Seminar

Explainable Machine Learning (SS18)

Understanding Black-box Predictions via Influence Functions

by Pang Wei Koh and Percy Liang

This seminar thesis has been carried out by

Philipp de Sombre (3502136)

under the supervision of

Prof. Dr. Ullrich Köthe

Contents

1 Introduction

1.1 Motivation

Machine Learning has reached a state where it can achieve superhuman performance for many applications, such as answering trivia questions [Markoff, 2011], diagnosing skin cancer [Kubota, 2017] and even performing legal work [Kohn, 2017]. These machine learning models and many others that are used in practice, work as a black box. Black-box models perform significantly better than their white-box counterparts, with the disadvantage of not giving the user a notion of how the prediction was formed. This leads to several downsides. Firstly, the new General Data Protection Regulation (GDPR) states that a subject that, *"[the data subject should have] the right [...] to obtain an explanation of the decision reached"* [European Union, 2016, Recital 71]. Black-box models can not provide such an explanation. Secondly, if scientists understand what models do, they can potentially improve the models performance [Saleema et al., 2015]. Furthermore, scientists can use the explanations of a well performing model to possibly make new scientific discoveries [Shrikumar et al., 2016].

To tackle the disadvantages of black-box models and make their actions more comprehensible, we will introduce a method of explaining the results, to the user.

1.2 Related work

Because of the relevance of this issue, other approaches on how to explain a prediction of a model have already been published.

One approach is fitting another model locally around a test point. This other model is a simple approximation of the original model and therefor easier to understand by humans. This approach was published for example in [Ribeiro et al., 2016].

Another approach is to modify the attribute values of a test point and then look at the change in the prediction. This approach is closely related to trying to find attributes that maximize the confidence of the model for a certain target value. Results of research using this approach was published in [Simonyan et al., 2013], [Datta et al., 2016], [Adler et al., 2016] and [Li et al., 2016].

These approaches have in common that they treat a black-box model and its prediction as a fixed entity. They do not explain how the model came to be in the first place.

1.3 Overview

First we will lay the theoretical groundwork in the chapter Theoretical Foundations. Here we explain the approach and show how it can be adapted to solve real-world problems. In the next chapter (3), we present experiments and their results to show that the theory works. Furthermore, we also present some practical applications. Finally, in the Conclusion we summarize the content of this seminar thesis and also highlight some disadvantages.

2 Theoretical Foundations

2.1 Approach

2.1.1 Idea

As the term machine learning suggests, the predictions for a data point are learned and not known a priori. For supervised learning, this is done using training data where the correct values or labels of the prediction task are known. Instead of treating the model as a fixed entity we trace the prediction back to the training data. Here, the model parameters were learned. We then can use the training data to give an explanation of how much an individual part of the training data influenced a given result.

2.1.2 Calculating Influence

To have a fine notion of the influence a training point has on a given test point, we derive two measurements of influence. Firstly, the total impact a training point has on a test point. Secondly, the impact individual attributes have on a test point. The total impact of a training point can be calculated by looking at the impact of up- or down-weighting the importance of a training point, whereas for the individual features a training point has to be perturbed.

In general, we have an input space X where we want to predict the correct values of an output space Y for a prediction problem. We have obtained training points $z_1, \ldots, z_n$ where each $z_i = (x_i, y_i) \in X \times Y$. We define a loss function $L(z, \theta)$ with a training point z and parameters $\theta \in \Theta$. We define our average loss over all training points, given by $\frac{1}{n}\Sigma_{i=1}^{n} L(z_i, \theta)$, as the empirical risk. The goal is to find an empirical risk minimizer $\hat{\theta}$, which is given by $\hat{\theta} \overset{\text{def}}{=} argmin_{\theta \in \Theta} \frac{1}{n}\Sigma_{i=1}^{n} L(z_i, \theta)$ with any regularization term being folded into L. For this section we assume that the empirical risk is twice-differentiable as well as strictly convex in θ. In the section Generalization and Optimization we relax these restrictions.

Upweighting of a training point First, we want to understand the total impact of a training point on a prediction of the model. We begin by studying the change of parameters θ, given an upweighting of a training point z by some factor ϵ. The empirical risk minimizer is then defined as $\hat{\theta}_{\epsilon,z} \overset{\text{def}}{=} argmin_{\theta \in \Theta} \frac{1}{n}\Sigma_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta)$. Note that the impact of the removal of a training point z can be calculated by setting $\epsilon \overset{\text{def}}{=} -\frac{1}{n}$. We now derive the impact for an infinitesimally small ϵ using the idea of influence from [Cook and Weisberg, 1982]. This gives us:

$$\begin{aligned} I_{up,params}(z) &\overset{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\ &= -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta}) \end{aligned} \tag{2.1}$$

[1].

Here we derive the empirical risk minimizer with respect to ϵ at $\epsilon = 0$. This represents the infinitesimally small upweighting. The hessian matrix of the empirical risk minimizer is given by $H_{\hat{\theta}} \overset{\text{def}}{=} \frac{1}{n}\Sigma_{i=1}^{n}\nabla_\theta^2 L(z_i,\hat{\theta})$ and positive definite by assumption. Since we noted earlier that the removal of a point is equivalent to upweighting it by $\epsilon \overset{\text{def}}{=} -\frac{1}{n}$, we derive that $I_{remove,params}(z) \overset{\text{def}}{=} -\frac{1}{n}I_{up,params}(z)$. This can be calculated without having to retrain the model. After we calculated the influence of the upweighting on the parameter, we now need to calculate the influence on the loss for a test point. This gives us a notion of how much impact a training point has on a prediciton for the specific test point (z_{test}). We derive this using the chain rule.

$$\begin{aligned} I_{up,loss}(z,z_{test}) &\overset{\text{def}}{=} \left.\frac{dL(z_{test}\hat{\theta}_{\epsilon,z})}{d\epsilon}\right|_{\epsilon=0} \\ &= \nabla_\theta L(z_{test},\hat{\theta})^\top \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\ &= -\nabla_\theta L(z_{test},\hat{\theta})^\top H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta}) \end{aligned} \tag{2.2}$$

Perturbing of a training point In the second step we develop a finer notion of influence by looking at the influence of the individual features of a training point. To find the solution to this problem we answer the question: *How does a prediction change, if we alter the features of a training point?* .

Consider an original training point $z = (x, y)$, for which we define the altered point $z_\delta \overset{\text{def}}{=} (x + \delta, y)$. Then, let $\hat{\theta}_{z_\delta,-z}$ be the empirical risk minimizer for when training point z is replaced by altered point z_δ. An approximation of the new parameters caused by moving ϵ mass from z onto z_δ is given by: $\hat{\theta}_{\epsilon,z_\delta,-z} \overset{\text{def}}{=} argmin_{\theta\in\Theta}\frac{1}{n}\Sigma_{i=1}^{n}L(z_i,\theta)+ L(z_\delta,\theta) - L(z,\theta)$. Performing analogous calculations as in equation 2.1 gives us:

$$\begin{aligned} I_{pert,params}(z) &\overset{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}\right|_{\epsilon=0} \\ &= I_{up,params}(z_\delta) - I_{up,params}(z) \\ &= -H_{\hat{\theta}}^{-1}(\nabla_\theta L(z_\delta,\hat{\theta}) - \nabla_\theta L(z,\hat{\theta})) \end{aligned} \tag{2.3}$$

[1]The calculations are detailed in Appendix A of [Koh and Liang, 2017]

.

If the attribute values of x are continuous and δ is small, we can approximate $I_{pert,params}(z)$ further. If the input space $X \subseteq \mathbb{R}^d$, the parameter space $\Theta \subseteq \mathbb{R}^p$ and for the loss L exists a derivative in θ and x.

We can approximate $\nabla_\theta L(z_\delta, \hat{\theta}) - \nabla_\theta L(z, \hat{\theta}) \approx [\nabla_x \nabla_\theta L(z, \hat{\theta})]\delta$ from equation 2.4 as the absolute value of δ goes towards 0. Using this approximation, we can now apply the chain rule and calculate the influence of attributes of a training point on a test point ($I_{pert,loss}(z, z_{test})$). This gives us:

$$\begin{aligned} I_{pert,loss}(z, z_{test})^\top &\overset{\text{def}}{=} \nabla_\delta L(z_{test}, \hat{\theta}_{z_\delta,-z})^\top \Big|_{\delta=0} \\ &= -\nabla_\theta L(z_{test}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta L(z, \hat{\theta}) \end{aligned} \tag{2.4}$$

.

This formula lets us identify the most influential features on the loss for z_{test}. This can be interpreted as being the features that have the biggest impact on the prediction of the model. To be precise $I_{pert,loss}(z, z_{test})^\top \delta$ gives us an approximation of the effect that $z \mapsto z + \delta$ has on the loss at the point z_{test}.

2.2 Generalization and Optimization

2.2.1 Non-differentiability of the loss function

One assumption we made, was that the empirical risk is twice-differentiable. Hence, the loss function has to be twice-differentiable. This is not the case for all loss functions. For example the first and second derivative of the Hinge-loss function, which is defined as $Hinge(s) = \max(0, 1 - s)$, is undefined at $s = 1$. We can use a smooth approximation of the Hinge loss function which in turn is twice-differentiable. The smooth Hinge loss is dependent on a smoothing hyper-parameter t and defined as: $SmoothHinge(s, t) = t \log(1 + \exp(\frac{1-s}{t}))$. When setting t to 0, the smooth Hinge loss is equivalent to the normal Hinge loss function. Both functions are plotted in figure 2.1. The practical implications of using the smooth approximation of loss functions is discussed in more detail in [Koh and Liang, 2017, chapter 4.3].

2.2.2 Non-convexity of the Loss

Another assumption we made, was that the empirical risk is strictly convex in θ. In practice this is generally not the case. Instead of a global minimum $\hat{\theta}$, we obtain an approximation $\tilde{\theta}$ by running some variant of gradient descent. Because $\hat{\theta} \neq \tilde{\theta}$ the Hessian $H_{\tilde{\theta}}$ can have negative eigenvalues. By constructing a convex quadratic approximation of the loss around $\tilde{\theta}$, we can negate this effect. This method is illustrated in more detail in the original paper ([Koh and Liang, 2017, chapter 4.2]).

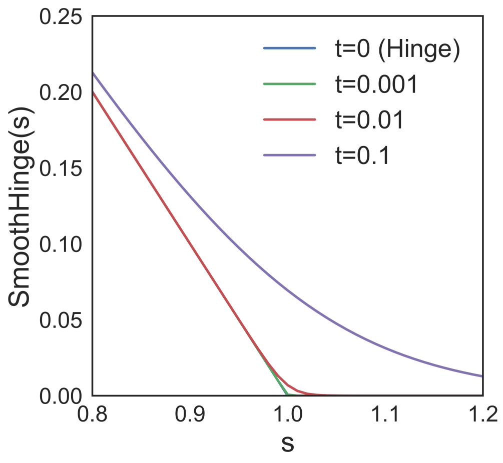

Figure 2.1: A plot of the Hinge and smooth Hinge loss functions using different values for smoothing parameter t

2.2.3 Computational complexity

When calculating the influence functions we face two complexity problems. First, the calculation and inversion of the Hessian matrix $H_{\hat{\theta}}$ has a complexity of $O(np^2 + p^3)$, with n being the number of training points and $\theta \in \mathbb{R}^p$. Second, in practice it is often necessary or useful to calculate $I_{up,loss}(z_i, z_{test})$ for all training points z_i.

The first problem can be handled by using implicit Hessian-vector products ([Pearlmutter, 1994]) to get an approximation of $s_{test} \overset{\text{def}}{=} H_{\hat{\theta}}^{-}1\nabla_\theta L(z_{test}, \hat{\theta})$. We use this intermediate step to calculate $I_{up,loss}(z_i, z_{test}) = -s_{test}\nabla_\theta L(z, \hat{\theta})$. We can precompute s_{test} and then calculate $-s_{test}\nabla_\theta L(z_i, \hat{\theta})$ for each z_i to solve the second problem. In the paper ([Koh and Liang, 2017]) the authors present two possible techniques to approximate s_{test}, namely Conjugate gradients by [Martens, 2010] and Stochastic estimation by [Agarwal et al., 2016].

3 Experiments

3.1 Validation using leave-one-out retraining

To show that the theory works, the authors conduct several experiments. Here they compare the computed influence to the actual change of prediction, when removing a training point and retraining the model. In theory the values for $-\frac{1}{n}I_{up,loss}(z, z_{test})$ (the effect of removing a training point) and $L(z_{test}, \hat{\theta}_{-z}) - L(z_{test}, \hat{\theta})$ (actually removing a training point and retraining the model) should be approximately equal.

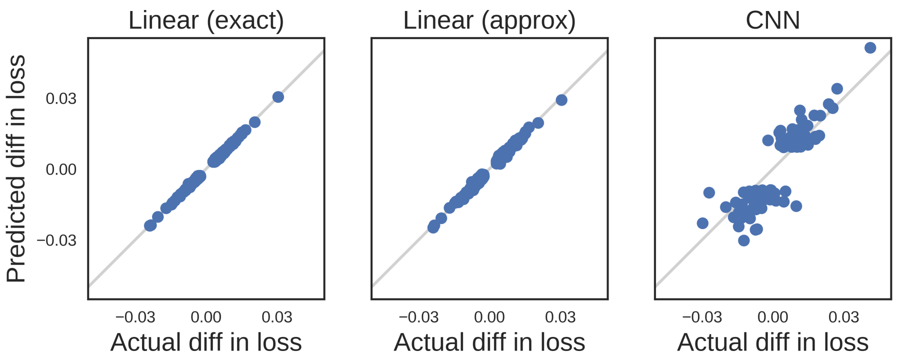

Figure 3.1: Influence vs. Leave-one-out error for different scenarios

In figure 3.1 the results of influence and leave-one-out retraining are shown. For each experiment the MNIST dataset was used and for z_{test} an arbitrary randomly misclassified test point was chosen. If the assumptions from the previous chapter are correct, we should see all data points on or close to the identity function $(f(x) = x)$.

The left plot shows the 500 training points with the highest absolute influence on z_{test} for a logistic regression model. In this experiment the Hessian-vector products were solved exactly using the Conjugate gradients method. Here the points are very close to the ideal result.

In the center plot the Hessian-vector products were approximated with Stochastic estimation. Otherwise, the experiment is identical to the first one. For the Stochastic approximation the parameters were set to: repeats $= 10$ and iterations $= 5{,}000$. This approximation saves computational resources, but still gives quiet accurate influence values.

The right plot shows the top most 100 influential points for z_{test}. In this case a convolutional neural network was the underlying model. Just like in the first experiment, the Hessian-vector products were solved exactly using the Conjugate gradients method. For the retraining, that was necessary to find the actual different in loss, starting at $\tilde{\theta}$ 30,000 steps were used to find $\tilde{\theta}_{-z}$.

We can see in all three figures that the calculated influence and the change after retraining are approximately equal. Therefore, the method works as expected.

3.2 Practical applications

In the original publication by Pang Wei Koh and Percy Liang, the authors presented a handful of applications using the derived method. For each application they also provide the results of a corresponding experiment.

Finding important training points With the presented approach, we can find the influence of a training point on a prediction. We can use this to find out how much the model relied on a training point for its prediction. This can then be used as a way to explain the prediction to a user.

Generating adversarial training points The goal of an adversarial training point attack is to generate visually-indistinguishable images in the training set that change the prediction for a test image.

Using the influence we can determine the most important training points for a given test point. When perturbing the training points for our attack, we only have to focus on the most influential ones. The results of the successful attack are detailed in [Koh and Liang, 2017, chapter 5.3].

Finding a domain mismatch The notion of influence can also be used to identify a possible domain mismatch between the training and test data sets. The authors of [Koh and Liang, 2017] achieve this by identifying the training points that have the highest influence for false predictions. They then compare them to the data from the test set.

Fixing mislabeled training points Furthermore, the approach can be applied to identify mislabeled training data. For this use-case the idea is to calculate the influence of training point z_i on itself, which is given by $I_{up,loss}(z_i, z_i)$. This gives us an approximation of the error on z_i of we remove said point from our training set. In a nutshell, this influence describes how unique a training point is. The points are the ordered by uniqueness and can then be checked manually. This reduces the time and cost of data cleansing.

4 Conclusion

In this report, we introduced an approach of explaining black-box models using the training data. This approach is chosen because the training data is the origin of the learned parameters for a machine learning model. Furthermore, we showed practical applications and verified the theory using experiments.

Unfortunately we could not reproduce the results that were conducted independently using a modified version of the source code published by the original authors. We used a smaller dataset and fewer computations when approximating the Hessian-vector products due to resource limitations. The covariance between predicted and actual difference in loss was only around 0.294 and is therefore much smaller than in the original paper.

One shortcoming of the presented method is that it can not detect global trends in the data. This is owed to the nature of the method. If a pattern is spread out evenly in the training data, there no longer exist a subset of points which can be considered special.

Another possible issue for the application of this method is linked to privacy concerns. In some cases the training data can not be or should not be made public. Let's assume we have a model that generates a treatment plan for a patient given some features (age, symptoms, medical history, etc.) of that patient. In this scenario the training data used, is historical patient data. When someone wants to know how the model came to its proposed treatment plan, we would output medical data of a stranger. This is not legal in most cases.

Part I

Appendix

A Lists

A.1 List of Figures

B Bibliography

P. Adler, C. Falk, S. A. Friedler, G. Rybeck, C. Scheidegger, B. Smith, and S. Venkatasubramanian. Auditing Black-box Models for Indirect Influence. *ArXiv e-prints*, February 2016.

Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization in linear time. *stat*, 1050:15, 2016.

R. D. Cook and S. Weisberg. Residuals and influence in regression. chapman and hall, new york — london 1982. viii, 229 pp., £ 12,-. *Biometrical Journal*, 27(1):80–80, 1982. doi: 10.1002/bimj.4710270110. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/bimj.4710270110`.

A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617, May 2016. doi: 10.1109/SP.2016.42.

European Union. General Data Protection Regulation. *Official Journal of the European Union*, pages 1–88, May 2016. URL `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC`.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, August 2017. PMLR.

Alice Kohn. An ai law firm wants to 'automate the entire legal world'. 2017. URL `https://futurism.com/an-ai-law-firm-wants-to-automate-the-entire-legal-world/`.

Taylor Kubota. Deep learning algorithm does as well as dermatologists in identifying skin cancer. 2017. URL `https://news.stanford.edu/2017/01/25/artificial-intelligence-used-identify-skin-cancer/`.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016. URL `http://arxiv.org/abs/1612.08220`.

John Markoff. Computer wins on 'jeopardy!': Trivial, it's not. 2011. URL `https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html`.

James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

Amershi Saleema, Chickering Max, Drucker Steven, Lee Bongshin, Simard Patrice, and Suh Jina. Modeltracker: Redesigning performance analysis tools for machine learning. April 2015. URL https://www.microsoft.com/en-us/research/publication/modeltracker-redesigning-performance-analysis-tools-for-machine-learning/.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016. URL http://arxiv.org/abs/1605.01713.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.