

Security and Fairness of Deep Learning

Privacy Attacks on Deep Networks II

Klas Leino

CMU

Spring 2019

Overview

- Review Membership Inference
- Understanding Overfitting
- Bayes-optimal Membership Inference
- Extending to Deep Models
- Homework 4

Recall

- Membership Inference
- Black-box Attacks
 - Naïve
 - Shadow Models

Membership Inference

- Universe, U , of points, (x, y) , of features (x) and labels ($y \in [C]$), distributed according to distribution, θ^* .
- Training set, S , of N points drawn from θ^* .
 - (x, y) drawn from the training set: (x, y) chosen uniformly at random from the elements of S .
 - (x, y) drawn from the general population (or test set): (x, y) drawn directly from θ^* .
- Target model, \hat{g} , learned by algorithm, \mathcal{A} , which takes a training set and produces a model.

Membership Inference

- Draw a point, (x, y) , with $\frac{1}{2}$ probability from the training set, and with $\frac{1}{2}$ probability from the general population.
- Adversary predicts 1 ((x, y) was a training point) or 0 ((x, y) was not a training point).
- Advantage: *true positive rate* – *false positive rate*, or equivalently, $2(\text{accuracy} - \frac{1}{2})$.

Membership Inference: Threat Models

- Black-box: adversary has black-box access to \hat{g} , i.e., given features, x , the adversary can obtain $\hat{y} = \hat{g}(x)$.
 - Adversary doesn't have access to weights or internal activations.
 - Typically, we do assume adversary has access to \mathcal{A} .
 - We also assume adversary has access to some set of points, \tilde{S} , also drawn from θ^* (but disjoint from S).
- White-box: adversary additionally has access to the internals of \hat{g} , e.g., weights and biases.

Membership Inference: Threat Models

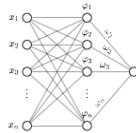
- Black-box: adversary has black-box access to \hat{g} , i.e., given features, x , the adversary can obtain $\hat{y} = \hat{g}(x)$.
 - Adversary doesn't have access to weights or internal activations.
 - Typically, we do assume adversary has access to \mathcal{A} .
 - We also assume adversary has access to some set of points, \tilde{S} , also drawn from θ^* (but disjoint from S).
- White-box: adversary additionally has access to the internals of \hat{g} , e.g., weights and biases.

Example

Non-sensitive information of patients with sensitive condition X



Model predicting non-sensitive target, e.g., height of patient



Membership inference



Bob was in training set (has condition X)

Adversary learns Bob has condition X

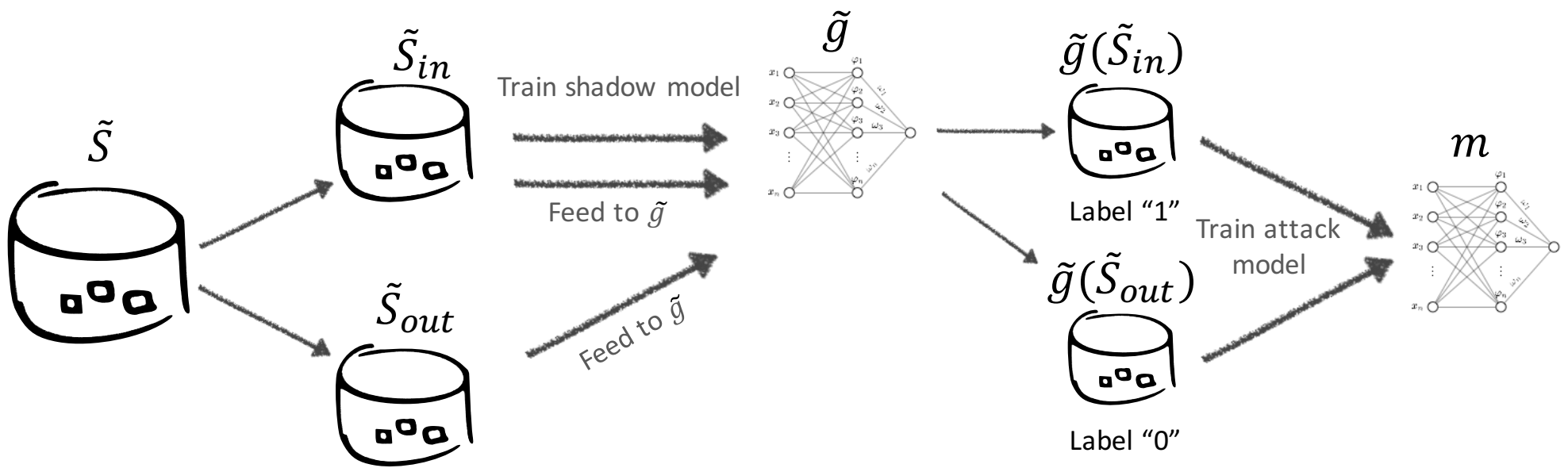
Naïve Attack

- If $\hat{y} = y$, predict 1, else predict 0. In other words we assume correctly classified points are training members, and incorrectly classified points are not.
- Advantage: *train accuracy* – *test accuracy*.
- Surprisingly, this attack is quite effective, i.e., compares similarly to more sophisticated attacks.

What's Wrong with the Naïve Attack?

- High false positive rate (bad precision)
- Doesn't quantify confidence in inference

Shadow Model Approach [1]



(do this for each class)

Overview

- Review Membership Inference
- **Understanding Overfitting**
- Bayes-optimal Membership Inference
- Extending to Deep Models
- Homework 4

Can We Extend Shadow Models to White-Box Setting?

- Use internal outputs (activations) at each layer?
 - Not clear this would generalize because a shadow model may learn an entirely different internal representation
 - Was shown not to perform better than black-box approach
- Might want a more fundamental understanding of overfitting...

How Does Overfitting Manifest Itself?

- Idiosyncratic use of features
 - Some features happen to be useful for classification on training data but not on general distribution – these are evidence of overfitting
- We would like to use this intuition when designing our attack

Example

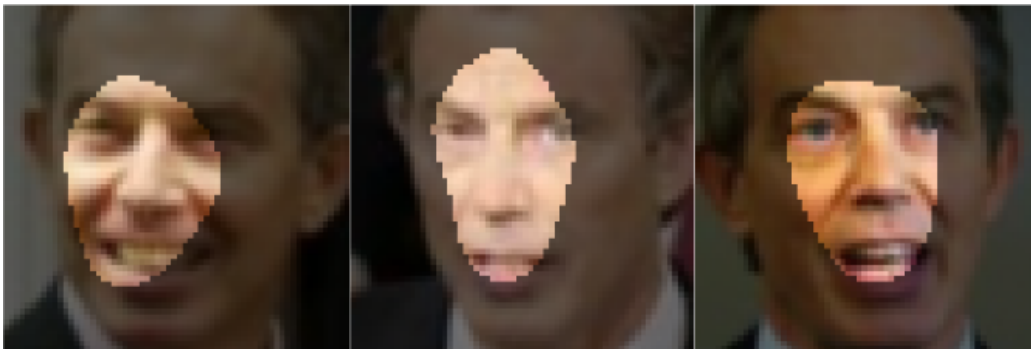
notice the distinctive pink background



Sample of LFW training instances



Explanation [2] on training instance of Tony Blair with distinctive pink background. The model uses the background to classify the instance as Tony Blair.



Typical explanations on test instances of Tony Blair

Overview

- Review Membership Inference
- Understanding Overfitting
- **Bayes-optimal Membership Inference**
- Extending to Deep Models
- Homework 4

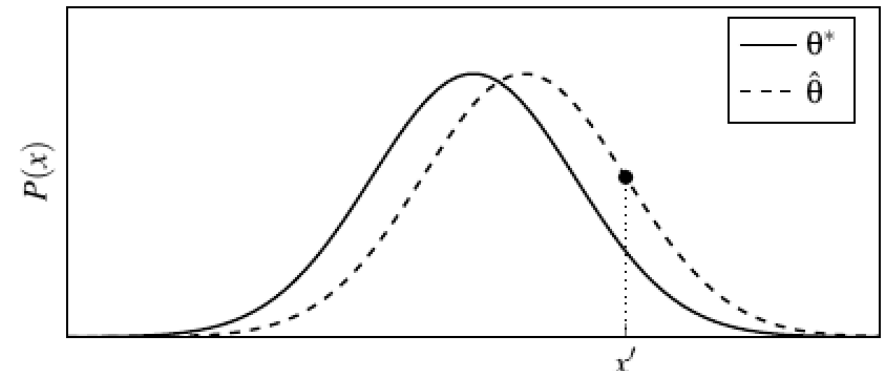
Key Idea

Membership information is leaked through the target model's idiosyncratic use of features. Features that are distributed differently in the training data from how they are distributed in the general population provide evidence for or against membership.

Next we would like to formalize this intuition...

Example

- Suppose we have two distributions, θ^* and $\hat{\theta}$.
- Graph shows $\Pr[x|\theta]$, i.e., the probability of getting value x from either θ^* or $\hat{\theta}$.
- When $\Pr[x|\theta^*] < \Pr[x|\hat{\theta}]$, we are more likely to have drawn x from $\hat{\theta}$ than from θ^* .



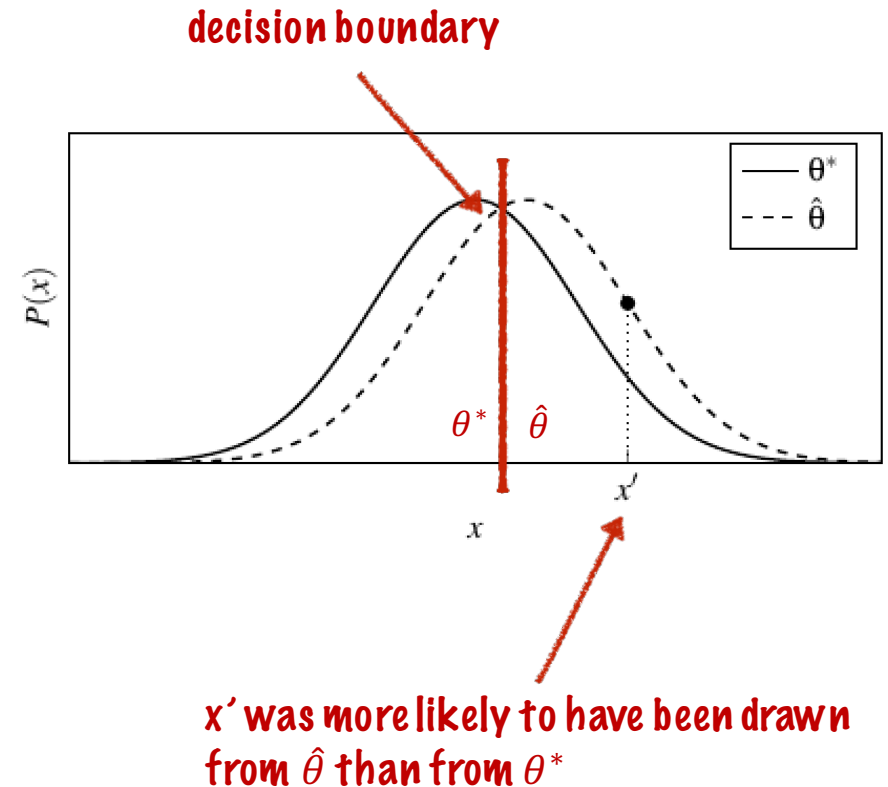
x' was more likely to have been drawn from $\hat{\theta}$ than from θ^*

Bayes-optimal Membership Inference Idea

- Universe, U , of points, (x, y) , of features (x) and labels ($y \in [C]$), distributed according to distribution, θ^* .
- Training set, S , of N points drawn from θ^* .
 - Assume $S \sim \hat{\theta}$, i.e., the training set is distributed according to some distribution, $\hat{\theta}$.
 - In expectation, $\hat{\theta} = \theta^*$, but in general the distributions may be different. For now, we take $\hat{\theta}$ to be the *empirical* distribution of S .
- Idea: we want to make a model that predicts whether (x, y) was more likely to have been drawn from $\hat{\theta}$ than from θ^* .

Example

- Suppose we have two distributions, θ^* and $\hat{\theta}$.
- Graph shows $\Pr[x|\theta]$, i.e., the probability of getting value x from either θ^* or $\hat{\theta}$.
- When $\Pr[x|\theta^*] < \Pr[x|\hat{\theta}]$, we are more likely to have drawn x from $\hat{\theta}$ than from θ^* .



Some Simplifications

- Assume data follows Gaussian Naïve Bayes assumption:
 - $\theta^* = \mathcal{N}(x|\mu_y^*, \Sigma^*)$ where Σ^* is a diagonal matrix, which we will write as a vector, σ^{*2} .
 - Same goes for $\hat{\theta}$
- Assume target model, \hat{g} , is linear softmax model, i.e.,
$$\hat{g} = \text{softmax}(\hat{W}x + \hat{b}).$$
- We will eventually relax these assumptions.

\mathcal{N} is the Gaussian function, i.e., $\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$.

Derivation

- Let X and Y be random variables that represent randomly drawn features and labels from either the training set (with 50% probability) or the general population (with 50% probability).
- Let T be the event that $(X, Y) \in S$.
- Want to discover $m^y(x)$, an attack model that predicts the probability that $(x, y) \in S$. I.e.,


$$m^y = \Pr[T \mid X = x, Y = y]$$

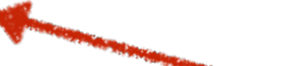
Derivation

$$m^y(x) = \Pr[T \mid X = x, Y = y]$$

 we can apply Bayes' rule to this

$$= \frac{\Pr[X = x \mid T, Y = y] \Pr[T]}{\Pr[X = x \mid Y = y]}$$

 1/2

 we can apply the rule of total probability here, partitioning on whether or not (x,y) was in the training set

$$= \frac{\Pr[X = x \mid T, Y = y] \frac{1}{2}}{\frac{1}{2} (\Pr[X = x \mid T, Y = y] + \Pr[X = x \mid \neg T, Y = y])}$$

Derivation

we can divide the top and bottom by the numerator, so the numerator becomes 1

$$= \frac{\Pr[X = x | T, Y = y] \cancel{\frac{1}{2}}}{\cancel{\frac{1}{2}} (\Pr[X = x | T, Y = y] + \Pr[X = x | \neg T, Y = y])}$$

this also becomes 1

$$= \frac{1}{1 + \frac{\Pr[X=x|\neg T, Y=y]}{\Pr[X=x|T, Y=y]}}$$

this is a fraction of two probabilities, which are both in the range $[0,1]$, thus it is non-negative so we can use the identity $v = \exp(\log(v))$ for $v > 0$

$$= \frac{1}{1 + \exp\left(\log \frac{\Pr[X=x|\neg T, Y=y]}{\Pr[X=x|T, Y=y]}\right)} = \sigma\left(\log \frac{\Pr[X = x | T, Y = y]}{\Pr[X = x | \neg T, Y = y]}\right)$$

we can multiply the log by -1 and flip the fraction, to get an equivalent expression in the form of a sigmoid function.

sigmoid function

Derivation

$$= \Delta \left(\log \frac{\Pr[X = x \mid T, Y = y]}{\Pr[X = x \mid \neg T, Y = y]} \right)$$

using the Naïve Bayes assumption, we can write each of these probabilities as a product of the probabilities of observing each individual feature

$$= \Delta \left(\log \prod_j \frac{\mathcal{N}(x_j \mid \hat{\mu}_{yj}, \hat{\sigma}_j^2)}{\mathcal{N}(x_j \mid \mu_{yj}^*, \sigma_j^{*2})} \right)$$

then we can use the fact that the log of a product is the sum of the logs

$$= \Delta \left(\sum_j \frac{(x_j - \mu_{yj}^*)^2}{2\sigma_j^{*2}} - \frac{(x_j - \hat{\mu}_{yj})^2}{2\hat{\sigma}_j^2} + \log \left(\frac{\sigma_j^*}{\hat{\sigma}_j} \right) \right)$$

here we have also applied the log to the Gaussian function

Derivation

Notice that if we expand the polynomial, we get a term (call it v_j^y) multiplied by x_j^2 , plus a term (call it w_j^y) multiplied by x_j , plus a constant (call it b_j^y). Thus we can think of this sum as a dot product.

$$= \Delta \left(\sum_j \frac{(x_j - \mu_{yj}^*)^2}{2\sigma_j^{*2}} - \frac{(x_j - \hat{\mu}_{yj})^2}{2\hat{\sigma}_j^2} + \log \left(\frac{\sigma_j^*}{\hat{\sigma}_j} \right) \right)$$

$$= \Delta (v^y T x^2 + w^y T x + b^y)$$

this is a quadratic model
(quadratic decision boundary)

where

$$v_j^y = \frac{1}{2\sigma_j^{*2}} - \frac{1}{2\hat{\sigma}_j^2}, \quad w_j^y = \frac{\hat{\mu}_{yj}}{\hat{\sigma}_j^2} - \frac{\mu_{yj}^*}{\sigma_j^{*2}},$$

$$b^y = \sum_j \left(\frac{\mu_{yj}^{*2}}{2\sigma_j^{*2}} - \frac{\hat{\mu}_{yj}^2}{2\hat{\sigma}_j^2} \right) + \log \left(\frac{\sigma_j^*}{\hat{\sigma}_j} \right)$$

One More Simplification

- We notice that in the previous equation gives us a quadratic decision boundary
- In expectation, $\hat{\sigma} = \sigma^*$. If we assume $\hat{\sigma} = \sigma^* = \sigma$, then v^y becomes zero, and we get a linear model, with

$$w^y = \frac{\hat{\mu}_y - \mu_y^*}{\sigma^2} \quad b^y = \sum_j \frac{\mu_{yj}^{*2} - \hat{\mu}_{yj}^2}{2\sigma_j^2}$$

Something's Missing

- This analysis tells us that the optimal membership predictor in this case is a linear model with weights $w^y = \frac{\hat{\mu}_y - \mu_y^*}{\sigma^2}$, and intercepts,

$b^y = \sum_j \frac{\mu_y^{*2} - \hat{\mu}_y^2}{2\sigma^2}$. However, we still have a problem...

- We don't know the actual parameters of $\hat{\theta}$!
- Data may not be well-modeled by a Gaussian distribution anyway.
- Idea: we would like to express w^y and b^y in a way that doesn't depend on the parameters of the distribution.

Getting Around Not Knowing $\hat{\theta}$ and θ^*

- We will make use of the fact that softmax regression on Gaussian Naïve Bayes data converges to the Bayes-optimal classifier, which has weights and intercepts, $W = \frac{\mu}{\sigma^2}$ and $b = \sum_j \frac{-\mu_j^2}{2\sigma^2}$.
- Thus, if \hat{g} has converged, $\hat{W}_{:,y} = \frac{\hat{\mu}_y}{\sigma^2}$, and $\hat{b}_y = \sum_j \frac{-\hat{\mu}_{yj}^2}{2\sigma^2}$.
- We will also use \tilde{S} as a *proxy* for θ^* , i.e., we assume that $\tilde{\theta} \approx \theta^*$.
- Similarly, if we train a *proxy model*, \tilde{g} , to convergence on \tilde{S} , $\tilde{W}_{:,y} \approx \frac{\mu_y^*}{\sigma^2}$, and $\tilde{b}_y \approx \sum_j \frac{-\mu_{yj}^{*2}}{2\sigma^2}$.
- Finally, we conclude that $w^y = \hat{W}_{:,y} - \tilde{W}_{:,y}$ and $b^y = \hat{b}_y - \tilde{b}_y$.

doesn't depend on the distribution parameters!

Linear Bayesian Membership Inference

Algorithm 1: The Linear bayes MI Attack

```
def createAttackModel ( $\hat{g}$ ,  $\tilde{S}$ ):  
     $\tilde{g} \leftarrow \text{trainProxy}(\tilde{S})$   
     $w^y \leftarrow \hat{g}.W_{:y} - \tilde{g}.W_{:y} \quad \forall y \in [C]$   
     $b^y \leftarrow \hat{g}.b_{:y} - \tilde{g}.b_{:y} \quad \forall y \in [C]$   
    return  $\lambda(x, y) : \delta(w^{yT}x + b^y)$   
  
def predictMembership ( $m$ ,  $x$ ,  $y$ ):  
    return 1 if  $m^y(x) > \frac{1}{2}$  else 0
```

Optimizations

- We can train multiple proxy models on various subsets of the proxy data (\tilde{S}) and average their weights to get a better approximation of the true distribution.

Overview

- Review Membership Inference
- Understanding Overfitting
- Bayes-optimal Membership Inference
- **Extending to Deep Models**
- Homework 4

How Would We Apply this to an Arbitrary Layer of a Network?

- Recall internal influence [2].
 - One axiom of Internal Influence was *linear agreement*: for linear models, the influence of a feature is its weight in the linear model.
 - Idea: use influence instead of weight for attacking internal layers.

Local Linear Approximations

- Intuitively the gradient gives a local linear approximation of a function.
- For *slice*, $\langle g, h \rangle$, of a deep network, f , we can locally approximate g using internal influence.
- Recall, internal influence is given by

$$\chi_j(g \circ h, P') = \int_{z \in h(\mathcal{X})} \left. \frac{\partial g}{\partial z_j} \right|_z P'(z) dz$$

**distribution of interest
over internal points**



Recall, a slice, $\langle g, h \rangle$, of a deep network, f , satisfies $f = g \circ h$.

Local Linear Approximations

- When we set the distribution of interest to $P_{z^0}^z$, a uniform distribution over the linear interpolation from z^0 to z (essentially, this recovers Aumann-Shapley for point, z , and baseline z^0), internal influence has a property called *efficiency* (sometimes called *completeness*).
Completeness states

$$\sum_j \chi_j(g \circ h, P') (z_j - z_j^0) = g(z) - g(z^0)$$

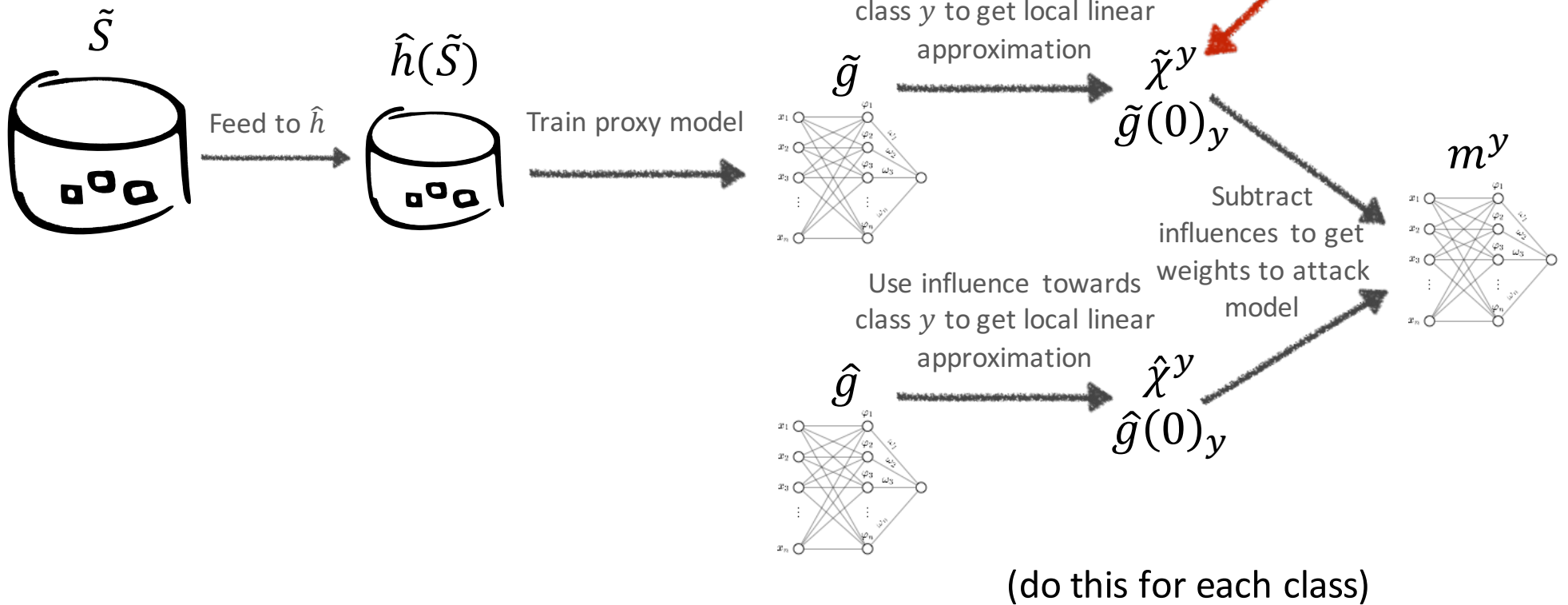
- Thus, when we set the baseline to 0, $g(z)$ is approximated by $\bar{g}(z) = \bar{W}z + \bar{b}$ where $\bar{W} = \chi(g \circ h, P_0^z)$ and $\bar{b} = g(0)$.

Deep Bayesian Membership Inference

Algorithm 3: The Deep bayes MI Attack

```
def createAttackModel ( $\hat{g} \circ \hat{h}, \tilde{S}$ ):  
     $\tilde{S}' \leftarrow [(\hat{h}(x), y) \text{ for } (x, y) \in \tilde{S}]$   
     $\tilde{g} \leftarrow \text{trainProxy}(\tilde{S}')$   
     $w^y \leftarrow \lambda(z) : \chi(\hat{g} \circ \hat{h}, P_0^z) - \chi(\tilde{g} \circ \hat{h}, P_0^z) \quad \forall y \in [C]$   
     $b^y \leftarrow \hat{g}(0)_y - \tilde{g}(0)_y \quad \forall y \in [C]$   
    return  $\lambda(x, y) : \delta \left( w^y (\hat{h}(x))^T \hat{h}(x) + b^y \right)$   
  
def predictMembership ( $m, x, y$ ):  
    return 1 if  $m^y(x) > \frac{1}{2}$  else 0
```

Illustration



Combining Attacks on Multiple Layers

- AND or OR of predictions at each layer.
- Majority vote on predictions at each layer.
- Learn a network to combine the outputs.
- Work out a creative solution for the homework!

Overview

- Review Membership Inference
- Understanding Overfitting
- Bayes-optimal Membership Inference
- Extending to Deep Models
- **Homework 4**

Homework 4 (part II)

- You will implement shadow models and the deep Bayesian attack for arbitrary layers.
 - Influence measure will be provided
- For final part (extra credit) combine attacks on multiple layers to attack LeNet model trained on LFW.
- Main function tests accuracy of the attack.

Shadow Attack Starter Code

Implement:

```
def build_attack_model(  
    target_model,  
    shadow_data,  
    shadow_labels,  
    num_shadow_models=10)  
def evaluate_membership(attack_model, y_pred, y)
```

You should make use of:

```
split = DataSplit(labels, seed)  
split.in_idx, split.out_idx
```

Deep Bayesian Attack Starter Code

Implement:

```
def build_attack_model(  
    target_model,  
    shadow_data,  
    shadow_labels,  
    attack_layer,  
    num_shadow_models=10)  
def evaluate_membership(attack_model, y_pred, y)
```

Deep Bayesian Attack Starter Code

You should make use of:

```
split = DataSplit(labels, seed)
split.in_idx # Use in_idx for proxy models!
```

```
attack_model = AttackModelInfo(W, b)
```

```
infl_measure = InfluenceMeasure(model, c, layer)
influence = infl_measure(Z)
```

```
g = TopOfModel(model, layer)
y_hat = g(Z)
```

References

- [1] Shokri et al. *Membership Inference Attacks on Deep Learning Models*. 2016
- [2] Leino et al. *Influence-directed Explanations for Deep Convolutional Networks*. 2018