# Machine Learning and Verification

Rémi Gilleron

Inria Lille & CRIStAL & Univ. Lille

Jan. 2018

# Introduction

I made a brief overview of recent tutorials

- ML in Verification still limited
- Verification of ML models still in its infancy

# Plan


1. Machine Learning
2. Verification in ML
3. ML in Verification
4. Conclusion

# Machine Learning

*Algorithms that improve automatically their performance on some task through experience (Mitchell 97)*

## Success Stories

- Image processing
- Speech recognition
- Automatic translation

## Many Problems

- Learning to transform, learning to class
- Learning to rank
- Learning probability distributions
- Learning strategies
- ...

# Learning from Data

## Learning to Transform

- Task: function from data input into data output
- Experience: data as pairs (input, output)
- Performance: function performs well on new input data

## Learning to class

- Task: function from data input into a finite set of labels
- Experience: data as pairs (input, label)
- Performance: correctly classify new inputs

# Challenges for ML

## Ill-posed Problem

What is the meaning of "correctly classify new inputs"

Let us consider a probabilistic framework and let us suppose a probability distribution $P$.

- Experience: pairs are drawn i.i.d. according to $P$
- Performance: expected value w.r.t. $P$ of the error on a new input

$P$ is fixed but unknown

## Consequences

- No Free Lunch Theorem: (Wolpert 96) over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.
- A learning bias is mandatory: it can be a probabilistic bias, an algorithmic bias, a bias on the class of functions, ...

# Computational Learning Theory

## PAC learning

A concept class is PAC-learnable if there exists an algorithm $\mathcal{A}$ such that, for every distribution $P$, $\mathcal{A}$ is able to learn in polynomial time an hypothesis with a low error.

## Negative Results

- No interesting PAC-learnable concept class
- Learning algorithms are consistency algorithms.
- Inherently Unpredictable: polynomial size Boolean Formulas, DFAs, ...

## Positive Results

- Weak learning = strong learning leads to boosting algorithms
- PAC learning models with queries
- PAC learning and Property testing (Goldreich et al, ACM, 98)

# Learning Regular Languages

## Main results

- Existence of a consistent language is NP-complete
- DFAs are inherently unpredictable
- DFAs are learnable with membership queries

## Algorithms

- Every algorithm aims to construct the Myhill-Nerode congruence defining the minimal DFA of a target regular language.
- The Hankel matrix $H$ of a regular language $L$ is the biinfinite matrix in $\{0,1\}^{\Sigma^* \times \Sigma^*}$ défined by $H(u,v) = 1$ iff $uv \in L$.
- The number of distinct rows of a binary Hankel matrix $H$ equals the minimal number of states of a DFA recognizing the language of $H$.

# Learning Weighted Automata

A new approach to learning languages: spectral learning of automata by Hsu, Denis, Balle, Mohri, ... since 2009

## Principles

- (empirical) Hankel Matrices of counts or frequencies
- Linear Algebra, mainly matrix factorization
- Learning complexity in terms of the smallest singular value of some Hankel matrix

## References


- Balle et al, Spectral Learning of Weighted Automata, Machine Learning, (2014)
- Balle tutorial at Logic and Learning Workshop, `https://borjaballe.github.io/slides/turing18.pdf`
- Mohri tutorial at LICS'17, Learning Weighted Automata `https://cs.nyu.edu/~mohri/talks/LICS2017.pdf`

# Deep Networks

## Success Stories

- Image processing
- Speech recognition
- Automatic translation

## What is new? Nothing except

- lot of data, software systems, distributed computing
- stochastic gradient descent, neuron ReLU, new loss functions, new regularization methods (dropout), specialized architectures for images and texts.
- Representation learning: unsupervised or supervised learning of vectorial representations. For instance, word (phrase, sentence) embeddings in NLP: Vectors of reals allow to do translation in NLP

# PMCs

## Base structure for neural networks

- input $\mathbf{x} \in \mathbb{R}^d$, output $f^p(\ldots(f^1(\mathbf{x})) \in \mathbb{R}^q$
- each layer $i$ computes a function $f^i$ as follows
  - a linear transformation $\mathbf{W}^i\mathbf{x}$
  - followed by a function $g$, i.e. $f^i(\mathbf{x}) = g(\mathbf{W}^i\mathbf{x})$
- Neurons: linear: $g = Id$; sigmoid: $g = \sigma$; ReLU: $g = max\{0, x\}$; ...
- At least, one layer is non linear

## Some results

- a very large class of functions is computable by PMCs
- only 2 layers (one sigmoid and one linear), but exponential numbers of neurons. Extended to ReLU
- more layers, less neurons but learning more difficult

# Learning PMCs

## Back-Propagation Algorithm

- A set of pairs $(\mathbf{x}, \mathbf{y})$, a PMC with unknown parameters $\mathbf{W}^1, \ldots, \mathbf{W}^p$, a differentiable loss function $E(f(\mathbf{x}), \mathbf{y})$.
- The chain rule of calculus (suppose $y = g(x)$ and $z = f(y)$, then $\frac{dz}{dx} = \frac{dz}{dy}\ \frac{dy}{dx}$) allow to compute every $\frac{\partial E}{\partial w}$ where $w$ is a parameter
- The Back-Prop Algorithm is an efficient implementation
- Then use stochastic (with mini-batch) gradient descent to learn the parameters of the PMC.

## It works!

- larger datasets, better software infrastructure, powerful computers,
- new loss functions, ReLU, regularization methods, adaptive SGD, ...
- Unexplained: SGD avoiding saddle points and plateaux; local minima are good enough; ...

# Representation Learning

Supervised: every $f^i(\dots(f^1(\mathbf{x}))$ is a representation of $\mathbf{x}$; Unsupervised:

## Autoencoders

- PMC with one hidden layer, i.e. $f(\mathbf{x}) = f^2(f^1(\mathbf{x}))$
- Learn to copy the input in the output
- Choose the size of the hidden layer to be smaller than $d$ or add constraints for the hidden layer when regularizing the loss
- $f^1(\mathbf{x})$ is a new representation of $\mathbf{x}$ capturing the essence of $\mathbf{x}$

## Word Embeddings

- Learn a matrix $\mathbf{W}$ with $\#V$ rows and $d$ columns ($d = 100, 200, 300$)
- $\mathbf{x}$ one-shot embedding of $w$, $f^1(\mathbf{x}) = \mathbf{W}^t\mathbf{x}$ is the representation of $w$, i.e. the row of $\mathbf{W}$ indexed by $w$
- PMC output 1 for pairs in a same context, 0 for random pairs.
- has been extended to node, edge and graph embeddings

# Beyond PMCs

## Convolutional Networks

- Modeling properties of grid-like structured data : images with sliding windows, sentences with sliding $n$-grams,
- Convolution, Pooling and ReLUs

## Recurrent Networks

- Beyond $n$-grams with sentences of variable lengths
- Defining $\mathbf{y}^{i+1} = f(\mathbf{x}^{i+1}, \mathbf{y}^i)$, then $\mathbf{y}^n$ depends on all $\mathbf{x}^1, \ldots, \mathbf{x}^n$ while sharing parameters

## Learning Languages with Neural Networks

- PMCs, convolutional networks, and –variants of– recurrent networks
- NLP research is dominated by NNs because of impressive results for many NLP tasks, among them automatic translation.

# Conclusion

## Machine Learning Results

- Learning from vectorial data: DTs & RFs, SVMs, NNs
- Beyond vectorial data: sequences, trees, graphs, ...
- Online learning, bandits algorithms (UCB), reinforcement learning
- Success stories for NNs in AI applications

## Machine Learning Research

- Transfer, never ending, self-supervised, intelligent learning
- Generative models with NNs (GANs)
- More theoretical results
- Ethics in ML

Neural networks and neural machines

# Plan


1. Machine Learning
2. Verification in ML
3. ML in Verification
4. Conclusion

# Security – it is easy to break ML-systems

Main source: cleverhans blog about security and privacy in machine learning `http://www.cleverhans.io/`

## ML is ill-defined

- theory: for every distribution, but nothing learnable
- practice: a (large) set of data, learning on a subset and testing on the rest, works well for many applications with ML systems

## Breaking an image recognition system

- a convolutional NN for classifying images
- works very well on large databases of images
- but a panda + adversarial noise looks like a panda for a human but is classified as a cat

# Testing or verification

## Types of guarantees

- testing: evaluating the system in several conditions
- verification: providing a compelling argument that the ML system will not misbehave under a very broad range of circumstances
- ML relies on testing, mainly on testing for "natural" inputs
- verification analog of testing is statistical learning theory (Vapnik) giving bounds for the generalization error. But, large upper bounds and w.r.t. the same distribution

## Challenge

- Use verification rather than testing
- Bring adversaries in the equation

# Verification in ML

## Claims

- testing against adversarial examples is insufficient to provide security guarantees because
  - an attacker can send unseen inputs
  - testing provide lower bounds on the failure rate
- We should verify but so far we only know how to test.
- The development of a guarantee characterizing the space of inputs that are processed correctly is central to the future of ML in adversarial settings, and it will almost certainly be grounded in formal verification

# Approaches to verification in ML

## Model verification

- Proving that small perturbations to a correctly classified input to the network cannot cause it to be misclassified.
- Safety Verification of Deep Neural Networks, Huang et al, 16. Based on Satisfiability Modulo Theory (SMT). Hypothesis on DNNs.
- Towards Proving the Adversarial Robustness of Deep Neural Networks, Barrett et al, 17. Based on LP solver. DNNs with ReLUs.
- Limitations: verify only that the class remains constant in some specified neighborhood (which one) of some specific point $x$ (which $x$)

## Verification in ML – security

- Verification in ML still in its infancy
- It is possible that no verification system will ever exist because no machine learning model will ever be fully robust and accurate

# Ethics in ML

## Main questions

- Privacy: insure that identity can not be inferred
- Fairness: insure fair treatment w.r.t. sensible attributs like race or gender

## Research problems

- Defining the notions is already challenging
- Differential privacy defines privacy in a PAC-like style
- Defining private or fair learning algorithms
- Testing or verification of such properties

in Magnet: defining privacy-preserving algorithms in a distributed context such that personal data remains private.

# Plan


1. Machine Learning
2. Verification in ML
3. ML in Verification
4. Conclusion

# ML in verification – motivations

Source: Learning Algorithms and Formal Verification (Madhusudan)
`http://madhu.cs.illinois.edu/learning.pdf`

## Verification of systems is a very hard problem

But there is promise in solving it because

- The programmer isn't an adversary.
- Simple concepts usually lie behind the correctness of code.
- Examples: simple loop invariants; simple shape invariants of structures; simple predicates that control flow; simple agreements between components; simple concurrency conventions.
- These simple concepts are however hidden.

Learning may be a way to mine these simple concepts back, and use them to verify the system.

# ML in verification – personal remarks

## Learning regular languages

Source: Model Learning, F. Vaandrager (2017), Communications of the ACM

- Implantation and improvements of the $L^*$-algorithm for applications
- Learning register automata, visibly pushdown automata
- Verification of the learned model

## Other stuff

- Basic learning systems (decision trees) and bandit algorithms (UCB)
- Specification Mining, ML-aided Theorem proving, Tests and Testing

ML in Verification still limited

# Plan


1. Machine Learning
2. Verification in ML
3. ML in Verification
4. Conclusion

# Conclusion

**In my opinion**

- Avenues to use up to date ML systems for verification
- Verification of ML systems raises many interesting but quite challenging problems.