Understanding Black-Box Predictions via Influence Functions

Statistical Machine Final Project

Ze Yang Zhengyang Qi Yundong Liu Yuze Liu

Carnegie Mellon University

2018.05.05

Outline

Outline

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

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:** Idealy, minimizes EPE: A statistical functional.

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

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

$$\boldsymbol{\theta}(\widehat{F}_Z) = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{z}_i, \boldsymbol{\theta})$$

What is Influence Function: Gâteaux Derivative

Generalizes the idea of **directional derivatives**, but the inputs are CDFs!

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

Meaning: the sensitivity of the statistical functional $\boldsymbol{\theta}(F_Z)$ 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 $\boldsymbol{z}_{tr}$.

$$\mathcal{I}_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{z}_{tr}) = \lim_{h \to 0} \frac{\boldsymbol{\theta}((1-h)\widehat{F}_Z + h\delta_{\boldsymbol{z}_{tr}}) - \boldsymbol{\theta}(\widehat{F}_Z)}{h} \tag{2}$$

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

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

$$\mathcal{I}_{\mathcal{L}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te}) = \lim_{h \to 0} \frac{\mathcal{L}(\boldsymbol{z}_{te}, \boldsymbol{\theta}((1-h)\widehat{F}_Z + h\delta_{\boldsymbol{z}_{tr}})) - \mathcal{L}(\boldsymbol{z}_{te}, \boldsymbol{\theta}(\widehat{F}_Z))}{h} \tag{3}$$

Outline

Naive Approach: LOO Refitting, But...

Take $h = -\frac{1}{n} \to 0$, the "contaminated CDF" is:

$$(1 + 1/n)\widehat{F}_Z - \delta_{\boldsymbol{z}_i}/n = (1 + 1/n)\widehat{F}_{-i} + o(1/n) \approx \widehat{F}_{-i}$$

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

$$\begin{aligned} \mathcal{I}_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{z}_i) &= \lim_{n\to\infty} \frac{\boldsymbol{\theta}(\widehat{F}_{-i}) - \boldsymbol{\theta}(\widehat{F}_Z)}{-1/n} \\ &= \lim_{n\to\infty} -n(\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i} - \widehat{\boldsymbol{\theta}}) \end{aligned} \tag{4}$$

- $\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i}$ 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 $\Rightarrow$ A closed-form solution for LOO estimator $\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i}$.
- Taylor approximation, drop high order infinitesimals...
- The LOO difference is approximately

$$(\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i} - \widehat{\boldsymbol{\theta}}) \approx -\frac{1}{n}\nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}})^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_i, \widehat{\boldsymbol{\theta}}) \tag{5}$$

- Plug into the asymptotic formula:

$$\mathcal{I}_{\widehat{\boldsymbol{\theta}}} = \lim_{n\to\infty} -n(\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i} - \widehat{\boldsymbol{\theta}}) \tag{6}$$

$$\approx -\nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}})^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_i, \widehat{\boldsymbol{\theta}}) \tag{7}$$

No retraining, only hessian and gradients.

Same for the Influence on Loss

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

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

Outline

How to Compute $\mathcal{I}_{\mathcal{L}}(z_i, z_{te})$?

Recall n = number of observations, p = number of parameters.

$$\mathcal{I}_{\mathcal{L}}(z_i, z_{te}) \approx -\underbrace{\nabla_\theta \mathcal{L}(z_{te}, \widehat{\theta})^\top \nabla^2_\theta R(\widehat{\theta})^{-1}}_{\text{Does not depend on } i} \nabla_\theta \mathcal{L}(z_i, \widehat{\theta}) \qquad (9)$$

- We want this for all training points $z_1, z_2, ..., z_n$.
- 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?

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

- $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}_{te}, \widehat{\boldsymbol{\theta}})^\top$: just a 1 to p gradient. $O(p)$.
- $\nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}})^{-1}$: 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

Conjugate Gradients

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

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

- 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 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.
- Only requires quantities like $\boldsymbol{A}\boldsymbol{v}$ for arbitrary vectors $\boldsymbol{v}$.
- No need for $\boldsymbol{A}$ itself!

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

Hessain-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

Stochastic Taylor Approximations

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

The idea is to use power series (which converges in this case) to approximate hessian inverse.
$(\boldsymbol{I} - (\boldsymbol{I} - \widehat{\boldsymbol{H}}))^{-1} = \widehat{\boldsymbol{H}}^{-1} = \sum_{i=0}^{\infty}(\boldsymbol{I} - \widehat{\boldsymbol{H}})^i$. The recursive form:

$$\widehat{\boldsymbol{H}}_j^{-1}\boldsymbol{v} = \left[\boldsymbol{I} + (\boldsymbol{I} - \widehat{\boldsymbol{H}})\widehat{\boldsymbol{H}}_{j-1}^{-1}\right]\boldsymbol{v}$$
$$\boldsymbol{p}_j = \boldsymbol{v} + \boldsymbol{p}_{j-1} - \widehat{\boldsymbol{H}}\boldsymbol{p}_{j-1} \tag{10}$$

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

Outline

Experiments

- Recall, $\mathcal{I}_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{z}_i) = \lim_{n\to\infty} -n(\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i} - \widehat{\boldsymbol{\theta}})$.
- influence function is an asymptotic approximation of the quantity $-n(\widehat{\boldsymbol{\theta}}_{-\frac{1}{n},\boldsymbol{z}_i} - \widehat{\boldsymbol{\theta}})$, the difference between leave-one-out refitted parameter and the full empirical risk minimizer, scaled by the sample size.
- Same argument holds for $\mathcal{I}_{\mathcal{L}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te})$.
- We test our influence function by plotting it against the LOO retraining difference.

Ridge Regression

- Ridge regression empirical risk:

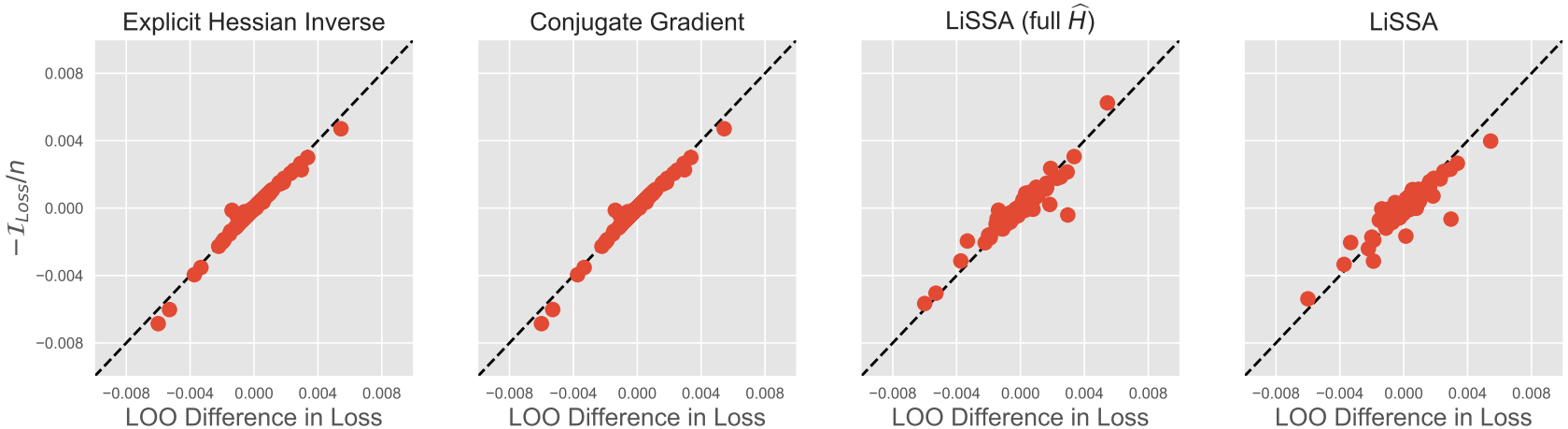
$$R_{\text{ridge}}(\widehat{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i=1}^{n}\left((y_i - \boldsymbol{x}_i^\top\widehat{\boldsymbol{\theta}})^2 + \frac{\lambda}{n}\left\|\widehat{\boldsymbol{\theta}}\right\|_2^2\right) \tag{11}$$

- The influence has a closed-form:

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

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 $\widehat{\theta}$, 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

Logistic Regression

- The logistic empirical risk is:

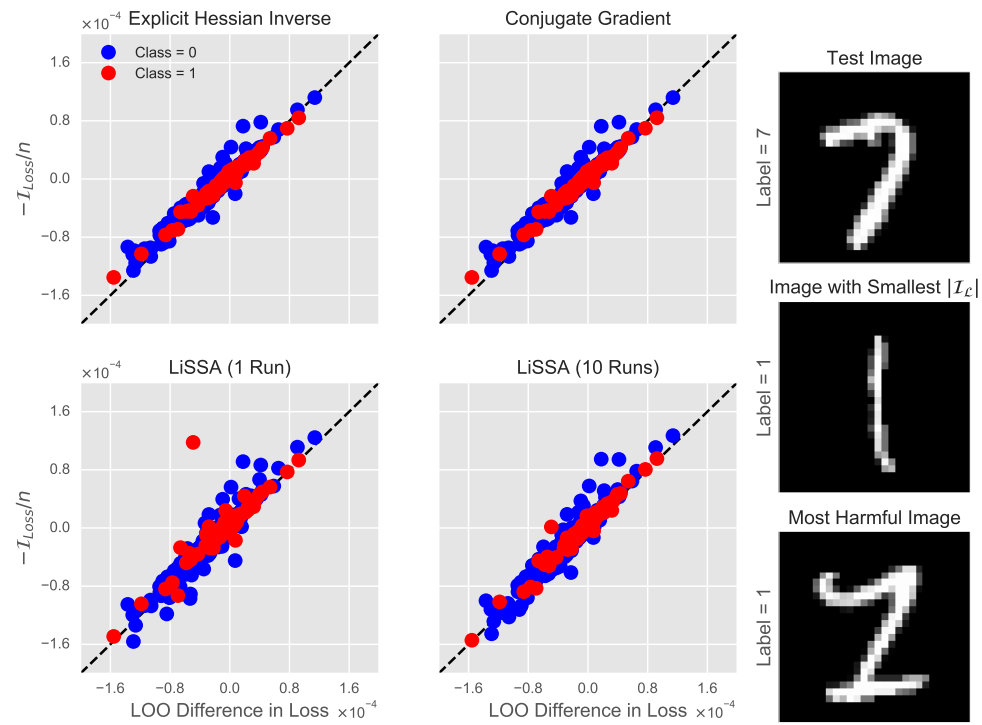
$$R_{\text{logit}}(\widehat{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i=1}^{n}\left(-y_i\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_i + \log(1+e^{\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_i})\right) \tag{13}$$

- Let $\sigma(t) = 1/(1+e^{-t})$, label $y \in \{0,1\}$, the influence of a training point $\boldsymbol{z}_{tr}$ on that validation loss is:

$$\mathcal{I}_{\mathcal{L}_{logit}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te}) = -\boldsymbol{x}_{tr}^\top \nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}})^{-1} \boldsymbol{x}_{te} \cdot \tag{14}$$
$$(\sigma(\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_{tr}) - y_{tr})(\sigma(\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_{te}) - y_{te})$$
$$\nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i=1}^{n} \sigma(\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_i)\sigma(-\widehat{\boldsymbol{\theta}}^\top \boldsymbol{x}_i)\boldsymbol{x}_i \boldsymbol{x}_i^\top$$

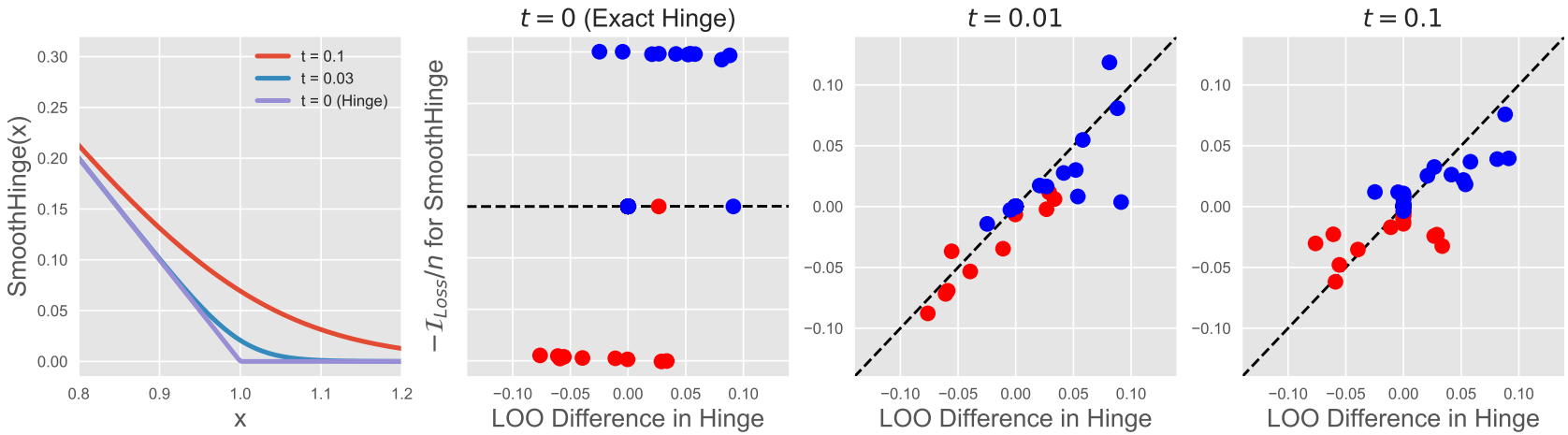
Logistic Regression

- We fit an L_2-regularized logistic regression model with $\lambda = 1000$ on the `MNIST` dataset. For simplicity we only include digits 1 and 7, and we let the label $y = 1$ stand for digit 1 and $y = 0$ for digit 7.
- Then we plot the actual LOO difference against its asymptotic approximation



Support Vector Machine

What if the loss function is not differentiable, such as hinge loss and L_1 regularization? The idea to make things work is to construct a smooth function that converges to the non-differentiable components of the loss function. Then, we minimize the smoothed version of the empirical risk, and compute $\mathcal{I}_{\mathcal{L}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te})$.



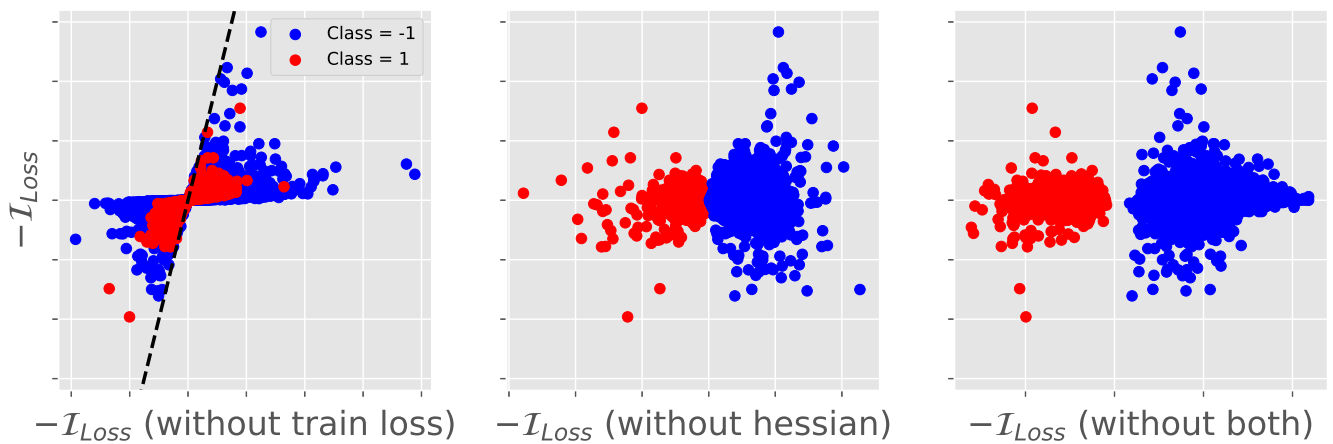
$h(x) = \max\{0, 1 - x\}$ is the hinge loss. We use $h_t(x) := t \log(1 + \exp(\frac{1-x}{t}))$, the "smooth hinge" function.

Outline

Understanding the Components of Influence

$$\mathcal{I}'_{\mathcal{L}_{logit}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te}) = -\boldsymbol{x}_{tr}^{\top} \nabla^2_{\boldsymbol{\theta}} R(\widehat{\boldsymbol{\theta}})^{-1} \boldsymbol{x}_{te} \cdot \tag{15}$$
$$y_{tr}\sigma(-y_{tr}\widehat{\boldsymbol{\theta}}^{\top}\boldsymbol{x}_{tr}) \cdot y_{te}\sigma(-y_{te}\widehat{\boldsymbol{\theta}}^{\top}\boldsymbol{x}_{te})$$

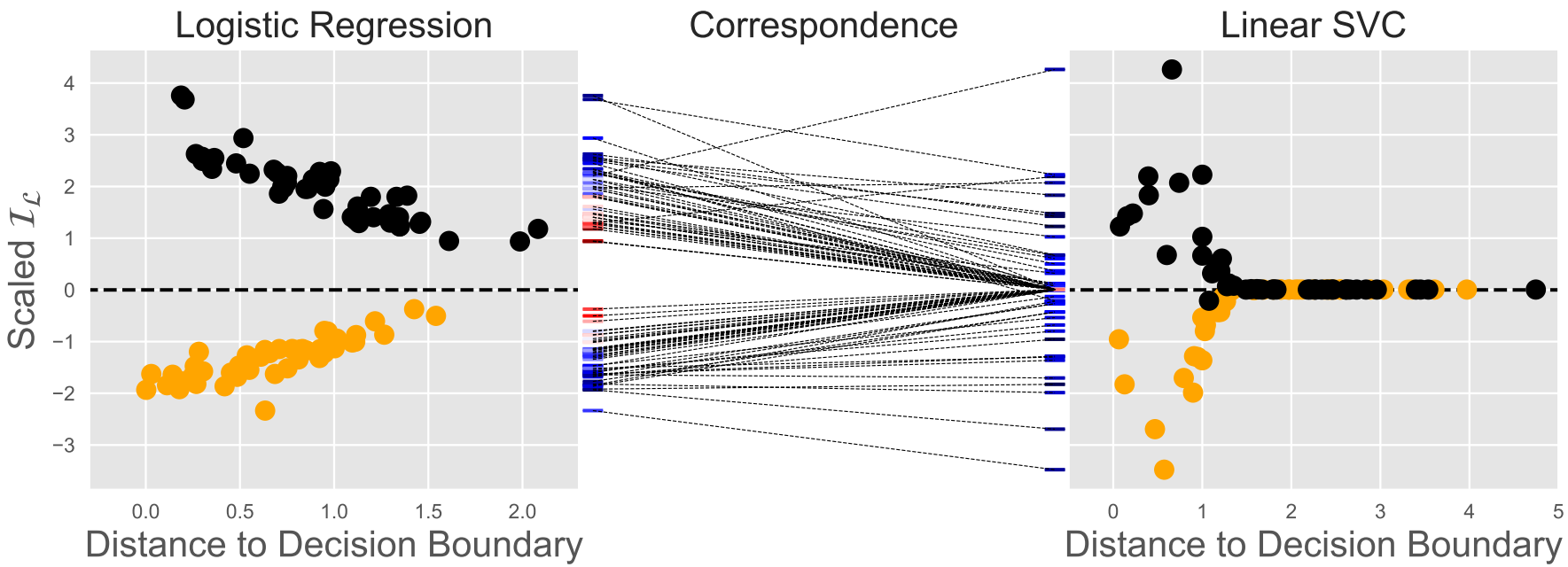
$$\mathcal{I}'_{\mathcal{L}_{logit}}(\boldsymbol{z}_{tr}, \boldsymbol{z}_{te}) = -[\text{Hessian Inv scaled inner product}] \cdot \tag{16}$$
$$(\text{training leverage})(\text{sgn}(y_{tr}y_{te}))(\text{constant})$$



Outline

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

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 funtion 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