Brokered Agreements in Multi-Party Machine Learning

10th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2019)

Clement Fung, Ivan Beschastnikh
University of British Columbia
The emerging ML economy

- With the explosion of machine learning (ML), data is the new currency!
  - Good quality data is vital to the health of ML ecosystems
- Improve models with more data from more sources!
Actors in the ML economy

- **Data providers:**
  - Owners of potentially private datasets
  - Contribute data to the ML process

- **Model owners:**
  - Define model task and goals
  - Deploy and profit from trained model

- **Infrastructure providers:**
  - Host training process and model
  - Expose APIs for training and prediction
Actors in today’s ML economy

- Data providers supply data for model owners
- Model owners:
  - Manage infrastructure to host computation
  - Provide privacy and security for data providers
  - Use the model for profit once training is complete
In-House privacy solutions

In-House privacy solutions

Incentive trade-off in the ML economy

- Not only correctness, but there is an issue with incentives:
  - Data providers want to keep their data as private as possible
  - Model owners want to extract as much value from the data as possible
- Service providers lack incentives to provide fairness [1]
  - Need solutions that can work without cooperation from the system provider and are deployed from outside the system itself

Incentive trade-off in the ML economy

- Not only correctness, but there is an issue with incentives:
  - Data providers want to keep their data as private as possible
  - Model owners want to extract as much value from the data as possible
- We cannot trust model owners to control the ML incentive tradeoff!

Incentives in today’s ML economy

- Data providers supply data for model owners
- Model owners:
  - Manage infrastructure to host computation
  - Provide privacy and security for data providers
  - Use the model for profit once training is complete
Incentives in today’s ML economy

- Data providers supply data for model owners
- Model owners have incentive to:
  - Manage infrastructure to host computation
  - Provide privacy and security for data providers
  - Use the model for profit once training is complete
Our contribution: Brokered learning

- Introduce a broker as a neutral infrastructure provider:
  - Manage infrastructure to host ML computation
  - Provide privacy and security for data providers and model owners
Federated learning

- A recent push for privacy-preserving multi-party ML [1]:
  - Send model updates over network
  - Aggregate updates across multiple clients
  - Client-side differential privacy [2]
  - Better speed, no data transfer
  - State of the art in multi-party ML
  - Brokered learning builds on federated learning

Data providers are not to be trusted

- Giving data providers unmonitored control over compute:
  - Providers can maximize privacy, **give zero utility or attack system**
  - Providers can attack ML model, compromising integrity [1]
  - Providers can attack other providers, compromising privacy [2]

Data providers are not to be trusted

- Giving data providers unmonitored control over compute:
  - Providers can maximize privacy, **give zero utility or attack system**
  - Providers can attack ML model, compromising integrity [1]

**We also cannot trust data providers to control the ML incentive tradeoff!**

Putting it all together

- The state of the art in multi-party ML
  - Gives too much control to model owners
  - Not privacy focused and vulnerable
- State of the art in private multi-party ML (federated learning)
  - Require trust in model owners or data providers
  - But there is no incentive for either to do so
- Data marketplaces (blockchains) [1]
  - Security and system overkill
  - Much too slow for modern use cases

Putting it all together

More Centralized
Less Private/Secure

Less Centralized
More Private/Secure
Putting it all together

Centralized Parameter Server

Putting it all together

Centralized Parameter Server

More Centralized
Less Private/Secure

Federated Learning

Less Centralized
More Private/Secure
Putting it all together

Centralized Parameter Server

Federated Learning

Blockchain-based Multi-party ML

More Centralized
Less Private/Secure

Less Centralized
More Private/Secure
Putting it all together

- Centralized Parameter Server
- Federated Learning
- Brokered Learning
- Blockchain-based Multi-party ML

More Centralized
Less Private/Secure

Less Centralized
More Private/Secure
Our contributions

- Current multi-party ML systems use unsophisticated threat/incentive model:
  - Trust the model owner
- New brokered learning setting for privacy-preserving ML
- New defences against known ML attacks for this setting
- TorMentor: A brokered learning example of an anonymous ML system

**Brokered Learning**: A new standard for incentives in secure ML
Brokered Learning
Brokered agreements in the ML economy

● Federated learning:
  ○ Communicate with model owner
  ○ Trust that model owner is not malicious
  ○ Model owners have full control over model and process

● Brokered learning
  ○ Communicate with neutral broker
  ○ Broker executes model owner’s validation services
  ○ **Decouple model owners and infrastructure**
Brokered learning components

- Deployment verifier
  - Interface for model owners ("curators")
- Provider verifier
  - Interface for data providers
- Aggregator
  - Host ML deployments
  - Collect and aggregate model updates
  - Same as federated learning
Deployment verifier API

- Serves as model owner interface
  - `curate()`: Launch curator deployment
  - Set provider verifier parameters
  - `fetch()`: Access to model once trained
- Protects the ML model from abuse from curator during training
- E.g. Blockchain smart contracts [1]

Provider verifier API

- Serves as data provider interface
  - Defined by curator
  - `join()`: Verify identity and allow provider join
  - `update()`: Verify and allow model update
- Protect model from malicious data providers
- E.g. Access tokens and statistical tests
Brokere learning workflow

- Curator: Create deployment
  - Define model and provide deployment parameters
  - Define verification services
Brokered learning workflow

- **Curator: Create deployment**
  - Define model and provide deployment parameters
  - Define verification services
- **Data providers: Join model**
  - Define personal privacy preferences (ε)
  - Pass verification on join
Brokered learning workflow

- Curator: Create deployment
  - Define model and provide deployment parameters
  - Define verification services
- Data providers: Join model and train
  - Define personal privacy preferences ($\varepsilon$)
  - Pass verification on join
  - Iterative model updates
  - Pass verification on model update
Brokered learning workflow

- **Curator**: Create deployment
  - Define model and provide deployment parameters
  - Define verification services
- **Data providers**: Join model and train
  - Define personal privacy preferences ($\varepsilon$)
  - Pass verification on join
  - Iterative model updates
  - Pass verification on model update
- **Complete training**
  - Return model to curator
Threat model

- **Assume:**
  - Broker honours verifier parameters
  - Users adhere to the given APIs for joining and model updates
  - Curators and data providers can collaborate

- **Trust is based on incentives:** broker is neutral to ML incentive trade-off
  - If broker attacks clients or violates curator specifications, reputation lost
  - Governments, large organizations, blockchains
TorMentor: An Example Brokered Learning System
TorMentor system goals

- Use brokered learning to **build the first anonymous ML system:**
  - Further support privacy in multi-party ML
  - Data provider and curator identity are hidden:
    - From each other and from the broker
- Meet defined learning objectives in reasonable time
  - Compared to WAN federated learning baseline
Implementation on Tor

- Onion routing protocols (Tor) [1]
  - Hide source and destination of messages by communicating through chain of random nodes in system
  - Hide identity of users in distributed ML!
  - Deploy broker as hidden Tor service

Implementation

- Libraries written in Python and Go
  - 1500 LOC Python, 600 LOC Go
- Tested on “credit card default” UCI dataset
  - Logistic classifier
  - 30000 examples, 24 features (14 MB / client)
- Deployment at scale on Azure (8 data centres)
  - Deploy curators and data providers as users over wide area network
Convergence at scale over Tor

With Tor

Without Tor

- 10 clients
- 50 clients
- 100 clients
- 200 clients

Training Error versus Time (s)
Convergence at scale over Tor

With Tor

Without Tor

TorMentor is within 4-10x baseline, and still converges while serving 200 clients on a WAN.
Provider verifier

- Reject on Negative Influence (RONI) [1]
  - Reject datasets with negative impact on “influence” metric
    - Typically, just use validation error

- Model curator defines a distributed RONI:
  - Evaluate influence of model updates instead of data
  - Use curator provided validation set
  - Tune using data provider proof-of-work [2]

Evaluation: Provider verifier
Evaluation: Provider verifier

The curator can define a service through the broker that rejects attacks under certain conditions.
Brokered learning opportunities and limitations

- Modern use cases:
  - Blockchain-based data marketplaces
  - Standardizing “ML as a service”
  - GDPR Compliance

- Limitations
  - Moving from 2 actors to 3
  - Adoption from big players
Summary of contributions

- Existing ML systems do not provide:
  - Incentives, privacy, security
- We propose **brokered learning** as an alternative to federated learning
  - APIs to protect process from model owners and data providers
- TorMentor prototype
  - Supports anonymous ML between data providers and curators
  - Allows curator defined process to reject malicious data providers

https://github.com/DistributedML/TorML