# Model Inversion Attacks.

- Develop an attack class that exploits confidence values revealed along with predictions.

To be explored in:

↳ decision trees in lifestyle survey

→ Neural network for facial recongnition.

## Background

- an ML model is a deterministic function

$f: R^d \rightarrow Y$ [from d features to a set of responses Y]

→ If Y is a finite set, [names of people in faceal recognition]

f is a classifier

→ I Y = R,

the f is a regression.

→ Many classifiers incorporate regression and selects the class estimating the maximum likelihood. → These esmitate outputs are confidence.

Consider formally, f is Composition of two function.

$f \rightarrow \tilde{f} + t$

$\rightarrow \tilde{f} : R^d \rightarrow [0,1]^m$, where $m$ is a parameter specifying the number of confidences.

[$m = |Y| - 1$, one less the number of class labels in eg.]

→ Second function is $t$.

$$t : [0,1]^m \rightarrow Y$$

~~When m=1~~ $$f(x) = t(\tilde{f}(x))$$

If classification call occurs, API returns both $f(x)$ and $\hat{f}(x)$.

ML APIs : Systems incorporates model $f$ via application programming interfaces (API).

↳ Typically in MLAAS such APIs are used.

$\boxed{\text{Threat Model}}$ :

→ adversary is assumed to have whatever information the API expresses.

White box setting - access to download a model $f$.

Black box setting - attacker has the ability to make prediction queries on feature vectors.

→ queries can be adaptive

⟹ queried features can be a function of previously retrieved predictions.

→ adversary has no access to training data [In some cases, data is public. However, here critical data is considered.]

<u>Fredrikson et al.'s attack</u>:

→ Consider a linear regression model $f$.

→ <u>predict</u>: a <u>real valued</u> suggested initial dose of a drug.

→ <u>feature vector</u>: patient's demographic information, medical history,

Sensitive attribute is genetic marker
↳ set as first feature $x_1$.

<u>Attacker access</u>: White box access to $f$.
Given auxiliary information <u>$\text{side}(x, y) = (x_2, \ldots x_t, y)$</u> for an instance $(x, y)$, attempts to infer genetic marker $x_1$.

Example steps of inversion attack:

→ Assume: aux gives empirically computed standard deviation $\sigma$ for a gaussian error model err and marginal priors

$$p = (p_1, \ldots p_t)$$

[Marginal priors ($p_i$) →

How to compute marginal priors?


· Partition real line in disjoint buckets
↳ ranges of values.

· each bucket $= v$

· each $p_i(v) = \frac{\text{No of times } x_i \text{ falls in } v}{\text{number of training vectors (ldb)}}$

→ Algorithm completes the target feature vector with possible values of $x_1$ and computes weighted probability estimate to get correct value.

→ Gaussian error model will penalize values of $x_1$ that force prediction to be far from label $y$.

Algorithm : Adversary $A^f(err, p_i, x_2, \ldots x_t, y)$

Input : Assume : • err : Gaussian error model with standard deviation $\sigma$.

• marginal priors $p = (p_1, \ldots p_t)$

• side info: $(x_2, \ldots x_t)$

adversary $A^f(err, p_i, x_2, \ldots x_t, y)$

1. for each value of $v$ of $x_1$ do
2. $x' = (v, x_2 \ldots x_t)$
3. $r_v = err(y, f(x')) . \Pi_i \, p_i(x_i)$
4. Return $\arg\max_v r_v$.

• Algo returns maximum a posteriori (MAP) estimate for $x_1$ given available information.

• minimizes adversary's misprediction rate.

$\arg\max_v r_v \longrightarrow$ Gets argument of the maximum

$\downarrow$

set of points for which $r_v$ attains maximum.

→ This algorithm produces least biased maximum a posteriori (MAP) estimate for $x_i$

↓

• MAP → • Maximum a posteriori probability (MAP) is an estimate of an unknown quantity.
• equals the mode of posterior distribution.

Posterior probability – The updated probability (prior probability) of an event taking into consideration some new event.

Probability Density function – (density of a continuous random variable)

→ whose value at any given sample in the sample space can be interpreted as the relative likelihood of random variable equal to the sample.

$\boxed{\text{MAP gives the value that maximises the posterior probability.}}$

MAP Inverters for TREES.

- Decision tree recursively partitions the feature space into disjoint regions $R_1, \dots R_m$.
  - for an instance $(x, y) \longrightarrow$ find region of $x$ $\longrightarrow$ return most likely value of $y$.

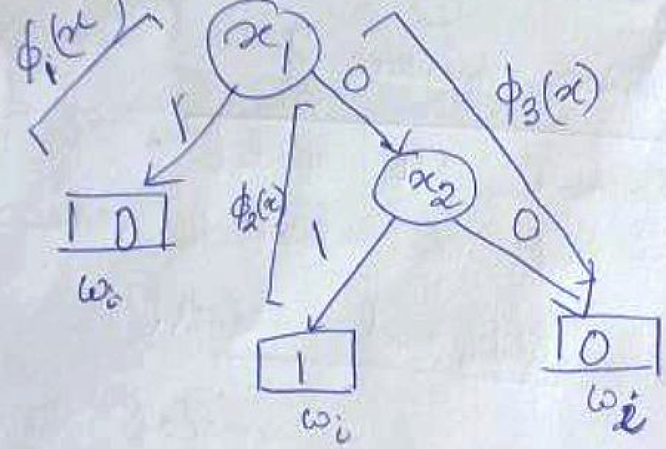
Eg: Let $y = \neg x_1 \wedge x_2$.

$\therefore$ If $x_1 = 1 \longrightarrow y = 0$

else if $x_1 = 0$,

$\longrightarrow$ if $x_2 = 1$, $y = 1$

else $x_2 = 0$, $y = 0$.




$\phi_1(x) = x_1$

$\phi_2(x) = (1 - x_1).x_2$

$\phi_3(x) = (1 - x_1)(1 - x_2)$

Trees can be mathematically characterized as follows;

$$f(x) = \sum_{i=1}^{m} \omega_i \, \phi_i(x), \text{ where } \phi_i(x) \in \{0, 1\}$$

[ $\phi_i \longrightarrow$ indicator of region $R_i$ and $\omega_i$ ]

## How to return confidence measures?

→ Classification measures by getting $\omega_i$

→ $\omega_i$ is set to a vector corresponding to distribution of class labels as observed in the training set. [In complex scenario, $\omega_i$ is set from training data]

→ Say you have added 89 samples for $x_1 = 1$ and $x_2 = 1, (11)$ or change the $11 \to 10$, get confidence measures for classification.

## What is confidence measure?

→ Confidence represents ML model's estimated probability that the extracted value is correct given document/inputs and labels.

Confidence = No of a correct prediction on a dataset.

## Classification results:

$$f(x) = \arg\max_j \left( \sum_{i=1}^{m} \omega_i[j]\, \phi_i(x) \right)$$

Confidences: $$\hat{f}(x) = \left[ \frac{\omega_{i^*}[1]}{\sum_i \omega_i[i]} \cdots \frac{\omega_{i^*}[|Y|]}{\sum_i \omega_{i^*}[i]} \right]$$

[$i^*$ takes ~~tat~~ values where $\phi_{i^*}(x) = 1$.]

## Decision tree APIs

- Several web APIs expose training and quering routines for decision trees to end users
  [eg. BigML, Microsoft machine learning etc]
- Users can upload their datasets to these services to train a decision tree

**Big ML**: publish trees in different modes:

- Black box
- White box

⇒ For both setting users has marginal priors for each feature of training set.
↳ user has Confusion matrix C
$C_{ij}$ = gives number of training instances for which ~~$C_{ij}$ gives number~~ of t
$y = i$ and predicted model label = j.

- - - - -

→ **White box setting**:
Attacker has to: $n_i$ = count of number of training instances that match $\phi_i$ path,
⇒ Computes the confidence.

The inversion problem :

- Fix a tree $f(x) = \sum_{i=1}^{m} w_i \phi_i(x)$

. Let $(x, y)$ be target instance that will be from target data db or not .

- For simplicity, there is one sensitive feature
  → make target feature set $T = \{1\}$;
  → [extending our attacks to one more one feature is straightforward]

  side info : $side_l(x_l, \dots x_d]$
  $[l \geq 2]$

  $K = \{l, \dots d\}$ be set of known feature indices]

  $[x_K$ represent $(d-1)$ dimensional vector]

Black - box ML :

→ decision tree produces discrete outputs
→ Use Confusion matrix $C$
→ define $err(y, y') \propto P_r[f(x) = y' | y$ is true label]

White-box ML :

- 1) Attacker knows each $\phi_i$ and $n_i$ = correspond to $\phi_i$.

2) from this $N = \sum_{i=1}^{m} n_i$ [Total number of samples in training set]

3) Known value $x_K$ induce a set of paths

$$S = \{s_i\}_{1 \leq i \leq m};$$

$$s = \{(\phi_i, n_i) \mid \exists x' \in R^d, x'_K = x_K \wedge \phi(x')\}$$

- Each path corresponds to a basis function $\phi_i$.
- Let $p_i = \frac{n_i}{N}$

[$p_i$ gives information about the joint distribution on features used to build training set.]

$P_r[S_i]$ = probability of drawing a row from joint prior that traverses $s_i$.

$p_i$ = emperical estimate for this quantity, derived from draws to produce training set

Basis functions partition feature space

$\Longrightarrow$ $x$ traverses exactly one paths in $S$

- Indicator function $\exists x' \in R^d, x_1' = v \wedge \phi_i(x')$

Estimator (E) characterizes the probability that $\underline{x_1 = v}$, given $x$ traverses one of paths $s_1, \ldots s_m$, and $x_K = v_K$:
[known values]

$$\Pr\left[x_1 = v \mid (s_1 \vee \ldots \vee s_m) \wedge x_K = v_K\right]$$

$$\propto \sum_{i=1}^{m} \frac{p_i\, \phi_i(v) \cdot \Pr[X_K = V_K] \cdot \Pr[X_1 = v]}{\sum_{j=1}^{m} p_j\, \phi_j(v)}$$

$$\propto \frac{1}{\sum_{j=1}^{m} p_j\, \phi_j(v)} \sum_{1 \le i \le m} p_i\, \phi_i(v) \cdot \Pr[X_1 = v] \quad \ldots (1)$$

= White-box with counts (WBWC) estimator

$\rightarrow$ adversary outputs a value $v$ that maximizes (1) as a guess for $x_1$.