Google Research

Scaling Up Influence Functions

Andrea Schioppa, Artem Sokolov, Polina Zablotskaia, David Vilar {arischioppa, artemsok, polinaz, vilar}@google.com

Introduction

Influence Functions

Questions that Influence Methods address:

- Which training examples are **most responsible** for the prediction at a given test point?
- Which training examples **advocate / oppose** the prediction at a test point?
- Which part of the training **data is corrupt** / of lower quality?
- Which training examples are **easier / harder** to memorize?

The formal answer [1]:

$$\mathcal{I}_{\text{true}}(x, z) = L(z|\Theta : x \notin \mathcal{D}) - L(z|\Theta : x \in \mathcal{D})$$

x is a training example; z is a test example; Θ is the parameters of a model; L is the loss function used in training; D is the corpus of all training examples.

However removing each training example and retraining the model is not feasible!

- Instead [1] compute the effect of the perturbation on the current optimum:

$$\mathcal{I}_H(x, z) = \langle \nabla_\Theta L(z), H^{-1} \nabla_\Theta L(x) \rangle$$

This is faster, but for DL models still not feasible due to the inverse of the Hessian $H = \nabla^2_\Theta L$

- [2] (LISSA) suggest iterative inversion of Hessian with:

$$H_r^{-1} v = v + (I - H) H_{r-1}^{-1} v$$

Hessian vector products can be estimated in $O(b \cdot p)$-time, where $b = |D|$, $p = |\Theta|$. The total $O(b \cdot p \cdot r)$-time complexity will be incurred at every x in whose influence on z we are interested.

Method is slow: either small models, reduction of parameters to the last layer, small datasets or a subset of points.

Method

Scalable Influence Functions

$O(p)$ complexity is the major bottleneck for efficient implementation of IFs.

We can use Random Projection $G \in \mathbb{R}^{p \times \tilde{p}}$ to create a subspace of size $\tilde{p}$ such that $\mathbb{E}[G^T G] = I$

$$\mathcal{I}_{\tilde{H}}(x, z) = \langle G \nabla_\Theta L(z), \tilde{H}^{-1} G \nabla_\Theta L(x) \rangle$$

$$\tilde{H} = G \cdot H \cdot G^T$$

However the image of G is not H-invariant. To solve this we propose using the standard the **Arnoldi iteration** [3] technique of building an approximately H-invariant subspace and constructing the n-th order Krylov subspace:

$$K_n(H; v) = \text{Span}\{v, Hv, H^2 v, \dots, H^n v\}$$

The procedure additionally builds an orthonormal basis for Krylov subspace so that the diagonalization of the restriction $\tilde{H}$ to H yields an approximation of the largest (in absolute value) eigenvalues of H and of the corresponding eigenvectors.

The matrix H gets replaced with now diagonal $\tilde{H}$ simplifying the matrix inversion appearing in the definition of $\mathcal{I}_H$ and dispensing with the expensive LISSA [2] procedure.

Algorithm 1 `Arnoldi`

```
procedure ARNOLDI(v, n)
▷ Build orthonormal basis for the Krylov subspaces K_n.
    w_0 ← v / ||v||_2
    A_{l,m} ← 0 for 0 ≤ l ≤ n and 0 ≤ m < n
    for i ← 1, n do
        w_i ← H · w_{i-1}                          ▷ HVP in fwd-over-rev mode
        Set A_{i,j} = ⟨w_i, w_j⟩ for j < i
        w_i ← w_i − Σ_{j<i} A_{i,j} w_j             ▷ Orthogonalize
        A_{i+1,i} ← ||w_i||_2, w_i ← w_i / ||w_i||_2
    return: A, {w_i}
procedure DISTILL(A, {w_i}, p̃)
▷ Distill A, {w_i} to its top-p̃ eigenvalues.
    Discard the last row of A and the last w_n
    Obtain A's eigenvalues {λ_i} and eigenvectors {e_i}
    Set {λ'_i} to the p̃-largest (in absolute value) of {λ_i}
    Set {e'_i} to the corresponding eigenvectors
    Set G to the projection onto the spans {e'_i} in {w_i}-basis
    return: {λ'_i}, G
procedure INFLUENCE(x, z, n, p̃)
▷ Influence of x on z with n iterations and top-p̃ eigenvalues.
    v ← N(0, 1)                      ⎫
    A, {w_i} = ARNOLDI(v, n)         ⎬ ▷ Executed once and cached
    {λ'_i}, G = DISTILL(A, {w_i}, p̃) ⎭
    g_x ← G · ∇_Θ L(x)               ▷ fwd JVP for x over G-rows
    g_z ← G · ∇_Θ L(z)               ▷ fwd JVP for z over G-rows
    g_x ← g_x / {λ'_i}               ▷ Multiply with diagonalized H̃^{-1}
    return: ⟨g_x, g_z⟩
```

Results

MNIST Benchmark

Corrupted MNIST -- Small Model (Figure 1 & 2)

- [2] considers only a 10% subset of data and a CNN with 3k parameters
- We evaluate ability to recall synthetically mislabeled examples [4].
- **IFs outperform gradient methods and RandProj does not capture correct eigenvalues.**

Corrupted MNIST -- Large Model (Table 1)

- CNN with 800k parameters..
- **LISSA is prohibitively slow; Arnoldi & RandProj scale well; no advantage in using curvature information.**

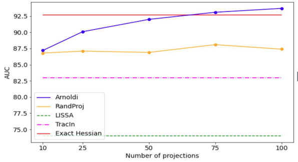

Figure 1

Method	$\tilde{p}$	T, secs	AUC	AP
`LISSA`, $r = 10$	-	4900	98.9	95.0
`LISSA`, $r = 100$	-	32300	98.8	94.8
`TracIn` [1]	-	5	98.7	94.0
`TracIn` [10]	-	42	**99.7**	**98.7**
`RandProj`	10	0.2	97.2	87.7
`RandProj`	100	1.9	98.6	93.9
`Arnoldi`	10	0.2	95.0	84.0
`Arnoldi`	100	1.9	98.2	92.9

Table 1

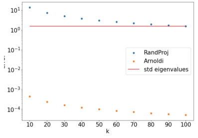

Figure 2

Machine Translation

Synthetic Noise (Table 2)

- A model on WMT17 German -> English.
- Noisy data where references where permuted (with 6% probability).
- **Subsetting the layers leads to worse performance.**

Natural Noise [Quality] (Table 3)

- Bad quality Paracrawl (100M pairs, ~1B words) from WMT'18 data cleaning task..
- **Applying Arnoldi on top of Microsoft winning submission Scores to get 50% of the selected clean data gives 37.2.**
- **Method has high throughput: 2.2M examples per hour using TPUv2 (32 cores)**

Layers	Method	$\tilde{p}$	AUC	AP
all layers	`RandProj`	10	85.6	31.3
	`RandProj`	20	85.2	28.0
	`Arnoldi`	10	92.0	47.8
	`Arnoldi`	20	**93.8**	**54.4**
last 3 decoder layers	`RandProj`	10	80.6	23.5
	`RandProj`	20	83.0	25.8
	`Arnoldi`	10	82.0	28.1
	`Arnoldi`	20	**83.7**	**28.5**

Table 2

Method	% selected	BLEU@10k	BLEU@200k
None	100	9.9	17.8
Pre-cleaning	14	11.7	22.6
`RandProj`	1	7.7	8.7
`Arnoldi`	1	17.8	19.6
`RandProj`	5	18.7	27.9
`Arnoldi`	5	24.0	30.3
`RandProj`	10	21.3	28.6
`Arnoldi`	10	**25.0**	**30.8**

Table 3

Conclusion

- We proposed a new way of calculating influence scores of [2] for large DNNs by approximate diagonalization of their Hessians and avoiding re-estimating them on every training example.
- We demonstrated finding influential or noisy examples in datasets of up to 100M training examples and models with up to 300M parameters.
- We showed IFs can be either superior or match random projection and gradient-based approaches, when measured on the benchmark of retrieving synthetically mislabeled data with self-influence scores.
- We will make our code publicly available.

References

[1] Cook, R. D.; and Weisberg, S. 1980. Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression. Technometrics 22(4): 495–508.

[2] Koh, P. W.; and Liang, P. 2017. Understanding Black-box Predictions via Influence Functions. In ICML.

[3] Arnoldi, W. E. 1951. The principle of minimized iterations in the solution of the matrix eigenvalue problem. Quarterly of Applied Mathematics 9(1): 17–29.

[4] Pruthi, G.; Liu, F.; Sundararajan, M.; and Kale, S. 2020. Estimating Training Data Influence by Tracing Gradient Descent. In NeurIPS.