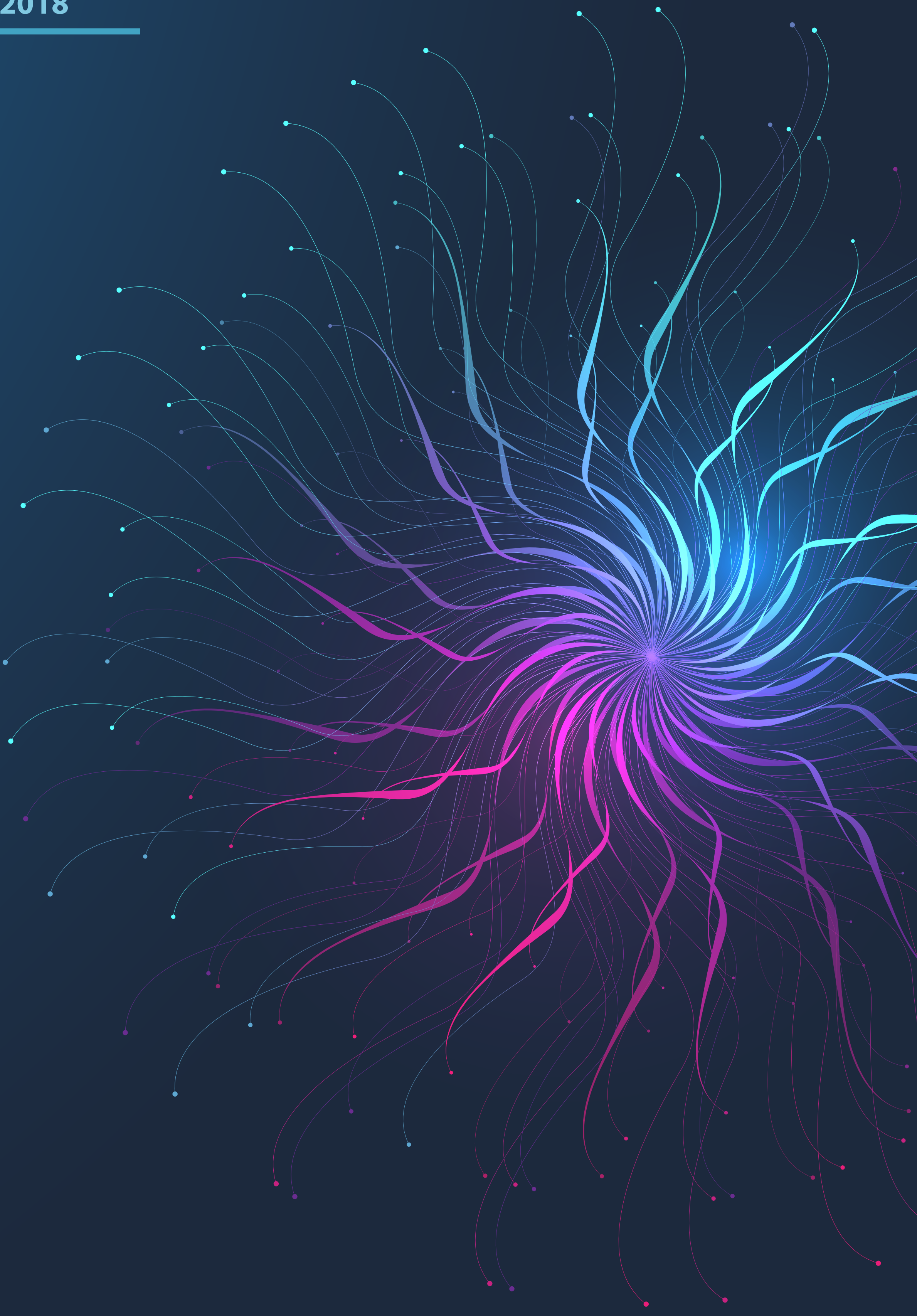
BUILDING A COMMERCIALLY VIABLE MACHINE LEARNING / DEEP LEARNING SOLUTION

Madhusudhan B A
Whirldata Inc. ©2018

Whirldata



A primer on factors to be considered for companies foraying into Machine Learning / Deep Learning and attempting to build experimental projects, solutions or Proof of Concepts based on Machine Learning.

In an effort to rapidly build Machine Learning/Deep Learning capabilities, several data analytics and data sciences organizations/startups go through a plethora of online training courses with Andrew Ng's Machine Learning course and Deep Learning Specialization (five MOOC courses) being the most sought after and popular ones.

The logical next step after completion of these courses is to look for real world problems (I shall discuss what constitutes a "real-world" problem in another article) to practice on or use for evaluating capabilities and continued learnings.

Once a problem has been identified (either for a potential client or has been created for the sake of solving) typical tendency has been to start evaluating approaches/algorithms that are most suited for solving the problem, and in parallel the process of data collection begins in an ambitious manner. There are those who also delve deep to understand the data and what story it tells before determining what algorithm to use. Data is then fed through the chosen algorithm to obtain results.

However, an important step that should not be missed is the analysis of the several comprehensive factors (discussed below) and the role they play in determining the architecture of the solution.

I define a solution as the most optimal pipeline comprising of

- Pre-processing of data
- Hand engineering (always required in some manner)
- Algorithms (a selection of several ones – many of which will seem right)
- Technologies/platforms for integration and deployment that are put together to get the desired outcome.

Factors that determine the architecture of a solution

While some level of brevity is attempted for this write-up, I can state with a reasonable degree of confidence that omitting even one factor below can lead to a non-optimal solution.

The corollary to the necessity to consider several factors is that one cannot develop a POC for a generalized ML solution and neither is it possible to develop a general solution that works for several applications. It is my aim to defend this statement through the rest of the write-up.

1. Cost: How much is the customer / sponsor willing to pay for this project? A ballpark is necessary to even determine various pieces of technologies and time that needs to be dedicated to such a project.
If one subscribes to the view that the market for ML based solutions are still maturing, it is quite likely that a customer's ability to define their need and accordingly allocate a budget will be limited. So better to know sooner than later the budgets available for this solution. A solution with a budget of US$10K will be vastly different from a solution with a budget of US$100K. It is impossible to assume a cost and build a solution – for there will always be an improvement possible in any of the factors below with a marginal increase in cost. So where does one draw the line? How does one defend the boundaries?

That's where knowing the budget comes into play.

2. Speed: The primary trade-offs are usually between speed(running time) and accuracy. Secondary trade-offs will be between speed and other factors. In some cases, the time taken for training may also become a factor - though rarely. The business case determines the trade-offs. It is essential that the running speeds required are known a-priori.
Speed will always be a trade-off with one or more factors below. For instance, a mobile app that helps to scan products in a retail store and use AR to provide additional information needs to work at lightning speeds as enhanced shopper's experience is the only benefit from such an app. However, a similar AR app built for use within a controlled environment (such as some applications in a manufacturing setup) would require Accuracy to be higher than Speed. The app could still reside on a mobile device but the need for speed can make a significant difference in the architecture of the solution. As the processing capability of mobile platforms are limited, various trade-offs including what is to be processed in-house (on the mobile platform) and what is faster to send out to a server and obtain (additional data transfer latency but with increased processing power of the server). I am trivializing the decisions for the sake of illustration, but as one goes through this decision-making process, the complexities concerning the trade-offs will be quite evident.

3. Accuracy: Very closely tied to speed, deployment platform and several other actors. For determining the accuracy required, it is necessary to find every evaluation metric that has an impact on the business outcomes for which this solution is being built (There is a completely different discussion on what

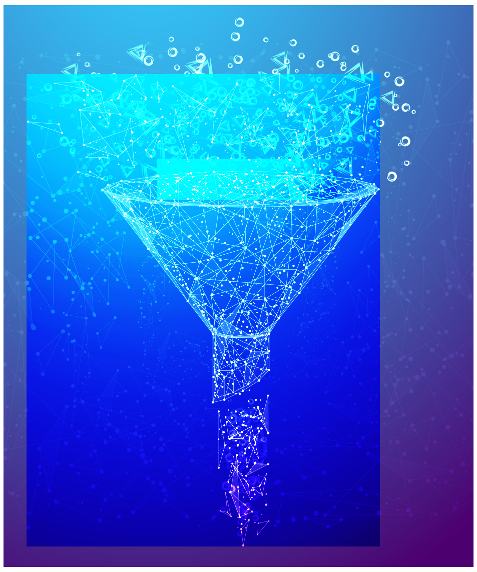
constitutes accuracy – precision, recall, F1 score, optimizing and satisficing conditions etc, and that's being saved for a later discussion). These evaluation metrics have to come from different stakeholders within the organization. Very often not having sufficient input from different stakeholders, skews evaluation metrics and outcomes of the solution may not end up being used productively and to demonstrate ROI. Only when the evaluation metrics are laid out on the table and determined, does it even allow for us to discuss the numerical values for these metrics. Some metrics may be qualitative. Only when numerical/qualitative values for these metrics and their acceptable bounds have been arrived at, does it help to even initiate efforts to determine what would be a single number evaluation metric that can capture the accuracy of the solution reliably and include all the metrics. This is the point where factors like precision, recall and everything else come into the picture.

The most significant point to note is that these evaluation metrics are different for different organizations even if the solution is identical in every other sense. This is another reason where efforts to develop one-size-fits-all solution efforts usually fail.

While such an endeavor works well in most domains, when it comes using Machine Learning, one-size-fits-all scenario is not the right strategic method to display capabilities or do POCs.

4. Availability & quality of data: While it's good to have a lot of training data, there are only certain conditions under which more data helps. (I will probably discuss a bit of this below and elaborate in

another write-up). If you look again at the constituents of a solution, the components of the pipeline
vary with the change in the determining factors. And this includes the change in requirement of the dataset. A dataset is of the form that it allows the learning algorithm to capture variations which eventually help in solving a business objective. For instance, a face recognition system for a controlled environment like an office building would be quite different from the one used in a shopping mall and so does the training data. The dataset ideally suited for a face identification system could be different from the one needed for a face recognition system. Wider variations need not necessarily mean a larger dataset, it means a different training philosophy for which a different dataset may be more amenable. In come cases a pretrained model will be essential. While I understand that there is a lot of generalization and ambiguity in the statements I have made, the crux is to make the point that datasets need a



very good thought that depends on the end business objective and very careful attention needs to be paid to its source, availability and quality.

5. Testing & Validation:

Generally accepted practices exist for determining the nature and size of the training set, development set and test set. It is an accepted rule that the training set and development set should contain data from the same distribution – which would also be an ideal representation of the test set (or the field set). Test set is a surrogate for expected inputs in real-world that determines performance - delving into the mechanics behind splitting data into these sets are beyond the scope of this article. However, there exists several scenarios where the training set and development set have to come from a different distribution than the test set. In such cases, expectations of performance metrics (discussed in point 3 above) will also have to be in sync with this fact. The ability for different stakeholders within the organization to understand the impact of the training, development and test set on the solution is essential and cannot be considered as a problem left only for the team championing the development of this solution – as the evaluation metrics have to be tailored to meet the nature of the training, development and test data available. Attempting to lay down evaluation metrics without considering the limitations here can affect the usability or benefits of the solution.

6. Alternate methods (non ML/DL) available for solving a similar problem:

Ideally, needs to be the first step, but practically has to be taken up in parallel. The first question is – why does this problem need a solution based on Machine Learning? There has to be a precise answer for that. There seems to be a slight tendency today to take every problem (especially for most non-natural perception tasks like recommendation, classification and prediction and Image analysis, a natural perception task) and force fit a Machine Learning algorithm onto it. If one were to take every prediction problem or a classification problem and subject it to structured data analysis methods and visualization, they would find that most of those tasks can be accomplished without an ML based solution. **Interestingly I have come across many challenges on Hackerearth and Kaggle as ones to be solved using ML/DL – but on closer scrutiny lend themselves to traditional methods of solving them that result in better accuracy.** The same situation applies to certain image recognition problems where Computer Vision would be able to provide a much better solution in all aspects than a ML based solution. I would like to highlight that for the purpose of this argument I do not consider regression problems as something based on ML (though many seem to consider so and I do not want to debate that yet).

7. Viability of alternate methods:

It is not just sufficient to identify alternate methods but also need to evaluate if the development cost and running costs are similar to a ML based solution provided the performance from both approaches are somewhat identical.
This may happen only in rare cases, but nevertheless something that needs to be considered.

8. Internal knowledge levels:

Closely tied to all factors that require evaluation metrics to be identified and values established. As I have said earlier, all stakeholders in an organization need to provide and accept the evaluation metrics that went into a solution. Some basic knowledge of ML is essential for quick turnaround of these metrics, else this process can take a long time. The easier way is to conduct a quick training session for the stakeholders on ML and provide a skeleton of the expected solution so that they can think in similar lines and help provide evaluation metrics. There are many cases today where I have seen no evaluation metrics provided but one or more efforts to develop a ML based
solution is in progress.

9. Deployment platform:

Encompasses several aspects that determines most other factors mentioned here. A major role in the expected cost too. I cannot fathom how one can even develop a POC without knowing the end deployment platform. One can develop very fancy POCs for a particular app to run on a mobile platform by outsourcing all processing to an external/powerful GPU. But if the same app were to be contained within a mobile platform, the entire architecture will change. It's not just the algorithms or models, it is pretty much everything mentioned here. It's not difficult to see why. Any constraint on the processing power will show up in the form of accuracy or speed or of the algorithm or the size of the trained model than can reside in a mobile device etc. Everything else linked to any of these factors will be affected and some point will impact evaluation metrics.

10. Solution's impact on business:

Mostly applicable for cases where a small development-ready-prototype is being developed for your customer. When there are no budgets and the development is to prove capability – what cost-bounds does one use? In such cases, attempt to find the commercial value of this solution for your customer/sponsor's business. Is it a revenue generating effort, a cost reduction
effort or a customer experience enhancing effort? What is the main purpose? Ask these questions and quantify their impact to the business. The rest of your solution should take this cost as a benchmark for determining other factors. This will also prevent you from taking on prototypes or POCs which will never
see the light of a large scale deployment due to lack of commercial viability. A solution has to fit the business case it is designed for. If you are doing it for free, make sure you analyze the business case before taking on an effort.

To conclude, please be aware that data sciences, Machine Learning and Artificial Intelligence are domains where the intersection of technical and business knowledge is the highest ever and traditional methods for evaluating projects and determining architectures do not work here. One may call it a categorical statement, but I am pretty certain that those who have gone down the path can empathize with it.

To know more visit www.whirldatascience.com or write to us at itspossible@whirldatascience.com