

# Understanding Model Predictions via Influence Functions

## Statistical Machine Final Project

Ze Yang   Zhengyang Qi   Yundong Liu   Yuze Liu

Carnegie Mellon University

2018.05.05

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Outline

## Backgrounds

### Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Introduction

- ▶ In this project, we reproduced the *Best Paper* awardee of ICML 2017: *Understanding Black-box Predictions via Influence Functions*, by Koh & Liang.
- ▶ Our main topic is **influence function**, a technique to trace a models prediction through the learning algorithm and back to its training data.
- ▶ We carried out efficient implementations with Tensorflow.
- ▶ We demonstrated its performance & applications on real-world datasets.

# Outline

## Backgrounds

Introduction

### What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

## What is Influence Function: Some Definitions

- ▶ **Statistical Functional:**  $T(F_Z)$ , a functional maps from the space of population CDFs to a field (such as  $\mathbb{R}$ ).
- ▶ **Risk Optimizer:** Ideally, minimizes EPE: A statistical functional.

$$\theta(\hat{F}_Z) = \operatorname{argmin}_{\theta \in \Theta} \int \mathcal{L}(z, \theta) d\hat{F}_Z$$

- ▶ **Empirical Risk Optimizer:** In reality, minimize empirical risk, a *plug-in* estimator to the risk optimizer.

$$\theta(\hat{F}_Z) = \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(z_i, \theta)$$

## What is Influence Function: Gâteaux Derivative

**Gâteaux Derivative** generalizes the idea of **directional derivatives** - the inputs are CDFs!

$$\begin{aligned}\mathcal{I}_{\theta}(G) &:= \nabla_G \theta(F_Z) \Big|_{F_Z=F_Z^*} \\ &= \lim_{h \rightarrow 0} \frac{\theta((1-h)F_Z^* + hG) - \theta(F_Z^*)}{h} \quad (1)\end{aligned}$$

Meaning: the sensitivity of the statistical functional  $\theta(F_Z)$  with respect to “mixing the CDF with a little bit  $G$ ”.

## Empirical Influence Function

Let  $\hat{F}_Z$  be the empirical CDF; let the direction  $G$  to be "some probability mass" at a training point  $\mathbf{z}_{tr}$ .

$$\mathcal{I}_{\hat{\theta}}(\mathbf{z}_{tr}) = \lim_{h \rightarrow 0} \frac{\theta((1-h)\hat{F}_Z + h\delta_{\mathbf{z}_{tr}}) - \theta(\hat{F}_Z)}{h} \quad (2)$$

- ▶  $\delta_{\mathbf{z}_{tr}}$  is the Heaviside step function centered at  $\mathbf{z}_{tr}$
- ▶ Meaning: the sensitivity of the statistical functional  $\theta(\hat{F}_Z)$  with respect to "adding a little bit probability mess at  $\mathbf{z}_{tr}$  into the empirical CDF".

Similar to other statistical functionals, the prediction loss, for example:

$$\mathcal{I}_{\mathcal{L}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) = \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{z}_{te}, \theta((1-h)\hat{F}_Z + h\delta_{\mathbf{z}_{tr}})) - \mathcal{L}(\mathbf{z}_{te}, \theta(\hat{F}_Z))}{h} \quad (3)$$

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

## Naive Approach: LOO Refitting, But...

Take  $h = -\frac{1}{n} \rightarrow 0$ , the “contaminated CDF” is:

$$(1 + 1/n)\hat{F}_Z - \delta_{z_i}/n = (1 + 1/n)\hat{F}_{-i} + o(1/n) \approx \hat{F}_{-i}$$

It's the leave- $i$ -out empirical CDF!

$$\begin{aligned}\mathcal{I}_{\hat{\theta}}(z_i) &= \lim_{n \rightarrow \infty} \frac{\theta(\hat{F}_{-i}) - \theta(\hat{F}_Z)}{-1/n} \\ &= \lim_{n \rightarrow \infty} -n(\hat{\theta}_{-\frac{1}{n}, z_i} - \hat{\theta})\end{aligned}\tag{4}$$

- ▶  $\hat{\theta}_{-\frac{1}{n}, z_i}$  is just the LOO estimate.
- ▶ Influence function is the asymptotic approximation of LOO difference in parameter estimate!

# But!

- ▶ If we want the influence from all training examples, we need to do LOO refit  $n$ -times!
- ▶ Super expensive. **NO!**

## Let's Solve the Easier Version First

Suppose the loss is strictly convex, twice differentiable.

- ▶ Use the FOC for the LOO empirical risk minimization problem  
⇒ A closed-form solution for LOO estimator  $\hat{\theta}_{-\frac{1}{n}, \mathbf{z}_i}$ .
- ▶ Taylor approximation, drop high order infinitesimals...
- ▶ The LOO difference is approximately

$$(\hat{\theta}_{-\frac{1}{n}, \mathbf{z}_i} - \hat{\theta}) \approx -\frac{1}{n} \nabla_{\theta}^2 R(\hat{\theta})^{-1} \nabla_{\theta} \mathcal{L}(\mathbf{z}_i, \hat{\theta}) \quad (5)$$

- ▶ Plug into the asymptotic formula:

$$\mathcal{I}_{\hat{\theta}} = \lim_{n \rightarrow \infty} -n(\hat{\theta}_{-\frac{1}{n}, \mathbf{z}_i} - \hat{\theta}) \quad (6)$$

$$\approx -\nabla_{\theta}^2 R(\hat{\theta})^{-1} \nabla_{\theta} \mathcal{L}(\mathbf{z}_i, \hat{\theta}) \quad (7)$$

No retraining, only hessian and gradients.

## Same for the Influence on Loss

$$\begin{aligned}\mathcal{I}_{\mathcal{L}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) &= \nabla_{\delta_{\mathbf{z}_{tr}}} \mathcal{L}(\mathbf{z}_{te}, \theta(F_Z)) \Big|_{F_Z = \widehat{F}_Z} \quad (8) \\ &= \nabla_{\theta} \mathcal{L}(\mathbf{z}_{te}, \widehat{\theta})^{\top} \lim_{h \rightarrow 0} \frac{\theta(\widehat{F}_{-\mathbf{z}_{tr}}) - \theta(\widehat{F}_Z)}{h} \\ &\approx -\nabla_{\theta} \mathcal{L}(\mathbf{z}_{te}, \widehat{\theta})^{\top} \nabla_{\theta}^2 R(\widehat{\theta})^{-1} \nabla_{\theta} \mathcal{L}(\mathbf{z}_{tr}, \widehat{\theta}) \\ &\quad (1 \times p) \quad (p \times p) \quad (p \times 1)\end{aligned}$$

- ▶ Nothing but a fancy version of *chain rule!*
- ▶ Meaning: the validation loss of point  $\mathbf{z}_{te}$ 's sensitivity (w.r.t. to adding a little prob. mass at) training point  $\mathbf{z}_{tr}$ .

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

## How to Compute $\mathcal{I}_{\mathcal{L}}(\mathbf{z}_i, \mathbf{z}_{te})$ ?

Recall  $n = \text{number of observations}$ ,  $p = \text{number of parameters}$ .

$$\mathcal{I}_{\mathcal{L}}(\mathbf{z}_i, \mathbf{z}_{te}) \approx - \underbrace{\nabla_{\theta} \mathcal{L}(\mathbf{z}_{te}, \hat{\theta})^\top \nabla_{\theta}^2 R(\hat{\theta})^{-1} \nabla_{\theta} \mathcal{L}(\mathbf{z}_i, \hat{\theta})}_{\text{Does not depend on } i} \quad (9)$$

- ▶ We want this for all training points  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n$ .
- ▶ Idea: evaluate the part that does not depend on  $i$  only **once**, then  $n$  inner products.  $O(np) + O(?)$ .
- ▶ What is  $O(?)$ ?

## How to Compute the part that does not depend on $i$ ?

$$\text{Let } \mathbf{s}_{te} := \underbrace{\nabla_{\theta} \mathcal{L}(\mathbf{z}_{te}, \hat{\theta})^\top \nabla_{\theta}^2 R(\hat{\theta})^{-1}}_{\text{Does not depend on } i}$$

- ▶  $\nabla_{\theta} \mathcal{L}(\mathbf{z}_{te}, \hat{\theta})^\top$ : just a 1 to  $p$  gradient.  $O(p)$ .
- ▶  $\nabla_{\theta}^2 R(\hat{\theta})^{-1}$ : full hessian of the empirical risk.
  - ▶ Empirical risk: `reduce_sum( n things )`.  $\times n$
  - ▶ Hessian of 1 “thing”: 1 to  $p$ ,  $O(p^2)$

Forming the hessian:  $O(np^2)$ . Inverting the hessian  $O(p^3)$ .

- ▶  $O(np^2 + p^3)$  in total.

Too expensive! On MNIST, even multi-classes logistic regression has  $p = (28^2 + 1)(10 - 1) = 7065$ .

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Conjugate Gradients

$\mathbf{s}_{te} = \nabla_{\theta}\mathcal{L}(\mathbf{z}_{te}, \hat{\boldsymbol{\theta}})^\top \nabla_{\theta}^2 R(\hat{\boldsymbol{\theta}})^{-1}$  is actually the solution to the linear system

$$\nabla_{\theta}^2 R(\hat{\boldsymbol{\theta}}) \mathbf{x} = \nabla_{\theta}\mathcal{L}(\mathbf{z}_{te}, \hat{\boldsymbol{\theta}})^\top$$

- ▶ This is a positive definite linear system, which is good.
- ▶ Linear conjugate gradients method (CG) is an efficient iterative algorithm to solve PD linear system  $\mathbf{A}\mathbf{x} = \mathbf{b}$ .
- ▶ Only requires quantities like  $\mathbf{A}\mathbf{v}$  for arbitrary vectors  $\mathbf{v}$ .
- ▶ No need for  $\mathbf{A}$  itself!

Can avoid explicitly forming and inverting hessian matrix. Only need to implement  **$H\mathbf{v}$  - hessian-vector products (HVPs)**.

## Hessian-Vector Product

- ▶ Easy to implement in auto-gradients systems like Tensorflow.
- ▶ Single HVP evaluation Cost is  $O(np)$ .
- ▶ If  $m$  CG iterations, total cost is  $O(nmp)$ . (typically,  $m = O(p)$ )
- ▶ Should be a big improvement from  $O(np^2 + p^3)$ .

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Stochastic Taylor Approximations

- ▶  $\widehat{\mathbf{H}}$  be the full hessian matrix of empirical risk.
- ▶ If the Jordan normal form of  $(\mathbf{I} - \widehat{\mathbf{H}})$  is  $\mathbf{J}$ , then  $\exists$  an invertible matrix  $\mathbf{P}$  such that  $(\mathbf{I} - \widehat{\mathbf{H}}) = \mathbf{P}\mathbf{J}\mathbf{P}^{-1}$ .
- ▶  $(\mathbf{I} - \widehat{\mathbf{H}})^m = \mathbf{P}\mathbf{J}^m\mathbf{P}^{-1} \rightarrow 0$  as  $m \rightarrow \infty$ , if and only if  $|\lambda| < 1$  for all the eigenvalues of  $(\mathbf{I} - \widehat{\mathbf{H}})$ .

The idea is to use power series (which converges in this case) to approximate hessian inverse.

$(\mathbf{I} - (\mathbf{I} - \widehat{\mathbf{H}}))^{-1} = \widehat{\mathbf{H}}^{-1} = \sum_{i=0}^{\infty} (\mathbf{I} - \widehat{\mathbf{H}})^i$ . The recursive form:

$$\begin{aligned}\widehat{\mathbf{H}}_j^{-1} \mathbf{v} &= [\mathbf{I} + (\mathbf{I} - \widehat{\mathbf{H}})\widehat{\mathbf{H}}_{j-1}^{-1}] \mathbf{v} \\ \mathbf{p}_j &= \mathbf{v} + \mathbf{p}_{j-1} - \widehat{\mathbf{H}}\mathbf{p}_{j-1}\end{aligned}\tag{10}$$

*LiSSA*: a stochastic (minibatch) version of the above. Look at only one training points in each iteration step.  $O(np + dp)$ .

# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Experiments

- ▶ Recall,  $\mathcal{I}_{\hat{\theta}}(\mathbf{z}_i) = \lim_{n \rightarrow \infty} -n(\hat{\theta}_{-\frac{1}{n}, \mathbf{z}_i} - \hat{\theta})$ .
- ▶ Influence function = Asymptotic approximation of the LOO difference (of the influenced statistical functional)!
- ▶ Same argument holds for  $\mathcal{I}_{\mathcal{L}}(\mathbf{z}_{tr}, \mathbf{z}_{te})$ .
- ▶ We test our influence function by plotting it against the LOO retraining difference.

# Ridge Regression

- ▶ Ridge regression empirical risk:

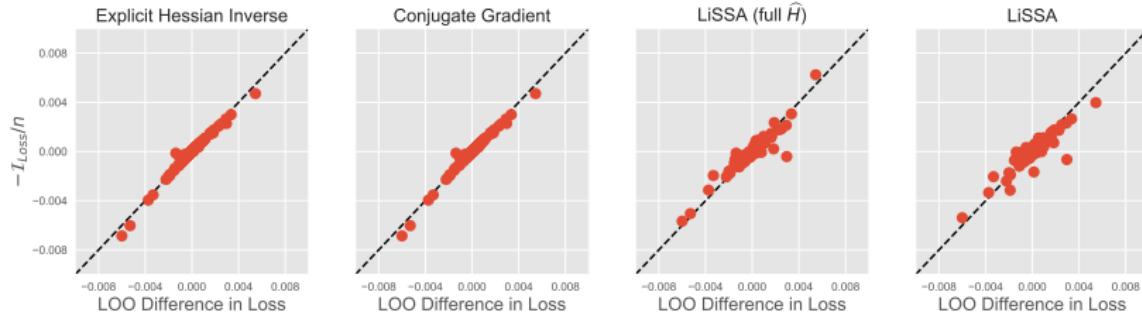
$$R_{\text{ridge}}(\hat{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^n \left( (y_i - \mathbf{x}_i^\top \hat{\boldsymbol{\theta}})^2 + \frac{\lambda}{n} \|\hat{\boldsymbol{\theta}}\|_2^2 \right) \quad (11)$$

- ▶ The influence has a closed-form:

$$\begin{aligned} \mathcal{I}_{\mathcal{L}_{\text{ridge}}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) &= \frac{n}{2} \left[ -2\mathbf{x}_{tr}^\top (y - \mathbf{x}_{tr}^\top \hat{\boldsymbol{\theta}}) + \frac{2\lambda}{n} \hat{\boldsymbol{\theta}} \right]^\top \left( \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p \right)^{-1} \cdot \\ &\quad \left[ -2\mathbf{x}_{te}^\top (y - \mathbf{x}_{te}^\top \hat{\boldsymbol{\theta}}) + \frac{2\lambda}{n} \hat{\boldsymbol{\theta}} \right] \end{aligned} \quad (12)$$

# Ridge Regression

- ▶ Tested on the ForestFires dataset, a regression task with 517 instances and 13 features.
- ▶ After training the model, compute the difference between the validation loss after doing actual LOO retaining and the original loss with  $\hat{\theta}$ , for all training points.
- ▶ Plot the actual LOO difference against the influence to assess the accuracy of our influence functions with all three methods that we discussed.



# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

# Logistic Regression

- ▶ The logistic empirical risk is:

$$R_{\text{logit}}(\hat{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^n \left( -y_i \hat{\boldsymbol{\theta}}^\top \mathbf{x}_i + \log(1 + e^{\hat{\boldsymbol{\theta}}^\top \mathbf{x}_i}) \right) \quad (13)$$

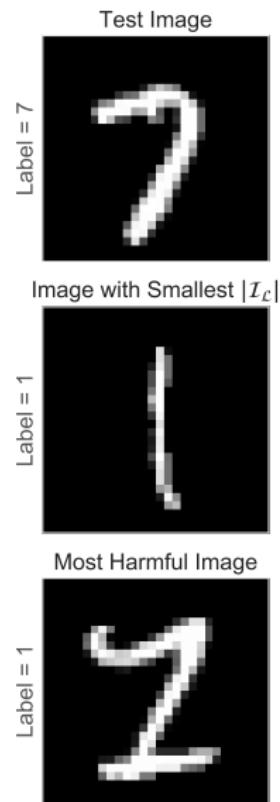
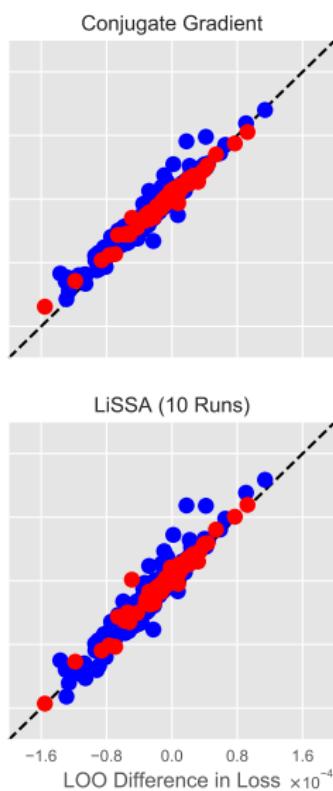
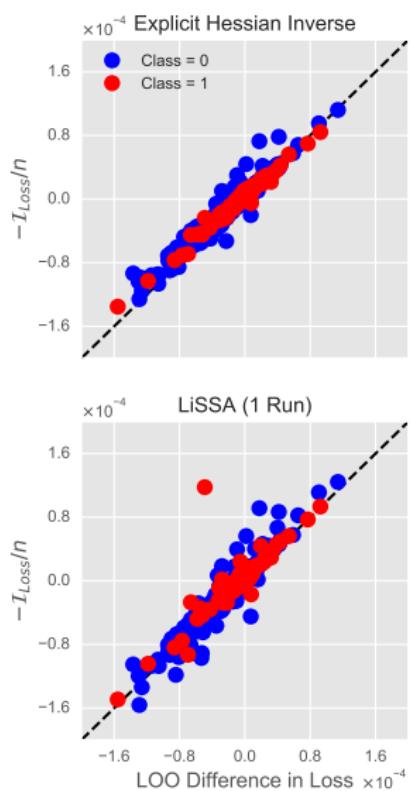
- ▶ Let  $\sigma(t) = 1/(1 + e^{-t})$ , label  $y \in \{0, 1\}$ . The influence also has a closed-form:

$$\begin{aligned} \mathcal{I}_{\mathcal{L}_{\text{logit}}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) &= -\mathbf{x}_{tr}^\top \left( \frac{1}{n} \sum_{i=1}^n \sigma(\hat{\boldsymbol{\theta}}^\top \mathbf{x}_i) \sigma(-\hat{\boldsymbol{\theta}}^\top \mathbf{x}_i) \mathbf{x}_i \mathbf{x}_i^\top \right)^{-1} \mathbf{x}_{te} \\ &\quad (\sigma(\hat{\boldsymbol{\theta}}^\top \mathbf{x}_{tr}) - y_{tr})(\sigma(\hat{\boldsymbol{\theta}}^\top \mathbf{x}_{te}) - y_{te}) \end{aligned} \quad (14)$$

# Logistic Regression

- ▶ We fit an  $L_2$ -regularized logistic regression on the MNIST dataset.
- ▶ Binary case for simplicity: digits 1 and 7. Let the label  $y = 1$  stand for digit 1 and  $y = 0$  for digit 7.
- ▶ We plot the actual LOO difference against the influence function.

# Logistic Regression

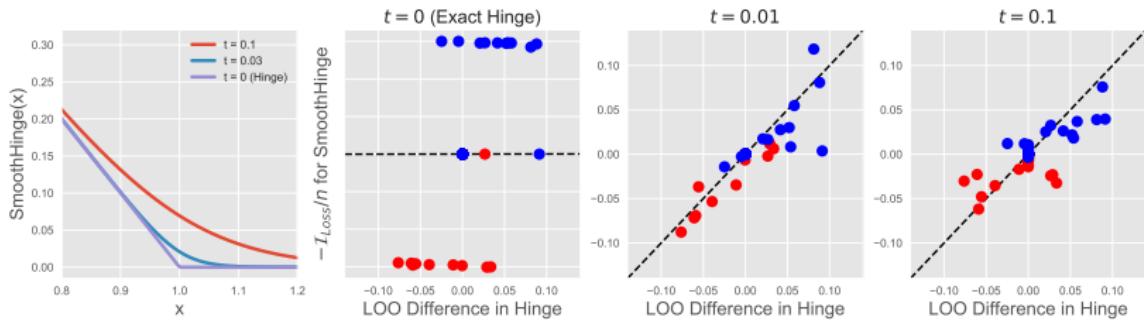


# Support Vector Machine

- ▶ Many loss functions are non-differentiable.
- ▶ Idea: construct a smooth function that converges to the non-differentiable components.
- ▶ Linear SVC Empirical Risk (Non-differentiable):

$$R_{\text{sv}}(\hat{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^n \left[ h(y_i(\mathbf{x}_i^\top \hat{\boldsymbol{\beta}} + \hat{\beta}_0)) + \frac{\lambda}{2n} \|\boldsymbol{\beta}\|_2^2 \right] \quad (15)$$

- ▶ “Smooth Hinge”: replace  $h(x) = \max\{0, 1 - x\}$  with  $h_t(x) := t \log(1 + \exp(\frac{1-x}{t}))$ .



# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

**Understanding the Components of Influence**

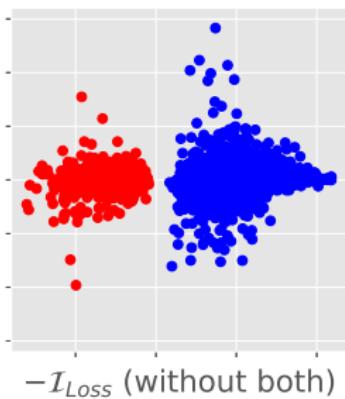
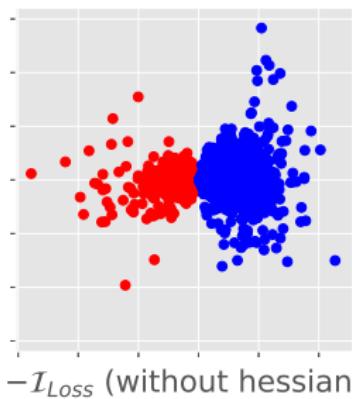
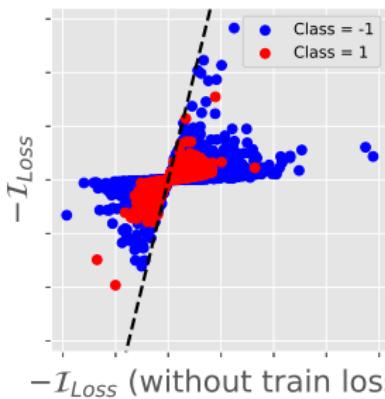
Understanding Model Behavior

Software Package

# Understanding the Components of Influence

$$\begin{aligned}\mathcal{I}'_{\mathcal{L}_{logit}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) &= -\mathbf{x}_{tr}^\top \nabla_\theta^2 R(\hat{\theta})^{-1} \mathbf{x}_{te} \cdot \\ &\quad y_{tr} \sigma(-y_{tr} \hat{\theta}^\top \mathbf{x}_{tr}) \cdot y_{te} \sigma(-y_{te} \hat{\theta}^\top \mathbf{x}_{te})\end{aligned}\tag{16}$$

$$\begin{aligned}\mathcal{I}'_{\mathcal{L}_{logit}}(\mathbf{z}_{tr}, \mathbf{z}_{te}) &= -[\text{Hessian Inv scaled inner product}] \cdot \\ &\quad (\text{training leverage})(\text{sgn}(y_{tr} y_{te}))(\text{constant})\end{aligned}\tag{17}$$



# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

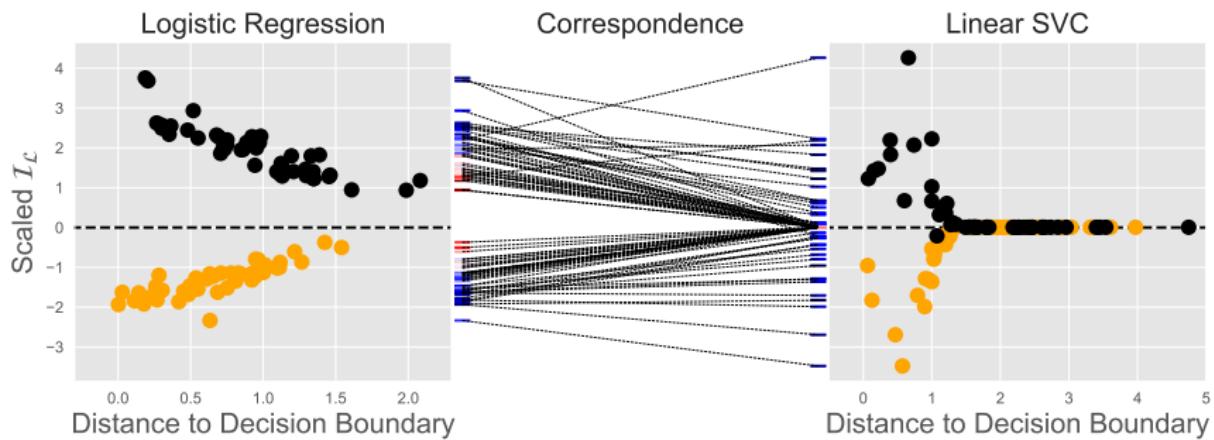
Understanding the Components of Influence

**Understanding Model Behavior**

Software Package

# Understanding Model Behavior

- ▶ The influence function can help us get a better understanding upon how black-box models rely on and extrapolate from the training data.
- ▶ Logistic regression V.S. linear SVC.



# Outline

## Backgrounds

Introduction

What is Influence Function?

How to Compute?

## Implementation

Exact Computation

Conjugate Gradients

Stochastic Taylor Approximations

## Experiments

Ridge Regression Models

Hyperplane Classification

## Applications

Understanding the Components of Influence

Understanding Model Behavior

Software Package

## Software Package

- ▶ We built a generic empirical risk optimizing framework with Tensorflow to compute influence functions for various models.
- ▶ With the help of auto-gradients system, the required gradients and hessian-vector products are automatically kept track of.
- ▶ The user only need to define the empirical risk function and its parametrization.
- ▶ All of our code, data, and reproducible experiments are available in our Github repository:  
<https://github.com/zedyang/46927-Project>.
- ▶ Feel free to download and playaround!

Thanks for your attention