Cryptography & Machine Learning: What Else?

SHAFI GOLDWASSER
Crypto 81

• Exciting
• Informal
• Art rather than a science
Simons Institute for Theory of Computing

Data Privacy: Foundations and Applications
Jan. 15 – May 17, 2019

Proofs, Consensus, and Decentralizing Society
Aug. 21 – Dec. 20, 2019

Integer Lattices: Algorithms, Complexity and Applications to Cryptography
Jan 15 – May 15, 2020
The Surprising Consequences
Of Basic Cryptographic Research

How NP got a new definition:
Probabilistically Checkable Proofs (PCPs) & Approximation Properties of NP-hard problems

Next Frontier:
Cryptography for Safe Machine Learning
Outline

• Historical connections between Cryptography and Machine Learning

• Safe Machine Learning: a Cryptographic Opportunity

• A sampling of what is done already today
“Explores the study and construction of algorithms that can learn from and make predictions on DATA without being explicitly programmed, through building a model from sample inputs.”
Phase 1: Learning/training

Given training data = \{(labeled) instances\}, drawn from an unknown distribution \(D\), generate an hypothesis/model, ordinarily tested against test data.

Phase 2: Hypothesis/model developed is used to

• **Classify** new data drawn from \(D\)
• **Generate** new data similar to \(D\)
• **Explain** the data.
Phase 1: Learning/training
Given training data \( \{(\text{labeled}) \text{ instances}\} \), drawn from an unknown distribution \( D \), generate a hypothesis/model.

Phase 2: Hypothesis/model developed is used to

- Classify new data drawn from \( D \)
- Generate new data similar to \( D \)
- Explain the data.

Many Machine Learning Models

Training
A *magic* DNF Boolean formula $c$ is hidden in a black box.

\[
c(x_1, x_2, x_3) = (x_1 \land x_3) \lor (x_1 \land x_2 \land \neg x_3)
\]

c could be used to answer:

- Is a tumor malignant
- Should a bank loan be approved
- Should a suspect be released on bail.
- Is an email message spam
Let's be more concrete

A magic DNF Boolean formula $c$ is hidden in a **black** box.

$$c(x_1, x_2, x_3) = (x_1 \land x_3) \lor (x_1 \land x_2 \land \text{not-}x_3)$$

$c$ could be used to answer:

- Is a tumor malignant?
- Should a bank loan be approved?
- Should a suspect be released on bail?
- Is an email message spam?

**Obviously, we would love to learn $c$.**

**But, how hard is it?**
To answer this question

Need to define:

What’s meant by successfully “learn”

What information is made available to the learner about the hidden c, aka “query model”

Given examples \( \{ x, c(x) \} \) for \( x \in X \) drawn according to unknown distribution \( D \) and concept \( c : X \rightarrow \text{Label} \), a successful efficient learning algorithm generates an hypothesis \( h \) that agrees with \( c \) approximately and with high probability on inputs drawn from \( D \).

Efficient = polynomial in input size \( n \) and concept size \( c \)
Agrees Approximately and with high probability =

Let \( \text{error} = \text{Prob}_{x \in D}[h(x) \neq c(x)] \). Then, \( \text{prob}[\text{error} > \varepsilon] < \delta \)
1984 Valiant PAPER: OPTIMISTIC

**DNF:** $c(x_1, x_2, x_3) = (x_1 \land x_3) \lor (x_1 \land x_2 \land \text{not-}x_3)$

- PAC-learn DNF with random examples from arbitrary $D$?
- PAC-learn DNF with random examples when $D=\text{uniform}$?
- PAC learn DNF by polynomial time $h$, not necessarily a DNF?
- PAC learn DNF if membership queries are allowed?

**Progress has been slow:**

<table>
<thead>
<tr>
<th>model</th>
<th>Time</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAC, hypothesis is DNF</td>
<td>NP-Hard</td>
<td></td>
</tr>
<tr>
<td>PAC, hypothesis is poly of degree $n^{1/3} \log n$</td>
<td>$2^{O(n^{1/3} \log^2 n)}$</td>
<td>[KS01]</td>
</tr>
<tr>
<td>PAC, $D=\text{Uniform Distribution}$</td>
<td>$n^{O(\log n)}$</td>
<td>[Ver90]</td>
</tr>
<tr>
<td>PAC, $D=\text{Uniform Distribution} + \text{Membership queries}$</td>
<td>$\text{poly}(n)$</td>
<td>[Jac94]</td>
</tr>
</tbody>
</table>
History of Cryptography & ML

Are there concepts which are not PAC-learnable?
PAC learnability (even representation independent) is crypto-hard for many query models

[ValiantKearns86] Secure RSA imply the existence of concepts in low level complexity classes (NC) which cannot be PAC-learnable even if hypothesis is any polynomial time algorithm

Proof: \(<e.N.X^e \bmod N, \text{label} = \text{lsb}(x)>\)

[PittWarmath90] Secure PRF \(f\) imply the existence of concepts in complexity class \(\text{Time}(f)\) which cannot be PAC-learnable with \(\text{membership queries} \& D\) uniform

[CohenGoldwasserVaikuntanathan14] Secure Aggregate-PRF \(f\) imply the existence of concepts in \(\text{Time}(f)\) not PAC-learnable even if can \(\text{request count of positive examples in an interval}\)

[BonehWaters13, BoyleGoldwasserIvan13] Constrained PRF imply non PAC-learnable \(c\) even if can \(\text{receive a circuit which computes a restriction of } c\).
Distribution $D=\{D_n\}$ computed by a family of polynomial time circuits $C=\{C_n\}$ is hidden in a black box.

Learner can request samples

D could be:
- Pictures of cats
- Successful college essays
- CV’s that get you a job
- Slides for Keynote talks
- Plays by Shakespeare

Goal: output polynomial size $C_n'$ which generates $D' \approx_\varepsilon D$

Naor95: if $\exists$ digital signatures Sig secure against CMA, then $\exists$ such family of distributions which are hard to generate.

$D = \{(m_i, \text{verification-key}), \text{Sig}(m_i)\}$
Modern cryptography has had considerable impact on the development of computational learning theory. Virtually every intractability result in Valiant’s model [13] (which is representation-independent in the sense that it does not rely on an artificial syntactic restriction on the learning algorithm’s hypotheses) has at its heart a cryptographic construction [4, 9, 1, 10]. In this paper, we give results in the reverse direction by showing how to construct several cryptographic primitives based on certain assumptions on the difficulty of learning. In doing so, we
**Learning Parity with Noise (LPN) [BFKL93]**

- Let \( s \) be a secret vector in \( \mathbb{Z}_2^n \)
- \( \text{LPN}_{n, \rho} \): Given an arbitrary number of “noisy” equations in \( s \), find \( s \)?

\[
\begin{align*}
0s_1+s_2+s_3+...+sx_n &\approx 0 \mod 2 & \text{Add noise vector } e: \\
1s_1+0s_2+s_3+...+1s_n &\approx 1 \mod 2 & \text{Bernulli with } \rho \\
1s_1+1s_2+0s_3+...+0s_n &\approx 0 \mod 2 & \Sigma|e_i| \text{ over } \mathbb{Z} \text{ is small} \\
1s_1+1s_2+0s_3+...+0s_n &\approx 0 \mod 2 \\
\vdots \\
0s_1+1s_2+0s_3+...+0s_n &\approx 1 \mod 2
\end{align*}
\]

- **Best-Algorithm [BKW03]**: Best known algorithm time \( 2^{O(n/\log n)} \)
- **Worst case to average reductions [BLVW18]**, noise: \( 1/2-1/poly(n) \)
- **“Easy” Hard problem**: decoding from relative distance \( \log^2(n)/n \)
The Learning with Errors Problem (LWE) [Regev05]

- Let $s$ be a secret vector in $\mathbb{Z}_q^n$
- $\text{LWE}_{n,\alpha}$: Given an arbitrary number of “noisy” equations in $s$, find $s$?

  
  \begin{align*}
  14s_1 + 15s_2 + 5s_3 + 2s_4 & \approx 8 \pmod{17} \\
  13s_1 + 14s_2 + 14s_3 + 6s_4 & \approx 16 \pmod{17} \\
  6s_1 + 10s_2 + 13s_3 + 1s_4 & \approx 3 \pmod{17} \\
  10s_1 + 4s_2 + 12s_3 + 16s_4 & \approx 12 \pmod{17} \\
  9s_1 + 5s_2 + 9s_3 + 6s_4 & \approx 9 \pmod{17} \\
  3s_1 + 6s_2 + 4s_3 + 5s_4 & \approx 16 \pmod{17} \\
  6s_1 + 7s_2 + 16s_3 + 2s_4 & \approx 3 \pmod{17}
  \end{align*}

  Add noise $e$: each $|e_i| < \text{small Gaussian in } [q/2,-q/2]$, std dev $\alpha q$

- Equivalent to approximating the size of the shortest vector in a worst-case integer lattice [Reg05, BLPRS13]
- Worst Case to Average [Ajtai98]
- Best known algorithm still $2^{O(n/\log n)}$ [BKW05]
- Revolutionary: Homomorphic Encryption, Leakage resilient Crypto, Functional/Attribute Encryption, and much more
Cryptographic Constructions from LWE and LPN

\[ \text{LWE}_{n,\alpha}, \alpha : \]

\[ \frac{1}{O(1)} \]

PRGs [BFKL93, GKL93], \ldots

\[ \frac{1}{\sqrt{n}} \]

PKE [Ale03, Reg05, MP12, KMP14], MPC [FMV18]

IBE, hashing [BLSV18, DGHM18]

\[ \frac{1}{\text{poly}(n)} \]

Lattice Trapdoors [GPV08], \ldots

\[ \frac{1}{2^{o(1)}} \]

PRFs/LWR [BPR12], \ldots

\[ \frac{1}{2^n} \]

easy

\[ \frac{1}{n} \]

\[ \frac{1}{\log^2(n)} \]

Thanks to Daniel Masny

\[ ^1 \text{or } n = \log^2(\kappa) \]
Quantum Significance

In 2017, Google, Microsoft, IBM and many other companies, as well as governments, are tracing toward building a quantum computer. NSA and NIST have started planning for post-quantum cryptography.
2017: Post Quantum Standardization has begun
82 submissions: 59 encryptions, 23 signatures

Essentially All Candidates are based on one version or another of LWE
Bliss for Crypto is a Nightmare for ML

Impossibility Results May be Positive News for Second Part of the Talk
The Evolution of Two Fields
Since the 1980s

**Cryptography**
Theory
Practice
Theory & Practice of cryptography are coming closer together

**Machine Learning**
Theory
Practice
Theory of ML alive and well, but the excitement in ML is in practice (DNN) lacking theory
The thing is... the practice of ML is too important to leave to practice.

**An Algorithm Is Now Helping Set Bail in New Jersey**

The algorithm looks at criminal history to calculate the likelihood of a defendant skipping town or committing another crime.

- **Health:** disease control by trend prediction
- **Finance:** predictions for financial markets
- **Economic Growth:** intelligent consumer targeting
- **Infrastructure:** traffic patterns and energy usage
- **Vision:** facial and image recognition
- **NLP:** speech recognition, machine translation
- **Security:** threat prediction models, spam
- **Policing:** decide which neighborhoods to police
- **Bail:** decide who is a flight risk
- **Credit Rating:** decide who gets a loan

**Sudden Shift of Power**
“Data is the new oil”
– Shivon Zilis, Bloomberg Beta

“Data will become a currency”
– David Kenny, IBM Watson
"Data is the new oil"
– Shivon Zilis, Bloomberg Beta

"Data will become a currency"
– David Kenny, IBM Watson

The Sudden Shift of Power
Can leave us unprotected and unregulated
The Thesis for the rest of the talk

After 30+ years of working on methods to ensure the privacy and correctness of computation as well as communication

Cryptography has the tools and models that should enable it to play a central role in ensuring power of algorithms is not abused
Challenges that Cryptography can help address (and is addressing)

1. Power of ML comes from Data of individuals

Ensure privacy of both data & model during training and classifying (even when not mandated by current regulations) to maintain “power to the people”

2. Models should not be tampered-with nor introduce bias for profit or control

Develop methods to minimize the influence of maliciously chosen training data and to prove models were derived from reported data.

Extra Benefit: Opportunity for using the last 30 years of “crypto computing” in practice
Challenges that Cryptography can help address and is not currently addressing

3. Adversarial ML where clever manipulations of an input by an adversary can cause misclassifications and fool applications emerges as a real threat in applications such as self driving cars or virus detection.
Challenges that Cryptography can help address and is not currently addressing

3. Adversarial ML emerges as a real threat in applications such as self driving cars or virus detection where clever manipulations of an input by an adversary can cause misclassifications and fool applications.

As cryptographers have vast experience in mathematically modeling of adversarial behavior may help in defining a class of attacks and techniques that defend against them.

Define a class of domain specific attacks and prove
- Adversarial Robustness via Robust Training [MMSTV2018]
- Adversarial Robustness requires more data [SSTTM18]
- Getting adversarial robustness to rotations/translations of an image [ETTSM10]
3. Adversarial ML emerges as a real threat in applications such as self driving cars or virus detection where clever manipulations of an input by an adversary can cause misclassifications and fool applications. As cryptographers have vast experience in mathematically modeling of adversarial behavior, it may help in defining a class of attacks and techniques that defend against them.

Reminiscent of early side channel attack days.
3. Adversarial ML emerges as a real threat in applications such as self driving cars or virus detection where clever manipulations of an input by an adversary can cause misclassifications and fool applications.

Holy Grail: build ML models where `misclassification’ requires learning a `cryptographically-hard’ task – fine grained cryptographic hardness would be necessary. Recall
Challenges that Cryptography can help address and is not currently addressing

4. **Trace** the unauthorized use of your data and model

Develop methods to trace training data used for learning a model without introducing new vulnerabilities.

SAN FRANCISCO — California has passed a digital privacy law granting consumers more control over and insight into the spread of their personal information online, creating one of the most significant regulations overseeing the data-collection practices of technology companies in the United States.

**Conjecture [reception]:** data tracing is possible unless “privacy-preserving” learning algorithm was used on data.

[Double edged sword]
Challenges that Cryptography can help address and is not currently addressing

4. Trace the unauthorized use of your data/model

How about tracing unauthorized use of the model? Develop methods to water mark (or leash) your models.

[ABCPK-Usenix18] “Turning your Weakness into your Strength”

Idea: Watermark DNN models by training the network to accept some “planted” adversarial examples = watermarks.
Challenges that Cryptography can help address and is not currently addressing

5. Fairness, accountability, and de-Biasing
Come up with computational Crypto-style definitions building on “real” vs. “ideal” paradigm rather than “similarity”.

6. Proper Use of Proper Randomness
Randomness seems key to training phase in DNN, what type of randomness? does it affect stability? Is secrecy of the randomness important?
Challenges that Cryptography can help address and is not currently addressing

7. Define specialized cryptographic functionalities which are ML complete

And then focus on efficient reductions between known ML classifiers to these functionalities.

8. Replace current ML algorithms with cryptographic friendly ones

... A Real Opportunity for developing new theory for cryptography motivated by ML
Challenge 1
Ensure Privacy of both data & model

• Classification
  • Performance

• Training
  • Approximate functionality
  • Trust models

• Model Stealing
  • Differential Privacy

Many Many works

Feasibility
Asymptotic efficiency
Concrete efficiency
Proof of concept
Uses Cryptographic Technologies of the Past

**Garbled circuits**

\[ A_0, A_1 \rightarrow E_0, E_1 \]
\[ B_0, B_1 \rightarrow F_0, F_1 \]
\[ C_0, C_1 \rightarrow E_0 \]
\[ D_0, D_1 \rightarrow F_0 \]

**Homomorphic Encryption**

Input Data → Encrypt → Evaluation → Decrypt → Output Response

**Secret sharing**

\[(0, s) \rightarrow (x_1, y_1) \rightarrow (x_3, y_3) \rightarrow (x_2, y_2)\]

**MPC**

Data_1 → Data_2 → Data_3 → Data_N

**Differential Privacy**

User’s data \( D \)

\[ X_1, X_2, \ldots, X_n \]

Curator → Statistic \( \hat{\theta} \)
Each Have Their Merit depending on particular ML model

A Pick and Choose Approach
Privacy during Classification Phase

The server’s model is sensitive financial model, genetic sequences, want to monitor it, ...

Client’s private data medical records, credit history, ...

MPC/2PC
General 2PC [Y, 80’s]

+ OWF Assumption
+ Efficient Computationally

- Large Communication
  ~ size of the Boolean circuit
- Have to convert your ML model to a Boolean circuits
- Inefficient for Arithmetic circuits
- Not easy to reuse effort

Garbled circuits

Using (F)HE [GM82, P86, BGV, G’09, BV’11, BGV’12, GSW’13]

+ Efficient Communication
  ~ size of input/output
+ Arithmetic Computation (built in)

- High Computation Cost
  ~ poly in depth of arith. circuit
- If your computation is not a low-degree polynomial, too bad
- QR/LWE vs. general assumption
### Simple Classifiers [BPTG15]

**Approach:** There are repeating **building blocks** across different classifiers. Find them, focus on building them, emphasizing performance.

<table>
<thead>
<tr>
<th>ML Algorithm</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>Linear</td>
</tr>
<tr>
<td>Least squares</td>
<td>Linear</td>
</tr>
<tr>
<td>Fischer linear discriminant</td>
<td>Linear</td>
</tr>
<tr>
<td>Support vector machine</td>
<td>Linear</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>Naïve Bayes</td>
</tr>
<tr>
<td>ID3/C4.5</td>
<td>Decision trees</td>
</tr>
</tbody>
</table>
Simple Classifiers [BPTG15]

Approach: There are repeating building blocks across different classifiers. Find them, focus on building them, emphasizing performance.

Choose and combine the best fitted primitives
Homomorphic Encryption, Garbled Circuits, ...

Diagram:
- Linear Classifier
- Naïve Bayes Classifier
- Decision Tree Classifier
- Dot Product
- Enc. Compare
- Enc. Argmax
- ES Switching
- Private Decision Trees
Linear Classifier

Separate two sets of points
Very common classifier

Dot product +
Encrypted compare

Client
\[ v \xrightarrow{PK} \text{Dot Product} \xrightarrow{\langle v, w \rangle} \xrightarrow{PK} \text{Enc. Compare} \xrightarrow{\langle v, w \rangle > 0} \]

Server
\[ w \xrightarrow{SK} \text{Dot Product} \xrightarrow{\langle v, w \rangle} \xrightarrow{SK} \text{Enc. Compare} \xrightarrow{\langle v, w \rangle > 0} \]
Moving from Simpler Model to Deep Neural Nets: what’s the challenge?

- Probabilities of Dog, Cat, Man, Neither
- Activation Function= Non-linear
  - e.g. g=logistic function, Max (ReLu), Tanh
And yet, yes, we can!
Neural Nets Private Classification

Using Lattice based FHE: CryptoNets [GLLNW16]
- convert fixed precision real numbers to integers
- use the square function: \( \text{sqr}(z) := z^2 \) activation function
- replacing

Big Idea: Trading Accuracy for Efficiency

Using MPC: DeepSecure [RRK17]
- Garbled Circuits-optimized implementation of Sigmoid, Tanh function

When is FHE better than MPC [Vinod’s rule]?

1. Computation is linear (deg 1) and
2. Circuit-size is super-linear (e.g. quadratic)

\( \text{MPC costs in bandwidth} \)
The Gazelle Approach [JVC18]

Convolutional Neural Networks: Alternating Linear and Non-linear Layers

Fast HE Library with Native Support for Neural Network Layers (extending the PALISADE lattice library)
Maintaining Privacy during Training Phase: more challenging

- **Non-Linearity Galore:** Training non-linear regressions and DNN’s involve multiple passes through the entire corpus of training data – each time computing a sequence of non-linear operations on “encrypted data”

Training with Privacy >> |Training Data| Classification with Privacy
Maintaining Privacy during Training Phase: more challenging

- **Non-Linearity Galore:** Training non-linear regressions and DNN’s involve multiple passes through the entire corpus of training data – each time computing a sequence of non-linear operations on “encrypted data”

Training with Privacy >> |Training Data| Classification with Privacy

- As LARGE cohorts of training examples are needed, often need training data from **multiple institutions or individuals** and must keep data private across contributors
Federated Learning for **Neural Nets** = Distributed training data with local training [BIKMMPRSS17]

Train a DNN by

1. local training by user
2. Report weight modifications to server, not your inputs
3. The loss gradient can be now computed as a **weighted sum** of local loss gradients of individual users

Not good enough...

Weight modification $\Delta w^i$ can leak information
Federated Learning for Neural Nets = Distributed training data with local training [BIKMMPRSS17]

Train a DNN by

(1) local training by user
(2) Report weight modifications to server, not your inputs
(3) The loss gradient can be now computed as a weighted sum of local loss gradients of individual users

Idea’: MPC among users each with
Inputs $\Delta w^i$ to compute the aggregate modification

Assumption: server does not collude with any single user
Train Approximate Logistic Regression

• iDash 2017 winning entry. Logistic Regression Model Training based on new Homomorphic Encryption for approximate arithmetic [KimSong KimLeeCheon17]

• iDash 2017 runner up. Use (F) HE with low-deg polynomial instead of a logistic function [ChenGiladBachrachHanHuanJalaliLaineLauter17]
Training Neural Nets

Multiple Non Colluding Servers: secure ML [MZ17] and (F)HE: secure NN [WGC18]

Hard (for me) to compare: which benchmarks, ability to process batches of data as they come, performance, training sample size, depth of network, precision of results
Output of the Model can Leak Training Data

Even with best guarantees on privacy of users training data, the output $c(x)$ may reveal information on training inputs.

Output $+ \text{Aux Information} \rightarrow \text{Model Inversion}$

**Solution:** Convert Training phase to output a Differentially Private Model/Hypothesis

**Def[KLN11]:** A Learning algorithm $L$ is $(\varepsilon, \delta)$-differentially private if $\forall S = \{(x_i, b_i)\}$, $S' = \{(x'_i, b'_i)\}$ which are identical except for 1 entry,

$$\forall \text{ set } T \quad \text{Prob}[L(S) \text{ in } T] < e^{\varepsilon} \text{Prob}[L(S') \text{ in } T] + \delta$$

DP learning was applied to Histograms, regressions, decision trees, SVM’s and Neural Nets: Gap in sample complexity is large

**Note:** still need to use (MPC or HE) to protect the training data input to $L$, even if output hypothesis will be differentially private
What about Model Stealing?

Table 1: Results of model extraction attacks on ML services. For each target model, we report the number of prediction queries made to the ML API in an attack that extracts a 100% equivalent model. The attack time is primarily influenced by the service’s prediction latency ($\approx 100\text{ms}/\text{query}$ for Amazon and $\approx 500\text{ms}/\text{query}$ for BigML).

Unnecessary Vulnerability? Services Report Confidence levels

Figures from “Stealing Machine Learning Models via Prediction APIs” [TZJRR16]
Are we done yet?  

Wait a second!

Why do we trust all these users and their training data (or the servers to follow the protocol)?

This is a Fundamental Question

The stakes are too high to pretend it doesn’t matter
Challenge 2: Need to ensure models reflect data accurately and are not tampered with and data is not poisoned.

• How to verify that everyone (servers and users) follows the protocol during the training phase

• How to make Learning robust to adversarial inputs
  • Distributed Optimization + Byzantine Agreement
  Toward achieving “Robust” and “Statistically-Optimal” gradient descent
  [BJK15, BMGS17, YCRB18]

• How to verify model is not modified post training phase
Verify Everyone Follows the protocol: build MPC for malicious parties

- Information theoretic [GW88] <1/3 Malicious colluders: efficient but may be too much interaction

- Add commitments + Zero Knowledge Proofs to implementations
  - Non-Interactive SNARK, STARK with setup
  - Or Some Interaction

- Dovetails work in the block chain world on adding zk-proofs for anonymity, privacy, enterprise proofs of correct supply chains
Verify Everyone Follows the protocol: build MPC for malicious parties

- Information theoretic [GW88] <1/3 Malicious colluders: efficient but too much interaction
- Add commitments + Zero Knowledge Proofs to implementations
- Non-Interactive SNARK, STARK with setup
- Or Some Interaction
- Dovetails work in the blockchain world on adding zk-proofs for anonymity, privacy, enterprise proofs of correct supply chains
Verify the Model/Findings are accurate (extending robust statistics to IP-land)

Extend Interactive Proofs + PCPs to the land of “proofs about distributions” [GRothblum18]

I have an hypothesis consistent with distribution $D$ (which I may own)
I claim 95% accuracy

I want to verify the model is 95% accurate on $D$ which I have a limited ability to sample
New ML Challenges: an opportunity

For using the last 30 years of “crypto computing” in practice

For developing new theory for crypto for ML
Thanks to

Peter Bartlett
Zvika Brakersky
Aloni Cohen
Ran Cohen
Adam Klivans
Alexander Madry
Daniel Masny
Raluca Popa
Guy Rothblum
Adi Shamir
Yonadav Shavit
Vinod Vaikuntanathan

And anyone else I bothered with questions on this topic...