SAND2019-6536PE

An Overview of Training Data Security Vulnerabilities: Machine Learning is a Leaky Black Box

Philip Kegelmeyer, Jeremy Wendt, Cosmin Safta

Sandia National Laboratories, Livermore, CA

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

LANL, June 13, 2019

Office of the Director of National Intelligence
IARPA
BE THE FUTURE

Doing Bad Things…

- **With** AI
- **To** AI
  - Reveal the wrong thing

INTELLIGENCE ADVANCED RESEARCH PROJECTS ACTIVITY (IARPA)

Outline

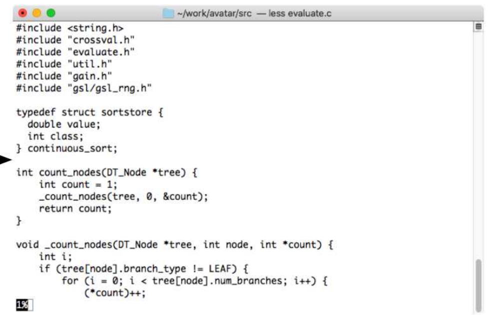
- **Components of a machine learning system**
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

Training and testing a machine learning model

Training Data

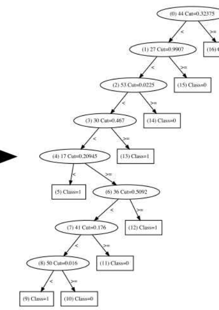
DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Machine Learning Code




```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
```

Learned Model



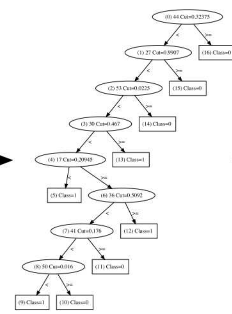
Private

Public

Test Data

CGINTX	CGINTY	SNR	…	PMIN
14	123	0.54	…	0.34

Learned Model



Classification with Weights

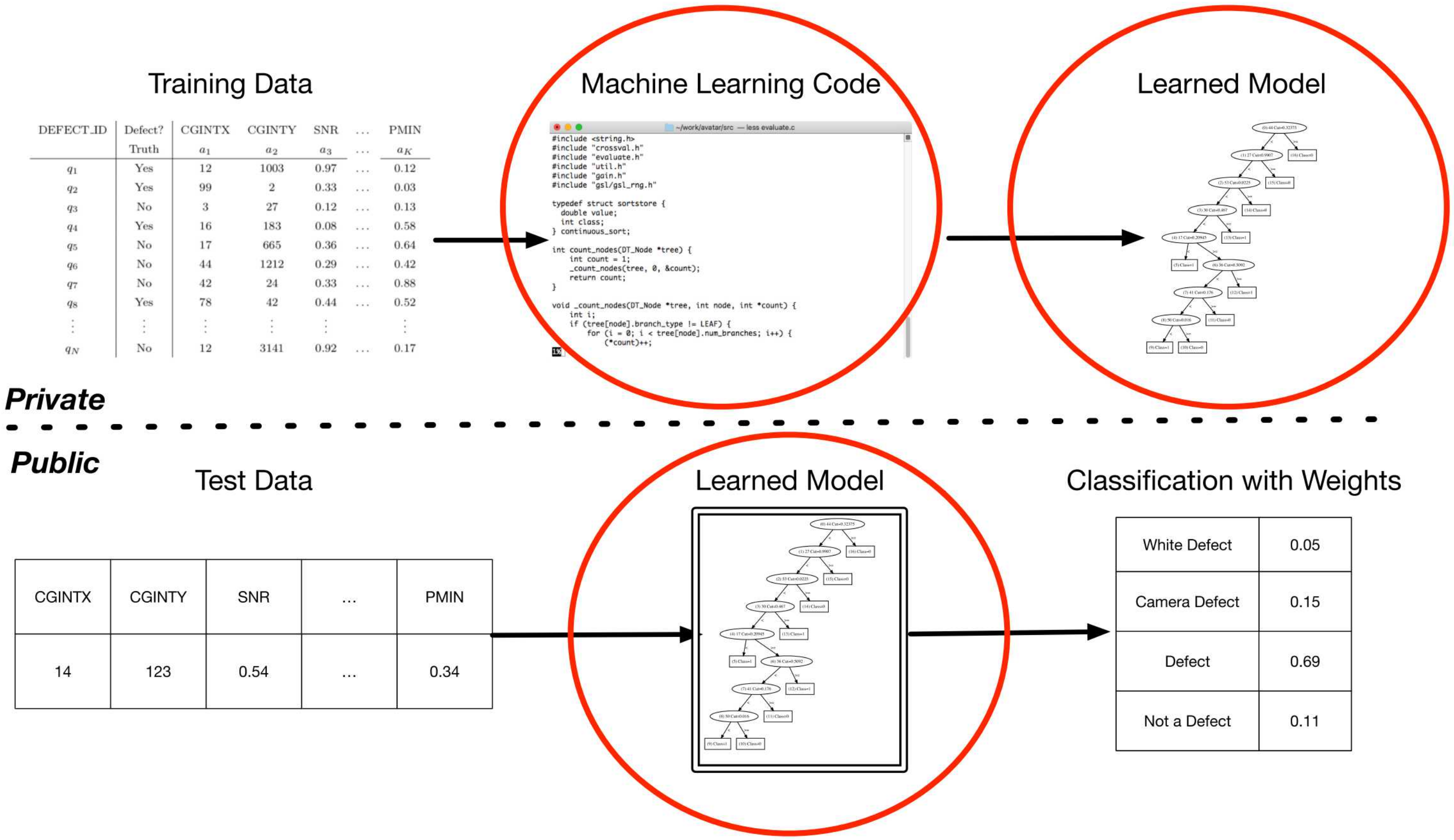
White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. **Exfiltration via model parameters**
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

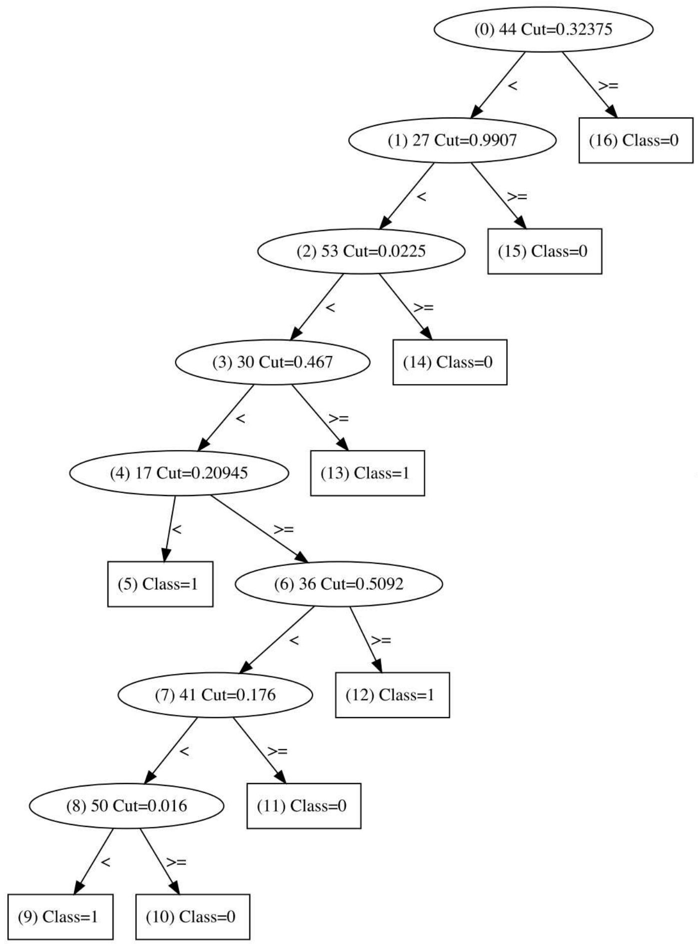
Exfiltration via model parameters

Attack: a code backdoor stashing training data in model parameters



Machine Learning Models That Remember Too Much[9]

A decision tree is a series of threshold parameters




```
SPLIT CONTINUOUS ATT# 44 < 0.323750
SPLIT CONTINUOUS ATT# 27 < 0.990700
SPLIT CONTINUOUS ATT# 53 < 0.022500
SPLIT CONTINUOUS ATT# 30 < 0.467000
SPLIT CONTINUOUS ATT# 17 < 0.209450
LEAF Class 1 Proportions 0 10
SPLIT CONTINUOUS ATT# 17 >= 0.209450
SPLIT CONTINUOUS ATT# 36 < 0.509200
SPLIT CONTINUOUS ATT# 41 < 0.176000
SPLIT CONTINUOUS ATT# 50 < 0.016000
LEAF Class 1 Proportions 2 11
SPLIT CONTINUOUS ATT# 50 >= 0.016000
LEAF Class 0 Proportions 10 3
SPLIT CONTINUOUS ATT# 41 >= 0.176000
LEAF Class 0 Proportions 22 0
SPLIT CONTINUOUS ATT# 36 >= 0.509200
LEAF Class 1 Proportions 1 9
SPLIT CONTINUOUS ATT# 30 >= 0.467000
LEAF Class 1 Proportions 2 72
SPLIT CONTINUOUS ATT# 53 >= 0.022500
LEAF Class 0 Proportions 16 1
SPLIT CONTINUOUS ATT# 27 >= 0.990700
LEAF Class 0 Proportions 17 1
SPLIT CONTINUOUS ATT# 44 >= 0.323750
LEAF Class 0 Proportions 30 1
```

Encode the training data as digits

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17

Compress,
Encrypt,
Serialize to Digits

9833, 6299, 3495, 4946,
3470, 0158, 2537, 2076,
1277, 3644, 9284, 4085,
4201, 4159, 8444, 7234, …

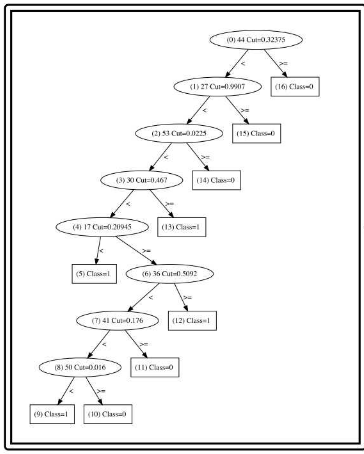
Stash the data in insignificant digits

9833, 6299, 3495, 4946, 3470, 0158, 2537, 2076, 1277, 3644, 9284, 4085, 4201, 4159, 8444, 7234, ...

```
SPLIT CONTINUOUS ATT# 44 < 0.323750
SPLIT CONTINUOUS ATT# 27 < 0.990700
SPLIT CONTINUOUS ATT# 53 < 0.022500
SPLIT CONTINUOUS ATT# 30 < 0.467000
SPLIT CONTINUOUS ATT# 17 < 0.209450
LEAF Class 1 Proportions 0 10
SPLIT CONTINUOUS ATT# 17 >= 0.209450
SPLIT CONTINUOUS ATT# 36 < 0.509200
SPLIT CONTINUOUS ATT# 41 < 0.176000
SPLIT CONTINUOUS ATT# 50 < 0.016000
LEAF Class 1 Proportions 2 11
SPLIT CONTINUOUS ATT# 50 >= 0.016000
LEAF Class 0 Proportions 10 3
SPLIT CONTINUOUS ATT# 41 >= 0.176000
LEAF Class 0 Proportions 22 0
SPLIT CONTINUOUS ATT# 36 >= 0.509200
LEAF Class 1 Proportions 1 9
SPLIT CONTINUOUS ATT# 30 >= 0.467000
LEAF Class 1 Proportions 2 72
SPLIT CONTINUOUS ATT# 53 >= 0.022500
LEAF Class 0 Proportions 16 1
SPLIT CONTINUOUS ATT# 27 >= 0.990700
LEAF Class 0 Proportions 17 1
SPLIT CONTINUOUS ATT# 44 >= 0.323750
LEAF Class 0 Proportions 30 1
```

```
SPLIT CONTINUOUS ATT# 44 < 0.329833
SPLIT CONTINUOUS ATT# 27 < 0.996299
SPLIT CONTINUOUS ATT# 53 < 0.023495
SPLIT CONTINUOUS ATT# 30 < 0.464946
SPLIT CONTINUOUS ATT# 17 < 0.203470
   LEAF Class 1 Proportions 0 10
SPLIT CONTINUOUS ATT# 17 >= 0.200158
SPLIT CONTINUOUS ATT# 36 < 0.502537
SPLIT CONTINUOUS ATT# 41 < 0.172076
SPLIT CONTINUOUS ATT# 50 < 0.011277
   LEAF Class 1 Proportions 2 11
SPLIT CONTINUOUS ATT# 50 >= 0.013644
   LEAF Class 0 Proportions 10 3
SPLIT CONTINUOUS ATT# 41 >= 0.179284
   LEAF Class 0 Proportions 22 0
SPLIT CONTINUOUS ATT# 36 >= 0.504085
   LEAF Class 1 Proportions 1 9
SPLIT CONTINUOUS ATT# 30 >= 0.464201
   LEAF Class 1 Proportions 2 72
SPLIT CONTINUOUS ATT# 53 >= 0.024159
   LEAF Class 0 Proportions 16 1
SPLIT CONTINUOUS ATT# 27 >= 0.998444
   LEAF Class 0 Proportions 17 1
SPLIT CONTINUOUS ATT# 44 >= 0.327234
   LEAF Class 0 Proportions 30 1
```

Recover the data by white box inspection

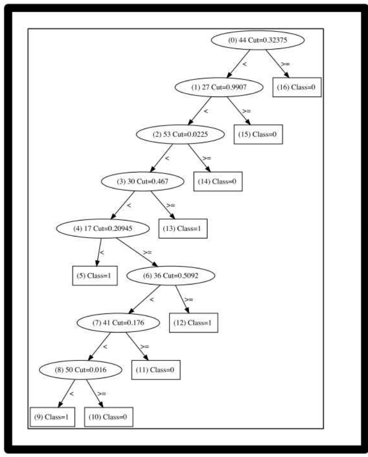


9833, 6299, 3495, 4946,
3470, 0158, 2537, 2076,
1277, 3644, 9284, 4085,
4201, 4159, 8444, 7234, …

Concatenate,
Deserialize,
Decrypt,
Uncompress

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Block exfiltration by providing only a black box?



????, ????, ????, ????,
????, ????, ????, ????,
????, ????, ????, ????,
????, ????, ????, ????, ...

Concatenate,
Deserialize,
Decrypt,
Uncompress

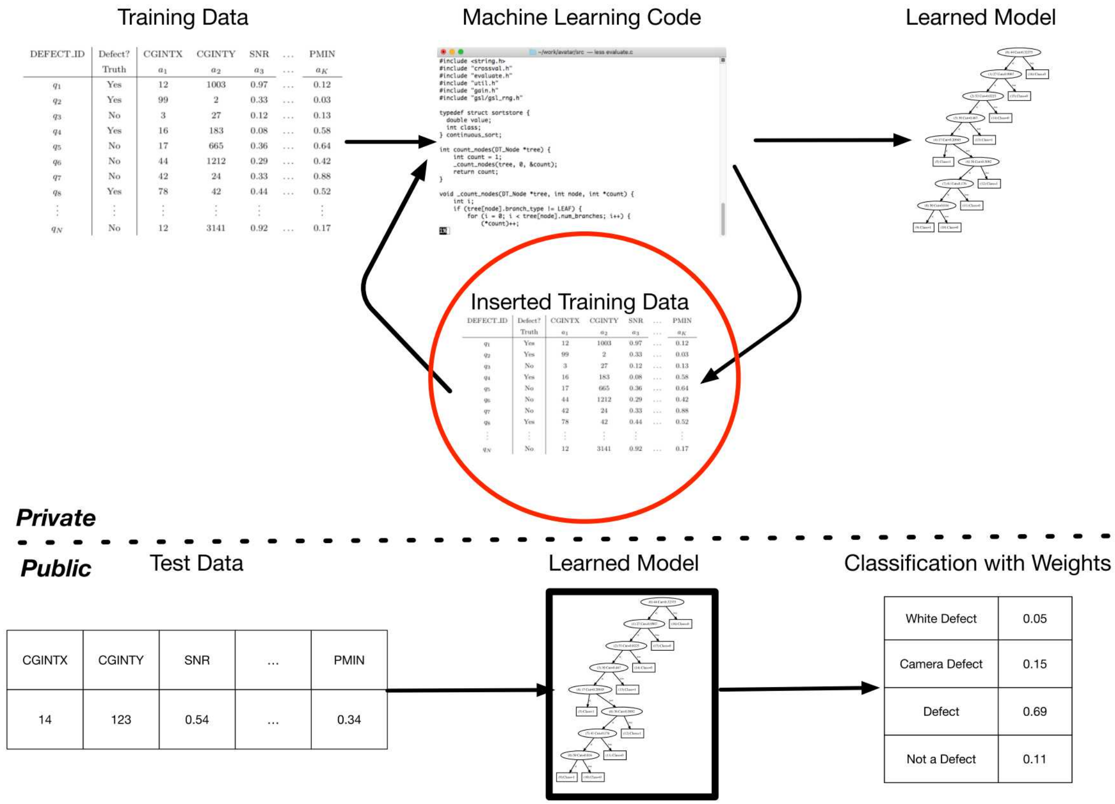
DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. **Exfiltration via model labels**
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

Exfiltration via model labels

Attack: a code backdoor adding carefully designed synthetic training data



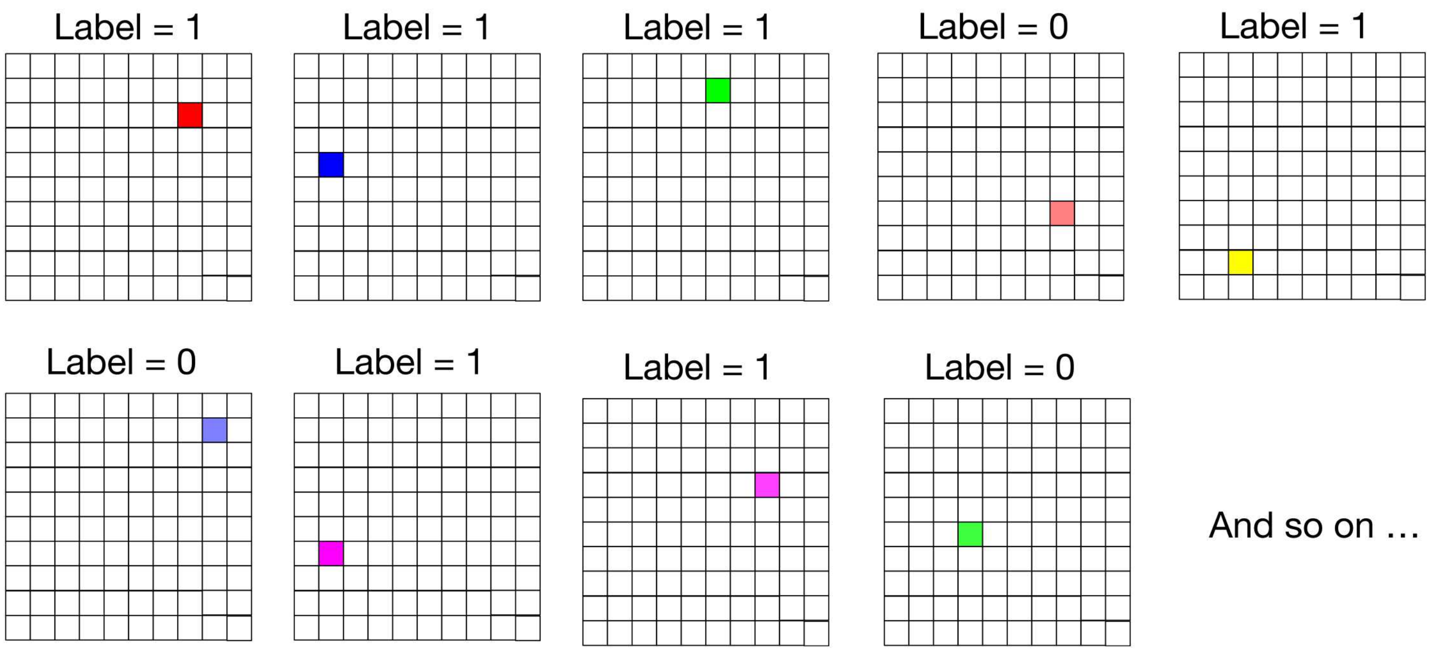
Machine Learning Models That Remember Too Much[9]

Exfiltration of a training image

Choose an image to exfiltrate.

Encode image pixel values as bits, say 1,1,1,0,1,0,1,1,0,. . . .

Create pseudo-random training images to encode those bits as *labels*.



Model learns the labels, dutifully emits them later when probed.

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. **Exploit inadvertent memorization**
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

Exploit inadvertent memorization

Attack: exploit rare string memorization in text prediction

Training Data

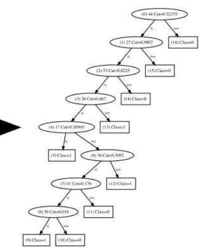
DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Machine Learning Code

~/work/avatar/src — less evaluate.c

```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
1%
```

Learned Model



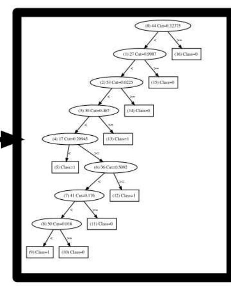
Private

Public

Test Data

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

Learned Model

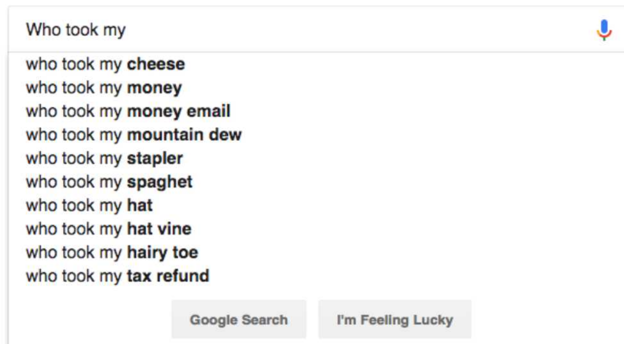


Classification with Weights

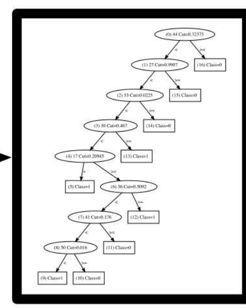
White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

The Secret Sharer: Measuring unintended neural network memorization and extracting secrets[2]

ML to predict the next token in a string



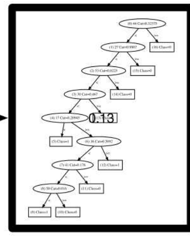
‘Who took my ?”



cheese	0.54
money	0.17
money email	0.12
mountain dew	0.05
stapler	0.03

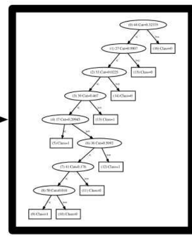
Probe with promising templates

“My SSN is ?”



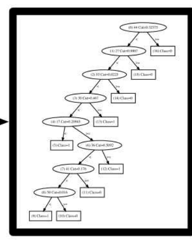
0	1	2	3	4	5	6	7	8	9
0.16	0.07	0.09	0.36	0.05	0.11	0.06	0.12	0.02	0.06

“My SSN is 3?”



0	1	2	3	4	5	6	7	8	9
0.07	0.13	0.03	0.07	0.10	0.20	0.17	0.09	0.08	0.06

“My SSN is 35?”



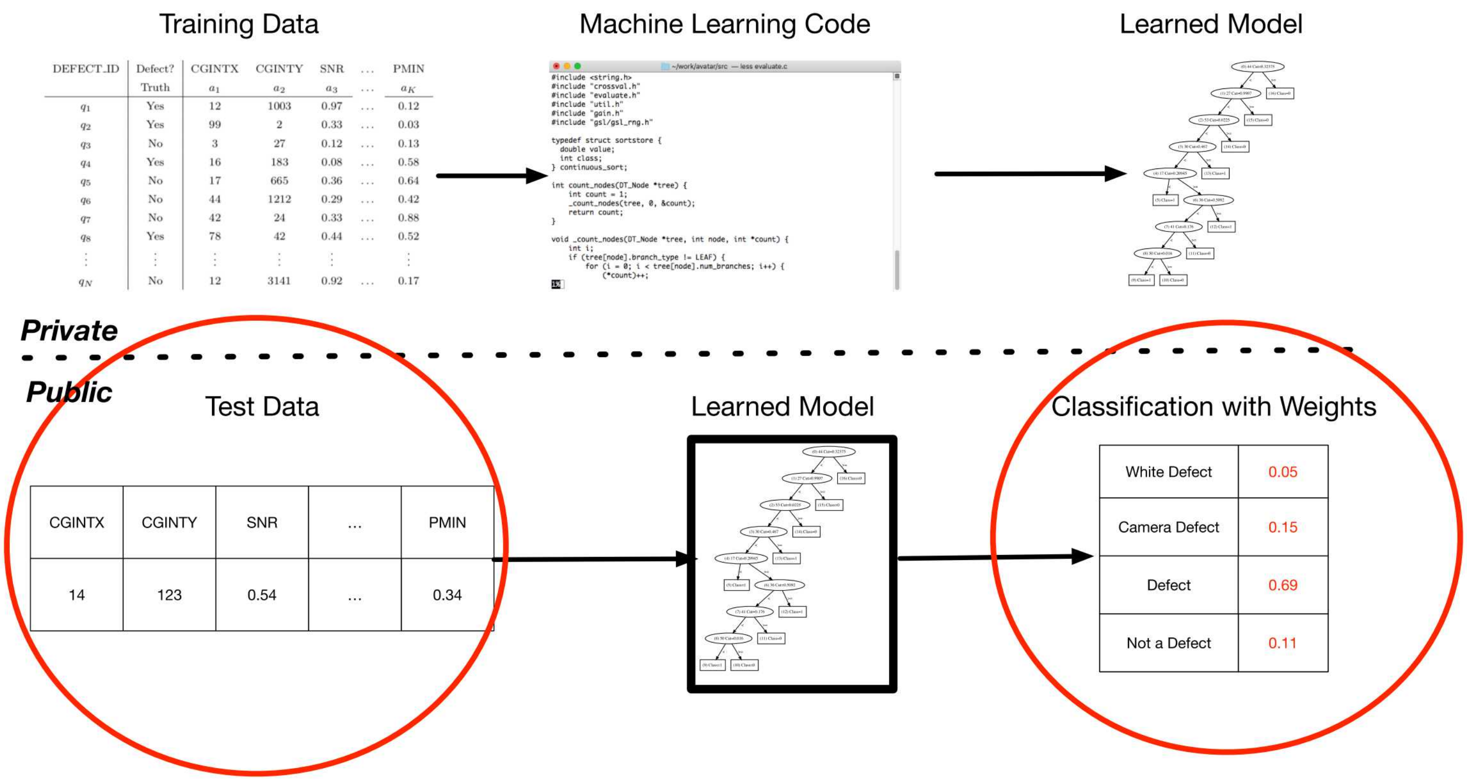
And so on

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. **Attribute inference: recovering training data**
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

Attribute inference #1: recovering training data

Attack: exploit **black box** class label weights to recover **feature vectors**



DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures[3]

Recovery of Part of a Feature Vector

Attacker knows part of feature vector used as training data.

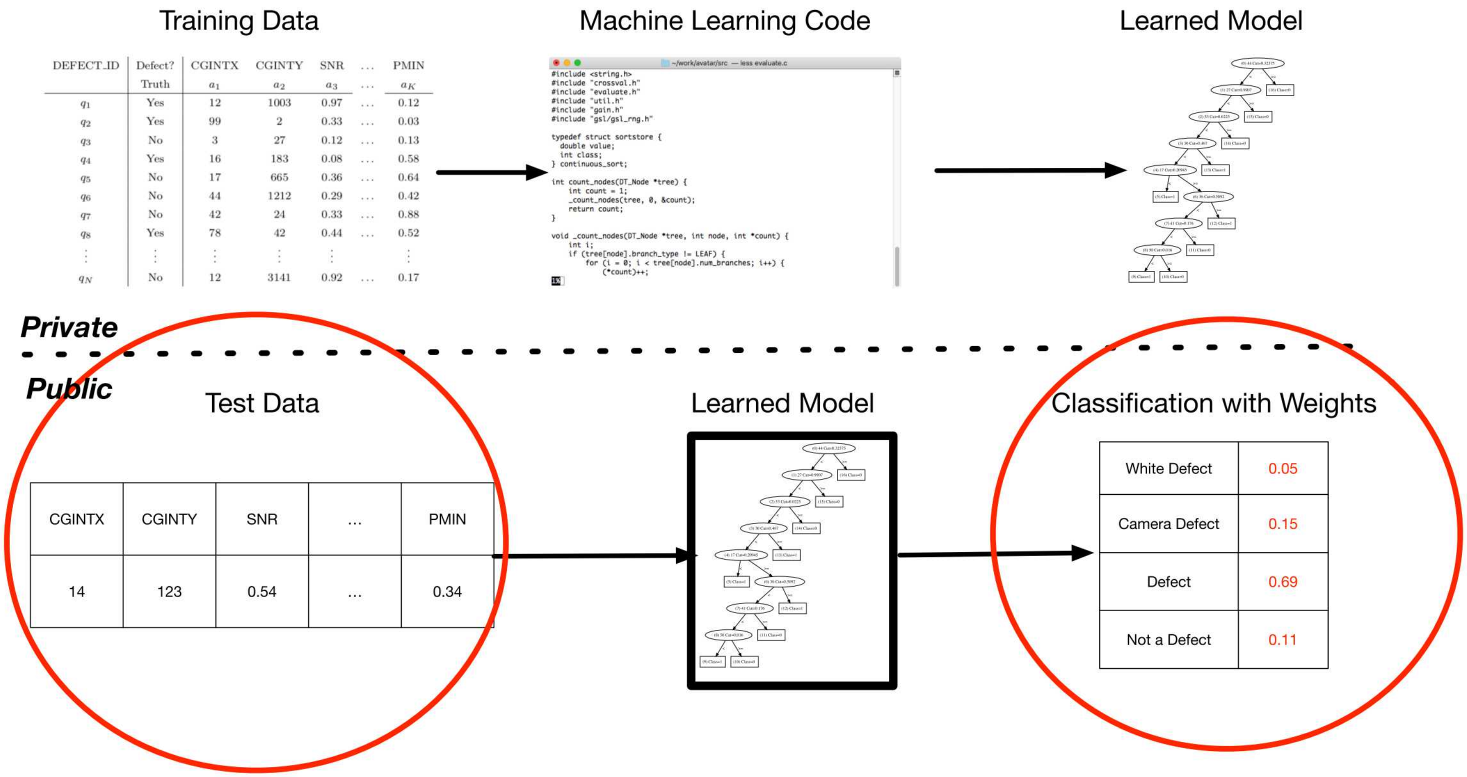
DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
—	—	12	1003	0.97	...	0.12
—	—	99	2	0.33	...	0.03
—	—	3	27	0.12	...	0.13
—	—	**?**	**183**	**0.08**	...	**0.58**
⋮	⋮	⋮	⋮	⋮		⋮
—	—	12	3141	0.92	...	0.17

Apply Maximum A Posteriori (MAP) analysis:

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
—	—	12	1003	0.97	...	0.12
—	—	99	2	0.33	...	0.03
—	—	3	27	0.12	...	0.13
—	—	**16**	**183**	**0.08**	...	**0.58**
⋮	⋮	⋮	⋮	⋮		⋮
—	—	12	3141	0.92	...	0.17

Attribute inference #2: recovering training data

Attack: exploit **black box** class label weights to recover **averaged raw data**



DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

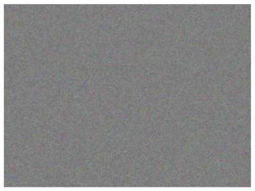
CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures[3]

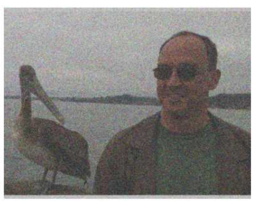
Recovery of a Representative Training Image

Biometric face recognition; attacker knows name, not face



Adam	Joe	Michelle	Dan	Jeremy	Laura	Philip	Katie	Steve	Dave
0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

Adam	Joe	Michelle	Dan	Jeremy	Laura	Philip	Katie	Steve	Dave
0.05	0.10	0.05	0.10	0.10	0.05	0.30	0.05	0.10	0.10



Adam	Joe	Michelle	Dan	Jeremy	Laura	Philip	Katie	Steve	Dave
0.00	0.10	0.00	0.10	0.10	0.00	0.60	0.00	0.10	0.10

Adam	Joe	Michelle	Dan	Jeremy	Laura	Philip	Katie	Steve	Dave
0.00	0.00	0.00	0.05	0.00	0.00	0.85	0.00	0.10	0.00

Attribute inference #3: recovering training data

Attack: use **white box** model knowledge recover **specific raw data**

Training Data

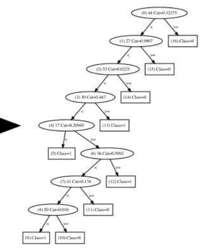
DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Machine Learning Code

~/work/avatar/src — less evaluate.c

```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
```

Learned Model



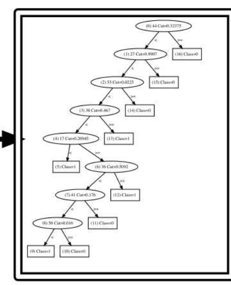
Private

Public

Test Data

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

Learned Model



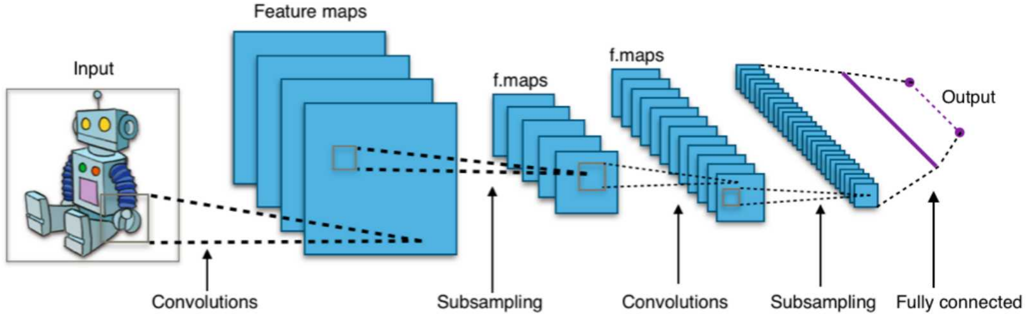
Classification with Weights

White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

Understanding Deep Image Representations by Inverting Them[4]

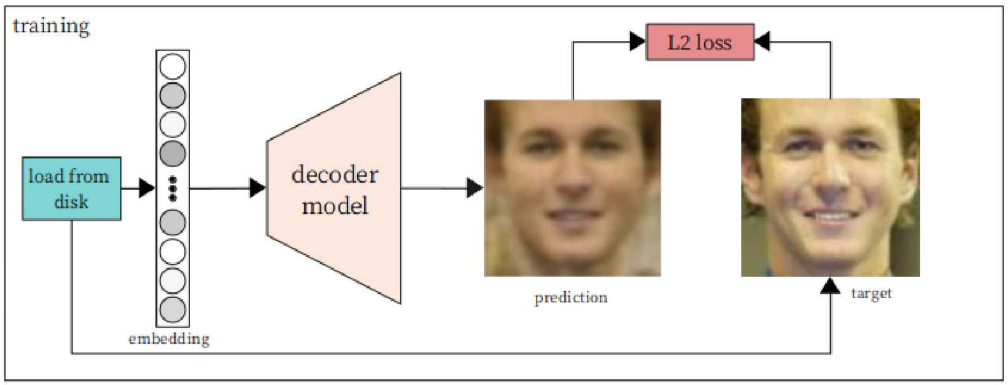
Recovery of an Exact Training Image

Attacker knows one level of a convolutional neural net or autoencoder:



(https://commons.wikimedia.org/wiki/File:Typical_cnn.png

Use gradient descent to find an input that would create that level:



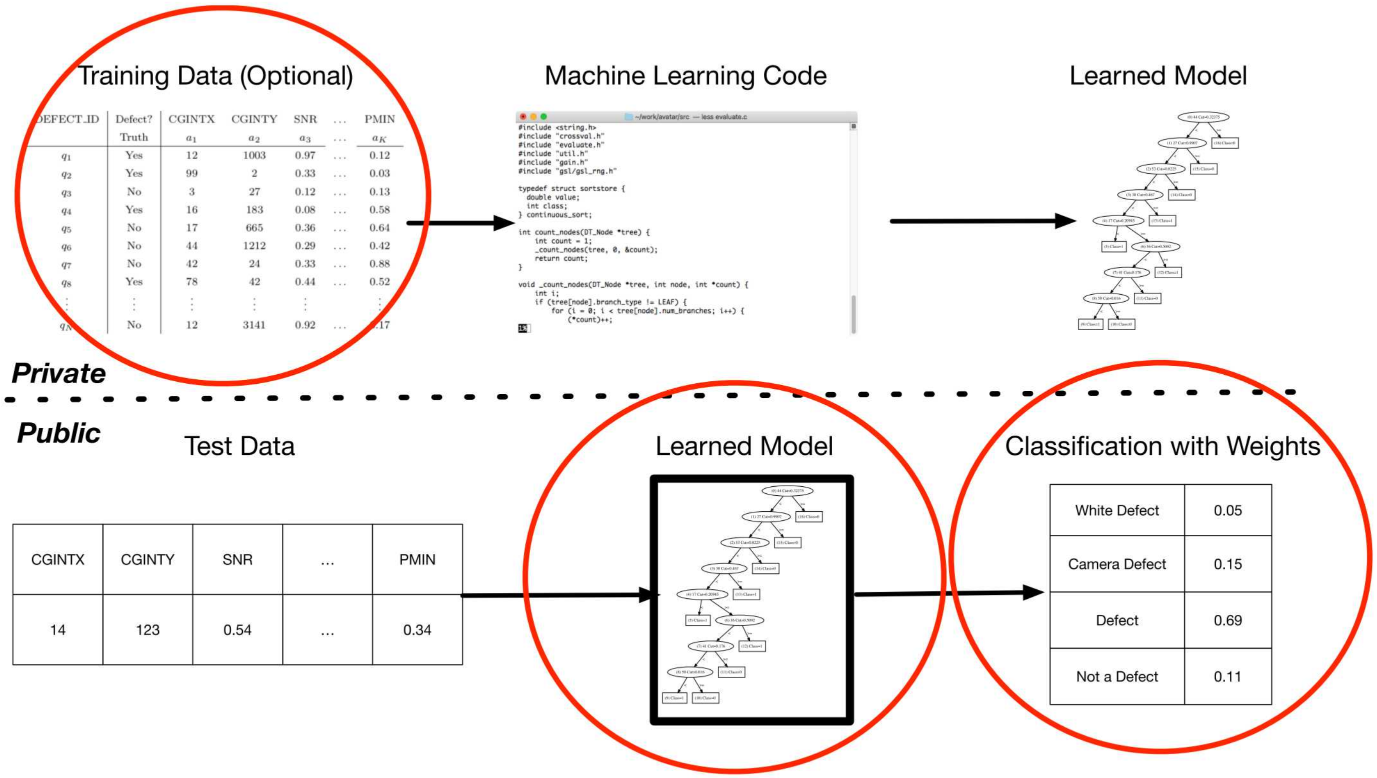
(https://blog.floydhub.com/inverting-facial-recognition-models)

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. **Membership inference: confirming training data**
  6. Model stealing: infer the model to better infer the training data
- What to do? A distressingly shallow set of ideas

Membership inference: confirming training data

Attack: build "shadow models" to *learn* to detect training data



Membership Inference Attacks Against Machine Learning Models[8],

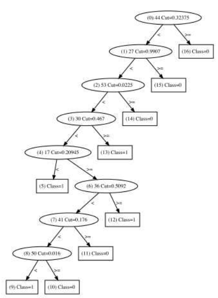
ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models[7]

Preview: attack involves *three* different ML models

Original Optics Training Data

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

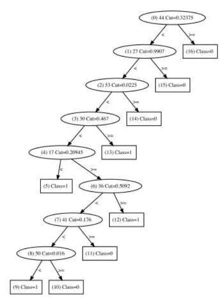
Original Model: Classify Defects



Surrogate Training Data

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

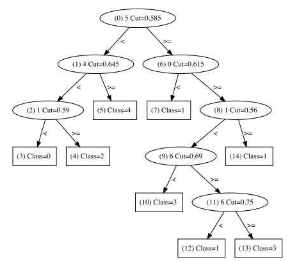
Surrogate Model



Membership Data (explanation coming …)

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	…	PMIN a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Membership Inference Model



Step 1: Adversary builds a surrogate model

Acquire training data, split in two, use both to build a surrogate model

Training Data: D_OTHER

DEFECT_ID	Defect?	CGINTX	CGINTY	SNR	…	PMIN
	Truth	a_1	a_2	a_3	…	a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Training Data: D_IN

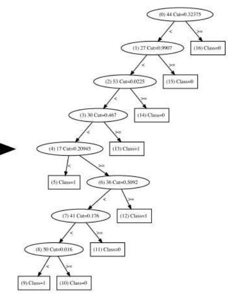
DEFECT_ID	Defect?	CGINTX	CGINTY	SNR	…	PMIN
	Truth	a_1	a_2	a_3	…	a_K
q_1	Yes	12	1003	0.97	…	0.12
q_2	Yes	99	2	0.33	…	0.03
q_3	No	3	27	0.12	…	0.13
q_4	Yes	16	183	0.08	…	0.58
q_5	No	17	665	0.36	…	0.64
q_6	No	44	1212	0.29	…	0.42
q_7	No	42	24	0.33	…	0.88
q_8	Yes	78	42	0.44	…	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	…	0.17

Machine Learning Code

~/work/avatar/src — less evaluate.c

```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
1%
```

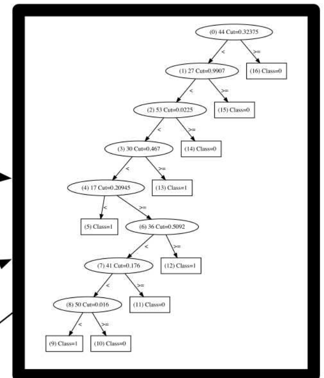
Surrogate Model



Step 2: Use surrogate model as a feature generator

Newly created “membership” data has bizarre features and “IN/OUT” labels

Surrogate Model



Test Data: D_IN

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

Test Data: D_OUT

CGINTX	CGINTY	SNR	...	PMIN
17	103	0.25	...	0.27

Normal Classification with Weights

White	Camera	Defect	Not
0.05	0.15	0.69	0.11

New Feature Data, with Labels

Truth	F1	F2	F3
IN	0.69	0.15	0.11

Normal Classification with Weights

White	Camera	Defect	Not
0.21	0.42	0.07	0.30

New Feature Data, with Labels

Truth	F1	F2	F3
OUT	0.42	0.30	0.21

Step 3: Generate *lots* of IN/OUT training data

Test Data: D_IN(1)

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

Test Data: D_IN(2)

CGINTX	CGINTY	SNR	...	PMIN
17	103	0.25	...	0.27

...

Test Data: D_IN(K)

CGINTX	CGINTY	SNR	...	PMIN
19	112	0.43	...	0.14

Test Data: D_OUT(1)

CGINTX	CGINTY	SNR	...	PMIN
11	132	0.31	...	0.29

Test Data: D_OUT(2)

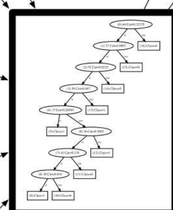
CGINTX	CGINTY	SNR	...	PMIN
16	121	0.45	...	0.92

...

Test Data: D_OUT(K)

CGINTX	CGINTY	SNR	...	PMIN
18	127	0.13	...	0.45

Surrogate Model



Classification with Weights

White	Camera	Defect	Not
0.05	0.15	0.69	0.11

Features with Labels

Truth	F1	F2	F3
IN	0.69	0.15	0.11

Classification with Weights

White	Camera	Defect	Not
0.65	0.20	0.07	0.08

Features with Labels

Truth	F1	F2	F3
IN	0.65	0.20	0.08

...

Classification with Weights

White	Camera	Defect	Not
0.35	0.65	0.00	0.00

Features with Labels

Truth	F1	F2	F3
IN	0.65	0.35	0.00

Classification with Weights

White	Camera	Defect	Not
0.21	0.42	0.07	0.30

Features with Labels

Truth	F1	F2	F3
OUT	0.42	0.30	0.21

Classification with Weights

White	Camera	Defect	Not
0.17	0.23	0.25	0.35

Features with Labels

Truth	F1	F2	F3
OUT	0.35	0.25	0.23

...

Classification with Weights

White	Camera	Defect	Not
0.10	0.20	0.30	0.40

Features with Labels

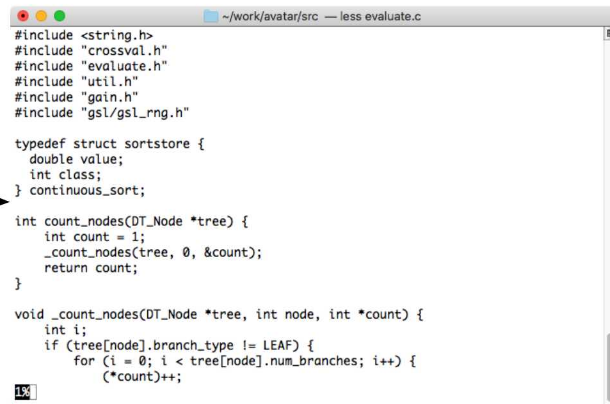
Truth	F1	F2	F3
OUT	0.40	0.30	0.20

Step 4: Use IN/OUT data to build *membership* model

Membership Features and Labels

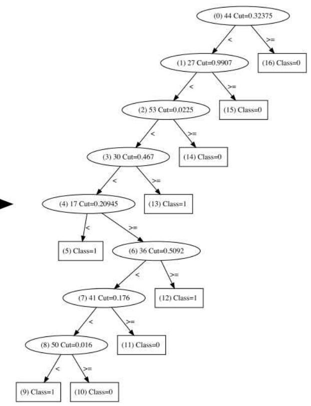
Truth	F1	F2	F3
IN	0.69	0.15	0.11
IN	0.65	0.20	0.08
IN	0.65	0.35	0.00
IN	...	...	...
OUT	0.42	0.30	0.21
OUT	0.35	0.25	0.23
OUT	0.40	0.30	0.20
OUT	...	...	...

Machine Learning Code



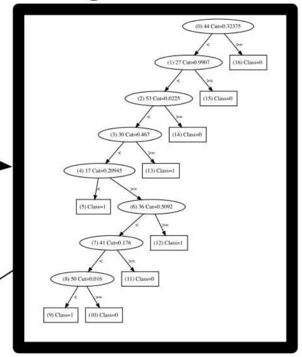
```
~/work/avatar/src — less evaluate.c
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
1%
```

Membership Inference Model



Step 5: Use membership model on original model

Original Model



Test Data: D_?

CGINTX	CGINTY	SNR	...	PMIN
16	141	0.62	...	0.43

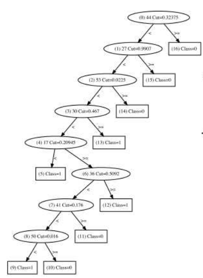
Normal Classification with Weights

White	Camera	Defect	Not
0.05	0.15	0.69	0.11

New Feature Data, Unlabeled

Truth	F1	F2	F3
?	0.69	0.15	0.11

Membership Inference Model



Membership Inference

IN	0.83
OUT	0.17

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. **Model stealing: infer the model to better infer the training data**
- What to do? A distressingly shallow set of ideas

Model stealing

Attack: probe the model with test data, deduce its structure

Training Data

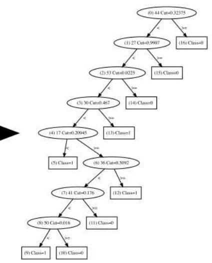
DEFECT_ID	Defect?	CGINTX	CGINTY	SNR	...	PMIN
	Truth	a_1	a_2	a_3	...	a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
⋮	⋮	⋮	⋮	⋮		⋮
q_N	No	12	3141	0.92	...	0.17

Machine Learning Code

~/work/avatar/src — less evaluate.c

```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
1%
```

Learned Model



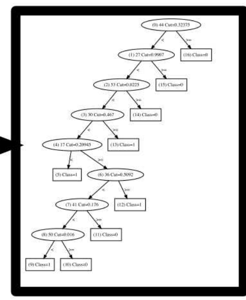
Private

Public

Test Data

CGINTX	CGINTY	SNR	...	PMIN
14	123	0.54	...	0.34

Learned Model



Classification with Weights

White Defect	0.05
Camera Defect	0.15
Defect	0.69
Not a Defect	0.11

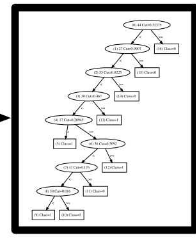
Replicating a black box model

Attack: use the model as a cheap labeler, build a new model

Lots of Unlabeled Test Data

14	123	0.54	...	0.34
23	197	0.17	...	0.54
81	101	0.16	...	0.76
51	314	0.27	...	0.29
63	163	0.72	...	0.17
12	145	0.31	...	0.91
31	415	0.92	...	0.6

Model to be Stolen



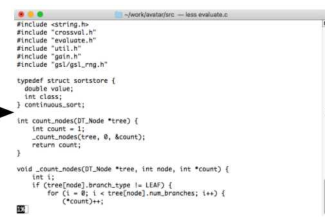
Newly Labeled Test Data

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17

Newly Labeled Test Data

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17

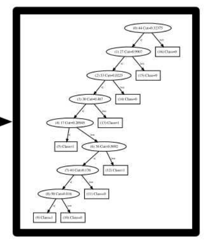
Machine Learning Code



~/work/avatar/src — less evaluate.c

```
#include <string.h>
#include "crossval.h"
#include "evaluate.h"
#include "util.h"
#include "gain.h"
#include "gsl/gsl_rng.h"

typedef struct sortstore {
  double value;
  int class;
} continuous_sort;

int count_nodes(DT_Node *tree) {
    int count = 1;
    _count_nodes(tree, 0, &count);
    return count;
}

void _count_nodes(DT_Node *tree, int node, int *count) {
    int i;
    if (tree[node].branch_type != LEAF) {
        for (i = 0; i < tree[node].num_branches; i++) {
            (*count)++;
```

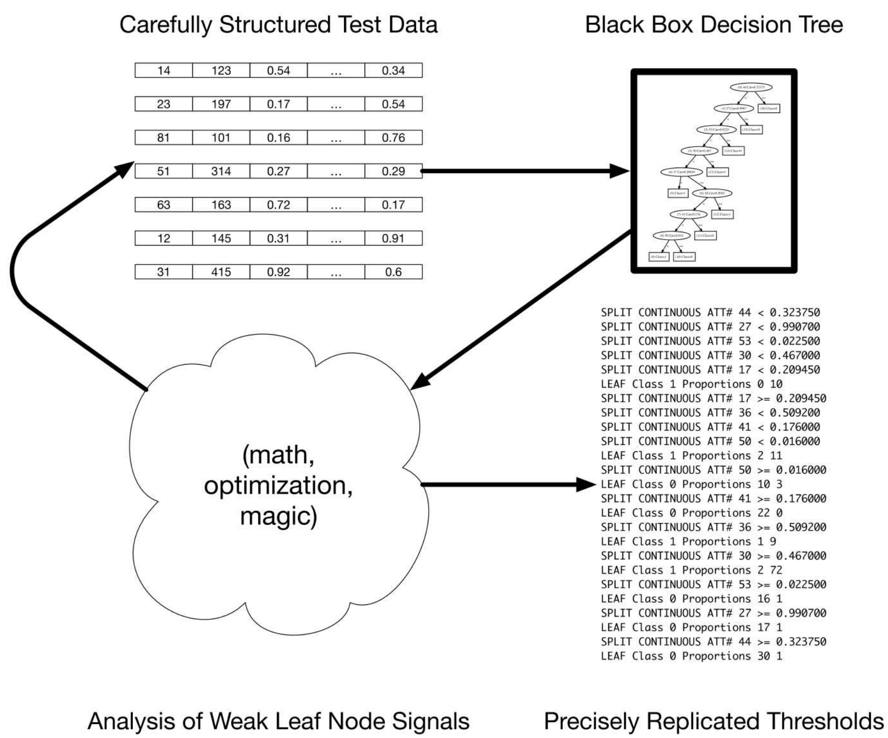
Replicated Model



Practical Black-Box Attacks Against Machine Learning[5], *Stealing Machine Learning Models via Prediction APIs*[10]

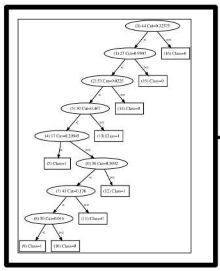
Precisely reproducing a model's parameters

Attack: use black box response discontinuities to detect thresholds



(Work in progress at Sandia)

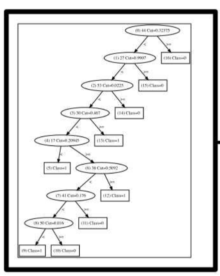
Therefore: can't block exfiltration with a black box

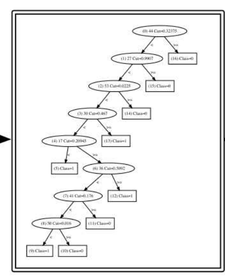


????, ????, ????, ????,
????, ????, ????, ????,
????, ????, ????, ????,
????, ????, ????, ????, ...

Concatenate,
Deserialize,
Decrypt,
Uncompress

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17





9833, 6299, 3495, 4946,
3470, 0158, 2537, 2076,
1277, 3644, 9284, 4085,
4201, 4159, 8444, 7234, ...

Concatenate,
Deserialize,
Decrypt,
Uncompress

DEFECT_ID	Defect? Truth	CGINTX a_1	CGINTY a_2	SNR a_3	...	PMIN a_K
q_1	Yes	12	1003	0.97	...	0.12
q_2	Yes	99	2	0.33	...	0.03
q_3	No	3	27	0.12	...	0.13
q_4	Yes	16	183	0.08	...	0.58
q_5	No	17	665	0.36	...	0.64
q_6	No	44	1212	0.29	...	0.42
q_7	No	42	24	0.33	...	0.88
q_8	Yes	78	42	0.44	...	0.52
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
q_N	No	12	3141	0.92	...	0.17

Outline

- Components of a machine learning system
- A variety of training data vulnerabilities
  1. Exfiltration via model parameters
  2. Exfiltration via model labels
  3. Exploit inadvertent memorization
  4. Attribute inference: recovering training data
  5. Membership inference: confirming training data
  6. Model stealing: infer the model to better infer the training data
- **What to do? A distressingly shallow set of ideas**

What to do? Some basic hygiene

- Know about differential privacy[1].
- Specifially, know about PATE[6] and DP-SGD[1].
- Be wary of code *you* didn't write.
- Don't use pre-trained NN architectures that *you* didn't train.
- Use only the parameters, and parameter precision, that you must.
  Don't use generic NN architectures as is, even untrained: adjust the architecture carefully.
- Expose no more model information than you have to.
  Think carefully about emitting anything more than a classification.
- Inspect the models you build. (Good luck; tools are scarce.)
- Maybe on the horizon: multi-party communication for information theoretic security, homomorphic encryption, garbled circuits …

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 308–318.

[2] Carlini, N., Liu, C., Kos, J., Erlingsson, U., and Song, D. The Secret Sharer: Measuring unintended neural network memorization and extracting secrets. Tech. Rep. arXiv:1802.08232, arXiv, 2018.

[3] Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), pp. 1322–1333.

[4] Mahendran, A., and Vedaldi, A. Understanding deep image representations by inverting them. *CoRR abs/1412.0035* (2014).

[5] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (New York, NY, USA, 2017), ASIA CCS '17, ACM, pp. 506–519.

[6] Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., and Erlingsson, U. Scalable private learning with PATE. In *International Conference on Learning Representations (ICLR)* (2018).

[7] Salem, A., Zhang, Y., Humbert, M., Fritz, M., and Backes, M. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. Tech. Rep. arXiv:1806.01246, arXiv, 2018.

[8] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy* (2017).

[9] Song, C., Ristenpart, T., and Shmatikov, V. Machine learning models that remember too much. In *ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 587–601.

[10] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction APIs. *25th USENIX Conference on Security Symposium* (2016), 601–618.