Academic versus Industrial Research in Machine Learning

Lucian M. Sasu, Ph.D.

Transilvania University of Brașov
Siemens Romania, Corporate Technology RTC
lmsasu@unitbv.ro, lucian.sasu@siemens.com

SEE Forum on Data Science, Belgrade
June 20th 2016

Outline

About me

- Teaching at Faculty of Mathematics and Informatics, Transilvania University of Brașov, since 2000
- Ph.D. degree in Computer Science since 2006, “Computational Intelligence Techniques in Data Mining”
- Joined as research engineer at Siemens Corporate Technology in 2012

Outline

Talk motivation – the obvious part

- Plenty of success stories and eye-catching results in ML; popular subject which attracts masses;
  - maybe too much emphasize from press and quite bombastic presentations
  - remember AI winters
- Proved (and sometimes: super–human) results in computer vision, adversarial games, voice recognition, recommendation engines, . . .
- Direct results:
  - gold-rush of faculties (not only IT-affiliated) towards ML; CS 229 (aka Machine Learning course) became the most popular course at Stanford
  - great expectations
  - main point: academia and industrial environments should leverage each other
  - realizing the challenges might speedup the process

Talk motivation – academic research

- Academic research in ML:
  - Improving existing ML models or creating new building blocks
    - how should one cleverly vary the learning rate coefficient in SGD–based algorithms?
    - how should I melt multiple models into an ensemble model to produce best predictions?
    - what are good choices for error functions?
  - Finding the weakest points in existing models (see deep learning manifesto, criticizing local models)
  - ML research complementing other data–targeting approaches — think about Breiman's "Statistical Modeling: The Two Cultures"
  - Overall, highly creative people are involved here, with low pressure of *pragmatic* initial results;

Talk motivation – industrial research

- ML–related research in industry:
    - Real–world and provoking questions;
    - Constant pressure to augment the *existing* functionalities through smart services;
    - Pre-existing infrastructure which should be considered as an environment (hard constraints)
    - Plenty of domain expertise which should be used for validation or encapsulation in ML models;
    - Plenty of dirty data, prioritized use cases and exceptions, tons of ill–defined problems;
    - (Seldom) NDAs and confidential/restricted use–cases + data sets;

Outline

Samples from academic research (1)

- Let us consider a quite popular and theoretical result: "The model M is a universal approximator"
- E.g. multilayer perceptron, radial basis functions, Fuzzy ARTMAP, Bayesian ARTMAP, Maxout, ...
- Informal: any given continuous function can be arbitrarily well approximated by a specific model (usually a neural network)
- The magic trick: show that the set of functions which can be produced by M is dense in the set of $C\left([0,1]^m\right)$

Samples from academic research (1)

- Let us consider a quite popular and theoretical result: "The model M is a universal approximator"
- E.g. multilayer perceptron, radial basis functions, Fuzzy ARTMAP, Bayesian ARTMAP, Maxout, ...
- Informal: any given continuous function can be arbitrarily well approximated by a specific model (usually a neural network)
- The magic trick: show that the set of functions which can be produced by M is dense in the set of $C\left([0,1]^m\right)$
- **Q1: How can one prove it?**
- **Q2: Is universal approximation the ultimate goal?**

Samples from academic research (2)

- **Q1: How can one prove it?**
- (A quite old result) Weierstrass approximation theorem states that every continuous function defined on an interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function;
- A more recent and general result is Stone–Weierstrass theorem: instead of set of polynomials, subalgebra of functions are considered; $[a, b]$ can be replaced by more general spaces;

Samples from academic research (2)

- **Q1: How can one prove it?**
- (A quite old result) Weierstrass approximation theorem states that every continuous function defined on an interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function;
- A more recent and general result is Stone–Weierstrass theorem: instead of set of polynomials, subalgebra of functions are considered; $[a, b]$ can be replaced by more general spaces;
- **Theorem (Stone-Weierstrass):** Let X be a compact metric space, and let A be a subalgebra of $C(X)$. If A separates points in X and vanishes at no point of X, then A is dense in $C(X)$.
- S-W theorem provides a widely used framework for proving universal approximation capabilities

Samples from academic research (3)

- **Q2: Is universal approximation the ultimate goal?**
- Actually, the result shows that the architecture itself is able to build a good approximation . . .

Samples from academic research (3)

- **Q2: Is universal approximation the ultimate goal?**
- Actually, the result shows that the architecture itself is able to build a good approximation ...
- ...while the learning algorithm is not discussed anywhere!
- Is there something more useful than universal approximation?
- The best approximation theory is defined as:

  An approximation scheme M is a best approximator if given a function f, there is always a choice of coefficients for M so that the distance between M and f reaches its absolute minimum (Girosi F. and Poggio T, *Networks and the Best Approximation Property*, Biological Cybernetic 63, 1989)

Samples from academic research (4)

- In other words: the model M is a best approximator if for every given continuous function, there is one in the set of M approximators which is situated at minimum distance.
- Not all universal approximators are also have the best approximator property: Radial Basis Function (RBF), Bayesian ARTMAP networks have this property, while Generalized RBFs or multilayer perceptron are not best approximators.
- The best approximation property is at least as important as universal approximation capability, but rarely studied or mentioned in research papers.

Samples from academic research (5)

- How about learning speed of an adaptive system? How fast does it learn? This would be a more practical result, actually
- Think about how we stress our students with algorithmic complexity; why does it matter?
- Difficult part in study of model's performance: learning models are generally stochastic systems, sometimes one has to come up with assumptions on data (independent and identically distributed r.v., stationarity, ...)
- While universal approximation can be understood by a freshman (a Mathematical Analysis course suffices), the learning speed might require a more in-depth view: probability, statistics, measure theory, ...see the next slide

Samples from academic research (6)

- An example of how learning speed looks like:

  If $q_0 \in [0, b]$, $q_t \in [a, b]$, $\forall t \geq 1$, for two real values $0 < a \leq b < \infty$, the following inequality occurs almost surely for every $\epsilon > 0$, $t \geq 1$:

$$|w_t(a_i) - p| \leq |w_0(a_i) - p| \frac{q_0}{Q_t} + (1+\epsilon)\frac{\sqrt{2p(1-p)\left(\sum_{i=1}^{t} q_i^2\right)\log\log\left(p(1-p)\sum_{i=1}^{t} q_i^2\right)}}{Q_t}$$

  excepting for a finite set of terms, where $p = P(a_i)$.

- Note: $w_t(a_i)$ are the weights, t is time step, p is the probability to be approximated (Andonie R. and Sasu L., *Fuzzy ARTMAP with Input Relevances*, IEEE Trans Neural Netw. 2006 Jul; 17(4))

Outline

Industrial research: a different world

- A world dominated by pragmatic requirements
- First things first:
  - "I have this problem which we failed to solve with our engineering knowledge; can you help us?" – understanding the context is quite challenging
  - "We follow this approach, which sometimes fail; can you come up with something complementary?"
  - "We have some constraints to fulfill": computational resources – cloud computing is only sometimes allowed; or there are pre-existing frameworks, or cross validation of the models through interpretability is required, . . .
  - "We want to get the human out of the loop" – smart automation, infield analytics

What an academic researcher should consider

- Avoid the "here I am, show me your differential equations" syndrome
- What remains after initial engineering attack is often hard; make sure you know about the previously failed approaches
- As found from personal experience:
  - some of the details (constraints, workflow, validity criteria) are obvious for the domain expert and not communicated in the first steps, but not obvious for the academic researcher
  - considerable "cultural" gap
  - lack of ML understanding (building a ML model needs lot of data; lack of knowledge on ML process workflow, overfitting/underfitting cases, etc.)

What an academic researcher should consider

- Sometimes: overly optimistic expectations from final customers, based on ML hype
- Often over-engineered approaches are proposed and should be avoided — similarly to overfitting in ML
- Frequently the problems are really ill–posed, e.g. the predictive features are later proved as not holding enough information; or there are too many possible answers for the question at hand; usually the customer realizes this and supports initial investigation, *if the costs are not too high*

How to prepare for industrial research (1)

- Traditional hot topics:
  - Descriptive analysis: “What happened?”: summaries, reports, outlier detection

How to prepare for industrial research (1)

- Traditional hot topics:
  - Descriptive analysis: “What happened?”: summaries, reports, outlier detection
  - Diagnosis, including root cause analysis: “Why did it happen?”, e.g. through finding correlations between subsystems or measurements

How to prepare for industrial research (1)

- Traditional hot topics:
  - Descriptive analysis: "What happened?": summaries, reports, outlier detection
  - Diagnosis, including root cause analysis: "Why did it happen?", e.g. through finding correlations between subsystems or measurements
  - Predictive analytics: "What will happen?", e.g. energy consumption, fault prediction

How to prepare for industrial research (1)

- Traditional hot topics:
  - Descriptive analysis: "What happened?": summaries, reports, outlier detection
  - Diagnosis, including root cause analysis: "Why did it happen?", e.g. through finding correlations between subsystems or measurements
  - Predictive analytics: "What will happen?", e.g. energy consumption, fault prediction
  - Prescriptive analytics: "What should I do?", e.g. production optimization, logistic maintenance, work loading

How to prepare for industrial research (2)

- Be ready to work in inter–disciplinary teams; while you are able to master the details of Stone–Weierstrass algorithm on your own, it is impossible to grasp all the knowledge in technical domains
- Be aware of the *inductive bias* of various ML models – everyone loves decision trees and rule inference because of explainability, but how well do they fit to the problem at hand?

How to prepare for industrial research (3)

- Get in touch with libraries/tools/frameworks readily available in ML — Python sklearn, Weka, KNIME, R, RapidMiner, Excel, Matlab/Octave, Theano, Tensorflow, Torch, DL4J, . . .

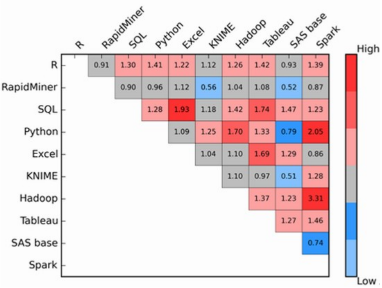

Figura 2: Popular tools used in Data Science; source: kdnuggets.com

How to prepare for industrial research (4)

- Mastering extract-transform-load and data cleansing tools

How to prepare for industrial research (4)

- Mastering extract-transform-load and data cleansing tools
- License attached to software tools; many open source tools, but not necessarily free for closed–source projects

How to prepare for industrial research (4)

- Mastering extract-transform-load and data cleansing tools
- License attached to software tools; many open source tools, but not necessarily free for closed–source projects
- Be aware of the technical limitations of various packages
    1. memory related problems; e.g. precompiled versions of Octave target x32 platforms $\rightarrow$ memory limitations; Excel can handle (only) about 1 million lines, etc.
    2. some libraries/frameworks are hard to be deployed on specific OSes

How to prepare for industrial research (5)

- Be aware about implementations details
  - “Spam Filtering with Naive Bayes - Which Naive Bayes?”, Vangelis Metsis, Ion Androutsopoulos, Georgios Paliouras, The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA:

    “[...]There are, however, several forms of Naive Bayes, something the anti-spam literature does not always acknowledge. We discuss five different versions of Naive Bayes[...]

  - Backpropagation: more than 65 versions of backprop, “Backpropagation Family Album”, Jondarr, G., Technical Report C/TR96-05, Department of Computing, Macquarie University, New South Wales, August, 1996.
- **We have an issue with reproducibility, not only inside academia!**

How to prepare for industrial research (6)

- Selected ML-related topics to be acquired (1):
  - **Outlier detection** (aka one–class–classification) – as manual labeling and is expensive, and a traditional classifier is hard to obtain; however, most of the time, a system works fine and the corresponding logs show "normal" behavior; one should use these cases to spot the abnormal cases

How to prepare for industrial research (6)

- Selected ML-related topics to be acquired (1):
  - **Outlier detection** (aka one–class–classification) – as manual labeling and is expensive, and a traditional classifier is hard to obtain; however, most of the time, a system works fine and the corresponding logs show "normal" behavior; one should use these cases to spot the abnormal cases
  - **Concept drifting** – the values are rarely identically distributed; the data dynamics should not be seen as anomaly

How to prepare for industrial research (6)

- Selected ML-related topics to be acquired (1):
  - **Outlier detection** (aka one–class–classification) – as manual labeling and is expensive, and a traditional classifier is hard to obtain; however, most of the time, a system works fine and the corresponding logs show "normal" behavior; one should use these cases to spot the abnormal cases
  - **Concept drifting** – the values are rarely identically distributed; the data dynamics should not be seen as anomaly
  - **Incremental (online, stream) learning** – {learn additional information from new data} + {do not require access to the original data used to train the existing system} + {preserve previously acquired knowledge} + {accommodate new data categories that may be introduced with new data}

How to prepare for industrial research (7)

Selected ML-related topics to be acquired (2):

- **Data fusion** – joining data from various sources; "more data beats better algorithms" is almost an axiom, and melting data sources leads to informational richer data sets

How to prepare for industrial research (7)

Selected ML-related topics to be acquired (2):

- **Data fusion** – joining data from various sources; "more data beats better algorithms" is almost an axiom, and melting data sources leads to informational richer data sets
- **Explaining the models** – statistical evidence is fine, but some customers want to see the "why is that" behind a model's decision;

How to prepare for industrial research (7)

Selected ML-related topics to be acquired (2):

- **Data fusion** – joining data from various sources; "more data beats better algorithms" is almost an axiom, and melting data sources leads to informational richer data sets
- **Explaining the models** – statistical evidence is fine, but some customers want to see the "why is that" behind a model's decision;
- **Metric learning** – if one uses local models or case based reasoning, how do you compute a proper similarity measure?

How to prepare for industrial research (8)

For a CS student:

- Linear algebra (including SVD), probability theory (why Bayes' rule is important?), statistics (frequentist and Bayesian statistics), operation research (esp. convex optimization), numerical computation, algorithms and data structure, procedural and OO-based programming are a must

How to prepare for industrial research (8)

For a CS student:

- Linear algebra (including SVD), probability theory (why Bayes' rule is important?), statistics (frequentist and Bayesian statistics), operation research (esp. convex optimization), numerical computation, algorithms and data structure, procedural and OO-based programming are a must
- Learn some rapid prototyping languages: Python, Matlab/Octave, R; read-eval-print loop environments; C/C++/Java/C# are not appropriate for rapid prototyping, but for production mode

How to prepare for industrial research (8)

For a CS student:

- Linear algebra (including SVD), probability theory (why Bayes' rule is important?), statistics (frequentist and Bayesian statistics), operation research (esp. convex optimization), numerical computation, algorithms and data structure, procedural and OO-based programming are a must
- Learn some rapid prototyping languages: Python, Matlab/Octave, R; read-eval-print loop environments; C/C++/Java/C# are not appropriate for rapid prototyping, but for production mode
- Fail fast and recover (frequent idea found in Kaggle's winners' interviews)

How to prepare for industrial research (8)

For a CS student:

- Linear algebra (including SVD), probability theory (why Bayes' rule is important?), statistics (frequentist and Bayesian statistics), operation research (esp. convex optimization), numerical computation, algorithms and data structure, procedural and OO-based programming are a must
- Learn some rapid prototyping languages: Python, Matlab/Octave, R; read-eval-print loop environments; C/C++/Java/C# are not appropriate for rapid prototyping, but for production mode
- Fail fast and recover (frequent idea found in Kaggle's winners' interviews)
- Distributed environments: Spark, Hadoop — plenty of MOOCs and virtual machines on Internet

How to prepare for industrial research (8)

For a CS student:

- Linear algebra (including SVD), probability theory (why Bayes' rule is important?), statistics (frequentist and Bayesian statistics), operation research (esp. convex optimization), numerical computation, algorithms and data structure, procedural and OO-based programming are a must
- Learn some rapid prototyping languages: Python, Matlab/Octave, R; read-eval-print loop environments; C/C++/Java/C# are not appropriate for rapid prototyping, but for production mode
- Fail fast and recover (frequent idea found in Kaggle's winners' interviews)
- Distributed environments: Spark, Hadoop — plenty of MOOCs and virtual machines on Internet
- Become a ML project contributor – plenty of them on GitHub; you will gain experience and become visible

Outline

What did I learn since being in industry?

- The user asks for an end-to-end solution, not just building blocks; make sure that you are able to add value to technological workflow
- Customized software architecture is often needed; hot topic in the area of data science
- We are living in a inter–disciplinary world; contrast this to the small focused teams found in academia
- Plenty of technologies to be considered: M2M connection protocols (message brokers), Complex Event Processing – e.g. for temporal alignment or near-real time complex feature creation in streams of data
- When in doubt, be pragmatic :)
- Be aware of the existing technical environments for rapid prototyping

Getting the best of two worlds

- Academy has cerebral resources, but often lacks contacts with real and stringent problems
- Industry has tons of challenging problems; some of them can be solved and become success stories
- Acting in small academic teams may lead to potential theoretical strong results, but for industrial research interdisciplinary teams are a must
- Think on how your experience can leverage students' curricula or their personal development
- Work towards the bridges between academia and industry — support internships, create data science labs
- Data science labs – e.g. collaboration between Siemens and Ludwig-Maximilians University Munich, http://dsl.ifi.lmu.de/data-science-lab