Stealing Machine Learning Models via Prediction APIs

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, Thomas Ristenpart

Usenix Security Symposium

Austin, Texas, USA

August, 11th 2016

CORNELL TECH

Machine Learning (ML) Systems

(1) Gather labeled data

$x^{(1)}, y^{(1)}$ $x^{(2)}, y^{(2)}$ ...

n-dimensional feature vector x

Dependent variable y

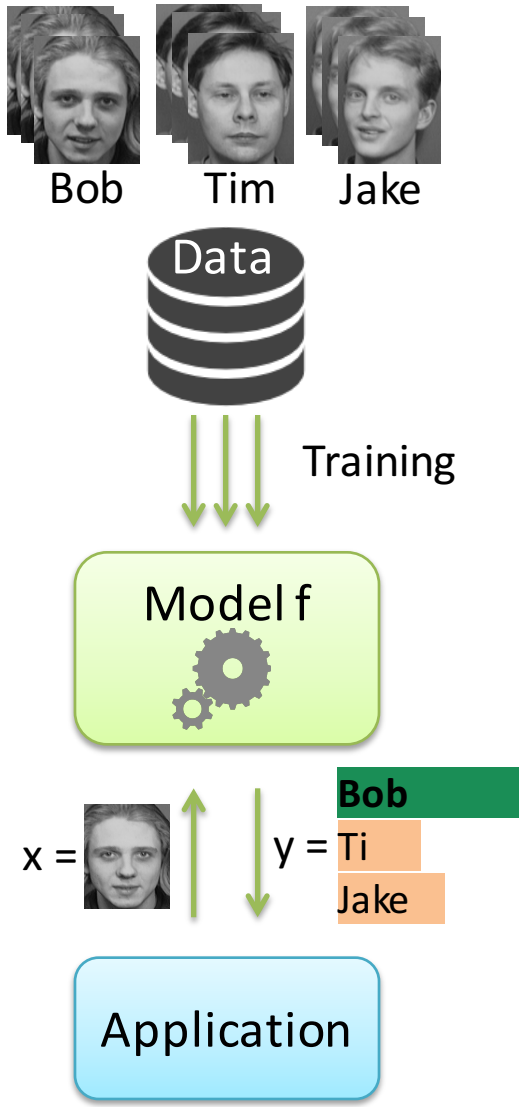
(2) Train ML model f from data

$f(x) = y$

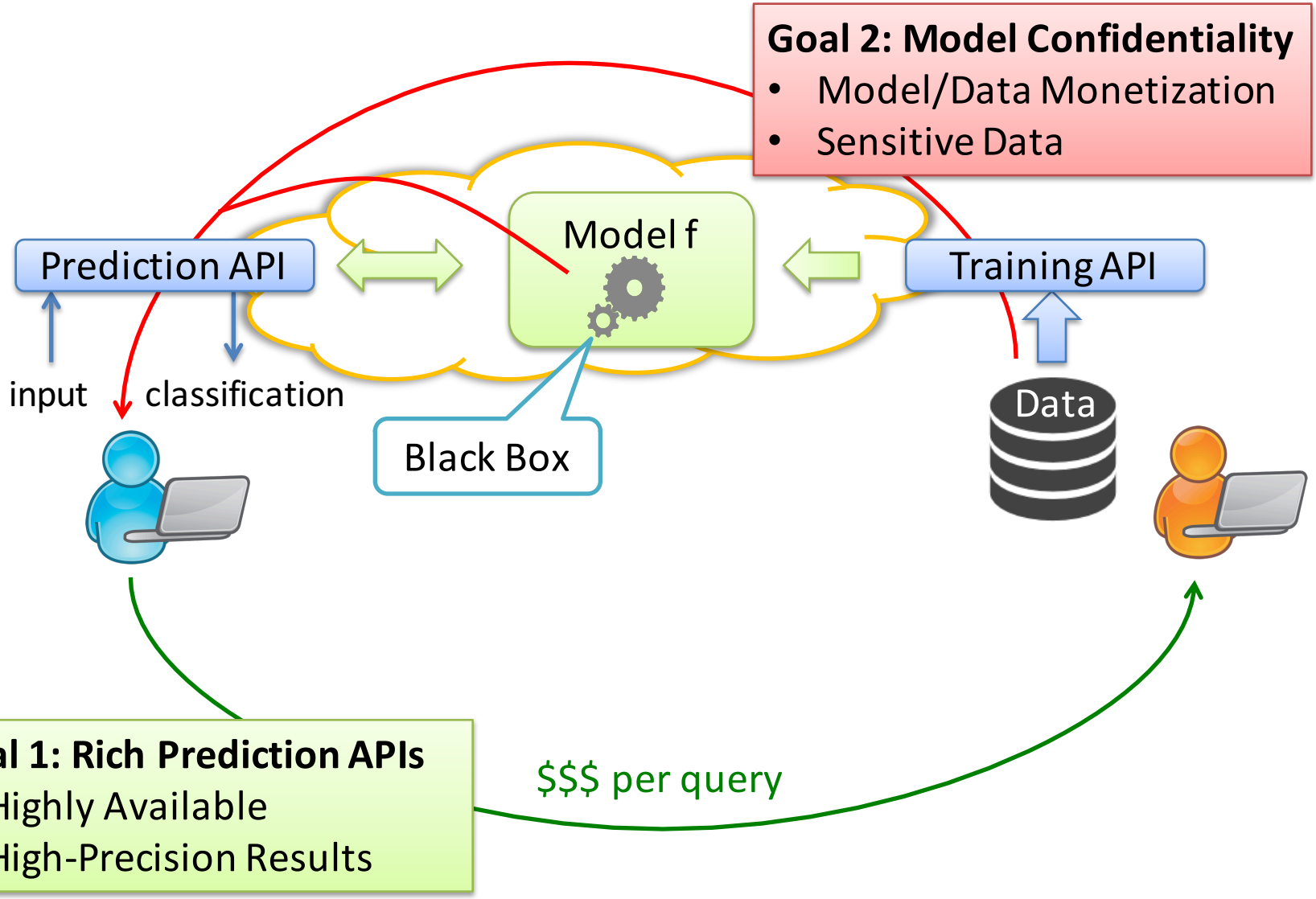
Prediction

Confidence

(3) Use f in some application or publish it for others to use



Machine Learning as a Service (MLaaS)



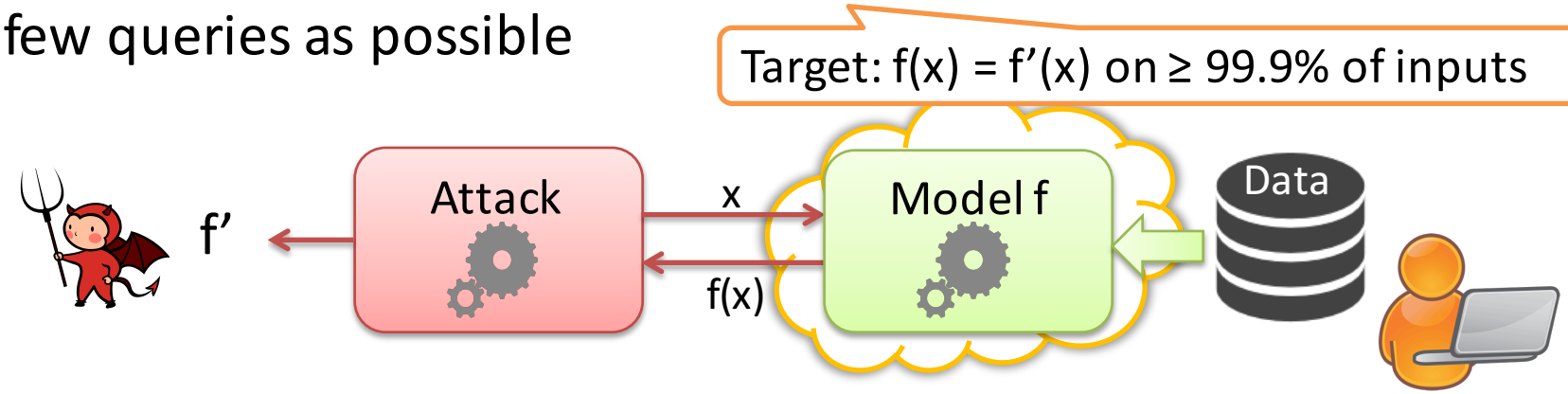
Machine Learning as a Service (MLaaS)

Service	Model types
Amazon	**Logistic regressions**
Google	??? (announced: logistic regressions, decision trees, neural networks, SVMs)
Microsoft	Logistic regressions, decision trees, neural networks, SVMs
PredictionIO	Logistic regressions, decision trees, SVMs (white-box)
BigML	**Logistic regressions, decision trees**

Sell Datasets – Models – Prediction Queries
$$$ to other users $$$

Model Extraction Attacks

Goal: Adversarial client learns close approximation of f using as few queries as possible

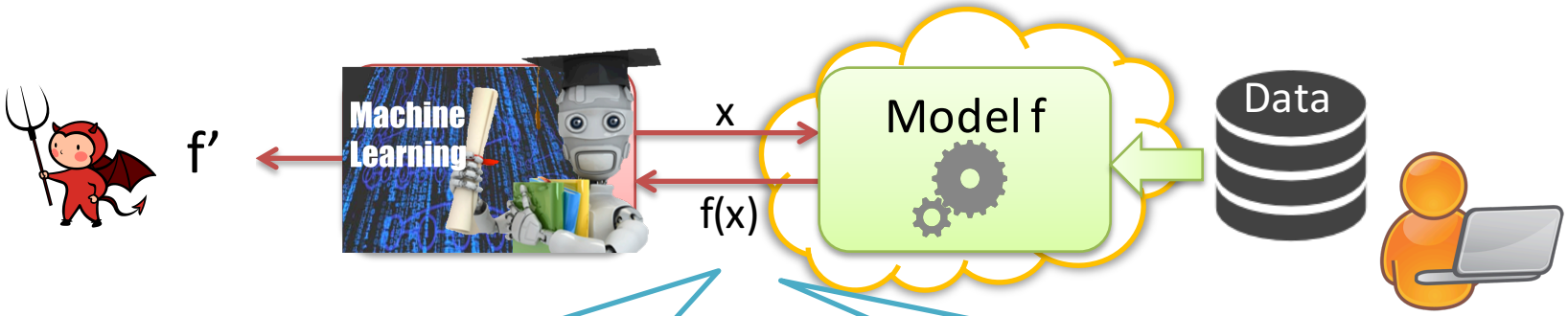


Applications:

1) Undermine pay-for-prediction pricing model
2) Facilitate privacy attacks (
3) Stepping stone to model-evasion
   [Lowd, Meek – 2005] [Srndic, Laskov – 2014]

Model Extraction Attacks (Prior Work)

Goal: Adversarial client learns close approximation of f using as few queries as possible



No! Prediction APIs return more information than assumed in prior work and "traditional" ML

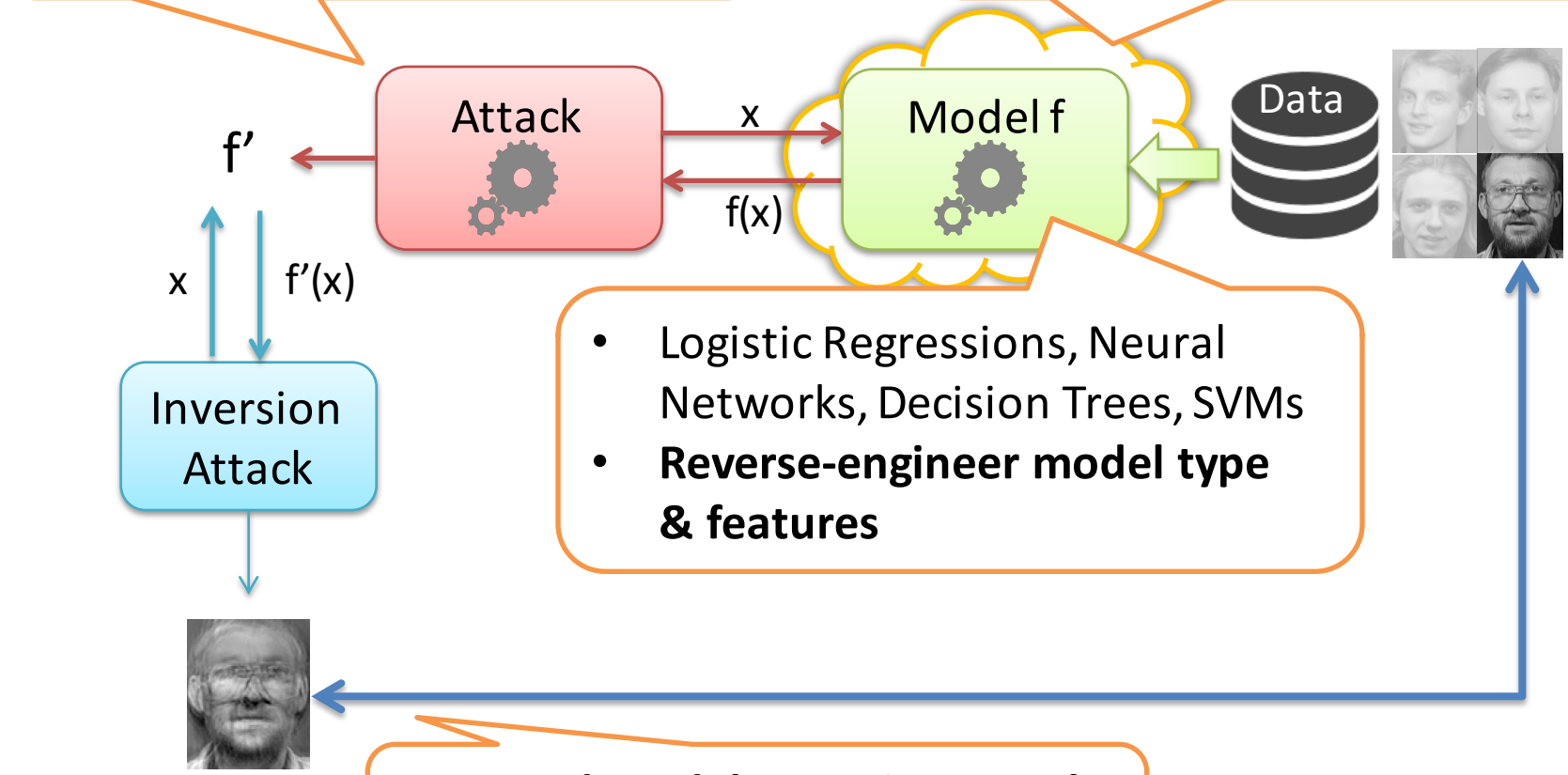
Isn't this "just Machine Learning"?

If f(x) is just a class label: learning with membership queries

- Boolean decision trees [Kushilevitz, Mansour – 1993]
- Linear models (e.g., binary regression) [Lowd, Meek – 2005]

Main Results

f'(x) = f(x) on 100% of inputs
100s-1000's of online queries



- Logistic Regressions, Neural Networks, Decision Trees, SVMs
- **Reverse-engineer model type & features**

Improved Model-Inversion Attacks
[Fredrikson et al. 2015]

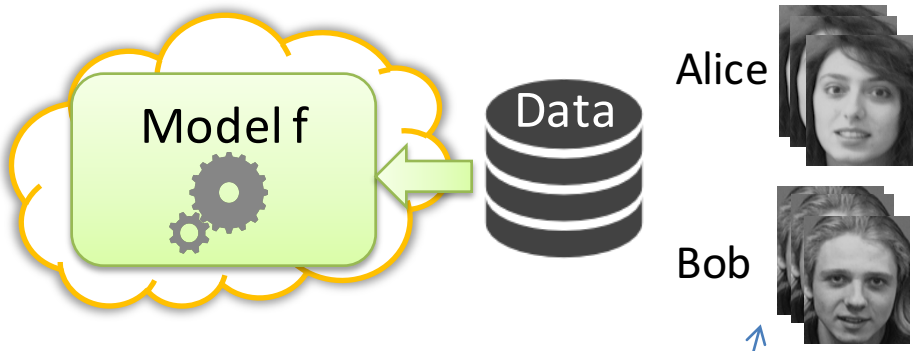
Model Extraction Example: Logistic Regression

Task: Facial Recognition of two people (binary classification)

n+1 parameters w,b chosen using training set to minimize expected error

$$f(x) = 1 / (1+e^{-(w*x + b)})$$

f maps features to predicted probability of being "Alice"
≤ 0.5 classify as "Bob"
> 0.5 classify as "Alice"

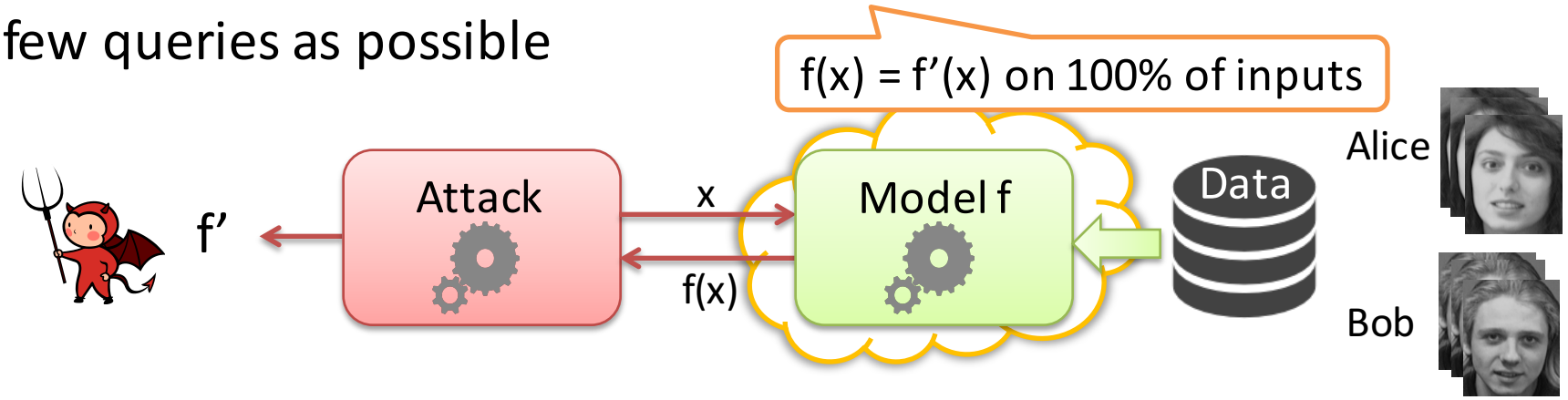


Feature vectors are pixel data
e.g., n = 92 * 112 = 10,304

Generalize to c > 2 classes with *multinomial logistic regression*
$f(x) = [p_1, p_2, \ldots, p_c]$ predict label as $\text{argmax}_i\ p_i$

Model Extraction Example: Logistic Regression

Goal: Adversarial client learns close approximation of f using as few queries as possible



$$f(x) = 1 / (1+e^{-(w*x+b)})$$

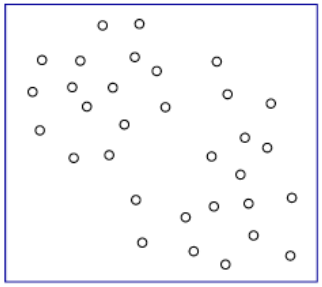
$$\ln\left(\frac{f(x)}{1-f(x)}\right) = w*x + b$$

Linear equation in n+1 unknowns w,b

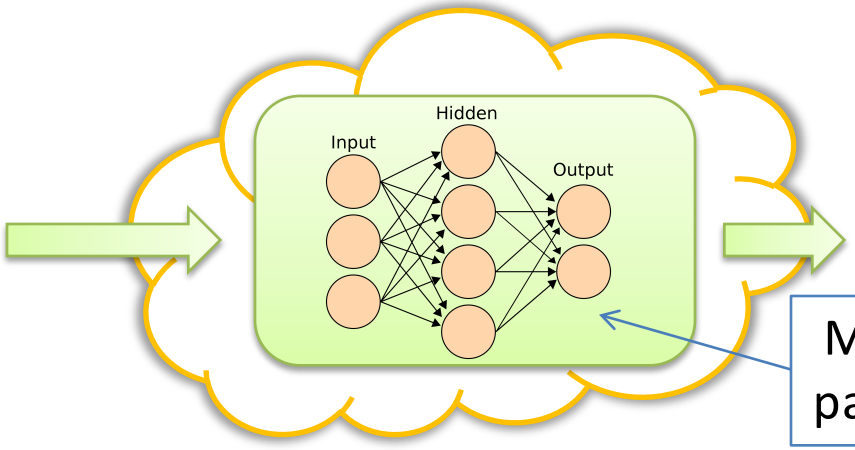
Query n+1 random points ⇒ solve a linear system of n+1 equations

Generic Equation-Solving Attacks

random inputs X



MLaaS Service



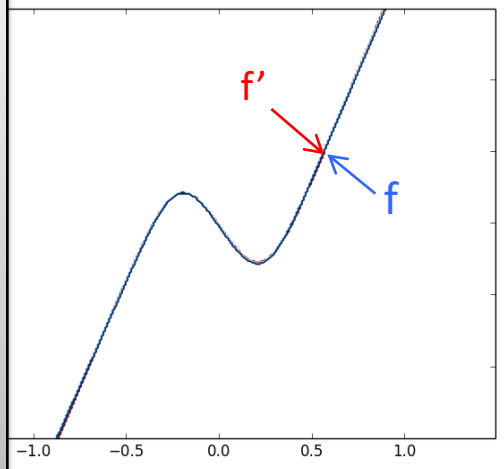
outputs Y

confidence values

$[f_1(x), f_2(x), \ldots, f_c(x)] \in [0,1]^c$

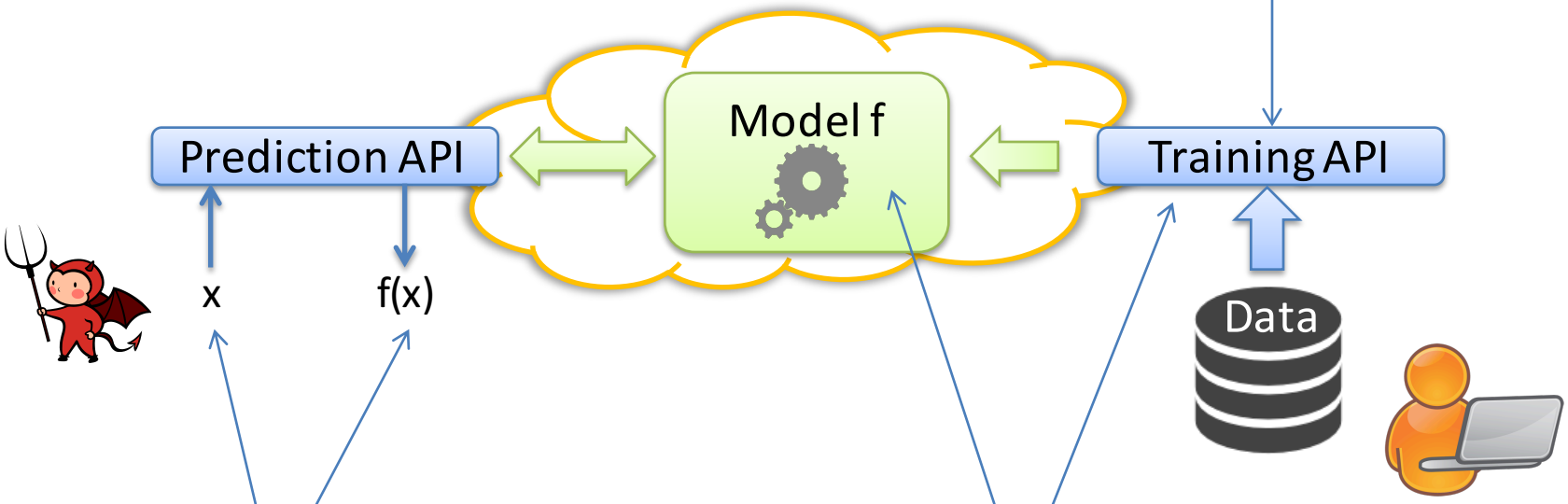
Model f has k parameters W

- Solve non-linear equation system in the weights W
  - Optimization problem + gradient descent
  - *"Noiseless Machine Learning"*
- Multinomial Regressions & Deep Neural Networks:
  - >99.9% agreement between f and f'
  - ≈ 1 query per model parameter of f
  - 100s - 1,000s of queries / seconds to minutes



MLaaS: A Closer Look

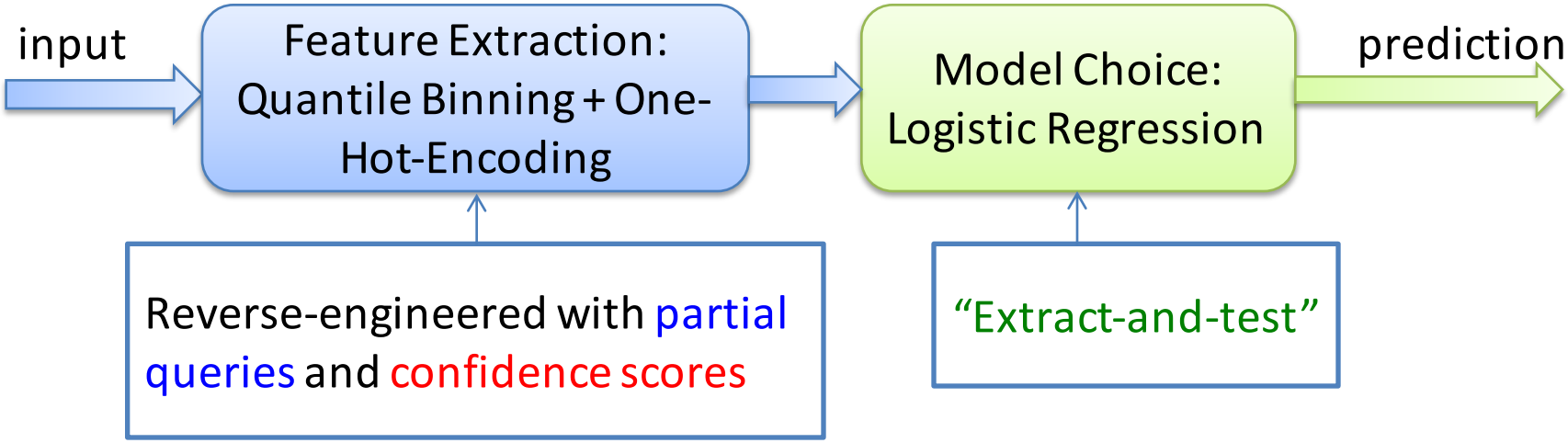
Feature Extraction:
(automated and partially documented)



- Class labels and confidence scores
- Support for partial inputs

ML Model Type Selection:
logistic or linear regression

Online Attack: AWS Machine Learning

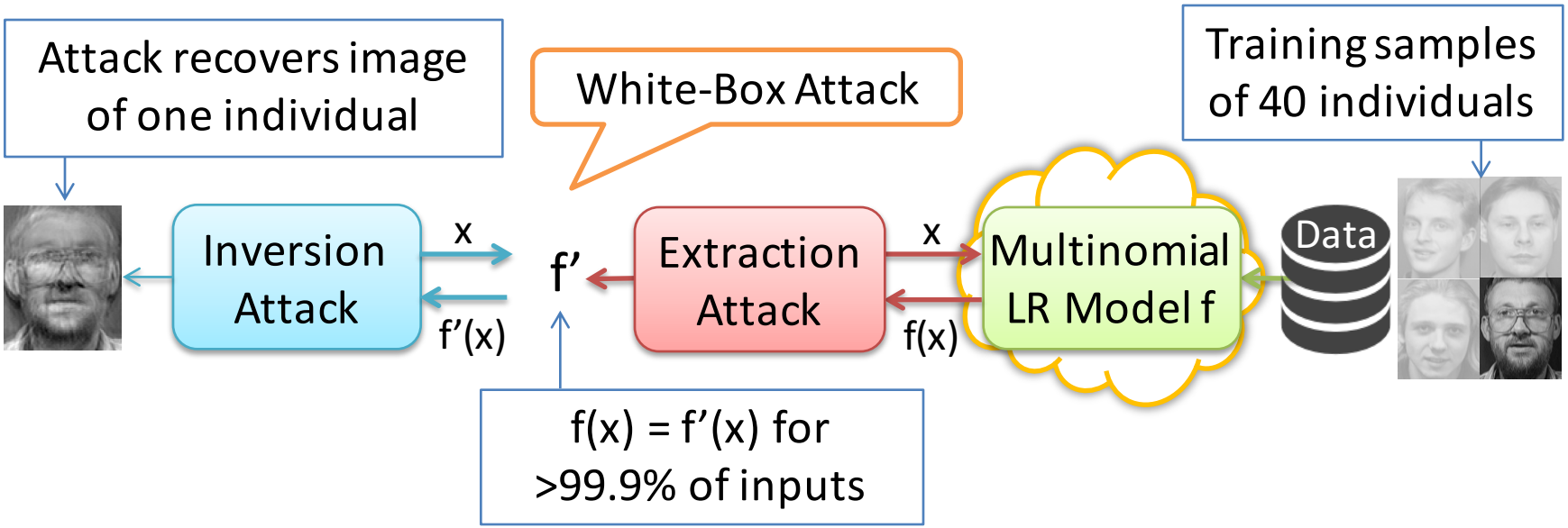


Model	Online Queries	Time (s)	Price ($)
Handwritten Digits	650	70	0.07
Adult Census	1,485	149	0.15

Extracted model f' agrees with f on 100% of tested inputs

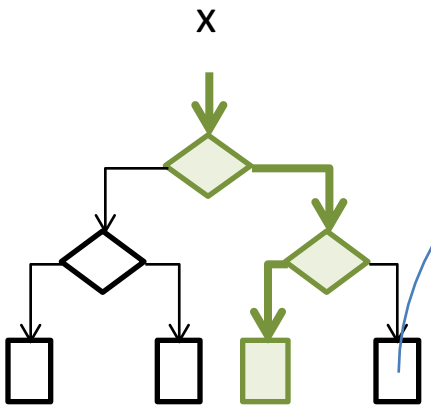
Application: Model-Inversion Attacks

Infer training data from trained models [Fredrikson et al. – 2015]



Strategy	Attack against 1 individual		Attack against all 40 individuals	
	Online Queries	Attack Time	Online Queries	Attack Time
Black-Box Inversion [Fredrikson et al.]	20,600	24 min	×40 800,000	16 hours
Extract-and-Invert (our work)	41,000	10 hours	×1 41,000	10 hours

Extracting a Decision Tree



Confidence value derived from class distribution in the training set

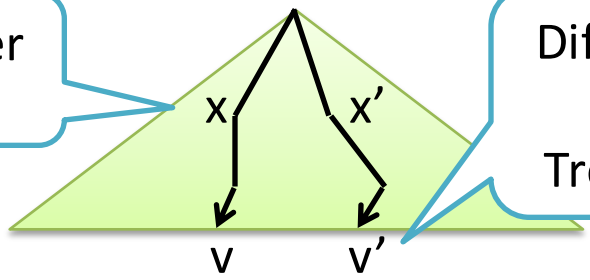
Kushilevitz-Mansour (1992)

- Poly-time algorithm with *membership queries* only
- Only for Boolean trees, impractical complexity

(Ab)using Confidence Values

- Assumption: all tree leaves have unique confidence values
- Reconstruct tree decisions with "differential testing"
- Online attacks on BigML

Inputs x and x' differ in a single feature



Different leaves are reached
⇔
Tree "splits" on this feature

Countermeasures

How to prevent extraction?

API Minimization

f (x) = y

Prediction

Confidence

- Prediction = class label only
- *Learning with Membership Queries*

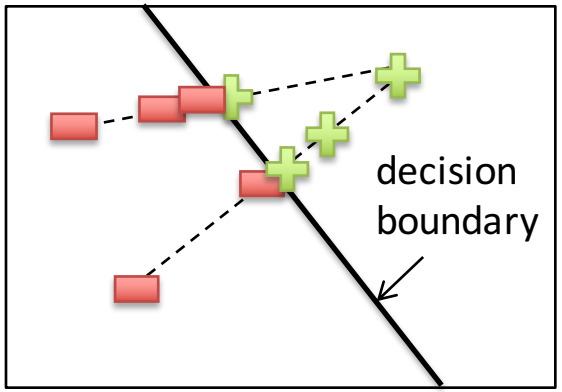
Attack on Linear Classifiers [Lowd,Meek – 2005]

classify as "**+**" if w*x + b > 0 and "**-**" otherwise

n+1 parameters w,b

f(x) = sign(w*x + b)

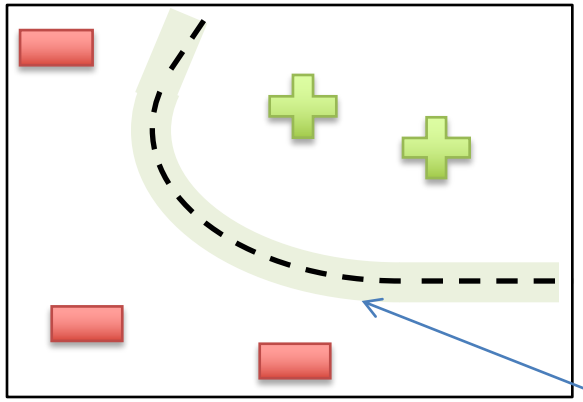
1. Find points on decision boundary (w*x + b = 0)
   - Find a "**+**" and a "**-**"
   - Line search between the two points
2. Reconstruct w and b (up to scaling factor)



Generic Model Retraining Attacks

- Extend the Lowd-Meek approach to non-linear models
- Active Learning:
  - Query points close to "decision boundary"
  - Update f' to fit these points
- Multinomial Regressions, Neural Networks, SVMs:
  - >99% agreement between f and f'
  - ≈ 100 queries per model parameter of f

≈ 100× less efficient than equation-solving



query more points here

Conclusion

Rich prediction APIs  **Model & data confidentiality**

Efficient Model-Extraction Attacks

- Logistic Regressions, Neural Networks, Decision Trees, SVMs
- Reverse-engineering of model type, feature extractors
- Active learning attacks in membership-query setting

Applications

- Sidestep model monetization
- Boost other attacks: privacy breaches, model evasion

Thanks! **Find out more: https://github.com/ftramer/Steal-ML**