Representer Point Selection for Explaining Deep Neural Networks

Chih-Kuan Yeh*, Joon Sik Kim*, Ian E.H. Yen, Pradeep Ravikumar

MACHINE LEARNING DEPARTMENT

Carnegie Mellon University

Overview

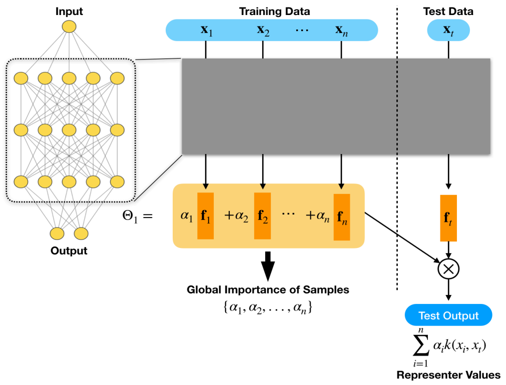
- We extend the representer theorem for RKHS to a DNN function Φ. For a testing point x_t and training points x_i, $\Phi(x_t) = \Sigma_i^n \alpha_i k(x_i, x_t)$ for some function k.
- $\alpha_i k(x_i, x_t)$ is called the *representer value* for prediction, and α_i captures the global importance for x_i.
- Our method can be used as an interpretable model with L2 regularization and a post-hoc explanation for any DNN-based model.

Methods

Theorem 3.1. *Let us denote the neural network prediction function by* $\hat{\mathbf{y}}_i = \sigma(\Phi(\mathbf{x}_i, \Theta))$, *where* $\Phi(\mathbf{x}_i, \Theta) = \Theta_1 \mathbf{f}_i$ *and* $\mathbf{f}_i = \Phi_2(\mathbf{x}_i, \Theta_2)$. *Suppose* Θ^* *is a stationary point of the optimization problem:* $\arg\min_\Theta \{\frac{1}{n}\sum_i^n L(\mathbf{x}_i, \mathbf{y}_i, \Theta)) + g(||\Theta_1||)\}$, *where* $g(||\Theta_1||) = \lambda ||\Theta_1||^2$ *for some* $\lambda > 0$. *Then we have the decomposition:*

$$\Phi(\mathbf{x}_t, \Theta^*) = \sum_i^n k(\mathbf{x}_t, \mathbf{x}_i, \alpha_i),$$

where $\alpha_i = \frac{1}{-2\lambda n}\frac{\partial L(\mathbf{x}_i, \mathbf{y}_i, \Theta)}{\partial \Phi(\mathbf{x}_i, \Theta)}$ *and* $k(\mathbf{x}_t, \mathbf{x}_i, \alpha_i) = \alpha_i \mathbf{f}_i^T \mathbf{f}_t$, *which we call a representer value for* $\mathbf{x}_i$ *given* $\mathbf{x}_t$.



Training an Interpretable Model with L2 Regularization

$$\Theta^* = \arg\min_{\Theta} \frac{1}{n}\sum_i^n L(\mathbf{y}_i, \Phi(\mathbf{x}_i, \Theta)) + \lambda ||\Theta_1||^2$$

Post-hoc Analysis of a Given Pre-trained Model

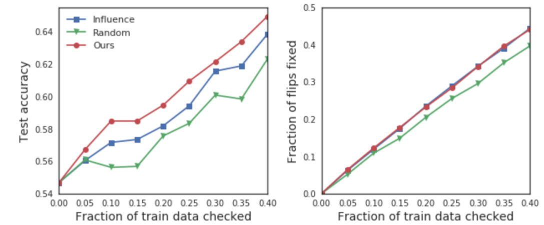
$$\Theta^* \in \arg\min_{\Theta} \left\{\frac{1}{n}\sum_i^n L(\Phi(\mathbf{x}_i, \Theta_{given}), \Phi(\mathbf{x}_i, \Theta)) + \lambda ||\Theta_1||^2\right\}$$

for any $\Theta^* \in \arg\min_\Theta L(\Phi(\mathbf{x}_i, \Theta_{given}), \Phi(\mathbf{x}_i, \Theta))$, we have $\sigma(\Phi(\mathbf{x}_i, \Theta^*)) = \sigma(\Phi(\mathbf{x}_i, \Theta_{given}))$.

Experiments

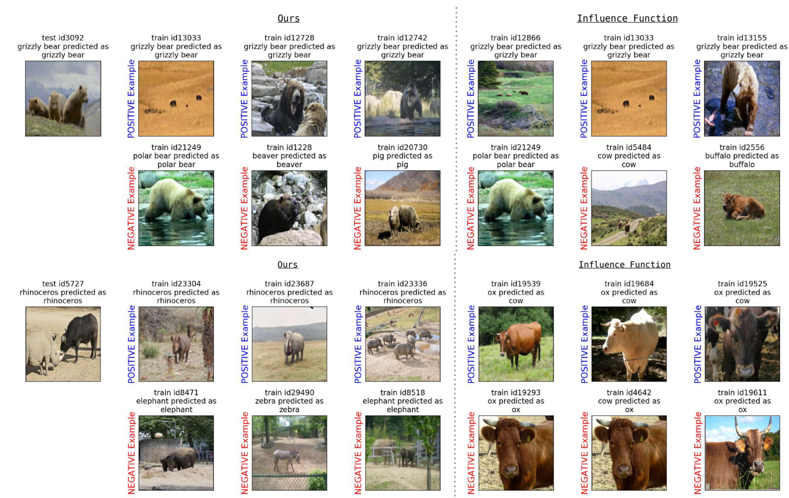
1. Dataset Debugging

The corrupted data has some labels flipped, and the goal is to find these incorrect points and correct them to perform better.



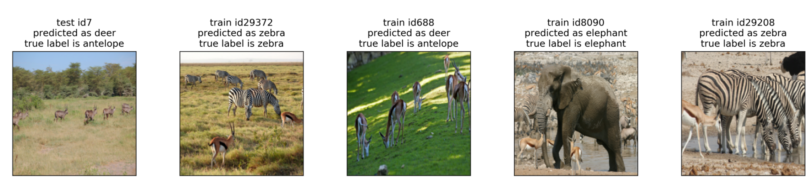
By inspecting the training points using the representer value, we recover the same amount of mislabeled training points as the influence function (right) with the highest test accuracy (left).

2. Positive/Negative Representer Points



Comparison of top three positive and negative influential training images for two test points (left-most column) using our method (left three columns) and influence functions (right three columns).

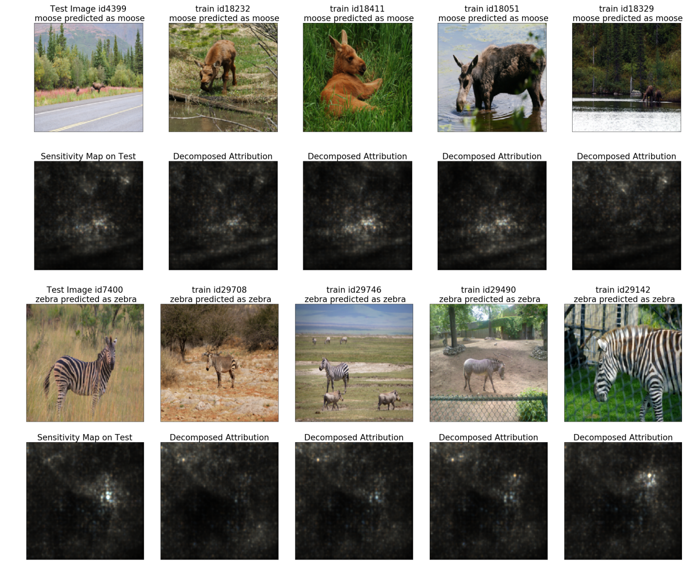
3. Misclassified Examples



A misclassified test image (left) and the set of four training images that had the most negative representer values for almost all test images in which the model made the same mistakes. The negative influential images all have antelopes in the image despite the label being a different animal.

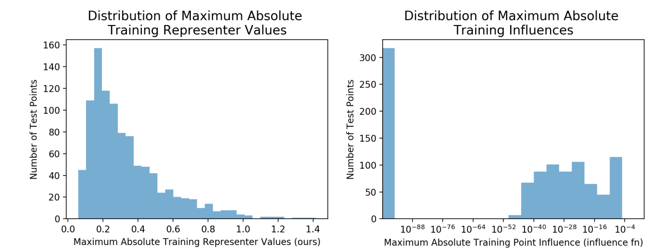
Experiments

4. Sensitivity Map Decomposition



The less the training point displays spurious features from the background and more of the features related to the object of interest, the more focused the decomposed sensitivity map of that training point is at the region the test sensitivity map highlights.

5. Numerical Stability / Computational Cost



The histogram of influence function/representer values for test points in CIFAR-10. While ours have more evenly spread out and larger values across different test points (left), the influence function values are either really small or zero for some points (right).

	Influence Function (Koh et al. 2017)		Representer Points (Ours)	
Dataset	Fine-Tuning	Computation	Fine-Tuning	Computation
CIFAR10	0	267.08 ± 248.20	7.09 ± 0.76	**0.10 ± 0.08**
AwA	0	172.71 ± 32.63	12.41 ± 2.37	**0.19 ± 0.12**

Average time (in seconds) required for computing values corresponding to all training points for a test point.