Understanding Black-box Predictions via Influence Functions

June 12, 2021

Content

Introduction

Given a high-accuracy, black-box model, and a predict from it. What did the model make this prediction?

1. Make better decisions.
2. Improve the model.
3. Discover new science.
4. Provide end-users explanations.

Introduction

Existing methods:

- Treat model as fixed.
- Explain prediction w.r.t (with respect to) model parameters or test input.

In this paper:

- Treat model as function of training data.
- Explain prediction w.r.t the training data "most responsible" for predict.

Introduction

To formalize the impact of a training point on a prediction:

- What would happen if we didn't have this training point?
- Or if the values of this training point were changed slightly?

The symbols:

- $\mathcal{X}$: input space
- $\mathcal{Y}$: output space
- $\{z_i = (x_i, y_i)\}_{i=1}^n$: training points in $\mathcal{X} \times \mathcal{Y}$
- For a point z and parameters $\theta \in \Theta$, let $L(z, \theta)$ be the loss, and let $\frac{1}{n}\sum_{i=1}^n L(z_i, \theta)$ be the empirical risk.

The empirical risk minimizer is given by:

$$\hat{\theta} := \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

Assume: Empirical risk is twice-differentiable and stricly convex in θ

Influence functions

Upweighting a training point

- New parameters: $F(\epsilon) = \hat{\theta}_{\epsilon,z} := \underset{\theta\in\Theta}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z,\theta)$
- Define:

$$\mathcal{I}_{\text{up,params}}(z) := \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = H_{\hat{\theta}}^{-1}\nabla_{\theta}L(z,\hat{\theta})$$

  with: $H_{\hat{\theta}} := \frac{1}{n}\sum_{i=1}^{n}\nabla_{\theta}^{2}L(z_i,\hat{\theta})$ is the Hessian and is positive definite (PD) by assumption.
- Linear approximation (Taylor): $F(\epsilon) \approx F(0) + \epsilon * \mathcal{I}_{\text{up,params}}(z)$
- So $\hat{\theta}_{\epsilon,z} - \hat{\theta} \approx \epsilon * \mathcal{I}_{\text{up,params}}(z)$ and $\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n}\mathcal{I}_{\text{up,params}}(z)$

Influence function

Deriving $\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$

- Recall that $\hat{\theta}$ minimizes the empirical risk:

$$R(\theta) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} L\left(z_i, \theta\right) \tag{1}$$

- We further assume that R is twice-differentiable and strongly convex in θ:

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \nabla^2 R(\hat{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\theta}^2 L\left(z_i, \hat{\theta}\right) \tag{2}$$

- The perturbed parameters $\hat{\theta}_{\epsilon,z}$ can be written as:

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta}\{R(\theta) + \epsilon L(z,\theta)\} \tag{3}$$

Influence functions

Deriving $\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$

- Define: $\Delta_\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$. We have:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon} \tag{4}$$

- Since $\hat{\theta}_{\epsilon,z}$ is minimizer of (3), let us examine its firstorder optimality conditions:

$$0 = \nabla R\left(\hat{\theta}_{\epsilon,z}\right) + \epsilon \nabla L\left(z, \hat{\theta}_{\epsilon,z}\right) \tag{5}$$

Influence functions

Deriving $\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$

- Since $\hat{\theta}_{\epsilon,z} \rightarrow \theta$ as $\epsilon \rightarrow 0$, we perform a Taylor expansion:

$$\begin{aligned} 0 \approx & [\nabla R(\hat{\theta}) + \epsilon \nabla L(z,\hat{\theta})] + \\ & \left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z,\hat{\theta})\right] \Delta_\epsilon \end{aligned} \tag{6}$$

- We have $\nabla R(\hat{\theta}) = 0$. Keeping only $O(\epsilon)$ terms, we have:

$$\Delta_\epsilon \approx -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z,\hat{\theta})\epsilon \tag{7}$$

- Finally:

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla L(z,\hat{\theta}) \stackrel{\text{def}}{=} \mathcal{I}_{\text{up,params}}(z) \tag{8}$$

Influence of upweighting z on the loss at the test point z_{test}:

$$
\begin{aligned}
\mathcal{I}_{\text{up,loss}}\left(z, z_{\text{test}}\right) &\stackrel{\text{def}}{=} \left.\frac{dL\left(z_{\text{test}}, \hat{\theta}_{\epsilon,z}\right)}{d\epsilon}\right|_{\epsilon=0} \\
&= \nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^\top \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\
&= -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})
\end{aligned}
$$

Influence functions

Perturbing a training input

- If we change $z = (x, y)$ to $z_\delta = (x + \delta, y)$, what will test loss change?
- $z = (x, y)$ to $z_\delta = (x + \delta, y)$ equals to delete z then add z_δ.
- New parameter:

$$\hat{\theta}_{\epsilon, z_\delta, -z} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z_\delta, \theta) - \epsilon L(z, \theta)$$

- We have:

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

Influence functions

Perturbing a training input

- Linear approximation:

$$\hat{\theta}_{z_\delta,-z} - \hat{\theta} \approx \frac{1}{n}\left(\mathcal{I}_{\text{up,params}}(z_\delta) - \mathcal{I}_{\text{up,params}}(z)\right)$$

- As $||\delta|| \to 0$, $\nabla_\theta L\left(z_\delta, \hat{\theta}\right) - \nabla_\theta L(z, \hat{\theta}) \approx \left[\nabla_x \nabla_\theta L(z, \hat{\theta})\right] \delta$
- So:

$$\left.\frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}\right|_{\epsilon=0} \approx -H_{\hat{\theta}}^{-1}\left[\nabla_x \nabla_\theta L(z, \hat{\theta})\right] \delta$$

$$\hat{\theta}_{z_\delta,-z} - \hat{\theta} \approx -\frac{1}{n} H_{\hat{\theta}}^{-1}\left[\nabla_x \nabla_\theta L(z, \hat{\theta})\right] \delta$$

Influence of perturbing $z \rightarrow z_\delta$ on the loss at the test point z_{test}:

$$
\begin{aligned}
\mathcal{I}_{\text{pert,loss}}\left(z, z_{\text{test}}\right) &\stackrel{\text{def}}{=} \nabla_\delta L\left(z_{\text{test}}, \hat{\theta}_{z_\delta,-z}\right)\Big|_{\delta=0} \\
&= -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^{\top} H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta L(z, \hat{\theta})
\end{aligned}
$$

- Challenges:
  - How to calculating Inverse Hessian Matrix?
  - How to calculating influence function $\mathcal{I}_{\text{up,loss}}(z_i, z_{\text{test}})$ on all training points?
- What happens when our assumptions are violated?

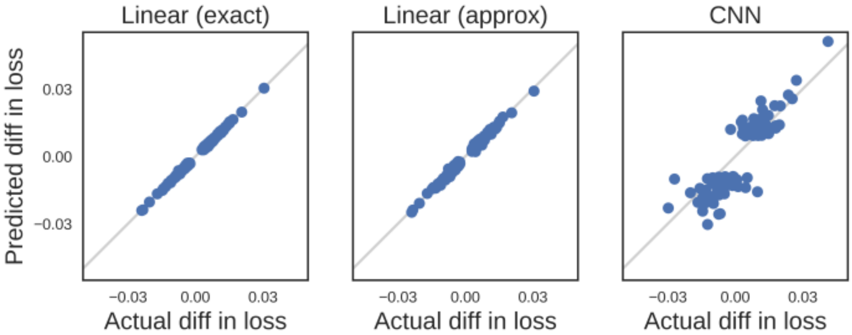
Scaling up

Calculating Inverse Hessian Matrix

We have: n training points, p parameters.

- Inverse Hessian Matrix requires $O(np^2 + p^3)$
- Use Conjugate gradients (CG) requires $O(np)$
- Use Stochastic estimation
  - Key idea: Don't explicity form H_θ^{-1}. Instead, compute $H_\theta^{-1} v$.
  - $s_{\text{test}} = H_\theta^{-1} \nabla_\theta L(z_{\text{test}}, \hat{\theta})$. Then, $\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \nabla_\theta L(z, \hat{\theta})$

- Compared $-\frac{1}{n}\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}})$ with $L(z_{\text{test}}, \hat{\theta}_{-z}) - L(z_{\text{test}}, \hat{\theta})$ using logistic regression, CNN model on 10-class MNIST.



Scaling up

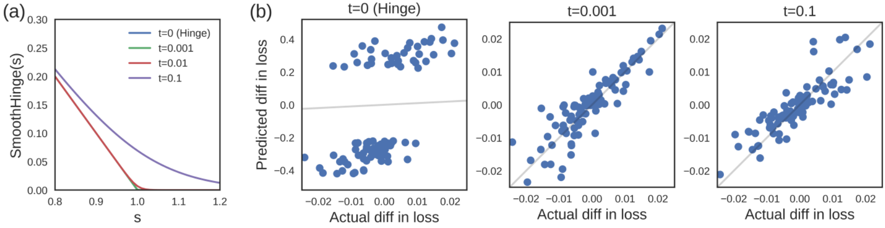
Non-convexity

- We took $\hat{\theta}$ as the global minimum. If we have $\widetilde{\theta}$ on non-convex objectives, $\widetilde{\theta} \neq \hat{\theta}$. As a result, $H_{\widetilde{\theta}}$ could have negative eigenvalues.
- Calculate $\mathcal{I}_{\text{up,loss}}$ using $\widetilde{L}$:

$$\widetilde{L}(z,\theta) = L(z,\widetilde{\theta}) + \nabla L(z,\widetilde{\theta})^T(\theta - \widetilde{\theta}) + \frac{1}{2}(\theta - \widetilde{\theta})^T(H_{\widetilde{\theta}} + \lambda I)(\theta - \widetilde{\theta})$$

- What happens when $\nabla_\theta L$ and $\nabla_\theta^2 L$, do not exits?
- Key idea: Replace origin L with smoothed version. E,g:

$$\begin{aligned} Hinge(s) &= \max(0, 1-s) \\ &\approx SmoothHinge(s,t) := t \log\left(1 + \exp\left(\frac{1-s}{t}\right)\right) \end{aligned}$$



Application

1. Understanding model behavior
2. Adversarial training examples
3. Debugging domain mismatch
4. Fixing mislabeled examples

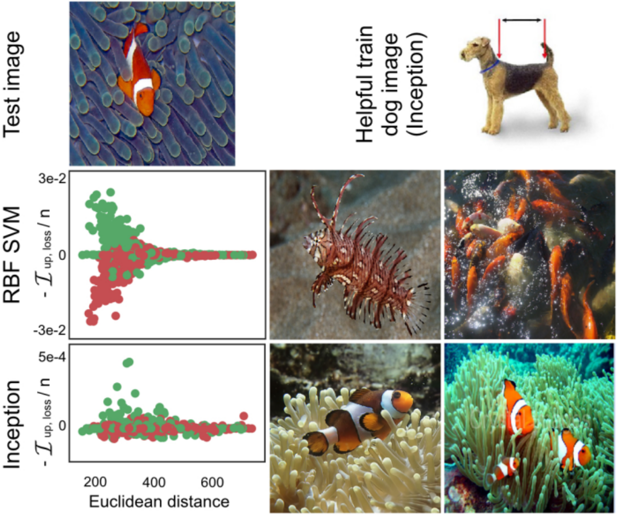
Applications

Understanding model behavior

- Model 1: Inception v2 with all but the top layer frozen (used pre-trained from Keras).
- Model 2: SVM with RBF kernel.
- Task: Binary classification of fish and dog.

Applications

Understanding model behavior



Applications

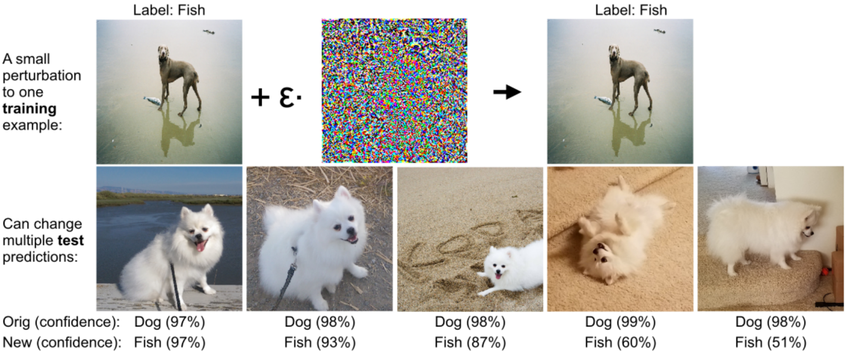
Adversarial training examples

- There exits some paper generating some adversarial test images that are visually indistinguishable but can fool a classifier.
- We demonstrate we can craft adversarial training images that can flip a model's prediction.
- Basically, the idea is iterating on training images on the direction of influence function.

- Data same as fish and dog
    - Origin correctly classified 591/600 test images.
    - For each test image, find only one training image and do 100 iterations.
    - 335 (57%) of the testing images were flipped.
    - Also, attack on one training image can influence multiple test images.

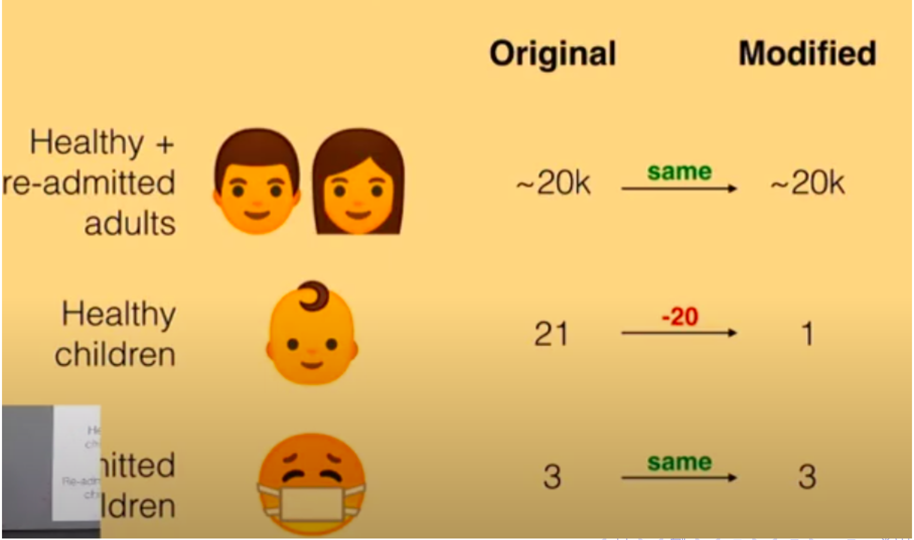
Applications

Adversarial training examples



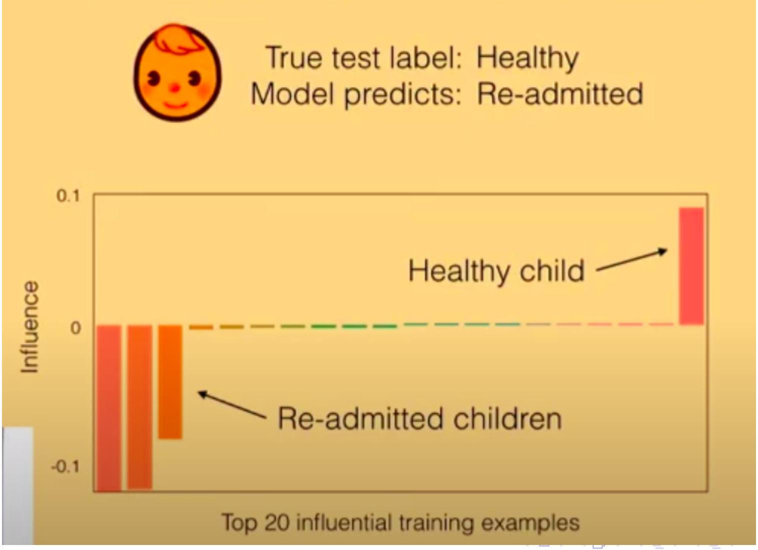
Applications

Debugging domain mismatch



Applications

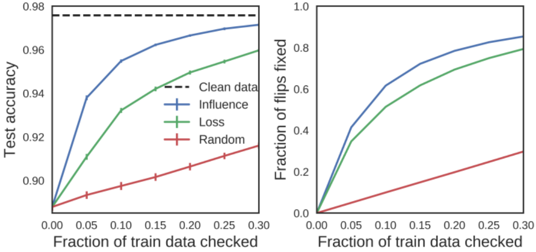
Debugging domain mismatch



Applications

Fixing mislabeled examples

- Only have training set. Find example with largest loss.
- We meansure the influence of z_i with $\mathcal{I}_{\text{up,loss}}(z_i, z_i)$ which approximates the error incurred on z_i if we remove z_i from training data.



Conclusion

- We can better understand the behavior of a model by looking at how it was derived from training data.
- Influence functions let us do this efficiently.
- Key: Differentiate through training and reply on asymptotics.
- Locality allows us to get a closed form expression. Can we get at a global notion of influence?
- Much more work to be done on these black box model tool.

Thank you for your attention!