The Size and Quality of a Data Set

"Garbage in, garbage out"

The preceding adage applies to machine learning. After all, your model is only as good as your data. But how do you measure your data set's quality and improve it? And how much data do you need to get useful results? The answers depend on the type of problem you're solving.

The Size of a Data Set

As a rough rule of thumb, your model should train on at least an order of magnitude more examples than trainable parameters. Simple models on large data sets generally beat fancy models on small data sets. Google has had great success training simple linear regression models on large data sets.

What counts as "a lot" of data? It depends on the project. Consider the relative size of these data sets:

Data set	Size (n examp
Iris flower data set (https://archive.ics.uci.edu/ml/datasets/iris)	150 (to
MovieLens (the 20M data set) (https://grouplens.org/datasets/movielens/20m/)	20,000 (total s
Google Gmail SmartReply (https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45189.pdf)	238,00 (trainin
Google Books Ngram	468,00 (total s
Google Translate	trillions

As you can see, data sets come in a variety of sizes.

The Quality of a Data Set

It’s no use having a lot of data if it’s bad data; quality matters, too. But what counts as "quality"? It's a fuzzy term. Consider taking an empirical approach and picking the option that produces the best outcome. With that mindset, a quality data set is one that lets you succeed with the business problem you care about. In other words, the data is *good* if it accomplishes its intended task.

However, while collecting data, it's helpful to have a more concrete definition of quality. Certain aspects of quality tend to correspond to better-performing models:

- reliability
- feature representation
- minimizing skew

Reliability

Reliability refers to the degree to which you can *trust* your data. A model trained on a reliable data set is more likely to yield useful predictions than a model trained on unreliable data. In measuring reliability, you must determine:

- How common are label errors? For example, if your data is labeled by humans, sometimes humans make mistakes.
- Are your features noisy? For example, GPS measurements fluctuate. Some noise is okay. You’ll never purge your data set of *all* noise. You can collect more examples too.
- Is the data properly filtered for your problem? For example, should your data set include search queries from bots? If you're building a spam-detection system, then likely the answer is yes, but if you're trying to improve search results for humans, then no.

What makes data unreliable? Recall from the Machine Learning Crash Course (/machine-learning/crash-course/representation/cleaning-data) that many examples in data sets are unreliable due to one or more of the following:

- Omitted values. For instance, a person forgot to enter a value for a house's age.
- Duplicate examples. For example, a server mistakenly uploaded the same logs twice.
- Bad labels. For instance, a person mislabeled a picture of an oak tree as a maple.
- Bad feature values. For example, someone typed an extra digit, or a thermometer was left out in the sun.

Google Translate focused on reliability to pick the "best subset" of its data; that is, some data had higher quality labels than other parts.

Feature Representation

Recall from the Machine Learning Crash Course (/machine-learning/crash-course/representation/feature-engineering) that representation is the mapping of data to useful features. You'll want to consider the following questions:

- How is data shown to the model?
- Should you normalize (/machine-learning/glossary#normalization) numeric values?
- How should you handle outliers (/machine-learning/glossary#outliers)?

The Transform Your Data (/machine-learning/data-prep/transform/introduction) section of this course will focus on feature representation.

Training versus Prediction

Let's say you get great results offline. Then in your live experiment, those results don't hold up. What could be happening?

This problem suggests training/serving skew—that is, different results are computed for your metrics at training time vs. serving time. Causes of skew can be subtle but have deadly effects on your results. Always consider what data is available to your model at prediction time. During training, use only the features that you'll have available in serving, and make sure your training set is representative of your serving traffic.

olden Rule: Do unto training as you would do unto prediction. That is, the more closely your training task es your prediction task, the better your ML system will perform.

Suppose you have an online store and want to predict how much money you'll make on given day. Your ML goal is to predict daily revenue using the number of customers as a feat What problem might you encounter? Click the plus icon to check your answer.

The problem is that you don't know the number of customers at prediction time, before the day's sales are complete. So, this feature isn't useful, even if it's strongly predictive of your daily revenue. Relatedly, when you're training a model and get amazing evaluation metrics (like 0.99 AUC (/machine-learning/glossary#AUC)), look for these sorts of features that can bleed into your label.

Previous
← Introduction to Constructing Your Dataset (/machine-learning/data-prep/construct/construct-intro)

Next
Joining Logs (/machine-learning/data-prep/construct/collect/joining-logs) →