An introduction to influence functions for machine learning

Aishwarya Mandyam

August 8th, 2020

Influence functions are useful to determine the effect that certain training data samples have on the predictions that a model makes. This post discusses the intuition behind influence functions in relation to machine learning, and how the use of influence functions can explain where a model's fairly black-box predictions come from and how they can be attributed to dataset error (i.e. mislabeled data points) and outliers. All influential data samples are outliers, but not all outliers are influential. The use of influence functions can help distinguish between these two types of samples and learning about the effect of the input dataset. Before we dig into the intuition, let us set up our problem space.

1 Setting up the problem

Suppose we want to classify the content of two types of images: cats and dogs. Our dataset is a bivariate distribution (i.e. we have two variables and they follow different distributions). Let us say that our model classifies cats with a 60% true positive rate and classifies dogs with 99% true positive rate. Why does our model perform poorly on images of cats and is it because of our data? Essentially we want to understand the effect of our training samples on our model's parameters.
We can answer this question by posing a counterfactual question: how would our model's parameters change if we removed a certain training sample? The easiest way to measure this is to create two models. One is the original model, and the second is a version of the same model trained with a particular data point, z, removed. The difference between the parameters of these two models represents the influence of z.
However, it would be prohibitively slow for us to do this for every data sample in our dataset. Influence functions can help us estimate the changes in our model when certain training samples removed, without having to retrain the model every time.

2 Formalizing the problem

2.1 Empirical risk minimization

Let us formalize our problem using the notion of empirical risk minimization. In a supervised learning algorithm, we aim to learn a function $h : X \to Y$ which maps training examples, X to labels Y. Suppose we have n training examples of the format (x_i, y_i) for $1 \leq i \leq n$ where $x_i \in X$ and $y_i \in Y$. An ideal model h^* performs such that $h^*(x_i) \approx y_i$.
In supervised learning we also have a loss function, $L(h(x_i), y_i)$, which estimates how close a predicted label is to the actual label. For the purposes of this post, let us assume that this loss

function is differentiable (i.e. we can take the derivative at every point).
Given this loss function, we now want to estimate the risk associated with a given model $\hat{h}$. For the purposes of this application, risk, written as $R(h)$, is a statistical measure that quantifies the degree to which an estimate from a given model $\hat{h}$ is likely to be inaccurate. For example, a higher risk means that $\hat{h}$ is likely to output inaccurate predictions, and a lower risk means that $\hat{h}$ is likely to output more accurate predictions. Risk is defined as the expected value of loss for this model. This can be calculated using the integral w.r.t the probability density function of our dataset, $P(x, y)$.

$$R(h) = E[L(h(x), y)] = \int L(h(x), y) dP(x, y) \tag{1}$$

However, we don't know the prior or underlying distribution $P(x, y)$; in the example above, we don't know what the distribution of all cat and dog images look like. This means that it is not possible to calculate $R(h)$ directly. Instead, we can calculate empirical risk or R_{emp}, which is the average of the loss values for each sample in our dataset. This doesn't require an understanding of the true underlying joint distribution $P(x, y)$.

$$R_{emp}(h) = \frac{1}{n} \sum_{i=1} L(h(x_i), y_i) \tag{2}$$

If you are familiar with the use of loss functions in machine learning, you know that we want to minimize loss as our model trains. This is analogous to minimizing empirical risk. An ideal model, h^*, minimizes empirical risk.

$$h^* = \arg\min_{h \in H} R_{emp}(h) \tag{3}$$

2.2 Understanding the effect of training samples on our model

Our original goal was to determine how our model would be affected if we removed a particular training samples. When we think about what a given model $\hat{h}$ has learned, we are referring to the parameters, or weights, written as $\hat{\theta}$. If we removed sample k, (x_k, y_k), from our training set, the corresponding change in parameters would be $\hat{\theta}_{-k} - \hat{\theta}$.
This change in parameters is what will allow us to determine the influence of the particular training sample k. $\hat{\theta}_{-k}$ corresponds to the weights of the model $\hat{h}_{-k}$ which can be expressed using the notation from above as

$$\hat{h}_{-k} = \frac{1}{n} \sum_{i=1, i \neq k} L(h(x_i), y_i) \tag{4}$$

It is important to note that we need to set the model that we are investigating, $\hat{h}$, prior to understanding the influence of a particular training sample. This means that $\hat{h}$ has the same architecture as $\hat{h}_{-k}$, the only difference between them being that $\hat{h}_{-k}$ was trained without the sample k.
The change in parameters represented by $\hat{\theta}_{-k} - \hat{\theta}$ is equivalent to computing the parameters $\hat{\theta}$ except by downweighting the sample k. When we weight this sample, we essentially change the importance of it in our dataset. If we choose to weight it by $\frac{-1}{n}$, this is equivalent to removing it from our dataset.
As of now, we know which sample in our training set we want to weight, and we know the parameters of our model $\hat{h}$. This leads us perfectly into what an influence function can tell us.

3 What do influence functions tell us?

For the purposes of this set of notes, let us assume that influence functions are a black box that take in the parameters of a model and the sample to weight. We will weight this sample by $-\frac{1}{n}$ which is equivalent to removing that sample from our training set. An influence function, I, can give us a tractable approximation ofthe following change in parameters without retraining the model:

$$\hat{\theta}_{-k} - \hat{\theta} \approx -\frac{1}{n} * I(\theta, k)$$

Notice that we multiply the value that the influence function gives us by the quantity to weight the sample k. The output of an influence function is a value that can tell us whether the input data sample was beneficial or detrimental to the performance of this model. But how does an influence function look? The following is the form of the function:

$$-H_{\hat{\theta}}^{-1} \delta_{\theta} L(k, \hat{\theta})$$

Note that this function uses $\hat{\theta}$, which are the parameters of our model $\hat{h}$, and k, which is the data sample we want to weight. It also references the loss function, L. This intuitively makes sense because the loss can tell us which samples change the parameters of the model the most. The samples that change the parameters of the model the most are influential, and taking a look at these influential samples can help us understand why our model performs the way that it does.
Let us take it back to our initial problem, with classifying cats and dogs. Recall that our goal is to classify images of cats and dogs with as high accuracy as possible, but that we misclassify cats 40% of the time, even though our classifier is good at recognizing dogs. Let's use influence functions to help us understand why our model performs in this way.

The first step is to use the influence function with a weight of $-\frac{1}{n}$ for each of the samples in our dataset; using this weight is analogous to taking each sample out of our dataset and retraining a model without it. For each sample, we get an associated influence function score. The higher the score, the more the corresponding sample influences our model. Suppose we do this for every sample and find that a portion of the images classified as cats have high influence scores. Upon closer inspection, we realize that these images contain wild cats, and not house cats. When we analyze the outputs of our model, we also notice that many of these influential samples were misclassified as dogs. We can now explain the performance of our classifier; it was confused because of outlier images in the training dataset.

4 Citations

https://arxiv.org/pdf/1703.04730.pdf
http://www.stat.ucla.edu/ nchristo/statistics100C/1268249.pdf
https://mlexplained.com/2018/06/01/paper-dissected-understanding-black-box-predictions-via-influence-functions/
TODO: https://arxiv.org/pdf/2005.06676.pdf