باسمه تعالی

# مقدمه

این گزارش به منظور بررسی و مقایسه تکنولوژی های موجود که برای پیاده سازی نسخه پایلوت یک سرویس ابری هوش مصنوعی[[1]](#footnote-1) مورد نیاز است تنظیم شده است. در این گزارش تلاش شده است تا چهارچوب ها[[2]](#footnote-2) و تکنولوژی های مختلفی که کاندیدای استفاده برای پیاده سازی راه حل های مورد نظر بوده اند بررسی شود.

# پیاده سازی الگوریتم های یادگیری ماشین[[3]](#footnote-3)

یکی از مهم ترین قابلیت هایی که یک سرویس هوش مصنوعی ابری باید داشته باشد امکان پیاده سازی، یادگیری و استفاده از الگوریتم های یادگیری ماشین به صورت توزیع شده بر روی زیرساخت ابری است. همچنین عمده سرویس های مطرح این حوزه نظیر IBM Watson، Amazon Sagemaker تعدادی مدل های از پیش یادگرفته ارائه می‌دهند که کاربرانشان تنها آنها را به منظور حصول نتیجه مورد نظرشان استفاده کنند. فلذا انتخاب یک چهارچوب و کتابخانه به عنوان مرجع اصلی که تمامی الگوریتم های یادگیری ماشین بر اساس قابلیت های این کتابخانه توسعه و به صورت توزیع شده اجرا شوند یکی از مهم ترین تصمیمات این پروژه است.

ابزارها و کتابخانه ‌های مختلفی اعم از scikit-learn، mlpack و Apache Spark ML به منظور پیاده سازی الگوریتم های یادگیری ماشین وجود دارند. اما مهم ترین و کاربردی ترین این کتابخانه ها scikit-learn و spark ml هستند که هر دوی آنها از زبان برنامه نویسی پایتون پشتیبانی میکنند که در ادامه به خصوصیات هرکدام می پردازیم.

مهم ترین خصوصیات کتابخانه Scikit Learn موارد زیر هستند:

* ابزارهای ساده و کارآمد برای تجزیه و تحلیل داده های پیش بینی شده
* قابل دسترسی برای همه و قابل استفاده مجدد  در زمینه های مختلف
* ساخته شده روی کتابخانه‌ی پرکاربرد NumPy ، SciPy و matplotlib
* منبع باز ، قابل استفاده تجاری - مجوز BSD

کتابخانه Scikit Learn عملکرد بسیار مناسبی در زمان استفاده از داده های با حجم کم که قابل ذخیره سازی بر روی RAM هستند ارائه می دهد و این کتابخانه قابلیت پردازش داده ها به صورت توزیع شده را ندارد. اما همانطور که میدانیم عموما حجم داده های تجاری که نیاز به استفاده از آنها برای مرحله یادگیری الگوریتم وجود دارد گاها به چندین ترابایت هم میرسد. این مقدار داده به طور کامل قابل ذخیره سازی روی حافظه RAM نیست. بنابراین باید به سراغ راه‌های جایگزین رفت.

مهم ترین ابزاری که برای پردازش توزیع شده داده های حجیم وجود دارد کتابخانه Apache Spark MLLib است که برخی ویژگی های این کتابخانه به شرح زیر هستند:

* سرعت تا ۱۰۰ برابر سریعتر از Apache Hadoop در یادگیری داده های حجیم
* MLLib بیش از ۸۰ اپراتور سطح بالا را ارائه می دهد که ساخت برنامه های موازی را آسان می کند.
* قابلیت پشتیبانی به صورت تعاملی از پوسته های Scala ، Python ، R و SQL
* قابل استفاده برای تمامی مراحل یادگیری ماشین اعم از استخراج و تحلیل های اولیه به کمک SQL، تحلیل های پیش پردازشی ، یادگیری الگوریتم و نهایتا اجرا
* قابلیت اجرا به صورت توزیع شده در بستر تکنولوژی های مختلف نظیر Hadoop Yarn، Apache Mesos، Kubernetes و یا حت به صورت مستقل در ابر پردازشی

نهایتا در این پروژه با توجه به خصوصیاتی که Spark ارائه میدهد و همچنین امکانات زیاد این ابزار برای کار با داده‌های حجیم، از ابزار Spark برای پردازش داده‌ها و همچنین پیاده‌سازی الگوریتم‌های یادگیری ماشین استفاده شده است.

# پیاده سازی الگوریتم های یادگیری عمیق[[4]](#footnote-4)

از آن جایی که تعداد پارامترهای مدل‌های ژرف روز به روز رو به افزایش است و با به وجود آمدن مجموعه دادگان بزرگ، آموزش توزیع شده این مدل‌ها یکی از نیاز‌های اصلی هر سیستم ابری آموزش شبکه‌های ژرف است. در این بخش ابتدا پارادایم‌های موجود برای موازی سازی پردازش و آموزش توزیع شده شبکه‌های ژرف را بررسی می کنیم.

## توازی داده[[5]](#footnote-5)

در این پارادایم گراف محاسباتی شبکه ژرف بر روی تمام  واحدهای پردازش کپی می‌شود و هر کدام بخشی از داده را پردازش می‌کند. سپس هر واحد پردازش اطلاعات مورد نیاز برای آموزش (گرادیان تابع هدف نسبت به پارامترهای شبکه) را به واحدهای پردازش دیگر ارسال کرده و با تجمیع این اطلاعات آموزش در هر واحد پردازش انجام میشود. در این پارادایم با افزایش واحد‌های پردازش، حجم داده پردازش شده در هر مرحله آموزش افزایش می‌یابد که باعث تسریع آموزش شبکه ژرف می شود. در این پارادایم، مدل ژرف و داده ورودی باید به طور کامل روی هر واحد پردازش قرار گیرد که برای آموزش شبکه‌های بزرگ محدود کننده است. از طرفی همزمان‌سازی پارامتر‌ها بین تمام واحدهای پردازشی بار زیادی بر پهنای باند شبکه ارتباطی بین واحدهای پردازشی وارد می‌کند.

## توازی مدل[[6]](#footnote-6)

در پارادایم توازی داده راهکاری برای آموزش شبکه‌های بسیار بزرگ که پارامتر‌های آن‌ها بر روی یک واحد پردازش جای نمی‌گیرد ارائه نمی‌کند. در پارادایم توازی مدل، شبکه ژرف به بخش‌های کوچک تر تقسیم می‌شود و هر بخش روی یک واحد پردازش قرار می‌گیرد. بسته به معماری مدل، هر واحد ورودی مورد نیاز را از واحدهای دیگر دریاف و خروجی را به واحد های‌دیگر ارسال می‌کند. می‌توان از ترکیب پارادایم‌های توازی مدل و داده برای آموزش توزیع شده شبکه‌های ژرف بسیار بزرگ بر روی مجموعه دادگانی بزرگ استفاده کرد.

## پارامتر سرور[[7]](#footnote-7)

پارادایم پارامتر سرور یک حالت خاص از توازی داده محسوب می‌شود که در آن برای کاهش بار پهنای باند شبکه و همزمان‌سازی بین واحدهای پردازش، از یک سرور پارامتر استفاده می‌شود. هر واحد پردازش گرادیان پارامترهای داده ورودی را محاسبه و به سرور ارسال می‌کند. سرور با تجمیع گرادیان‌ها، پارامتر‌ها را به روز کرده و پارامترهای جدید را به هر واحد پردازش ارسال می‌کند.

## پشتیبانی کتابخانه‌های یادگیری ژرف از پارادایم‌های یادگیری توزیع شده

از بین کتابخانه‌های یادگیری ژرف سه کتابخانه PyTorch، TensorFlow و MXNet را مورد بررسی قرار دادیم. کتابخانه Tensorflow تنها از پارادایم Data-Parallel پشتیبانی می‌کند. کتابخانه PyTorch از هر دو پارادایم data-parallel و model-parallel و همچنین انواع مختلفی از ترکیب‌های انها پشتیبانی میکند. کتابخانه MXNet تنها حالت Parameter Server از پارادایم Data-Parallel را پشتیبانی می‌کند. همچنین یادگیری هرسه کتابخانه با استفاده از Kubeflow بر روی بستر Kubernetes امکان پذیر است. نهایتا به دلیل بزرگ تر بودن جامعه کاربری، کامل بودن و پشتیبانی از پارادایم‌های مختلف پردازش توزیع شده، کتابخانه PyTorch را برای پیاده‌سازی مدل‌های ژرف انتخاب کرده ایم.

# تکنولوژي های اجرای توزیع شده

همانطور که میدانید یکی از مهم ترین ویژگی های یک سرویس هوش مصنوعی ابری امکان اجرای الگوریتم های هوش مصنوعی به صورت توزیع شده است. به منظور اجرای توزیع شده نیاز داریم تا از یکی از تکنولوژی هایی که به عوان زیرساخت مدیریت و نگهداری از خوشه ها ارائه شده اند استفاده کنیم.

در ادامه سه تا از معروف ترین این ابزارها که شامل Apache Mesos، Kubernetes و Docker Swarm است را با هم مقایسه میکنیم

|  |  |  |
| --- | --- | --- |
| **Supported Workloads** | **Apache Mesos** | برای کار با انواع مختلف workload طراحی شده است که هم در حالت containered و هم در حالت non-containered قابل استفاده است. |
| **Kubernetes** | برای کار با containerized workload طراحی شده است. در حال حاضر از تکنولوژي docker پشتیانی می کند. در آینده نیز از سایر کانتیتر ها و انواع دیگر workloads ها پشتیبانی خواهد کرد. |
| **Docker Swarm** | تمام دستورات داکر را در بر نمی گیرد اما قابلیت های عمومی داکر را دارد |
| **Scalability** | **Apache Mesos** | از مقیاس پذیری از طریق تعریف برنامه یا رابط کاربر پشتیبانی می کند. |
| **Kubernetes** | از طریق قابلیت deployment میتوان به صورت اتوماتیک یا دستی به هر میزان که مورد نیاز از یک application نمونه های مختلفی در قالب pod ها ساخت. |
| **Docker Swarm** | در مقایسه با k8s می تواند کانتینر سریعتر ایجاد کند و باعث می شود که زمان واکنش بر اساس تقاضا سریعتر شود |
| **Handling High-Availability** | **Apache Mesos** | با استفاده از قابلیتی به نام Mesos Agent نمونه های مختلفی از برنامه کاربردی را در هر زمان میتوان ایجاد کرد. |
| **Kubernetes** | به صورت معمول در هر سرور فیزیکی یک نسخه از برنامه کاربردی در قالب یک POD میتوان ایجاد کرد که در صورت بروز هر مشکلی در کارکرد کلی سیستم مشکلی به وجود نیاید. |
| **Docker Swarm** | دسترسی بالایی را بر اساس امکان ایجاد تکرار یک سرویس دارد. |
| **Service Discovery and Load Balancing** | **Apache Mesos** | از طریق Mesos-DNS می تواند توازن بار برای برنامه ها فراهم کند و بک رکورد SRV برای هر تسک Mesos ایجاد می کند که به یک آدرس IP و پورت ماشینی که task در آن اجرا می شود متصل است. |
| **Kubernetes** | Deployment در k8s ، pod ها را به صورت خودکار تولید و از بین می برد و Pod ها از طریق سرویس ها نشان داده می شود و امکان کشف سرویس را فراهم می کند. |
| **Docker Swarm** | دارای یک عنصر DNS است که می تواند درخواست های ورودی توزیع شده را به یک سرویس اختصاص دهد و سرویس می تواند به صورت خودکار تخصیص داده شود یا در پورت های مشخص شده توسط کاربر اجرا شود |
| **Logging and Monitoring** | **Apache Mesos** | دارای ابزاری است که تمام اجزای خوشه را اسکن می کند . داده ها از طریق API قابل دسترسی اند. |
| **Kubernetes** | اطلاعات دقیق مربوط با اجزای مختلف خوشه را از طریق یک رابط کاربری منتشر می کند. همچنین این اطلاعات از طریق ابزارهای خارجی مانند Prometheus و grafana قابل دسترسی هست و به کمک این ابزارها میتواند یک زیرساخت برای مانیتورینگ خوشه داشت. |
| **Networking** | **Apache Mesos** | دو نوع کلی پروتکل ارتباطی را پشتیبانی میکند. نوع اول که با نام IP-per-Container شناخته می شود و نوع دوم که با نام Network-Port-Mapping معروف است. |
| **Kubernetes** | برای هر POD یک IP داخلی و همچنین یک نگاشت از پورت container به پورت Host وجود دارد. |

نهایتا به دلیل ویژگی های بالا و جامعه کاربری بیشتر که امکان رفع مشکلات و جستجوی راه حل های موجود از طریق منابع آنلاین را به ما میداد ما از تکنولوژي Kubernetes به منظور زیرساخت مدیریت خوشه های برنامه های کاربردی مان استفاده کردیم.

1. AI Platform as a Service [↑](#footnote-ref-1)
2. framework [↑](#footnote-ref-2)
3. Machine Learning [↑](#footnote-ref-3)
4. Deep Learning [↑](#footnote-ref-4)
5. Data Parallelism [↑](#footnote-ref-5)
6. Model Parallelism [↑](#footnote-ref-6)
7. Parameter Server [↑](#footnote-ref-7)