Machine Learning Survey

Candidate Criteria

Relevance, Recency

How do we choose what we evaluate? **This field is undergoing a lot of change and activity. To make this survey/comparison as relevant as possible, we will not consider ML solutions that are not under active development.**

In the case of libraries, this means we'll ignore any software project that doesn't show either a reasonable pace of recent releases and updates.

In the case of services, this means active evolution, support, up-to-date documentation.

Language or Platform Support (?)

We could limit our analysis to the following specific languages:

- Which languages?

OR do we want to approach this as a 'platform' issue, that is, say,

if you're developing for Windows, we look at X

If you're developing for the web (mobile or otherwise) we look at Z

If you're developing for mobile/android, mobile/iOS, mobile/Windows… etc

But this seems like it could easily get out of hand.

OR, the other option is focus on the systems and not the languages. Dev environment and languages change all the time, and after all the whole point of web services is that you just switch to a new wrapper.

We'd then end up with examples of use cases in different languages or environments, since the point is to evaluate the ML technology itself. When the platform or language makes a big difference we could note it. For example, I'd expect Google stuff to have better Go, Java and Python support — (Java for Android)

Evaluation Criteria

It's important to note that the elements discussed here are not plainly "good" or "bad". Any conclusion regarding a service must take into account the whole package.

1. **Solution focus.** General? Vision? NLP?
2. **Target user.** Developers? Data Scientists? etc.
3. **Programming Language.** For Services, this can be less of an issue since you can theoretically write a wrapper on any language to

a remote API, but we will rely on the service's provided wrappers for evaluation.

4. **Service control & requirements.** Fully hosted (e.g. TensorFlow) or something you can download and install? (e.g. Apache Singa). Sidenote to this is whether you Can you evolve from cloud-hosted to in-house? ie., can you have set up hosting yourself? Is there a choice in this regard?
5. **Management and operational resources needs.** Does it require you to manage the details of the service, or is it maintained and scaled for you? Do you need monitoring for it? etc.
6. **SLAs.** constraints? Limitations? And very important: Service level commitment? Uptime commitment? Max failure rate commitment?
7. **Data control & access.** This could be fall under the umbrella of SLAs but it has enormous implications for any project so we'll keep it a separate point. It involves ownership and access of data, privacy issues, backups, exports, etc. For example, if Google kills TensorFlow tomorrow, do you lose everything? What can you export out of it that is immediately/directly usable? (many services allow you to export data that is then not really usable because of format issues, etc). Who 'owns' the training set data? etc. etc.
8. **Price & Terms.** Not just for the product, but in terms of evolution. Can you start free or small and grow? Or does it force you to talk to a sales rep to run Hello World?
9. **Activity, relevance.** How actively is it in development? How many resources are being put against it? How crucial is it for the company? E.g. for a startup that only does that it would be critical, for IBM it may be a drop in a bucket
10. **"Bonus." various items** IDE support? Web interface? Ability to experiment without full implementation? etc.

Questions/TBD

OK that we don't care about "libraries"?

Arguably deep learning requires deploying and running something at relatively large scale. Libraries by themselves are very far from something that is "usable" in the context that we're interested in.

What we wouldn't include: For example, FANN ("Fast Artificial Neural Network Library", see http://leenissen.dk/fann/wp/) is a fairly comprehensive open source library written in C with bindings for many other languages (http://leenissen.dk/fann/wp/language-bindings/). This is the type of project we would not include in our survey.

What we would include: systems that, while possibly requiring deployment and many resources, provide a solution that is as close to workable as possible to implement ML solutions. Example: Apache Singa http://singa.apache.org/docs/overview.html, a "general distributed deep learning platform for training big deep learning models over large datasets".

We will also skip Foundation-type projects or services, such as Apache Spark, which are not specifically ML-oriented but may be frequently used as part of providing ML solutions. We will look at a project as a whole. In some cases the underlying framework in use could have visible effects on efficiency, speed, etc (e.g., Spark vs. Hadoop) and we may note that but in the context of the system in question, so it's conceivable that in one type of project a dependency on Spark could be a net plus and in another neutral or slightly negative, depending on libraries, use, etc.

Terminology

We'll include terms in this section even if they are arguably well-known.

- **Library.** A collection of resources used to develop software, typically embedded within the final product. Libraries can include configuration, templates, sources, data, and any amount of code that can be used directly with knowledge of an API. We'll also consider projects that require internal setup and management Libraries. For example, under this definition, Apache Spark is a library, not a service, even though technically it is a service that you deploy and manage yourself.
- **Service.** We'll generally refer to a service as something *only* accessible remotely to the developer, and that is *not* controller by the developer/organization.
- **SLA.** Service level agreement. Particular elements of a contract between the service provider and the service user that cover scope, expectations and responsibilities of each party. These may include uptime expectations, mean time between failures (MTBF), mean time to repair or mean time to recovery (MTTR); responsibility for data rates, throughput, error rates, and other measurable details.