# Review of "Understanding Black-box Predictions via Influence Functions"

**Eole Cervenka** [1] **Geovani Rizk** [1]

## Abstract

Pang Wei Koh and Percy Liang introduce a new way to analyze a model predictions via its training data. () Pang Wei Koh & Percy Liang, 2017) We explain how influence functions can be used to efficiently track the impact of each training points on any test point prediction, even when using a non-convex and/or non-differentiable loss function. We reproduce use cases of tracking influence in understanding model behavior, learning adversarial training, debugging a model and checking the labels efficiently. We show the usefulness of using influence functions on other domain like NLP, etc...

## 1. Introduction

State-of-the-art learning models such as deep neural networks are often black boxes (Krizhevsky et al., 2012) and understanding model predictions pose a challenge that limits our ability to interpret, debug existing models or create new models. Prior efforts in understanding model predictions have always started with the assumption that model where fixed functions; taking an input and producing an output.This paper takes a new approach by measuring the influence of training data in the learning process and using this information to measure the influence of training data on the prediction process. The brute-force method to measure how changing the training data can affect the model predictions is prohibitively expensive as it requires retraining for each modified training set. Influence functions offer a closed-form solution to approximate new model parameters based on the previous models after training set perturbation without retraining from scratch. Using influence function requires an expensive second derivative calculations (Hessian matrix). Well see how Hessian Vector Product can be used to make influence functions tractable even in models with millions of parameters. Influence functions require differentiability and convexity constraints that are not satisfied in state of the art models (e.g. deep learning). The authors present how influence functions can be accurately approximated using 2nd order optimization techniques. Finally, the direct applications of this idea will be experimented through use cases of debugging a model, detecting dataset errors and engineering adversarial training points.

[*]Equal contribution [1]University of Paris Dauphine, Paris, France. Correspondence to: Eole Cervenka <eole.cervenka@dauphine.eu>, Geovani Rizk <geovani.rizk@dauphine.eu>.

*Proceedings of the $35^{th}$ International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. 

## 2. Influence Function approach

Suppose that we want to build a model for a prediction problem from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. We use the same notation that the original authors and give the training points $z_1, ..., z_n$ where $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Let $L(z, \theta)$ be the loss value for a certain training point z and the parameters $\theta$ of the model, and let $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$ be the empirical risk. By definition, the empirical risk minimizer is given by $\hat{\theta} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(z_i, \theta)$. In the first parts of this section we suppose that the empirical risk is twice differentiable and strictly convex. In 2.X, we explain how influence function can be approximated accurately and provide useful informations even in non-convex and/or non-differentiable empirical risk context.

### 2.1. Training set modification

In this section, we will demonstrate how influence functions can measure the impact of a modification in the training set on the parameters and the model prediction. Two kinds of dataset modification can be tracked this way:

- removing a training point which can be used to debug the model and to detect dataset errors.
- perturbing a training point which can be used to create adversarial training examples.

In both types of dataset modification, measuring the impact with the brute-force method requires retraining the model which is not tractable.

#### 2.1.1. Removing a training point

Suppose we want to see the effect of removing a point from the training set on parameters; it is equivalent to upweigth-

ing said point by $\epsilon = -\frac{1}{n}$. Influence functions allows us to express as a closed form expression the variation in $\theta$ with respect to a variation in $\epsilon$. Thus, it can be used to linearly approximate the effect of upweighting a point in the training set on $\theta$.

Influence of upweighting a point z on parameters is given by :

$$\begin{aligned}\mathcal{I}_{\text{up,params}}(z) &\stackrel{\text{def}}{=} \frac{d\hat{\theta_{\epsilon,z}}}{d\epsilon}|_{\epsilon=0} \\ &= -\mathcal{H}_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z,\hat{\theta}) \end{aligned} \quad (1)$$

The variation of the loss on $z_{\text{test}}$ with respect to the parameters is given by $\nabla_\theta \mathcal{L}(z_{\text{test}},\hat{\theta})$. By chaining, we can express the variation of the loss on $z_{\text{test}}$ with respect to $\epsilon$ :

$$\begin{aligned}\mathcal{I}_{\text{up,loss}}(z) &= \nabla_\theta \mathcal{L}(z_{\text{test}},\hat{\theta})\ \frac{d\hat{\theta_{\epsilon,z}}}{d\epsilon}|_{\epsilon=0} \\ &= -\nabla_\theta \mathcal{L}(z_{\text{test}},\hat{\theta})\mathcal{H}_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z,\hat{\theta}) \end{aligned} \quad (2)$$

#### 2.1.2. Perturbing a training point

Perturbing a training point z is equivalent to remove it and add its pertubed version $z_\delta = (x+\delta, y)$. Using what we have previously introduce about adding weight to the point, the empirical risk minimizer can be expressed by $\hat{\theta}_{\epsilon,z_\delta,-z} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(z_i,\theta) + \epsilon\mathcal{L}(z_\delta,\theta) - \epsilon\mathcal{L}(z,\theta)$.

Analogically, influence of perturbing a point z on parameters can be expressed as the influence of transfering the weight from point z to point $z_\delta$ :

$$\begin{aligned}\mathcal{I}_{\text{pert,params}}(z) &= \frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}|_{\epsilon=0} \\ &= \mathcal{I}_{\text{up,params}}(z_\delta) - \mathcal{I}_{\text{up,params}}(z) \\ &= -\mathcal{H}_{\hat{\theta}}^{-1}\left(\nabla_\theta \mathcal{L}(z_\delta,\hat{\theta}) - \nabla_\theta \mathcal{L}(z,\hat{\theta})\right) \end{aligned} \quad (3)$$

Suppose that x is continuous and $\delta$ small enough, we can linearly approximate $\nabla_\theta \mathcal{L}(z_\delta,\hat{\theta}) - \nabla_\theta \mathcal{L}(z,\hat{\theta})$ by $\nabla_x \nabla_\theta \mathcal{L}(z,\hat{\theta})\delta$.

???

### 2.2. Additionnal insights

The authors compare the insights from using euclidian distance versus using influence function in the context of searching for influential training point for a given prediction in a logistic regression model.

The euclidian distance (or cosine distance if the points are normalized) is given by $x \cdot x_{test}$.

The influence function that approximates the impact of upweighting a training point on the loss, in a logit regression is given by:

$$y_{\text{test}} y \cdot \sigma(-y_{\text{test}}\theta^\top x_{\text{test}}) \cdot \sigma(-y\theta^\top x) x_{test}^\top \mathcal{H}_{\hat{\theta}}^{-1} x \quad (4)$$

where $\sigma(t) = \frac{1}{1+\exp(-t)}$

- $\mathcal{H}_{\hat{\theta}}^{-1}$ : The inverse Hessian (weighted covariance matrix) is responsible for upweighting the influence of points which add information that is not redundant with information added by other points of the training set
- $\sigma(-y\theta^\top x)$ : The loss term is responsible for downweighting the influence of points which do not impact the predictions of the model

We will reproduce this experience... ?

### 2.3. Making influence function calculation tractable

Computing influence function pose a computational challenge for state-of-the-art models with typically millions of parameters. Specifically, forming the Hessian costs $O(np^2)$ and inverting it costs: $O(p^3)$. Thus, getting the hessian inverse in $\mathcal{I}_{\text{up,loss}}(z,z_{\text{test}}) = -\nabla_\theta \mathcal{L}(z_{\text{test}},\hat{\theta})^\top \mathcal{H}_{\hat{\theta}}^{-1}\mathcal{L}(z,\hat{\theta})$ costs $O(np^2+p^3)$.

The authors present two techniques for approximating $s_{\text{test}} = \mathcal{H}_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z_{\text{test}},\hat{\theta})$; relying on a Hessian Vector Product technique introduced by Pearlmutter.

#### 2.3.1. Conjugate gradients (CG)

The idea is to get $\mathcal{H}_{\hat{\theta}}^{-1}v$ by solving a minimisation problem. The function to minimise is define by $\mathcal{H}_{\hat{\theta}}^{-1}v = \arg\min_t \left(\frac{1}{2}t^\top \mathcal{H}_{\hat{\theta}} t - v^\top t\right)$. The CG approaches can iteratively gives the value of the solution by evaluate $\mathcal{H}_{\hat{\theta}}t$ in $O(np)$ p times with $\theta \in \mathbb{R}^p$. However CG approaches can give us a good estimation in less iterations than p.

#### 2.3.2. Stochastic estimation

While CG requires to go through n training points per iteration, stochastic estimation only samples a single point per iteration, reducing the cost dramatically on large datasets. We use the Taylor expansion to approximate $H^{-1}$. In order to do that, we use the Taylor expansion of ln :

We can express the $j^{\text{th}}$ of the Inverse Hessian as per Taylor Expension as $\mathcal{H}_j^{-1} = I + (I - \mathcal{H}_{j-1}^{-1})$

By definition $\lim_{j->\infty} \mathcal{H}_j^{-1} = \mathcal{H}^{-1}$ and $E\left[\mathcal{H}_j^{-1}\right] = \mathcal{H}_j^{-1}$
Hence, $\mathcal{H}_j^{-1} -> \mathcal{H}^{-1}$

In order to calculate $\mathcal{H}^{-1}v$, stochastic estimation tells us to sample $z_i$ and estimate whithout bias as gradient de $\mathcal{L}(z_i,\theta)$

and to apply the following procedure : We initialize$\mathcal{H}_0^{-1}v = v$, and apply recursively the expression $\mathcal{H}_j^{-1}v = v + (I - \nabla_{\hat{\theta}}\mathcal{L}(z_i, \theta))\mathcal{H}_{j-1}^{-1}v$.

We recurse enough that the value of the Hesian stabilizes and we repeat the procedure r times and average the results.

Empirically, stochastic estimation is faster than CG.

### 2.4. Relaxation of convexity and differentiability constraints

We remind that, in order to use the influence function instead of leave-one-out retraining, we made the assumptions that the parameters $\theta$ of the model reach the global minimum of the empirical risk and that the empirical risk is twice differentiable and strictly convex. In this section we will see how we can relax those asumptions.

#### 2.4.1. Non convex functions

The authors show by experiment that in logistic regression (GLM), the influence function does track change in loss for leave one out training. However, it is not obvious influence functions can be used when the loss function is non convex and does not converge.

The idea of the authors is to use quadratic approximation of the empirirical risk which is by definition smooth and convex, and apply influence function on the risk approximation function. The authors argue that the influence function track training effects even when they use a quadratic approximation of the empirical risk. They experiment with comparing the influence of training point in a CNN learning model as per the influence function versus as per the actual leave-one-out training loss delta. We note that in this case, the influence function tended to overestimate the absolute influence of influential points and underestimate the influence of less influential points in the CNN model.

#### 2.4.2. Non smooth functions

The authors also discuss working around a non smooth objective function. The idea presented in the paper consists in making a smooth approximation of the loss. They support this using the hinge function which cannot be derived in 0. They show that taking the hinge'(0) == 0 does not allow the influence function to provide useful information about training effect. They then proceed to show how using smooth hinge with small enough temperature term allows to approximate an influence function that correlates highly with the actual influence of points.

### 2.5. Use cases

### 2.6. Understanding model behavior

While the inverse of pixel wise euclidean distance is meaningful in SVM to track influence of training points in the prediction of a given test point, it is less so in Inception (CNN). This is due to the fact that SVM will use points that are matching superficial patterns of the class while Inception matches distinctive characteristics of the class. For SVMs, helpful training point belong to the class of the test point and harmful training points belong to some other class. For Inception, helpful training point may be of a different class than that of the test point.

We will produce experiment to track the most influential training point in a similar prediction problems, across two distinct models.

### 2.7. Adversarial training examples

### 2.8. Debugging domain mismatch

### 2.9. Fixing mislabeled examples

USE CASES

1 : Components of influence 2 : Understanding model behavior 3 : Non-convexity and non-convergence 4 :

---

Reproduce experiences (run original code) Reproduce experiences using different: - datasets - data type - objective function

---

==========

## References

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

## A. Do *not* have an appendix here

***Do not put content after the references.*** Put anything that you might normally include after the references in a separate supplementary file.

We recommend that you build supplementary material in a separate document. If you must create one PDF and cut it up, please be careful to use a tool that doesn't alter the margins, and that doesn't aggressively rewrite the PDF file. pdftk usually works fine.

**Please do not use Apple's preview to cut off supplemen-**

**tary material.** In previous years it has altered margins, and created headaches at the camera-ready stage.