Intersection

Where Machine Learning Meets Product Management and Design

Slides and Summary: Gene Olafsen

Reference Material

- These slides are based on the following post:
  - https://towardsdatascience.com/machine-learning-for-product-managers-part-i-problem-mapping-5436132c3a6e

Misconception

- Machine Learning fundamentally changes the product design process and the role of the project manager.

Focus on User Needs

- Great products focus on user needs.
- Great products, leveraging machine learning technologies, focus on user needs.

Adam Nash (LinkedIn, Ebay, Apple...)

- "Done properly, a strong product leader acts as a force multiplier that can help a cross-functional team of great technologies and designers do their best work."
- Over the years he has reduced this communication to just three sets of responsibilities:
  - Product Strategy
  - Prioritization
  - Execution
- https://adamnash.blog/2011/12/16/be-a-great-product-leader/

Reflecting on Product Strategy

- Nash continues, "it's the product manager's job to articulate two simple things":
  - What game are we playing?
    - Vision for the product.
    - Value to the customer.
    - Differentiation from the competition.
    - How to win in the market.
  - How do we keep score?
    - Metrics used to judge success.

The "ML Everywhere" Tool

- Access to AI/ML technology does not automatically validate its usage.
- Every problem shouldn't look like a 'nail' because you have an ML hammer.
- Machine learning is not an end unto itself.

Where the "ML Everywhere" Tool May Be Appropriate...

- If you are actually in the business of developing cutting-edge machine learning technology.
- The product is the ML technology, the technology is the product.
- If you are a PM for such a technology company your role may involve working with companies to exploit your solution to support the needs of users outside your organization. These outside needs may feedback to your job function to help fine-tune the technology under development.

Which Business Problems are Appropriate for ML?

- Simply stated: ML is best suited for problems that involve some sort of **pattern recognition.**

Applicable Business Problems

- User Faces Too Much Data
- Complex Cognition Problems
- Predicting and Forecasting
- Anomaly Detection
- Recommendations
- Experience Creation/Augmentation

Too Much Data!

- Sifting through large quantities of data is a perfect job for ML/AI. In fact search engines use such algorithms to surface the *best* results.

Complex Cognition

- The source material speaks of self-driving car and photo location identification tasks as examples. Stating, *"These require complex cognition skills and the only way to build these kinds of smart machines is to feed it a lot of data and learn through pattern recognition."*
  - Complex – Certainly, autonomouse driving functions require vast compute resources to achieve *safe* outcomes.
  - Tedious - Photo object and location identification

Predicting and Forecasting

- ML models are *comfortable* in prediction and forecasting situations. But so are less flashy, traditional methods… linear regression.
- The article states that ML can be useful in forecasting sales if "business fundamentals haven’t changed drastically".

Anomaly Detection

- "Since ML is good at pattern recognition, anything that doesn't fit the pattern of behavior that is considered the norm, can be detected very easily."
- One such example offered by the article is *fraud detection.*
- Anomalies can extend to the physical/mechanical world:
  - Vibration
  - Direction
  - Speed
  - Traction

Recommendations

- Recommendation engines abound!
- Certainly there are plenty of non-ML techniques that can be employed to produce recommendations- and may be the only option you have in *cost start* situations.

Experience Creation / Augmentation

- So this application is interesting...
- "SnapChat filters are a great example of how experiences can be augmented using ML. By using facial recognition algorithms, SnapChat filters are able to detect the contours of a face, and thus overlay creative filters over a face, thus making photo sharing more fun."

Product Manager Skills

- Product managers typically use five core skills:
  - customer empathy/design chops
  - communication
  - collaboration
  - business strategy
  - technical understanding

Iron Triangle of Project Management

Time

- Managing time is closely related to managing tasks, as the overall timeline is broken down by individual tasks and their anticipated timing.

Cost

- The financial constraint of a project, or *budget.*
- Costs on a project can comprise of a variety of elements, including resources – both materials and people – as well as any external costs that influence completion.

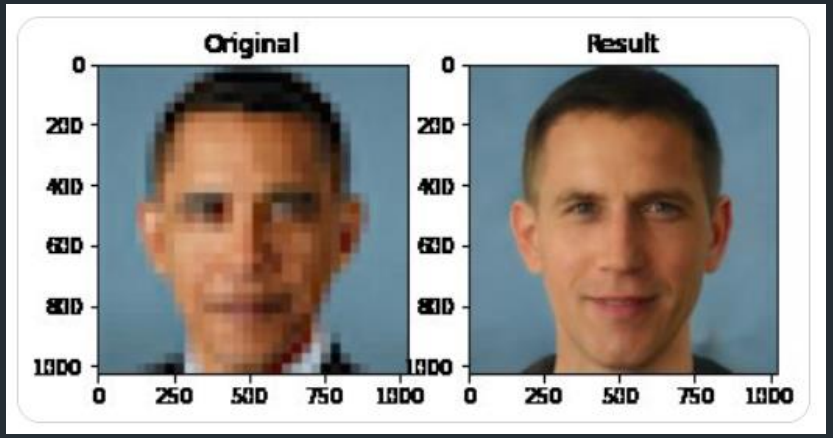
Scope

- Controlling scope is especially critical, as adjustments to scope almost guarantee an impact to cost and time.

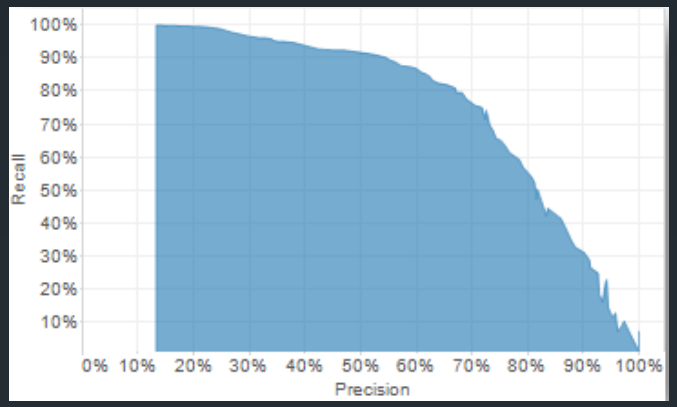
Mind the Gap

- The real world is a messy place and machine learning algorithms have limitations.
- "B*eing able to understand the impact and limitations of the algorithms, helps you understand what gaps would exist in the customer experience and solve for those gaps through product design or by choosing other ML techniques. This is an absolutely necessary skill-set if you are a PM in this space."*
  - Biases in Data
  - Precision/Recall Tradeoff
  - Cold Start Issues
  - User Feedback
  - Explore vs Exploit

Data Bias



- Machine learning learns patterns from the data on which it is trained.
- Ensure that the data is representative of the users who will use the product.
- In a recent example, a machine learning algorithm that is intended to enhance blurry images (or low resolution images), produced Obama with white features.



Precision/Recall

- **Precision and recall** are two extremely important model evaluation metrics.
- **Precision** refers to the percentage of your results which are relevant.
- **Recall** refers to the percentage of total relevant results correctly classified by your algorithm. Recall is also known as *sensitivity*.

Cold Start

- *Cold start* refers to situations where there is little information on which ML algorithms can make decisions.
- The author of the paper describes two types of cold starts: *user* and *item*.
  - In the *user* case, this is the first interaction that the user has with the system. There is little history with the user and recommendations may not be appropriate for this account.
  - For the *item* case, a new item is introduced to the system and it is difficult to identify users who may be interested in the product.
- I think that both of these cases are contrived and while both may present an issue, the issues can be better generalized.

Cold Start Revisited

- If the *user* and *item* cold starts are specific to all systems, or in fact, even to the system modeled in the paper- then how does the system even deal with "sub-optimal experiences"?
- The *userid* and *itemid* are unique to these records, so yes they will not be useful *features* when predicting outcomes for associated models. However, there are certainly many other features on which the model relies, which should provide an acceptable *instance* operation from the model.
- The larger question is- how to bootstrap an 'empty' system?

Initial Bootstrap

- For this example we will consider a recommendation engine.
  - An *ensemble* of models (or algorithms) may be employed to offer possible selections.
  - One *algorithm* may simply recommend another item in the same *category*.
  - Another *algorithm* may recommend another item in the same price-range, or color, or other attribute.
  - Finally a model considering all of the selections to date will be updated on a regular (nightly, weekly, etc.) basis and offer it's selection as part of the *ensemble* logic.

User Feedback

- Since the ML algorithms aren't perfect, the resulting predictions and pattern recognitions can be incorrect.
- You may create mechanisms in your product to provide feedback to the algorithms, such that they can improve their accuracy over time.
- Of course the user not taking a recommendation is feedback in itself, but a more explicit mechanism (thumbs up/down) are often unobtrusive mechanims by which such information is acquired.

Explore vs Exploit

- Imagine a system that is designed to recommend meals. After a few months, the model identifies that you like *filet mignon*.
- Now when you leave the choice to the system to choose a meal, it selects *filet mignon*, and you are satisfied. This model is acting in an *exploit* mode.
- This is perfect, however it will never change, you may never get the chance to try something new. If the system occasionally provides a random selection, you may now find out that you like *lobster* even better than *filet mignon*. This model is acting in an *explore* mode.

Runtime Exploring and Exploiting

- Supervised Learning:
  - ML models are most *comfortable* making recommendations (provide *instancing* output) based on strong model *signal strength*-- which is a typical *exploiting* mechanism.
  - Whereas an *exploration* option may be the domain of a simpler algorithmic process.

Mistakes with ML

- I would describe these as *challenges* when deploying ML systems. ML is difficult to implement *correctly* and produce the results that you intend for all of the situations your product will encounter.
  - Data – None, Small, Sparse, High Dimensional
  - Fitting – Overfitting, Underfitting
  - Computational Cost