LESSONS LEARNED

from building Practical AI Systems

Xavier Amatriain (@xamat)
www.quora.com/profile/Xavier-Amatriain
AI @Quora
To share and grow the world's knowledge

→ Millions of questions & answers
→ Millions of users
→ Thousands of topics
→ ...

Quora
The best answer to any question
What we care about

- Quality
- Demand
- Relevance
Lots of high-quality data
Lots of data relations
AI Applications At Quora

→ Answer ranking
→ Feed ranking
→ Search ranking
→ User recommendations
→ Topic recommendations
→ Duplicate questions
→ Email Digest
→ Request Answers
→ Trending now
→ Topic expertise prediction
→ Spam, abuse detection
→ …
Models

→ Logistic Regression
→ Elastic Nets
→ Gradient Boosted Decision Trees
→ Random Forests
→ LambdaMART
→ Matrix Factorization
→ LDA
→ Deep Neural Networks
→ ...

\[ P = \frac{e^{a + bx}}{1 + e^{a + bx}} \]

\[ nX = nU \times hV^T \]

\[ \hat{\beta} = \arg\min_{\beta} \left( ||y - X\beta||^2 + \lambda_2 ||\beta||^2 + \lambda_1 ||\beta||_1 \right) \]
Lessons Learned
More Data vs. Better Models
More data usually beats better algorithms.

I teach a class on Data Mining at Stanford. Students in my class are expected to do a project that does some non-trivial data mining. Many students opted to try their hand at the Netflix Challenge: to design a movie recommendations algorithm that does better than the one developed by Netflix.

Here’s how the competition works. Netflix has provided a large data set that tells you how nearly half a million people have rated about 18,000 movies. Based on these ratings, you are asked to predict the ratings of these users for movies in the set that they have not rated. The first team to beat the accuracy of Netflix’s proprietary algorithm by a certain margin wins a prize of $1 million!

Different student teams in my class adopted different approaches to the problem, using both published algorithms and novel ideas. Of these, the results from two of
More data or better models?

Sometimes, it’s not about more data.
More data or better models?

Norvig:
“Google does not have better Algorithms only more Data”

The Unreasonable Effectiveness of Data
Alon Halevy, Peter Norvig, and Fernando Pereira, Google

Many features/low-bias models

Figure 1. Learning Curves for Confusion Set Disambiguation
More data or better models?

Sometimes you might not need all your “Big Data”
### What about Deep Learning approaches?

<table>
<thead>
<tr>
<th>Year</th>
<th>Breakthrough in AI</th>
<th>Datasets (First Available)</th>
<th>Algorithms (First Proposal)</th>
</tr>
</thead>
</table>

**Average No. Of Years to Breakthrough**

<table>
<thead>
<tr>
<th>Year</th>
<th>Breakthrough in AI</th>
<th>Datasets (First Available)</th>
<th>Algorithms (First Proposal)</th>
</tr>
</thead>
</table>

The average elapsed time between key algorithm proposals and corresponding advances was about 18 years, whereas the average elapsed time between key dataset availabilities and corresponding advances was less than 3 years, or about 6 times faster.
**What about Deep Learning approaches?**

<table>
<thead>
<tr>
<th>Models and Recipes</th>
<th>Available models trained using OpenNMT</th>
<th>More models coming soon:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretrained</td>
<td>• English → German</td>
<td>• Ubuntu Dialog Dataset</td>
</tr>
<tr>
<td></td>
<td>• German → English</td>
<td>• Syntactic Parsing</td>
</tr>
<tr>
<td></td>
<td>• English Summarization</td>
<td>• Image-to-Text</td>
</tr>
<tr>
<td></td>
<td>• Multi-way – FR, ES, PT, IT, RO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• FR, ES, PT, IT, RO</td>
<td></td>
</tr>
</tbody>
</table>

**GoogLeNet in Keras**

*Joe Marino - June 2016*
Simple Models beat Complex Models
Football or Futbol?
Given two models that perform more or less equally, you should always prefer the less complex.

Deep Learning might not be preferred, even if it squeezes a +1% in accuracy.
Occam’s razor: reasons to prefer a simpler model

Why would you want to use a linear model?

Why would you want to use so simple a model when recent research has demonstrated the power of more complex neural networks with many layers?

Linear models:

- train quickly, compared to deep neural nets.
- can work well on very large feature sets.
- can be trained with algorithms that don't require a lot of fiddling with learning rates, etc.
- can be interpreted and debugged more easily than neural nets. You can examine the weights assigned to each feature to figure out what's having the biggest impact on a prediction.
- provide an excellent starting point for learning about machine learning.
- are widely used in industry.
Occam’s razor: reasons to prefer a simpler model

There are many others

- System complexity
- Maintenance
- Explainability

- “Why Should I Trust You?”
  Explaining the Predictions of Any Classifier

  Marco Tulio Ribeiro
  University of Washington
  Seattle, WA 98105, USA
  marcotcr@cs.uw.edu

  Sameer Singh
  University of Washington
  Seattle, WA 98105, USA
  sameer@cs.uw.edu

  Carlos Guestrin
  University of Washington
  Seattle, WA 98105, USA
  guestrin@cs.uw.edu

- Machine Learning:
  The High-Interest Credit Card of Technical Debt

  D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
  Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young

  {ds@sculley, gm@holt, dgg@edavdyrov}@google.com
  {toddphillips, ebner, vchaudhary, myoung}@google.com
  Google, Inc.

Figure 3: GoogLeNet network with all the bells and whistles
A real-life example

**Goal: Supervised Classification**
→ 40 features
→ 10k examples

**What did the ML Engineer choose?**
→ Multi-layer ANN trained with Tensor Flow

**What was his proposed next step?**
→ Try ConvNets

**Where is the problem?**
→ Hours to train, already looking into distributing
→ There are much simpler approaches

---

**Fizz Buzz in Tensorflow**

interviewer: Welcome, can I get you coffee or anything? Do you need a break?

me: No, I've probably had too much coffee already!

interviewer: Great, great. And are you OK with writing code on the whiteboard?
But, sometimes you do need a Complex Model
Better models and features that “don’t work”

E.g. You have a linear model and have been selecting and optimizing features for that model
→ More complex model with the same features -> improvement not likely
→ More expressive features -> improvement not likely

More complex features may require a more complex model

A more complex model may not show improvements with a feature set that is too simple
Supervised vs. Unsupervised Learning
Supervised/Unsupervised Learning

Unsupervised learning as dimensionality reduction

Unsupervised learning as feature engineering

The “magic” behind combining unsupervised/supervised learning

E.g.1
Clustering + knn

E.g.2
Matrix Factorization

Unsupervised:
• Dimensionality Reduction a la PCA
• Clustering (e.g. NMF)

Supervised:
• Labeled targets ~ regression
One of the “tricks” in Deep Learning is how it combines unsupervised / supervised learning

→ E.g. Stacked Autoencoders

→ E.g. training of convolutional nets

Convolutional Network (CovNet)

- Input 25x25
- Layer 1 64x25x25
- Layer 2 64x14x14
- Layer 3 256x6x6
- Layer 4 256x1x1
- Output 101

9x9 Convolution (64 kernels)
10x10 pooling
5x5 subsampling
9x9 Convolution (4096 kernels)
6x6 pooling
4x4 subsampling

→ Non-Linearity: half-wave rectification, shrinkage function, sigmoid
→ Pooling: average, L1, L2, max

Why Does Unsupervised Pre-training Help Deep Learning?

Damiru Erhan*
Yoshua Bengio
Aaron Courville
Pierre-Antoine Manzagol
Pascal Vincent

*Department of Computer Science and Operations Research
University of Montreal
Montréal, Québec, H3T 1J8, Canada

Samy Bengio
Google Research
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA

BENGIO@GOOGLE.COM
Everything is an ensemble
Ensembles

Netflix Prize was won by an ensemble

- Initially Bellkor was using GDBTs
- BigChaos introduced ANN-based ensemble

Most practical applications of ML run an ensemble

- Why wouldn’t you?
- At least as good as the best of your methods
- Can add completely different approaches (e.g. CF and content-based)
- You can use many different models at the ensemble layer: LR, GDBTs, RFs, ANNs...
Ensembles & Feature Engineering

Ensembles are the way to turn any model into a feature!

E.g. Don’t know if the way to go is to use Factorization Machines, Tensor Factorization, or RNNs?

→ Treat each model as a “feature”
→ Feed them into an ensemble

- Sigmoid
- Rectified Linear Units
- Output Units
- Hidden Layers
- Dense Embeddings
- Sparse Features

Wide Models
Deep Models
Wide & Deep Models
The output of your model will be the input of another one (and other system design problems)
Outputs will be inputs

Ensembles turn any model into a feature
→ That’s great!
→ That can be a mess!

Make sure the output of your model is ready to accept data dependencies
→ E.g. can you easily change the distribution of the value without affecting all other models depending on it?

Avoid feedback loops

Can you treat your ML infrastructure as you would your software one?

Two big challenges in machine learning

LÉON BOTTOU
FACEBOOK AI RESEARCH
ICML 2015 - LILLE

Feedback loops in machine learning

“Information (signal) feedback loops are everywhere.”
“They are central to adaptation and learning...”

*See (Bottou et al., JMLR 2013) for a possible treatment of causal loops.
Can you treat your ML infrastructure as you would your software one?  
→ Yes and No

You should apply best Software Engineering practices (e.g. encapsulation, abstraction, cohesion, low coupling...)

However, Design Patterns for Machine Learning software are not well known/documentated.
Yes, you should care about Feature Engineering
What is a good Quora answer?

- Truthful
- Reusable
- Provides explanation
- Well formatted

What music do data scientists usually listen to while working?

Paula Griffin, data scientist and biostatistics PhD ... (more)
13 upvotes by William Chen, Alexandr Wang (王浩), Sheila Christine Lee, (more)

I was figuring that this question was just fishing for someone to answer that Big Data is their favorite band. Unfortunately, the question log indicates this was asked about 6 months before their EP came out, so there goes that theory.

This is going to be a pretty odd list, but here's the list, in order of decreasing social acceptability:

- Electropop -- Banks and CHVRCHES are my favorites at the moment.
- Miscellaneous alt-rock -- this category basically includes anything I found out about from listening to Sirius XM in the car.
- Nerd rock -- What kind of geek would I be if Jonathan Coulton wasn't on this list?

Shankar Iyer, data scientist at Quora
10 upvotes by William Chen, Sheila Christine Lee, Don van der Drift, (more)

Based on the Pandora stations that I've been listening to, my recent work-time listening consists of:

1. **Acoustic folk music**: John Fahey, Leo Kottke, Six Organs of Admittance, etc.
2. **Post-Rock / Ambient Music**: Sigur Rós, Gregor Samsa, the Japanese Mono, Eluvium, El Ten Eleven, etc.
3. **Hindustani**: mostly Vishwa Mohan Bhatt
4. **Carnatic**: recently Rajeswari Parthi
5. **Classical Guitar**: recently Paul Galbraith, Konrad Ragossnig, etc.
### Feature Engineering Example - Answer Ranking

**How are those dimensions translated into features?**

<table>
<thead>
<tr>
<th>Feature Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features that relate to the answer</td>
</tr>
<tr>
<td>Quality itself</td>
</tr>
<tr>
<td>Interaction features (upvotes/downvotes, clicks, comments…)</td>
</tr>
<tr>
<td>User features (e.g. expertise in topic)</td>
</tr>
</tbody>
</table>

This is going to be a pretty odd list, but here’s the list, in order of decreasing social acceptability:

- Electropop -- Banks and CHVRCHES are my favorites at the moment.
- Miscellaneous alt-rock -- this category basically includes anything I found out about from listening to Sirius XM in the car.
- Nerd rock -- What kind of geek would I be if Jonathan Coulton wasn’t on this list?
- Straight-up nostalgia -- I have an admittedly weird habit of listening to the same album (sometimes just one song) over and over for hours on end which was formed during all-nighters in high school. Motion City Soundtrack, Jimmy Eat World, and Weezer are my go-to’s in this category.
- Soundtracks of all sorts -- *Chicago, Jurassic Park, Bastion, The Book of Mormon*, the Disney version of *Hercules…* again, basically anything that works on a repeat loop for ~3 hours.
- Pop -- don’t make me list the artists. I’ve already told you I listen to Disney soundtracks; you can’t possibly need more dirt on me. The general principle is that if you can dance to it, you can code to it.

Now, if you don’t mind, I’m just going to sit at my desk and be super-embarrassed that my coworkers know what’s in my headphones.
Feature Engineering

Properties of a well-behaved ML feature

- Reusable
- Transformable
- Interpretable
- Reliable

Deep Learning: Automating Feature Discovery

Deep Learning
NIPS'2015 Tutorial
Geoff Hinton, Yoshua Bengio & Yann LeCun

Deep Learning
Output
Output
Mapping from features
Mapping from features
Most complex features

Hand - designed program
Hand - designed features
Features
Simplest features

Rule - based systems
Classic machine learning
Representation learning
Deep learning

Fig; I. Goodfellow
Your Model will learn what you teach it
Training a model

Model will learn according to

→ Training data (e.g. implicit and explicit)
→ Target function (e.g. probability of user reading an answer)
→ Metric (e.g. precision vs. recall)
Example 2 - Quora’s feed

→ Training data = implicit + explicit
→ Target function: Value of showing a story to a user $\sim$ weighted sum of actions:
  \[ v = \sum a_1 \{ y_1 = 1 \} \]
  - predict probabilities for each action, then compute expected value:
  \[ v_{\text{pred}} = E[V|x] = \sum a_1 p(a|x) \]
→ Metric: any ranking metric
The curse of presentation bias

User can only click on what you decide to show
→ But, what you decide to show is the result of what your model predicted is good

Simply treating things you show as negatives is not likely to work

Better options
→ Correcting for the probability a user will click on a position
→ Attention models
→ Explore/exploit approaches such as MAB

More likely to see

Less likely
Data and Models are great. You know what’s even better? The right evaluation approach!
Offline/Online testing process

**Offline Experimentation**
- Initial Hypothesis
  - Reformulated Hypothesis
- Try different Model?
  - NO
  - YES: Hypothesis Validated?
    - NO
    - YES: Design AB Test
      - Choose Control
      - Deploy Prototype
      - Observe Behavior
      - Analyze Results
      - Significant Improvements?
        - YES: Deploy Feature
        - NO: Try different Model?
Executing A/B tests

Measure differences in metrics across statistically identical populations that each experience a different algorithm.

Decisions on the product always data-driven

Overall Evaluation Criteria (OEC) = member retention

→ Use long-term metrics whenever possible

→ Short-term metrics can be informative and allow faster decisions
  - But, not always aligned with OEC
Offline testing

Measure model performance, using (IR) metrics

Offline performance = indication to make decisions on follow-up A/B tests

A critical (and mostly unsolved) issue is how offline metrics correlate with A/B test results.
You don’t need to distribute your AI algorithm.
Distributing ML

→ Most of what people do in practice can fit into a multi-core machine
  → Smart data sampling
  → Offline schemes
  → Efficient parallel code

→ Dangers of “easy” distributed approaches such as Hadoop/Spark

→ Do you care about costs? How about latencies?
Distributing ML

Example of optimizing computations to fit them into one machine

- Spark implementation: 6 hours, 15 machines
- Developer time: 4 days
- C++ implementation: 10 minutes, 1 machine

Most practical applications of Big Data can fit into a (multicore) implementation
You should care about answering questions (about your model)
Model debuggability

- Value of a model = value it brings to the product
- Product owners/stakeholders have expectations on the product
- It is important to answer questions to why did something fail
- Bridge gap between product design and ML algos
- Model debuggability is so important it can determine:
  - Particular model to use
    - Features to rely on
    - Implementation of tools
Model debuggability

→ E.g. Why am I seeing or not seeing this on my homepage feed?

### feed / feature analysis using score / feature analysis using model score

This table shows feature values for the debug story (using feedStory or debug_aid/qid above) and for the top 10 comparison stories from the same leaf node. For each comparison story, the color (and hover text) of a feature cell shows how the score of the debug story would change if feature values were swapped between the debug story the comparison story. Feature rows are sorted by the maximum absolute score gain among the comparison stories.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>aid 14862324</th>
<th>aid 2546362</th>
<th>aid 2290</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER.LOC</td>
<td>0.0094568</td>
<td>0.2130826</td>
<td>0.213059</td>
</tr>
<tr>
<td>USER.LOC</td>
<td>0.0514654</td>
<td>0.2039645</td>
<td>0.203939</td>
</tr>
<tr>
<td>OBJECT</td>
<td>8</td>
<td>None</td>
<td>7</td>
</tr>
<tr>
<td>OBJECT</td>
<td>128263005100</td>
<td>70919435147759</td>
<td>7538566</td>
</tr>
<tr>
<td>USER.LOC</td>
<td>0.0648323</td>
<td>0.2112874</td>
<td>0.211278</td>
</tr>
<tr>
<td>USER.Tags</td>
<td>0</td>
<td>None</td>
<td>1</td>
</tr>
<tr>
<td>USER.LOC</td>
<td>0.0094568</td>
<td>0.0787334</td>
<td>0.078733</td>
</tr>
<tr>
<td>OBJECT</td>
<td>0</td>
<td>0.3824019</td>
<td>0.245166</td>
</tr>
<tr>
<td>OBJECT</td>
<td>0.1047419</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>NUM.RE</td>
<td>1</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Conclusions
01. Choose the right metric

02. Be thoughtful about your data

03. Understand dependencies between data, models & systems

04. Optimize only what matters

05. Invest in your AI infrastructure/tools
Questions?