در مجموع، AdaGrad با تطبیق نرخ یادگیری به وزن‌ها، سعي مي كند مشکلاتی که در نرخ یادگیری ثابت در دوره‌های طولانی ممکن است به وجود بیاید را حل کند و باعث بهبود کارایی و سرعت آموزش مدل می‌شود.



**آموزش با استفاده از روش بهينه سازي FedSGD + RMSProp**

روابط مربوط به این روش به صورت زیر تعریف می شود:



Root mean square prop یا RMSprop نرخ یادگیری تطبیقی دیگری است که بهبود AdaGrad است و به جای گرفتن مجموع تجمعی گرادیان های مجذور مانند AdaGrad، میانگین(اثر دادن ضريب بتا) متحرک نمایی[[1]](#footnote-1) این گرادیان ها گرفته می شود.

روش RMSprop یک روش بهینه‌سازی برای آموزش شبکه‌های عصبی با گرادیان کاهشی است. در اینجا به توضیح بخش‌های مهم این تابع می‌پردازم:

**حالا نگاهی به کد مربوط به روش RMSprop داریم:**

Feed-forward

- مانند توابع دیگر، داده‌های آموزش به صورت دسته‌های کوچک (mini-batch) برداشته می‌شوند.

- خروجی‌های لایه‌های شبکه با استفاده از وزن‌های فعلی محاسبه می‌شوند.

Backpropagation

- خطای Cross-Entropy محاسبه می‌شود.

- گرادیان‌های خطا نسبت به وزن‌ها محاسبه شده و وزن‌ها به‌روزرسانی می‌شوند.

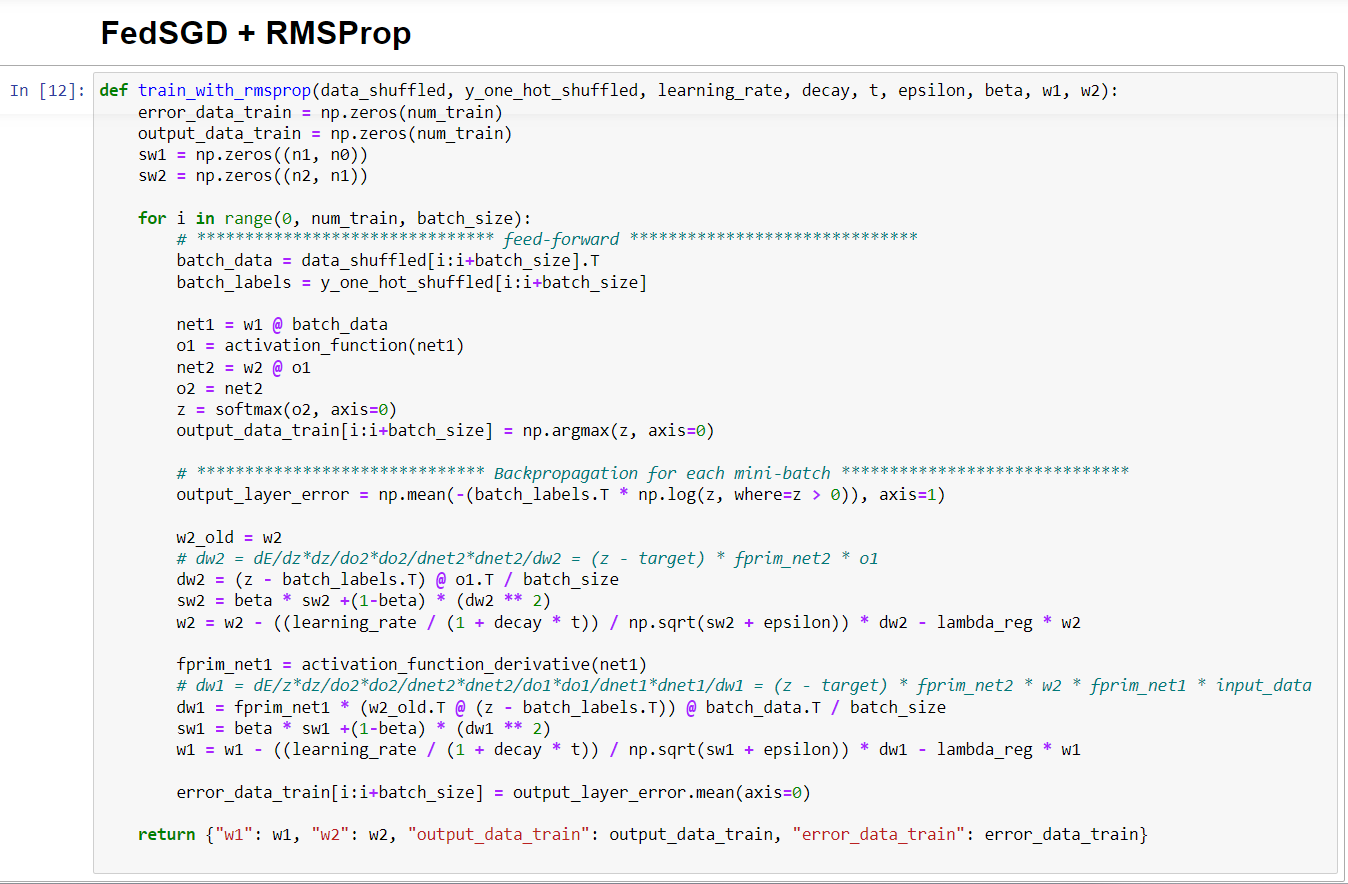
- در این روش، یک متغیر جدید به نام `sw1` و `sw2` برای هر وزن به منظور نگهداری مربع گرادیان‌ها در طول زمان ایجاد شده است.

آپدیت وزن با استفاده از RMSprop

- برای هر وزن w1 و w2 ، گرادیان مربع شده dw1 \*\* 2 و dw2 \*\* 2 با ضریب کاهش beta جمع می‌شود.

- وزن‌ها با استفاده از نرخ یادگیری به ازای هر وزن (`learning\_rate / (1 + decay \* t)`) و مقدار RMS (np.sqrt(sw1 + epsilon) یا np.sqrt(sw2 + epsilon)) به‌روزرسانی می‌شوند.

در کل، RMSprop با نگهداری یک متغیر جدید برای هر وزن، که مربع گرادیان‌ها را با یک ضریب کاهش جمع می‌کند، در کنترل نرخ یادگیری در دسته‌های داده با ویژگی‌های متغیر موثرند.

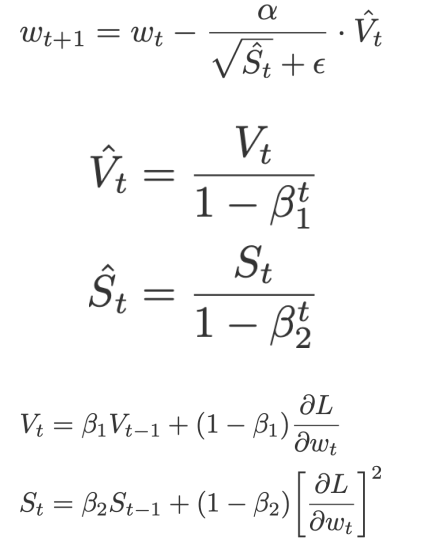


**آموزش با استفاده از روش بهينه سازي FedSGD + Adam**

**نکته:** لازم به ذکر است که برای اینکه مشاهده و بررسی فاز آموزش و تست روی هر یك از کلاینت ها جداگانه خیلی بهتر و راحت تر قابل رصد باشد، برای هر کلاینت فایل جداگانه ای در نظر گرفته شده که خروجی هر کدام جداگانه ذخیره شود.

## 

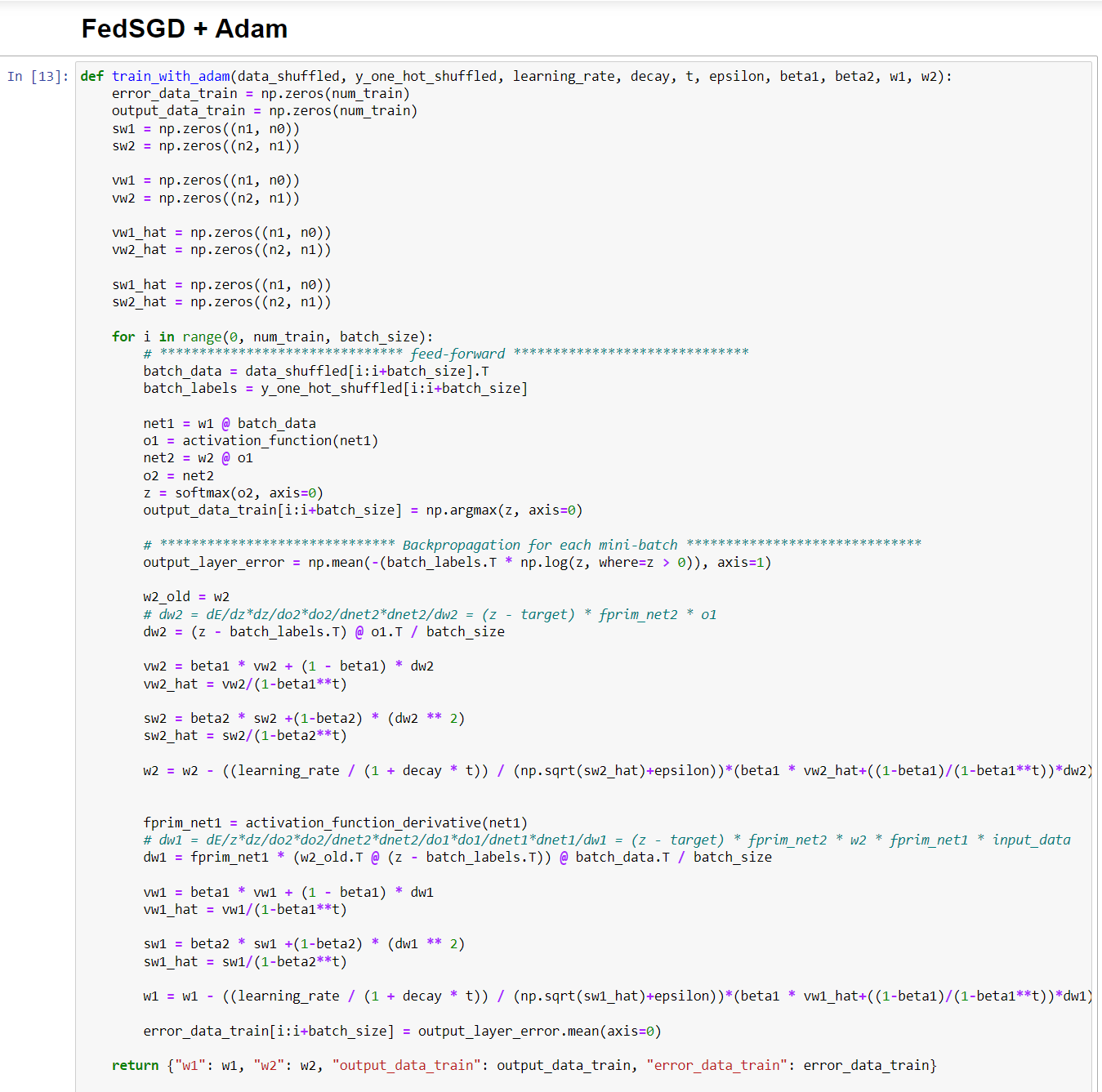
روابط مربوط به این روش به صورت زیر تعریف می شود:



این الگوریتم ترکیبی از روشهای Momentum و RMSprop است و بر اساس موارد زیر عمل می کند:

* مشابه روش Momentum، مولفه گرادیان[[2]](#footnote-2) با استفاده از V، میانگین متحرک نمایی گرادیان ها[[3]](#footnote-3)
* مشابه روش RMSprop مولفه نرخ یادگیری[[4]](#footnote-4) با تقسیم نرخ یادگیری α بر جذر S، میانگین متحرک نمایی مجذور گرادیان ها[[5]](#footnote-5)

در این کد، متغیرهای vw1، vw2، sw1، و sw2 به‌عنوان متغیرهای حرکت میانگین momentum و واریانس مربوط به گرادیان RMSprop برای هر یک از لایه‌ها استفاده می‌شوند. در هر مرحله آموزش، برای هر دسته داده (mini-batch)، مراحل feed-forward و backpropagation انجام می‌شوند. سپس گرادیان‌ها با استفاده از Adam به‌روزرسانی می‌شوند. متغیرهای vw1\_hat، vw2\_hat، sw1\_hat، و sw2\_hat مقادیر معیاری از حرکت میانگین و واریانس را نرمال‌سازی می‌کنند. این کار به کنترل نوسانات گرادیان کمک می‌کند. در نهایت، وزن‌های به‌روزرسانی شده به‌عنوان نتیجه تابع خروجی این تابع آموزش بازمی‌گردانده می‌شوند.



در بخش بعدي با توجه به اينكه روش بهينه سازي انتخاب شده از سمت سرور كدام روش بهينه سازي از روشهاي فوق باشد، تابع آن با مقادير پارامترهاي مختلفي كه براي آن روش با توجه به موارد تست شده در مقاله كه در فايل كانفيگ قرار گرفته به طور جداگانه آموزش داده مي شوند.

تابع فوق یک مدل محلی را با استفاده از یک الگوریتم بهینه‌سازی خاص و پارامترهای مشخص شده آموزش می‌دهد. برای هر ایپوک آموزش، داده‌ها به صورت تصادفی شافل می‌شوند. سپس برای هر تنظیمات مختلف بهینه‌سازی optimization\_params با استفاده از الگوریتم بهینه‌سازی مشخص (مثل SGD، Nesterov Momentum، AdaGrad، RMSprop، یا Adam) مدل محلی آموزش داده می‌شود.

در هر ایپوک، همچنين اطلاعات آموزش شامل خطای مربعات آموزش، خروجی مدل در هر ایپوک، نمودار خروجی مدل و confusion matrix برای داده‌های آموزش چاپ می‌شوند. همچنین، دقت آموزش (train accuracy) نیز محاسبه و نمایش داده می‌شود.

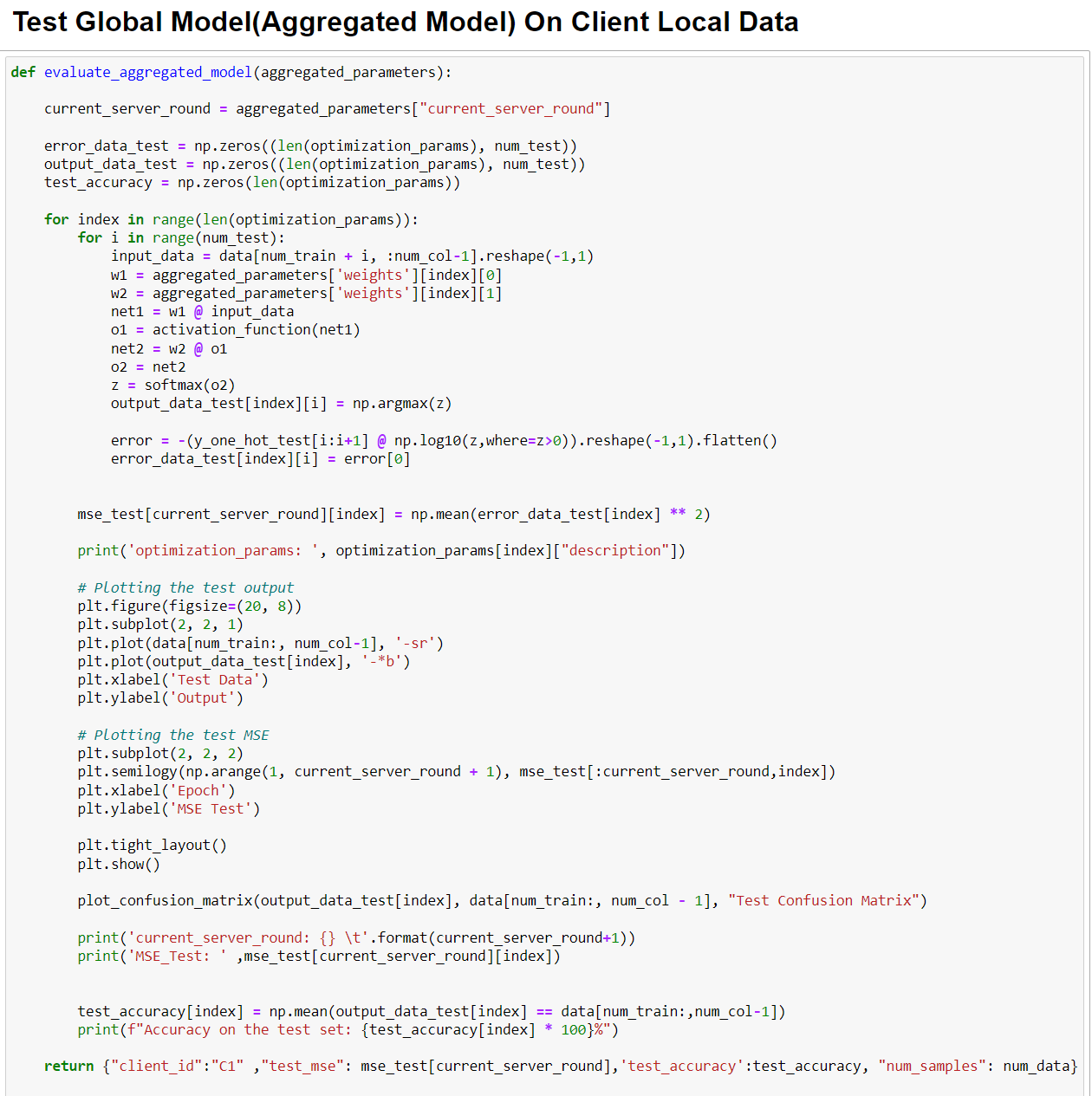
در نهایت، اطلاعات مربوط به آموزش شامل وزن‌های آخرین ایپوک، خطای مربعات آخرین ایپوک، دقت آموزش و تعداد نمونه‌ها به عنوان یک دیکشنری به عنوان خروجی از تابع برگردانده می‌شود.

بخش تابع نمايش ماتريس confusion:



در بخش بعدي نیز ارزیابی مدل گلوبال سرور با کمك دیتای تستی موجود در کلاینت ها انجام می شود. این تیکه کد برای ارزیابی مدل گلوبال در یک مرحله خاص از آموزش اجرا می‌شود. خروجی و خطاهای تست محاسبه می‌شوند و سپس عملکرد مدل در داده‌های آزمون ارزیابی می‌شود. سپس، خطای میانگین مربعات (MSE) براي هر يك از حالات مختلف روش بهينه سازيِ انتخاب شده محاسبه و نمودار خروجی برای داده‌های تست رسم می‌شود. همچنین ماتریس confusion نمایش داده شده و دقت مدل بر روی داده‌های تست محاسبه می‌شود.

سرانجام، اطلاعات ارزیابی شامل مقادیر خطا و دقت مدل بر روی داده‌های تست برگردانده می‌شود.



در سکشن زیر، فرآیند کلی یادگیری فدرالی و ارتباط کلاینت با سرور هندل می شود:

در حلقه زیر، فرآیند یادگیری فدرالی در کلاینت تا دریافت سیگنال terminate ادامه پیدا می کند. در غیر این صورت به

محض دریافت سیگنال ready، فرآیند آموزش بر روی دیتای محلیِ کلاینت آغاز شده و نتایج به همراه تعداد نمونه های موجود

در کلاینت به سرور برای اجماع ارسال میشود.

پس از تجمیع مدل و بدست آمدن پارامترهای مدل گلوبال، ارزیابی مدل بر روی دیتای تستی آغاز شده و بعنوان مقادیر اولیه راند بعدی ست میشود.

**نکته:** لازم به ذکر است که برای اینکه مشاهده و بررسی فاز آموزش و تست روی هر یك از کلاینت ها جداگانه خیلی بهتر و راحت تر قابل رصد باشد، برای هر کلاینت فایل جداگانه ای در نظر گرفته شده که خروجی هر کدام جداگانه ذخیره شود.



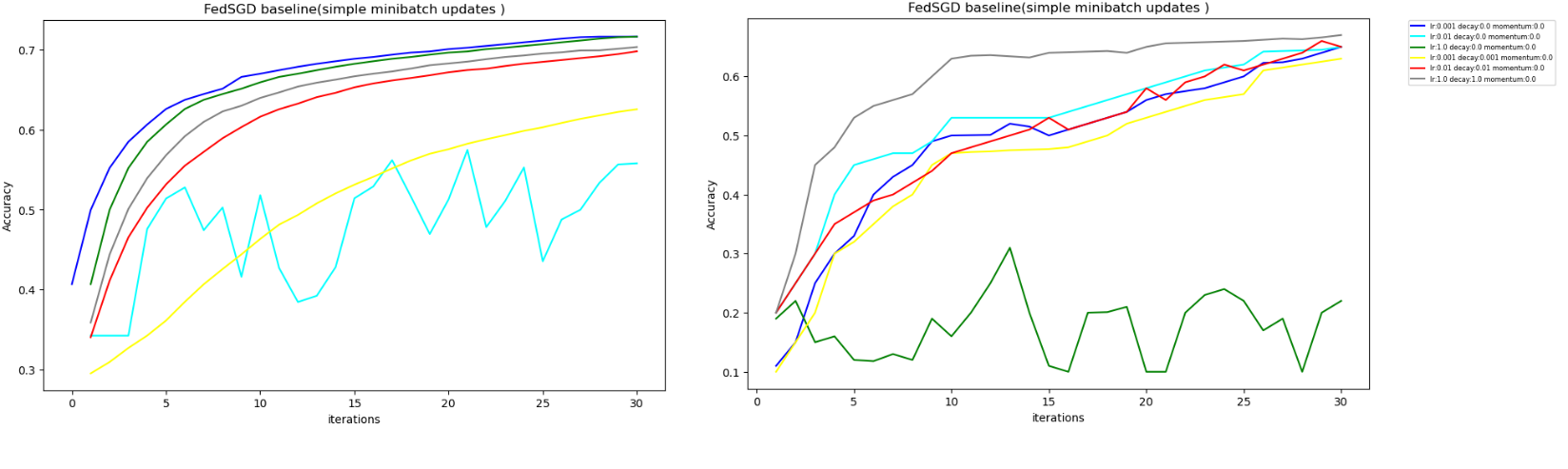
تا اينجا كدهاي پياده سازي شده براي تست عملكرد روشهاي مختلف بهينه سازي در كاربرد يادگيري فدرالي مورد بررسي قرار گرفت، در ادامه نتيجه اجراي كد پياده سازي شده براي هر يك از اين 5 روش را با توجه به كانفيگ هاي مختلف بيان شده در مقاله را گزارش مي دهيم.

# [گزارش نتايج آزمايش](#_top)

پس از بررسي نكات مهم مقاله و كد پياده سازي شده، در اين بخش نتايج اجراي كد و آزمايشات خود را گزارش مي دهيم.

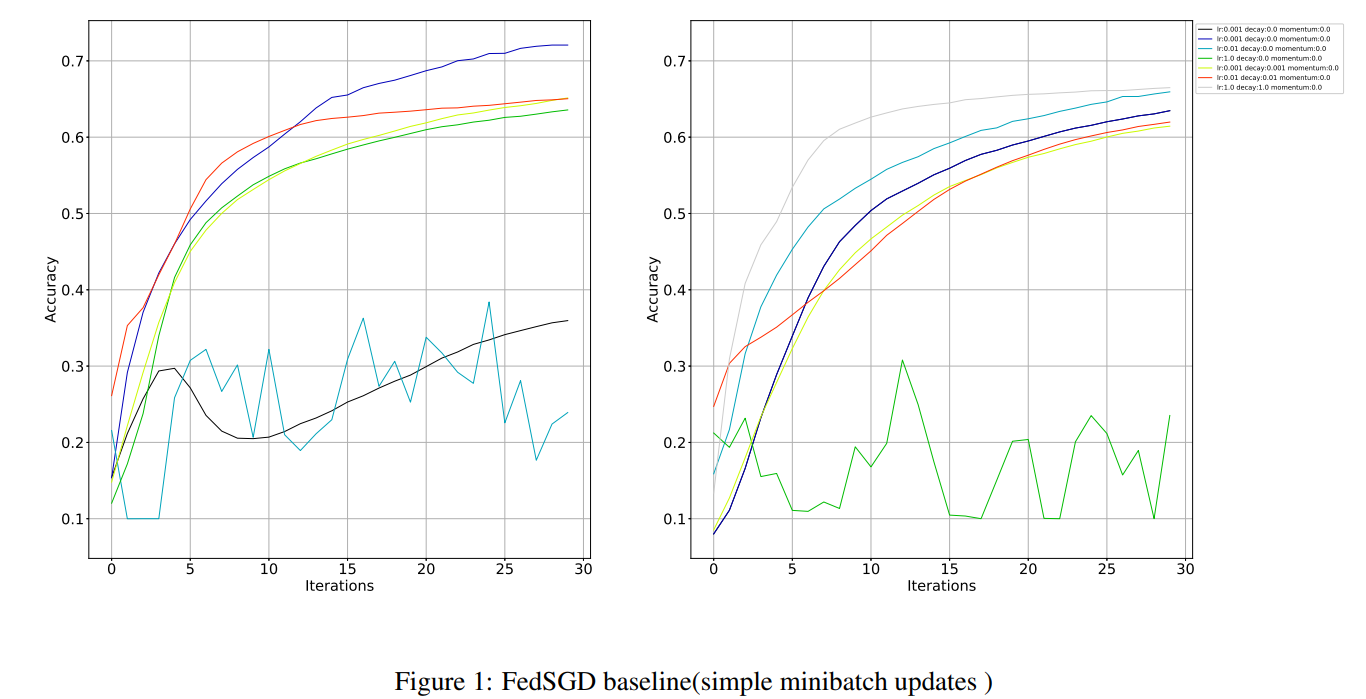
**FedSGD - Simple Minibatch Gradient Descent**: ازMBGD به عنوان baseline استفاده می شود. نتایج آزمایش نشان می دهد که MBGD به طور شگفت انگیزی خوب عمل می کند، به ویژه در مجموعه داده هایی که 99٪ غیر iid (غیر مستقل و به طور یکسان توزیع شده) و همچنیم برای مجموعه داده full-non-iid نیزخوب عمل می کند. مجموعه داده های غیر iid با بهترین پیکربندی فراپارامترها (نرخ یادگیری η = 0.001، بدون decay)، MBGD در 30 تکرار به دقت نزدیک به 75 درصد دست می یابد.با این حال، همچنین خاطرنشان می کند که استفاده از نرخ یادگیری بسیار بالا بدون decay منجر به عملکرد ضعیف در هر دو توزیع می شود.

نتایج زیر از پیاده سازی این قسمت به دست آمد:



نمودار سمت راست در حالت Full-non-iid و نمودار سمت چپ 99-non-iid است.

در شکل زیر نموداری که در مقاله آورده شده است نمایش داده می شود:



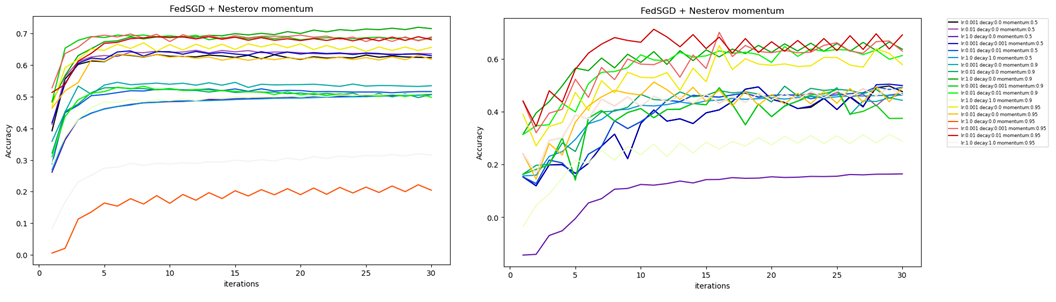
*با توجه بخاطر متفاوت بودن برخي از پارامترهاي آموزشي كه در مقاله دقيق مقدار آنها گزارش نشده بود و نصف كردن حجم ديتاست به علت محدوديت سخت افزاري كه وجود داشت نتايج به دست آمده کاملا با مقاله مشابه نشد. اما همانطور که ملاحظه می شود شباهت نسبتا زیادی بین نتایج وجود دارد.*

**FedSGD + Nesterov momentum**: ترکیب local momentum در محاسبه جهت‌های جزئی به‌روزرسانی‌ها تأثیر مثبت قوی بر روی عملکرد و نرخ همگرایی مدل تجمیع شده در هر دو توزیع داده دارد.

configuration هایی که بهترین عملکرد را داشتند به بالاترین دقتی رسیدند که baseline در طول کل آزمایش در چند تکرار اول به دست آورد. با توجه به نتایج، به نظر می رسد که هر چه مقدار β (که نشان دهنده تأثیر جهت های گذشته بر به روز رسانی است) بیشتر باشد، عملکرد بهتری خواهد داشت. این مشاهدات بدون توجه به مقادیر η (نرخ یادگیری) و نرخ decay صادق است.

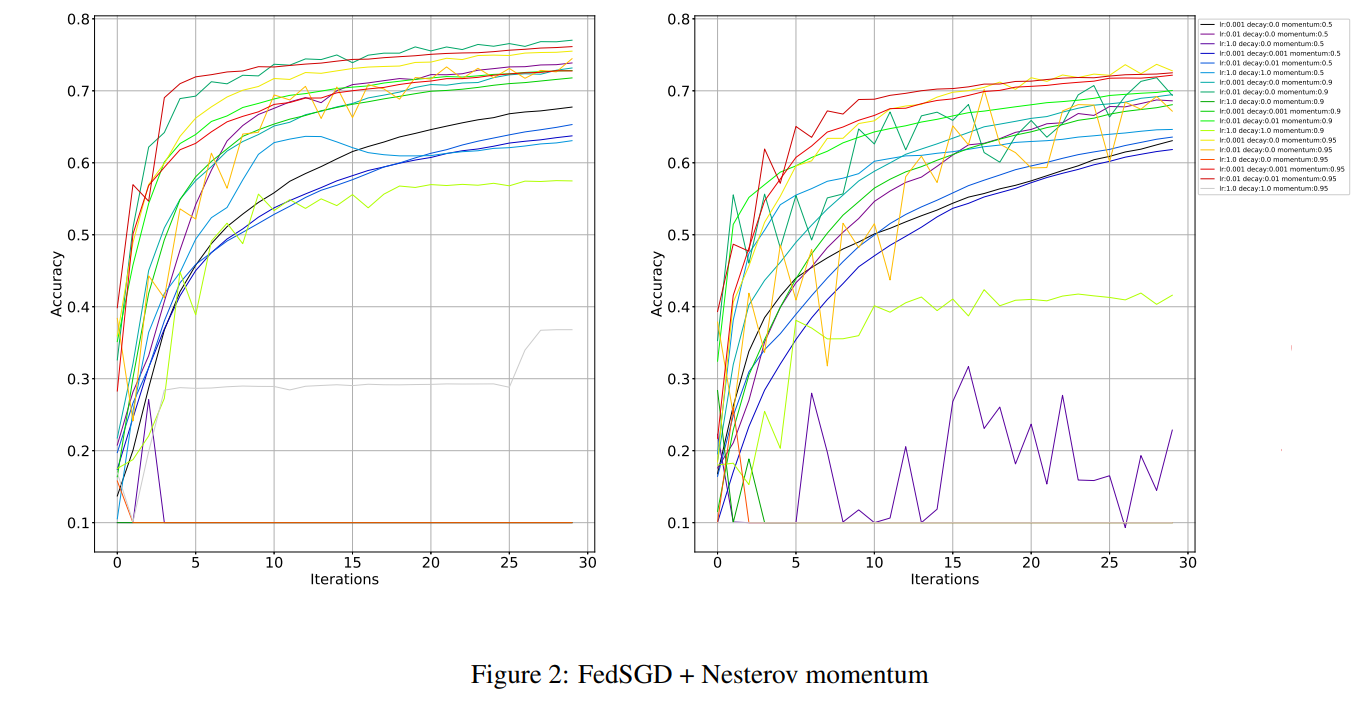
به طور خلاصه، گنجاندن مومنتوم Nesterov در FedSGD هم عملکرد و هم نرخ همگرایی را افزایش می‌دهد، با مقادیر بالاتر پارامتر مومنتوم β معمولاً عملکرد بهتری را به همراه دارد.

نتایج زیر از پیاده سازی این قسمت به دست آمد:



نمودار سمت راست در حالت Full-non-iid و نمودار سمت چپ 99-non-iid است.

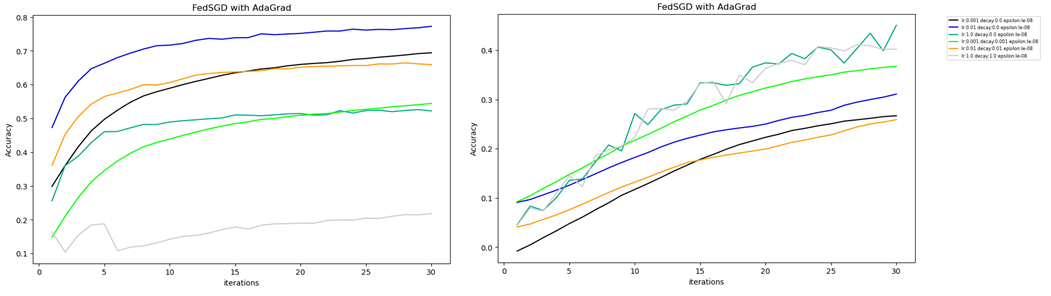
در شکل زیر نموداری که در مقاله آورده شده است نمایش داده می شود:

****

*با توجه بخاطر متفاوت بودن برخي از پارامترهاي آموزشي كه در مقاله دقيق مقدار آنها گزارش نشده بود و نصف كردن حجم ديتاست به علت محدوديت سخت افزاري كه وجود داشت نتايج به دست آمده کاملا با مقاله مشابه نشد. اما همانطور که ملاحظه می شود شباهت نسبتا زیادی بین نتایج وجود دارد.*

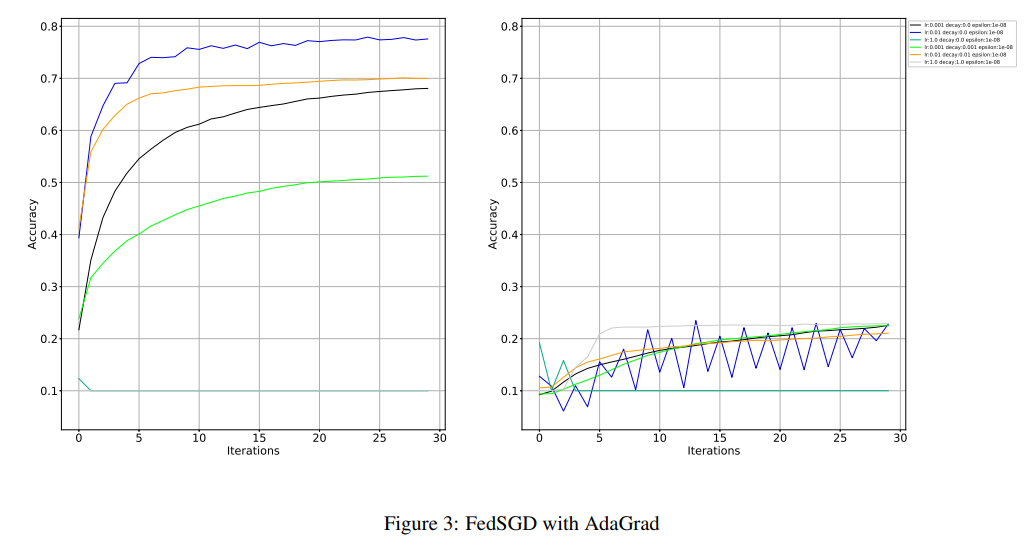
**FedSGD + AdaGrad**: الگوریتم AdaGrad حتی بهتر از سایر روش‌ها عمل می‌کند، به‌ویژه در مجموعه‌های داده‌ای که ۹۹ درصد داده‌ها به طور مستقل و یکسان توزیع نشده‌اند (non-iid). در ابتدا، AdaGrad در مقایسه با روش پایه سریعتر همگرا می شود و تا 70٪ از عملکرد خود می رسد. با این حال، در مجموعه‌های داده‌ای که تمام داده‌ها غیر iid هستند، عملکرد AdaGrad به طور قابل‌توجهی کاهش می‌یابد و تنها حدود ۲۵ درصد دقت را بدون هیچ نشانه‌ای از بهبود به دست می‌آورد. این مقاله پیشنهاد می‌کند که بررسی تعداد نمونه‌های آموزشی تصادفی باید به مجموعه داده‌های غیرiid اضافه شود تا عملکرد عالی AdaGrad مشاهده شده بر روی 99٪ مجموعه داده‌های غیرiid، جالب باشد.

نتایج زیر از پیاده سازی این قسمت به دست آمد:



نمودار سمت راست در حالت Full-non-iid و نمودار سمت چپ 99-non-iid است.

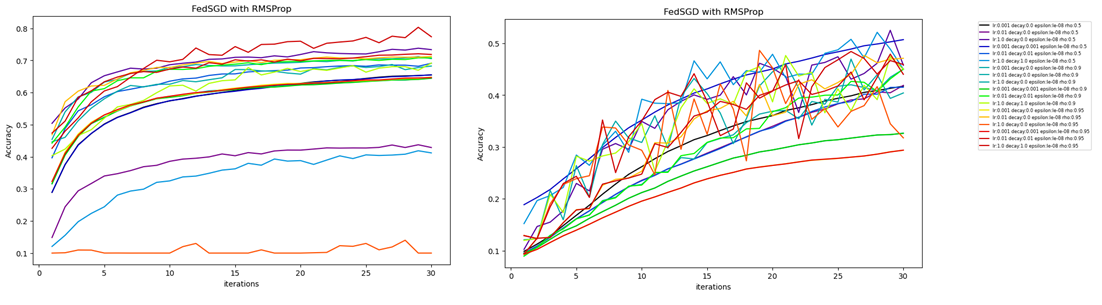
در شکل زیر نموداری که در مقاله آورده شده است نمایش داده می شود:



*با توجه بخاطر متفاوت بودن برخي از پارامترهاي آموزشي كه در مقاله دقيق مقدار آنها گزارش نشده بود و نصف كردن حجم ديتاست به علت محدوديت سخت افزاري كه وجود داشت نتايج به دست آمده کاملا با مقاله مشابه نشد. اما همانطور که ملاحظه می شود شباهت نسبتا زیادی بین نتایج وجود دارد.*

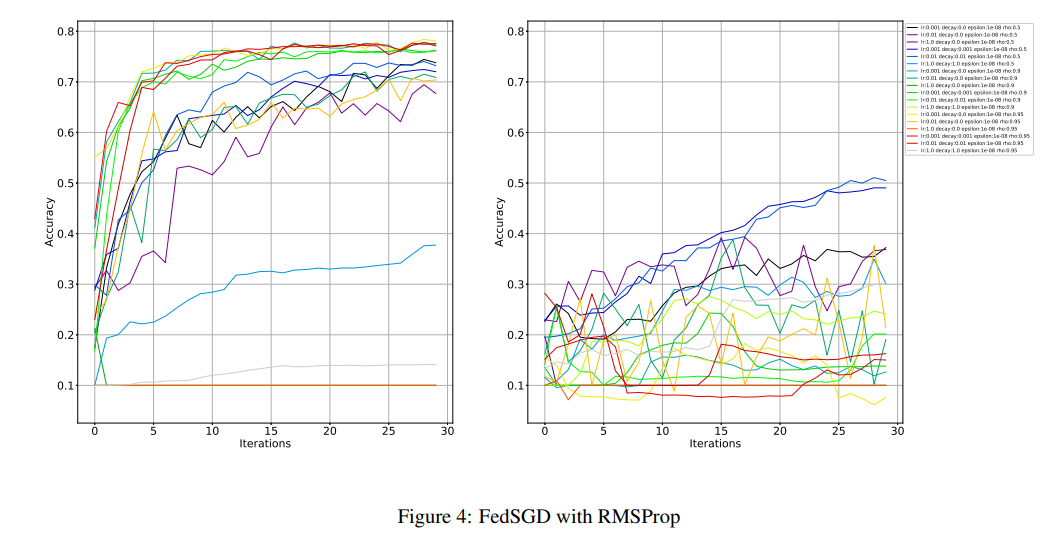
**FedSGD + RMSProp**: نتایج آزمایش نشان می دهد که الگوریتم بهینه ساز RMSProp علیرغم داشتن دامنه عملکردی گسترده تر، به سطوح دقت به دست آمده توسط روش های حرکت AdaGrad وNesterov نزدیک می شود. همچنین از الگوریتم پایه MBGD بهتر عمل می کند، به ویژه در مجموعه داده هایی که 99٪ از داده ها به طور مستقل و یکسان توزیع نشده اند (non-iid).

با این حال، در مجموعه‌های داده‌ای که تمام داده‌ها غیر iid هستند، RMSProp در مقایسه با MBGD و Nesterov Momentum همچنان به دقت کمتری دست می‌یابد. با این وجود، منحنی یادگیری برای RMSProp امیدوار کننده است و در بهترین تنظیمات به دقت 50 درصد می‌رسد و نشانه‌هایی از بهبود را نشان می‌دهد.



نمودار سمت راست در حالت Full-non-iid و نمودار سمت چپ 99-non-iid است.

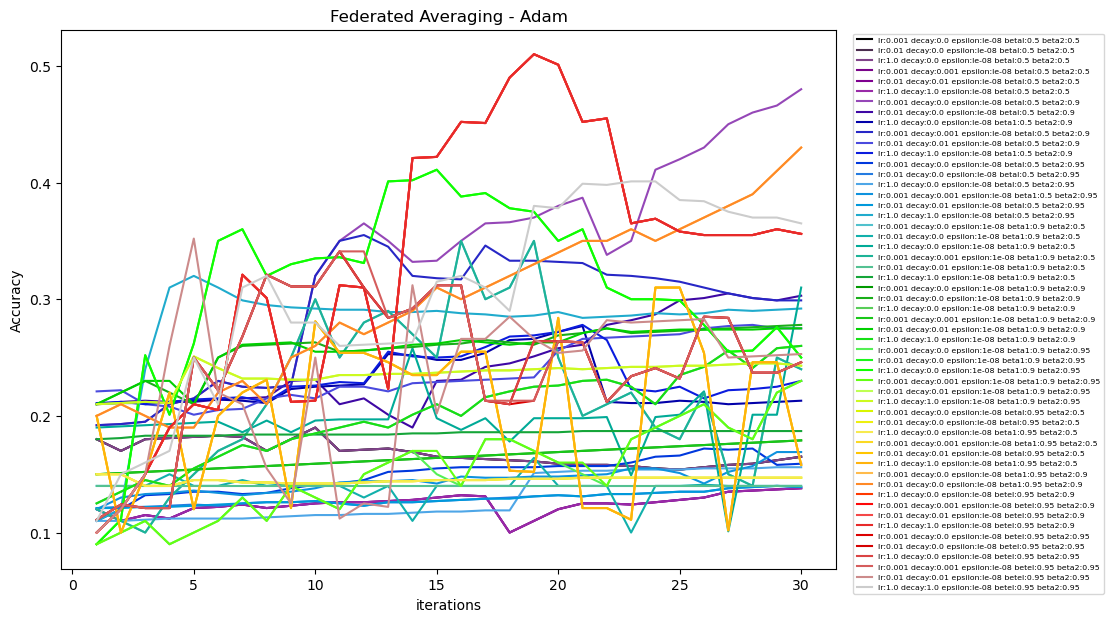
در شکل زیر نموداری که در مقاله آورده شده است نمایش داده می شود:

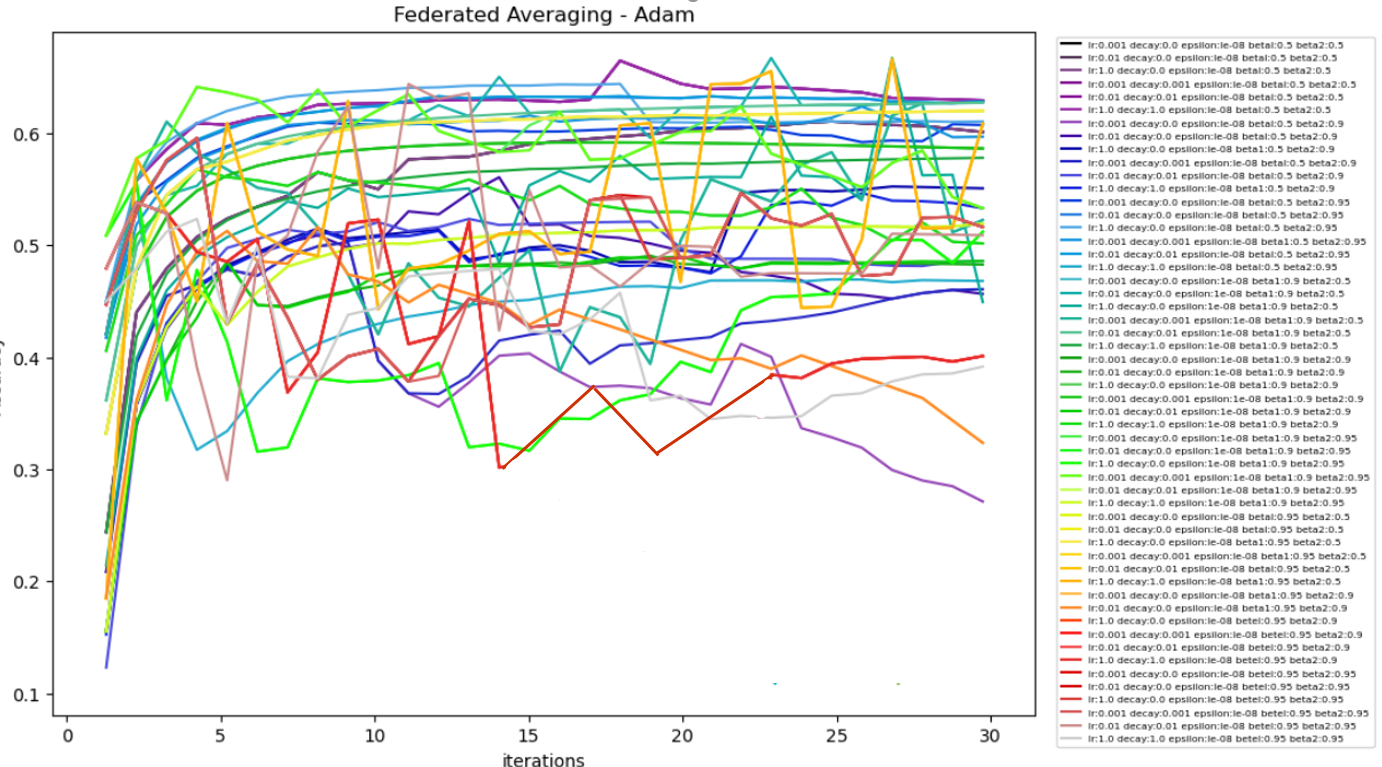


*با توجه بخاطر متفاوت بودن برخي از پارامترهاي آموزشي كه در مقاله دقيق مقدار آنها گزارش نشده بود و نصف كردن حجم ديتاست به علت محدوديت سخت افزاري كه وجود داشت نتايج به دست آمده کاملا با مقاله مشابه نشد. اما همانطور که ملاحظه می شود شباهت نسبتا زیادی بین نتایج وجود دارد.*

**FedSGD + Adam**: در آخرین آزمایش از بهینه ساز Adam استفاده شد. Adam به دلیل همگرایی سریع، دقت بالا در یادگیری شبکه های عصبی سنتی و استحکام در تنظیمات فراپارامتر به طور گسترده ای شناخته شده است. با این حال، در این آزمایش خاص، آدام اثربخشی مشابه RMSProp را نشان داد. به وضوح توسط MBGD و شتاب Nesterov از نظر عملکرد و روان بودن یادگیری پیشی گرفته است، به خصوص در مجموعه داده هایی که تمام داده ها غیر iid هستند.

نمودار اول در حالت Full-non-iid و نمودار دوم 99-non-iid است.





# [نتيجه گيري](#_top)

در اين مقاله هدف بررسي عملكرد روش هاي مختلف بهينه سازي بر روي عملكرد يادگيري فدرالي بود كه ما با درنظرگرفتن نكات بيان شده در مقاله پياده سازي آن را انجام داديم و به نتايجي مشابه با نتايج مقاله دست يافتيم.

در آزمایش خود، متوجه شدیم که بهترین الگوریتم‌های بهینه‌ساز برای هر دو توزیع داده، Mini-batch Gradient Descent، با و بدون Nesterov momentum است. Adadelta و RMSProp امیدوار کننده هستند اما در مجموعه داده های کاملاً غیر iid عملکرد ضعیفی دارند. وجود بخش غیر iid از داده های آموزشی دارای یک اثر regularizing قوی است، حتی اگر وزن آن در مقایسه با کلاس غالب ناچیز به نظر می رسد.

روش‌هایی با هدف کاهش واریانس جهت گرادیان در سناریوی خاص ما به خوبی کار می‌کنند. تکنیک‌های مومنتوم، به‌ویژه، به‌عنوان میانگین‌گیری نسبت به گرادیان‌های بعدی عمل می‌کنند، که منجر به تخمین سوگیری کمتری از جهت به‌روزرسانی بهینه می‌شود. حرکت قوی در اکثر پیکربندی ها کمک می کند.

از سوی دیگر، به نظر می‌رسد روش‌هایی که بزرگی گرادیان‌ها را تطبیق می‌دهند، مانع از فرآیند یادگیری می‌شوند، به‌ویژه در موارد کامل بدون iid. این احتمالاً به این دلیل است که بهینه‌سازهای محلی وضعیت درونی خود را، مانند نرخ یادگیری، بر اساس گرادیان‌های محلی به‌روزرسانی می‌کنند. نوسانات بزرگی گرادیان در طول دوره های آموزشی می تواند بهینه سازی را در تکرارهای بعدی کند کند.

عملکرد ضعیف AdaGrad در مجموعه داده‌های کامل بدون iid از این شهود پشتیبانی می‌کند زیرا به‌جای انطباق با نوسانات، به طور مداوم نرخ یادگیری را کاهش می‌دهد.

این الگوریتم‌ها به دلیل وجود گرادیان‌هایی با مقادیر زیاد در میانگین decay که نرخ یادگیری را به طور موثر کنترل می‌کند، روی 99٪ مجموعه داده‌های غیر iid عملکرد خوبی دارند.

در مورد آدام، که در آن هر دو نرخ یادگیری تکانه و تطبیقی اعمال می‌شوند، به نظر می‌رسد که اثر کاهش‌دهنده انطباق نرخ یادگیری بر مزایای حرکت غلبه می‌کند. با این حال، تفاوت عملکرد این مشاهده را توجیه می کند.

در حالی که این بهینه سازها ممکن است به وضوح در یافتن بهترین مدل Global نقش نداشته باشند، اما همچنان می توانند برای بهینه سازی مدل ها در مشتریان فردی مفید باشند. هدف نهایی بهینه‌سازی فدرال ارائه مدل‌هایی به مشتریان است که بر اساس داده‌های خود عملکرد خوبی داشته باشند.

1. exponential moving average [↑](#footnote-ref-1)
2. gradient component [↑](#footnote-ref-2)
3. the exponential moving average of gradients [↑](#footnote-ref-3)
4. learning rate component [↑](#footnote-ref-4)
5. exponential moving average of squared gradients [↑](#footnote-ref-5)