Overview of Machine Learning Systems

MLOps is a set of tools and best practices for bringing ML into production.

“ML algorithms” is usually what people think of when they say machine learning, but it’s only a small part of the entire system.

A diagram of a computer

Description automatically generated

Machine learning is an approach to   
(1) learn   
(2) complex patterns from   
(3) existing data and use these patterns to make   
(4) predictions on   
(5) unseen data

*------------------------------- (8/1/2024) Continue*

1. Learn: the system has the capacity to learn

if you want to build an ML system to learn to predict the rental price for Airbnb listings, you need to provide a dataset where each input is a listing with relevant characteristics (square footage, number of rooms, neighborhood, amenities, rating of that listing, etc.) and the associated output is the rental price of that listing. Once learned, this ML system should be able to predict the price of a new listing given its characteristics

*------------------------------- (8/2/2024) (Pg 3)Continue*

1. Complex patterns: there are patterns to learn, and they are complex ML solutions are only useful when there are patterns to learn. Sane people don’t invest money into building an ML system to predict the next outcome of a fair die because there’s no pattern in how these outcomes are generated.4 However, there are patterns in how stocks are priced, and therefore companies have invested billions of dollars in building ML systems to learn those patterns

*------------------------------- (8/3/2024) (Pg 4)Continue*

there might be a pattern in how Elon Musk’s tweets affect cryptocurrency prices. How‐ ever, you wouldn’t know until you’ve rigorously trained and evaluated your ML models on his tweets. Even if all your models fail to make reasonable predictions of cryptocurrency prices, it doesn’t mean there’s no pattern.

*------------------------------- (8/4/2024) (Pg 4)Continue*

A diagram of software

Description automatically generated

*------------------------------- (8/5/2024) (Pg 5)Continue*

1. Existing data: data is available, or it’s possible to collect data Because ML learns from data, there must be data for it to learn from. It’s amusing to think about building a model to predict how much tax a person should pay a year, but it’s not possible unless you have access to tax and income data of a large population. In the zero-shot learning (sometimes known as zero-data learning) context, it’s possible for an ML system to make good predictions for a task without having been trained on data for that task. However, this ML system was previously trained on data for other tasks, often related to the task in consideration

*------------------------------- (8/6/2024) (Pg 5)Continue*

1. Predictions: it’s a predictive problem ML models make predictions, so they can only solve problems that require predictive answers. ML can be especially appealing when you can benefit from a large quantity of cheap but approximate predictions. In English, “predict” means “estimate a value in the future.” For example, what will the weather be like tomorrow? Who will win the Super Bowl this year? What movie will a user want to watch next?

*------------------------------- (8/7/2024) (Pg 6)Continue*

1. Unseen data: unseen data shares patterns with the training data The patterns your model learns from existing data are only useful if unseen data also share these patterns. A model to predict whether an app will get downloaded on Christmas 2020 won’t perform very well if it’s trained on data from 2008, when the most popular app on the App Store was Koi Pond. What’s Koi Pond? Exactly

*------------------------------- (8/8/2024) (Pg 6)Continue*

6. It’s repetitive

Humans are great at few-shot learning: you can show kids a few pictures of cats and most of them will recognize a cat the next time they see one. Despite exciting progress in few-shot learning research, most ML algorithms still require many examples to learn a pattern. When a task is repetitive, each pattern is repeated multiple times, which makes it easier for machines to learn it.

*--------------------- 8/9/2024 up to page 06*

1. The cost of wrong predictions is cheap Unless your ML model’s performance is 100% all the time, which is highly unlikely for any meaningful tasks, your model is going to make mistakes. ML is especially suitable when the cost of a wrong prediction is low. For example, one of the biggest use cases of ML today is in recommender systems because with recommender systems, a bad recommendation is usually forgiving—the user just won’t click on the recommendation.

*--------------------- 8/10/2024 up to page 06*

1. It’s at scale:

Having a problem at scale also means that there’s a lot of data for you to collect, which is useful for training ML models

1. The patterns are constantly changing:

Today an indication of a spam email is a Nigerian prince, but tomorrow it might be a distraught Vietnamese writer.

Machine Learning Use Cases

Recommendation system, assisting people in many of their daily activity.

Enterprise applications might have stricter accuracy requirements but be more forgiving with latency requirements.

improving a speech recognition system’s accuracy from 95% to 95.5% might not be noticeable to most consumers, but improving a resource allocation system’s efficiency by just 0.1% can help a corporation like Google or General Motors save millions of dollars.

*--------------------- 8/11/2024 up to page 10*

Deciding how much to charge for your product or service is probably one of the hardest business decisions; why not let ML do it for you? Price optimization is the process of estimating a price at a certain time period to maximize a defined objective function, such as the company’s margin, revenue, or growth rate.

for example, internet ads, flight tickets, accommodation bookings, ride-sharing, and events.

*--------------------- 8/12/2024 up to page 11*

The cost of acquiring a new user is approximated to be 5 to 25 times more expensive than retaining an existing one.12 Churn prediction is predicting when a specific customer is about to stop using your products or services so that you can take appropriate actions to win them back.

*--------------------- 8/13/2024 up to page 11*

Machine Learning in Research Versus in Production

As ML usage in the industry is still fairly new, most people with ML expertise have gained it through academia: taking courses, doing research, reading academic papers. If that describes your background, it might be a steep learning curve for you to understand the challenges of deploying ML systems in the wild and navigate an overwhelming set of solutions to these challenges

*--------------------- 8/14/2024 up to page 12*

A black and white text on a black background

Description automatically generated

*--------------------- 8/15/2024 up to page 13*

ML engineers   
Want a model that recommends restaurants that users will most likely order from, and they believe they can do so by using a more complex model with more data.

Sales team   
Wants a model that recommends the more expensive restaurants since these restaurants bring in more service fees.

Product team   
Notices that every increase in latency leads to a drop in orders through the service, so they want a model that can return the recommended restaurants in less than 100 milliseconds.

ML platform team   
As the traffic grows, this team has been woken up in the middle of the night because of problems with scaling their existing system, so they want to hold off on model updates to prioritize improving the ML platform.

Manager   
Wants to maximize the margin, and one way to achieve this might be to let go of the ML team

*--------------------- 8/16/2024 up to page 14*

Production having different requirements from research is one of the reasons why successful research projects might not always be used in production. For example, ensembling is a technique popular among the winners of many ML competitions, including the famed $1 million Netflix Prize, and yet it’s not widely used in produc‐ tion. Ensembling combines “multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.”15 While it can give your ML system a small performance improvement, ensembling tends to make a system too complex to be useful in production,

*--------------------- 8/17/2024 up to page 14*

The misalignment of interests between research and production has been noticed by researchers. In an EMNLP 2020 paper, Ethayarajh and Jurafsky argued that benchmarks have helped drive advances in natural language processing (NLP) by incentivizing the creation of more accurate models at the expense of other qualities valued by practitioners such as compactness, fairness, and energy efficiency

*--------------------- 8/18/2024 up to page 15*

Computational priorities

When designing an ML system, people who haven’t deployed an ML system often make the mistake of focusing too much on the model development part and not enough on the model deployment and maintenance part.

*--------------------- 8/19/2024 up to page 15*

The average latency of Google Translate is the average time it takes from when a user clicks Translate to when the translation is shown, and the throughput is how many queries it processes and serves a second. If your system always processes one query at a time, higher latency means lower throughput. If the average latency is 10 ms, which means it takes 10 ms to process a query, the throughput is 100 queries/second. If the average latency is 100 ms, the throughput is 10 queries/second. However, because most modern distributed systems batch queries to process them together, often concurrently, higher latency might also mean higher throughput.

*--------------------- 8/20/2024 up to page 16*

A diagram of a number of queries

Description automatically generated

*--------------------- 8/21/2024 up to page 17*

To reduce latency in production, you might have to reduce the number of queries you can process on the same hardware at a time. If your hardware is capable of processing many more queries at a time, using it to process fewer queries means underutilizing your hardware, increasing the cost of processing each query

In 2017, an Akamai study found that a 100 ms delay can hurt conversion rates by 7%

*--------------------- 8/22/2024 up to page 17*

Use percentile to avoid the outliers and find the average of the latency. Like 50th percentile.

*--------------------- 8/23/2024 up to page 18*

Data During the research phase, the datasets you work with are often clean and wellformatted, freeing you to focus on developing models. They are static by nature so that the community can use them to benchmark new architectures and techniques

*--------------------- 8/24/2024 up to page 18*

means that many people might have used and discussed the same datasets, and quirks of the dataset are known. You might even find open source scripts to process and feed the data directly into your models

*--------------------- 8/25/2024 up to page 18*

In research, you mostly work with historical data, e.g., data that already exists and is stored somewhere. In production, most likely you’ll also have to work with data that is being constantly generated by users, systems, and third-party data.

*--------------------- 8/26/2024 up to page 19*



*--------------------- 8/27/2024 up to page 19*

Fairness During the research phase, a model is not yet used on people, so it’s easy for research‐ ers to put off fairness as an afterthought: “Let’s try to get state of the art first and worry about fairness when we get to production.” When it gets to production, it’s too late. If you optimize your models for better accuracy or lower latency, you can show that your models beat state of the art. But, as of writing this book, there’s no equivalent state of the art for fairness metrics

*--------------------- 8/28/2024 up to page 19*

You or someone in your life might already be a victim of biased mathematical algorithms without knowing it. Your loan application might be rejected because the ML algorithm picks on your zip code, which embodies biases about one’s soci‐ oeconomic background. Your resume might be ranked lower because the ranking system employers use picks on the spelling of your name. Your mortgage might get a higher interest rate because it relies partially on credit scores, which favor the rich and punish the poor. Other examples of ML biases in the real world are in predictive policing algorithms, personality tests administered by potential employers, and college rankings.

*--------------------- 8/29/2024 up to page 19*

ML algorithms don’t predict the future, but encode the past, thus perpetuating the biases in the data and more. When ML algorithms are deployed at scale, they can discriminate against people at scale. If a human operator might only make sweeping judgments about a few individuals at a time, an ML algorithm can make sweeping judgments about millions in split seconds. This can especially hurt members of minority groups because misclassification on them could only have a minor effect on models’ overall performance metrics

*--------------------- 8/30/2024 up to page 20*

If an algorithm can already make correct predictions on 98% of the population, and improving the predictions on the other 2% would incur multiples of cost, some companies might, unfortunately, choose not to do it. During a McKinsey & Company research study in 2019, only 13% of the large companies surveyed said they are taking steps to mitigate risks to equity and fairness, such as algorithmic bias and discrimination

*--------------------- 8/31/2024 up to page 20*

Interpretability

“Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?

While most of us are comfortable with using a microwave without understanding how it works, many don’t feel the same way about AI yet, especially if that AI makes important decisions about their lives.

First, interpretability is important for users, both business leaders and end users, to understand why a decision is made so that they can trust a model and detect potential biases

*--------------------- 9/1/2024 up to page 21*

Machine Learning Systems Versus Traditional Software

many challenges are unique to ML applications and require their own tools. In SWE, there’s an underlying assumption that code and data are separated. In fact, in SWE, we want to keep things as modular and separate as possible.

On the contrary, ML systems are part code, part data, and part artifacts created from the two. The trend in the last decade shows that applications developed with the most/best data win. Instead of focusing on improving ML algorithms, most companies will focus on improving their data

*--------------------- 9/2/2024 up to page 22*

In traditional SWE, you only need to focus on testing and versioning your code. With ML, we have to test and version our data too

*--------------------- 9/3/2024 up to page 22*

The size of ML models is another challenge. As of 2022, it’s common for ML models to have hundreds of millions, if not billions, of parameters, which requires gigabytes of random-access memory (RAM) to load them into memory. A few years from now, a billion parameters might seem quaint—like, “Can you believe the computer that sent men to the moon only had 32 MB of RAM?”

*--------------------- 9/4/2024 up to page 22*

As ML models get more complex, coupled with the lack of visibility into their work, it’s hard to figure out what went wrong or be alerted quickly enough when things go wrong.

*--------------------- 9/5/2024 up to page 23*

ML systems are complex, consisting of many different components. Data scientists and ML engineers working with ML systems in production will likely find that focusing only on the ML algorithms part is far from enough. It’s important to know about other aspects of the system, including the data stack, deployment, monitoring, maintenance, infrastructure, etc

*--------------------- 9/6/2024 up to page 23*

Introduction to Machine Learning Systems Design

ML systems design takes a system approach to MLOps, which means that we’ll consider an ML system holistically to ensure that all the components—the business requirements, the data stack, infrastructure, deployment, monitoring, etc.— and their stakeholders can work together to satisfy the specified objectives and requirements

*--------------------- 9/7/2024 up to page 25*

Before using ML algorithms to solve your problem, you first need to frame your problem into a task that ML can solve. The difficulty of your job can change significantly depending on how you frame your problem

*--------------------- 9/8/2024 up to page 25*

Business and ML Objectives

When working on an ML project, data scientists tend to care about the ML objectives: the metrics they can measure about the performance of their ML models such as accuracy, F1 score, inference latency, etc. They get excited about improving their model’s accuracy from 94% to 94.2% and might spend a ton of resources—data, compute, and engineering time—to achieve that.

*--------------------- 9/9/2024 up to page 26*

A pattern I see in many short-lived ML projects is that the data scientists become too focused on hacking ML metrics without paying attention to business metrics. Their managers, however, only care about business metrics and, after failing to see how an ML project can help push their business metrics, kill the projects prematurely (and possibly let go of the data science team involved).

*--------------------- 9/10/2024 up to page 26*

The ultimate goal of any project within a business is, therefore, to increase profits, either directly or indirectly: directly such as increasing sales (conversion rates) and cutting costs; indirectly such as higher customer satisfaction and increasing time spent on a website

*--------------------- 9/11/2024 up to page 26*

Example of Comparing ML metric to business metric

1. Increased Recommender system’s click through rate increase in purchase and revenue

For example, Netflix measures the performance of their recommender system using take-rate: the number of quality plays divided by the number of recommendations a user sees.4 The higher the take-rate, the better the recommender system. Netflix also put a recommender system’s take-rate in the context of their other business metrics like total streaming hours and subscription cancellation rate. They found that a higher take-rate also results in higher total streaming hours and lower subscription cancellation rates.

The same ML model can also solve their problems faster, which makes them spend less money on your services.

To gain a definite answer on the question of how ML metrics influence business metrics, experiments are often needed. Many companies do that with experiments like A/B testing and choose the model that leads to better business metrics, regardless of whether this model has better ML metrics.

*--------------------- 9/12/2024 up to page 27*

Yet, even rigorous experiments might not be sufficient to understand the relationship between an ML model’s outputs and business metrics

Actual threats will then go through another, different process aimed at stopping them. When this process fails to stop a threat, it might be impossible to figure out whether the ML component has anything to do with it.

Many companies like to say that they use ML in their systems because “being AIpowered” alone already helps them attract customers, regardless of whether the AI part actually does anything useful.

*--------------------- 9/13/2024 up to page 28*

Returns on investment in ML depend a lot on the maturity stage of adoption

*--------------------- 9/14/2024 up to page 28*

A graph with numbers and text

Description automatically generated with medium confidence

*--------------------- 9/15/2024 up to page 29*

Requirements for ML Systems We can’t say that we’ve successfully built an ML system without knowing what requirements the system has to satisfy. The specified requirements for an ML system vary from use case to use case. However, most systems should have these four charac‐ teristics: reliability, scalability, maintainability, and adaptability.

*--------------------- 9/16/2024 up to page 29*

Reliability

The system should continue to perform the correct function at the desired level of performance even in the face of adversity (hardware or software faults, and even human error).

*--------------------- 9/17/2024 up to page 29*

“Correctness” might be difficult to determine for ML systems. For example, your system might call the predict function—e.g., model.predict()—correctly, but the predictions are wrong. How do we know if a prediction is wrong if we don’t have ground truth labels to compare it with?

*--------------------- 9/18/2024 up to page 29*

ML systems can fail silently. End users don’t even know that the system has failed and might have kept on using it as if it were working. For example, if you use Google Translate to translate a sentence into a language you don’t know, it might be very hard for you to tell even if the translation is wrong

*--------------------- 9/19/2024 up to page 29*

Scalability There are multiple ways an ML system can grow. It can grow in complexity. Last year you used a logistic regression model that fit into an Amazon Web Services (AWS) free tier instance with 1 GB of RAM, but this year, you switched to a 100-millionparameter neural network that requires 16 GB of RAM to generate predictions.

*--------------------- 9/20/2024 up to page 30*

Your ML system can grow in traffic volume. When you started deploying an ML system, you only served 10,000 prediction requests daily. However, as your company’s user base grows, the number of prediction requests your ML system serves daily fluctuates between 1 million and 10 million

*--------------------- 9/21/2024 up to page 30*

An ML system might grow in ML model count. Initially, you might have only one model for one use case, such as detecting the trending hashtags on a social network site like Twitter. However, over time, you want to add more features to this use case, so you’ll add one more to filter out NSFW (not safe for work) content and another model to filter out tweets generated by bots

*--------------------- 9/22/2024 up to page 30*

This growth pattern is especially common in ML systems that target enterprise use cases. Initially, a startup might serve only one enterprise customer, which means this startup only has one model. However, as this startup gains more customers, they might have one model for each customer. A startup I worked with had 8,000 models in production for their 8,000 enterprise customers

*--------------------- 9/23/2024 up to page 30*

For example, at peak, your system might require 100 GPUs (graphics processing units). However, most of the time, it needs only 10 GPUs. Keeping 100 GPUs up all the time can be costly, so your system should be able to scale down to 10 GPUs

*--------------------- 9/24/2024 up to page 30*

However, with one hundred models, both the monitoring and retraining aspect will need to be automated. You’ll need a way to manage the code generation so that you can adequately reproduce a model when you need t

*--------------------- 9/25/2024 up to page 31*

Maintainability

It’s important to structure your workloads and set up your infrastructure in such a way that different contributors can work using tools that they are comfortable with, instead of one group of contributors forcing their tools onto other groups. Code should be documented

*--------------------- 9/26/2024 up to page 31*

Code, data, and artifacts should be versioned. Models

should be sufficiently reproducible so that even when the original authors are not

around, other contributors can have sufficient contexts to build on their work

*--------------------- 9/27/2024 up to page 31*

Adaptability   
To adapt to shifting data distributions and business requirements, the system should have some capacity for both discovering aspects for performance improvement and allowing updates without service interruption  
*--------------------- 9/28/2024 up to page 31*

Because ML systems are part code, part data, and data can change quickly, ML systems need to be able to evolve quickly

*--------------------- 9/29/2024 up to page 31*

Iterative Process

the process looks more like a cycle with a lot of back and forth between different steps.

1. Choose a metric to optimize. For example, you might want to optimize for impressions—the number of times an ad is shown.
2. During error analysis, you realize that errors are caused by the wrong labels, so you relabel the data.
3. During error analysis, you realize that your model always predicts that an ad shouldn’t be shown, and the reason is because 99.99% of the data you have NEGATIVE labels (ads that shouldn’t be shown). So, you must collect more data of ads that should be shown
4. The model performs well on your existing test data, which is by now two months old. However, it performs poorly on the data from yesterday. Your model is now stale, so you need to update it on more recent data
5. The model seems to be performing well, but then the businesspeople come knocking on your door asking why the revenue is decreasing.
6. So, you want to change your model to optimize for ad click-through rate instead and go to step 1.
7. A diagram of a process

   Description automatically generated

*--------------------- 9/30/2024 up to page 33*

Step 1. Project scoping A project starts with scoping the project, laying out goals, objectives, and constraints. Stakeholders should be identified and involved. Resources should be estimated and allocated.

*--------------------- 10/1/2024 up to page 34*

Step 2. Data engineering A vast majority of ML models today learn from data, so developing ML models starts with engineering data

*--------------------- 10/2/2024 up to page 34*

Step 3. ML model development With the initial set of training data, we’ll need to extract features and develop initial models leveraging these features

*--------------------- 10/3/2024 up to page 34*

Step 4. Deployment After a model is developed, it needs to be made accessible to users. Developing an ML system is like writing—you will never reach the point when your system is done. But you do reach the point when you have to put your system out there.

*--------------------- 10/4/2024 up to page 34*

Step 5. Monitoring and continual learning Once in production, models need to be monitored for performance decay and maintained to be adaptive to changing environments and changing requirements.

Step 6. Business analysis

Model performance needs to be evaluated against business goals and analyzed  
to generate business insights. These insights can then be used to eliminate unproductive projects or scope out new projects.

*--------------------- 10/5/2024 up to page 34*

Framing ML Problems

Frame the ML problems into inputs and output and this function going to improve what the businesspeople expecting.

Types of ML Tasks

A diagram of a task type

Description automatically generated

Classification versus regression

Classification models classify inputs into different categories. For example, you want to classify each email to be either spam or not spam. Regression models output a continuous value. An example is a house prediction model that outputs the price of a given house.

regression model can easily be framed as a classification model and vice versa.

* house prediction can become a classification task if we quantize the house prices into buckets such as under $100,000, $100,000–$200,000, $200,000–$500,000, and so forth
* The email classification model can become a regression model if we make it output values between 0 and 1, and decide on a threshold to determine which values should be SPAM (for example, if the value is above 0.5, the email is spam).

Binary versus multiclass classification

Within classification problems, the fewer classes there are to classify, the simpler the problem is.

where there are only two possible classes. Examples of binary classification include classifying whether a comment is toxic, whether a lung scan shows signs of cancer. When there are more than two classes, the problem becomes *multiclass classification*. Dealing with binary classification problems is much easier than dealing with multiclass problems.

When the number of classes is high, such as disease diagnosis where the number of diseases can go up to thousands or product classifications where the number of products can go up to tens of thousands, we say the classification task has *high cardinality*.

When the number of classes is large, hierarchical classification might be useful. In hierarchical classification, you have a classifier to first classify each example into one of the large groups. Then you have another classifier to classify this example into one of the subgroups.

Multiclass versus multilabel classification:

In both binary and multiclass classification, each example belongs to exactly one class. When an example can belong to multiple classes, we have a multilabel classification problem.

**label multiplicity problem**

For example, an annotator might believe an example belongs to two classes while another annotator might believe the same example to belong in only one class, and it might be difficult resolving their disagreements.

Multiple ways to frame a problem

*Given the problem of predicting the app a user will most likely open next, you can frame it as a classification problem. The input is the users features and environments features. The output is a distribution over all apps on the phone.*

This is a bad approach because whenever a new app is added, you might have to retrain your model from scratch, or at least retrain all the components of your model whose number of parameters depends on *N*.

A better approach is to frame this as a regression task.  
The input is the user’s, the environment’s, and the app’s features. The output is a single value between 0 and 1; the higher the value, the more likely the user will open the app given the context.

A diagram of a problem

Description automatically generated with medium confidence

In this new framing, whenever there’s a new app you want to consider recommending to a user, you simply need to use new inputs with this new app’s feature instead of having to retrain your model or part of your model from scratch.

Objective Functions:

To learn, an ML model needs an objective function to guide the learning process. An objective function is also called a loss function, because the objective of the learning process is usually to minimize (or optimize) the loss caused by wrong predictions. For supervised ML, this loss can be computed by comparing the model’s outputs with the ground truth labels using a measurement like root mean squared error (RMSE) or cross entropy.

------- page no: 40

Entropy code example

------- page no: 41

Decoupling objectives

Now two objectives are in conflict with each other. If a post is engaging but it’s of questionable quality, should that post rank high or low?

You can randomly test out different values of α and β to find the values that work best. If you want to be more systematic about tuning these values, you can check out Pareto optimization, “an area of multiple criteria decision making that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously.”15 A problem with this approach is that each time you tune α and β—for example, if the quality of your users’ newsfeeds goes up but users’ engagement goes down, you might want to decrease α and increase β—you’ll have to retrain your model.

Another approach is to train two different models, each optimizing one loss. So you have two models: quality\_model Minimizes quality\_loss and outputs the predicted quality of each post engagement\_model Minimizes engagement\_loss and outputs the predicted number of clicks of each post

You can combine the models’ outputs and rank posts by their combined scores: ɑ quality\_score + β engagement\_score Now you can tweak α and β without retraining your models! *In general, when there are multiple objectives, it’s a good idea to decouple them first because it makes model development and maintenance easier*. First, it’s easier to tweak your system without retraining models, as previously explained. Second, it’s easier for maintenance since different objectives might need different maintenance schedules. Spamming techniques evolve much faster than the way post quality is perceived, so spam filtering systems need updates at a much higher frequency than quality-ranking systems.

*--------------------- 1/2/2024 up to page 43*

Mind Versus Data

Progress in the last decade shows that the success of an ML system depends largely on the data it was trained on. Instead of focusing on improving ML algorithms, most companies focus on managing and improving their data.

The debate isn’t about whether finite data is necessary, but whether it’s sufficient. The term *finite* here is important, because if we had infinite data, it might be possible for us to look up the answer. Having a lot of data is different from having infinite data.

A graph with a line going up

Description automatically generated

Every project must start with why this project needs to happen, and ML projects are no exception. We started the chapter with an assumption that *most businesses don’t care about ML metrics unless they can move business metrics*. Therefore, if an ML system is built for a business, it must be motivated by business objectives, which need to be translated into ML objectives to guide the development of ML models.

There are still many people who believe that having intelligent algorithms will eventually trump having a large amount of data. *However, the success of systems including AlexNet, BERT, and GPT showed that the progress of ML in the last decade relies on having access to a large amount of data.* Regardless of whether data can overpower intelligent design, no one can deny the importance of data in ML.

*--------------------- 1/3/2024 up to page 48*

Data Engineering Fundamentals

Data models define how the data stored in a particular data format is structured.

*--------------------- 1/4/2024 up to page 50*

Data Sources

One source is *user input data*, data explicitly input by users. User input can be text, images, videos, uploaded files, etc

*--------------------- 1/5/2024 up to page 50*

Another source is *system-generated data*. This is the data generated by different components of your systems, which include various types of logs and system outputs such as model predictions.

*--------------------- 1/6/2024 up to page 50*

debugging ML systems is hard, its a common practice to log everything you can. This means that your volume of logs can grow very, very quickly. This leads to two problems. The first is that it can be hard to know where to look because signals are lost in the noise. There have been many services that process and analyze logs, such as Logstash, Datadog, Logz.io, etc. Many of them use ML models to help you process and make sense of your massive number of logs.

In most cases, you only have to store logs for as long as they are useful and can discard them when they are no longer relevant for you to debug your current system.

when users enter a search query on Amazon, one or more ML models process that query to detect its intention—if someone types in “frozen,” are they looking for frozen foods or Disney’s *Frozen* franchise?—then Amazon needs to check its internal databases for the availability of these products before ranking them and showing them to users

*--------------------- 1/7/2024 up to page 51*

The rise of the internet and smartphones has made it much easier for all types of data to be collected. It used to be especially easy with smartphones since each phone used to have a unique advertiser ID—iPhones with Apple’s Identifier for Advertisers (IDFA) and Android phones with their Android Advertising ID (AAID)—which acted as a unique ID to aggregate all activities on a phone.

*--------------------- 1/8/2024 up to page 52*

Data Formats

The process of converting a data structure or object state into a format that can be stored or transmitted and reconstructed later is *data serialization.*

A table with text on it

Description automatically generated

JSON

JSON, JavaScript Object Notation, is everywhere. Even though it was derived from JavaScript, it’s language-independent—most modern programming languages can generate and parse JSON. It’s human-readable. Its key-value pair paradigm is simple but powerful.

JSON files are text files, which means they take up a lot of space.

Row-Major Versus Column-Major Format

The two formats that are common and represent two distinct paradigms are CSV and Parquet. CSV (comma-separated values) is row-major, which means consecutive elements in a row are stored next to each other in memory. Parquet is column-major, which means consecutive elements in a column are stored next to each other.

Because modern computers process sequential data more efficiently than nonsequential data, if a table is row-major, accessing its rows will be faster than accessing its columns in expectation.

*--------------------- 1/9/2024 up to page 54*

A diagram of a column

Description automatically generated

Overall, row-major formats are better when you have to do a lot of writes, whereas column-major ones are better when you have to do a lot of column-based reads.

NumPy Versus pandas

pandas, library is built around the columnar format.

A screenshot of a computer code

Description automatically generated

CSV to be a horrible data format. It serializes nontext characters poorly. For example, when you write float values to a CSV file, some precision might be lost—0.12345678901232323 could be arbitrarily rounded up as “0.12345678901”.

*--------------------- 1/10/2024 up to page 56*