Weekly Presentation
DeltaGrad: Rapid retraining of machine learning models

Yinjun Wu Edgar Dobriban Susan B Davidson

September 30, 2020

Overview

Motivation

Retaining Problem

Regular Pipeline:

1. Train a ML model from data using a learning algorithm
2. Small change in training data occurs (deletions or additions)
3. Retrain ML model from scratch

Retaining Problem

Regular Pipeline:

1. Train a ML model from data using a learning algorithm
2. Small change in training data occurs (deletions or additions)
3. Retrain ML model from scratch

- Computationally expensive process

Retaining Problem

Regular Pipeline:

1. Train a ML model from data using a learning algorithm
2. Small change in training data occurs (deletions or additions)
3. Retrain ML model from scratch

- Computationally expensive process
- Throws away useful computations from initial training

Retaining Problem

Regular Pipeline:

1. Train a ML model from data using a learning algorithm
2. Small change in training data occurs (deletions or additions)
3. Retrain ML model from scratch

- Computationally expensive process
- Throws away useful computations from initial training

Research Question

Can we retrain models in an efficient manner?

Potential Applications

- **GDPR**: Deletion of private information from public datasets

Potential Applications

- **GDPR**: Deletion of private information from public datasets
- **Continuous Model Updating**: Handle additions, deletions and changes of training samples

Potential Applications

- **GDPR**: Deletion of private information from public datasets
- **Continuous Model Updating**: Handle additions, deletions and changes of training samples
- **Data Valuation**: *Leave One Out* tests to find important training samples

Potential Applications

- **GDPR**: Deletion of private information from public datasets
- **Continuous Model Updating**: Handle additions, deletions and changes of training samples
- **Data Valuation**: *Leave One Out* tests to find important training samples
- **Bias Reduction**: Speeds up jackknife resampling that requires retrained model parameters

Related Work

Prior Work

- Prior work for specialized problems and ML models, usually for deletion
  - Provenane Based deletions for linear and logistic regression [WTD20]
  - Newton step and noise for *certified data removal* [GGHv20]
  - K-means clustering [GGVZ19]

DeltaGrad

Gradient Descent

- Objective function

$$F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\mathbf{w})$$

- Stochastic Gradient Descent update rule, $\mathcal{B}_t$ is randomly sampled mini-batch of size B

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{\eta_t}{B}\sum_{i\in\mathcal{B}_t} \nabla F_i\left(\mathbf{w}_t\right)$$

- Full-batch gradient descent (GD) is on entire data

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{\eta_t}{n}\sum_{i=1}^{n} \nabla F_i\left(\mathbf{w}_t\right)$$

Removal of data

- After training, $R = \{i_1, i_2, \ldots, i_r\}$ is removed, where $r \ll n$
- Naive retraining is applying GD over remaining samples, $\mathbf{w}^U$ is resulting parameters

$$\mathbf{w}^U{}_{t+1} \leftarrow \mathbf{w}^U{}_t - \frac{\eta_t}{n-r} \sum_{i \notin R} \nabla F_i \left(\mathbf{w}^U{}_t \right) \qquad (1)$$

- The explicit gradient computation $\sum_{i \notin R} \nabla F_i \left(\mathbf{w}^U{}_t \right)$ is expensive
- Instead rewrite (1) as follows

$$\mathbf{w}^U{}_{t+1} = \mathbf{w}^U{}_t - \frac{\eta_t}{n-r} \left[\sum_{i=1}^{n} \nabla F_i \left(\mathbf{w}^U{}_t \right) - \sum_{i \in R} \nabla F_i \left(\mathbf{w}^U{}_t \right) \right]. \qquad (2)$$

- $\sum_{i \in R} \nabla F_i \left(\mathbf{w}^U{}_t \right)$ is cheaper to compute

Etymology

- After a small change to the data we need to redo the SGD computations
- We can achieve this by understanding the *delta* of the Gradient Descent

$$n\nabla F(\mathbf{w}) = \sum_{i=1}^{n} \nabla F_i(\mathbf{w}_t) \quad \& \quad n\nabla F(\mathbf{w}^U) = \sum_{i=1}^{n} \nabla F_i\left({\mathbf{w}^U}_t\right)$$

- Hence, the approach is called *DeltaGrad*

Aprroximating $\nabla F(\mathbf{w}^U)$

- $\mathbf{w}_0, \ldots, \mathbf{w}_t$ and $\nabla F(\mathbf{w}_0), \ldots, \nabla F(\mathbf{w}_t)$ are cached from training on initial dataset
- By Cauchy mean-value theorem[1]

$$\nabla F(\mathbf{w}^U{}_t) - \nabla F(\mathbf{w}_t) = \mathbf{H}_t \cdot (\mathbf{w}^U{}_t - \mathbf{w}_t)$$

Where $\mathbf{H}_t = \int_0^1 \mathbf{H}(\mathbf{w}_t + x(\mathbf{w}^U{}_t - \mathbf{w}_t))dx$ is the integrated hessian

- This requires a hessian $\mathbf{H}_t$ at each step, which is expensive to maintain and evaluate
- Leverage classical L-BFGS algorithm to approximate $\mathbf{H}_t$

Aprroximating $\nabla F(\mathbf{w}^U)$

- $\mathbf{w}_0, \ldots, \mathbf{w}_t$ and $\nabla F(\mathbf{w}_0), \ldots, \nabla F(\mathbf{w}_t)$ are cached from training on initial dataset
- By Cauchy mean-value theorem[1]

$$\nabla F(\mathbf{w}^U{}_t) - \nabla F(\mathbf{w}_t) = \mathbf{H}_t \cdot (\mathbf{w}^U{}_t - \mathbf{w}_t)$$

  Where $\mathbf{H}_t = \int_0^1 \mathbf{H}(\mathbf{w}_t + x(\mathbf{w}^U{}_t - \mathbf{w}_t))dx$ is the integrated hessian
- This requires a hessian $\mathbf{H}_t$ at each step, which is expensive to maintain and evaluate
- Leverage classical L-BFGS algorithm to approximate $\mathbf{H}_t$

[1] Seems to be a consequence of Fundamental theory of Calculus and mean-value theorem

Review of L-BFGS

- Traditional L-BFGS updates gradients using

$$\nabla F(\mathbf{w}_{t+1}) - \nabla F(\mathbf{w}_t) = \mathbf{B}_t \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t)$$

Where, $\mathbf{B}_t$ is the approximation of the hessian

Traditional L-BFGS

$$\begin{aligned}
&\nabla F\left(\mathbf{w}_{t+1}\right) - \nabla F\left(\mathbf{w}_t\right) \approx \mathrm{B}_t\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right) \\
&\mathrm{B}_t \approx \mathrm{H}_t \\
&= \int_0^1 \mathrm{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx \\
&\mathbf{s}_t = \mathbf{w}_{t+1} - \mathbf{w}_t \\
&\mathbf{y}_t = \nabla F\left(\mathbf{w}_{t+1}\right) - \nabla F\left(\mathbf{w}_t\right)
\end{aligned}$$

L-BFGS for approximating $\nabla F(\mathbf{w}^U)$

$$\begin{aligned}
&\nabla F\left({\mathbf{w}^U}_t\right) - \nabla F\left(\mathbf{w}_t\right) \approx \mathrm{B}_t\left({\mathbf{w}^U}_t - \mathbf{w}_t\right) \\
&\mathrm{B}_t \approx \mathrm{H}_t \\
&= \int_0^1 \mathrm{H}\left(\mathbf{w}_t + x\left({\mathbf{w}^U}_t - \mathbf{w}_t\right)\right) dx \\
&\mathbf{s}_t = {\mathbf{w}^U}_t - \mathbf{w}_t \\
&\mathbf{y}_t = \nabla F\left({\mathbf{w}^U}_t\right) - \nabla F\left(\mathbf{w}_t\right)
\end{aligned}$$

Using L-BFGS

- Maintain m historical observations of $\mathbf{Y} = (\mathbf{y}_t, \mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-m})$ and $\mathbf{S} = (\mathbf{s}_t, \mathbf{s}_{t-1}, \dots, \mathbf{s}_{t-m})$
- Let g be a function defined by L-BFGS, then we can approximate $\mathbf{B}_t \cdot \mathbf{v}$ using
$$g(\mathbf{Y}, \mathbf{S}, \mathbf{v})$$
Where, $\mathbf{v}$ is an arbitrary vector.
- Therefore,
$$\mathbf{B}_t \cdot ({\mathbf{w}^U}_t - \mathbf{w}_t) = g(\mathbf{Y}, \mathbf{S}, {\mathbf{w}^U}_t - \mathbf{w}_t)$$
- Hence we obtain the approximation as
$$\nabla F({\mathbf{w}^U}_t) \approx \nabla F(\mathbf{w}_t) + \mathbf{B}_t \cdot ({\mathbf{w}^U}_t - \mathbf{w}_t)$$

Rewriting

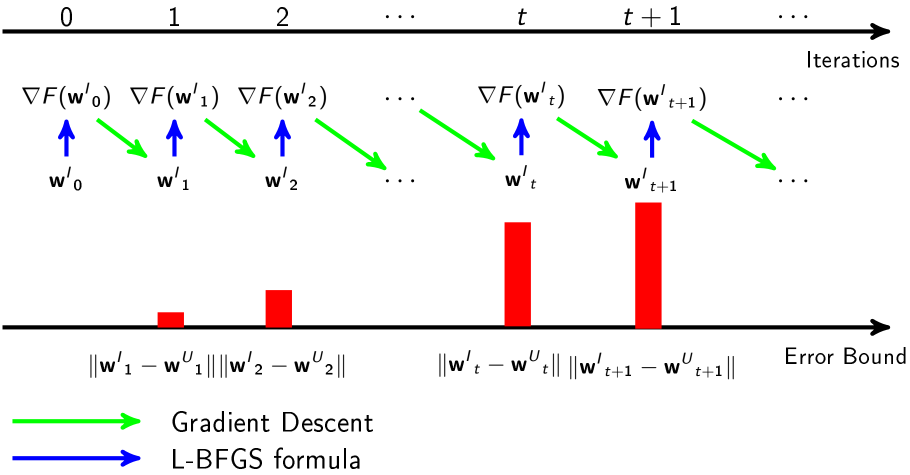
- Denoting $\mathbf{w}^I$ as the approximate $\mathbf{w}^U$ we have

$$\nabla F({\mathbf{w}^I}_t) \approx \nabla F(\mathbf{w}_t) + \mathbf{B}_t \cdot ({\mathbf{w}^I}_t - \mathbf{w}_t).$$

- replacing in (2)

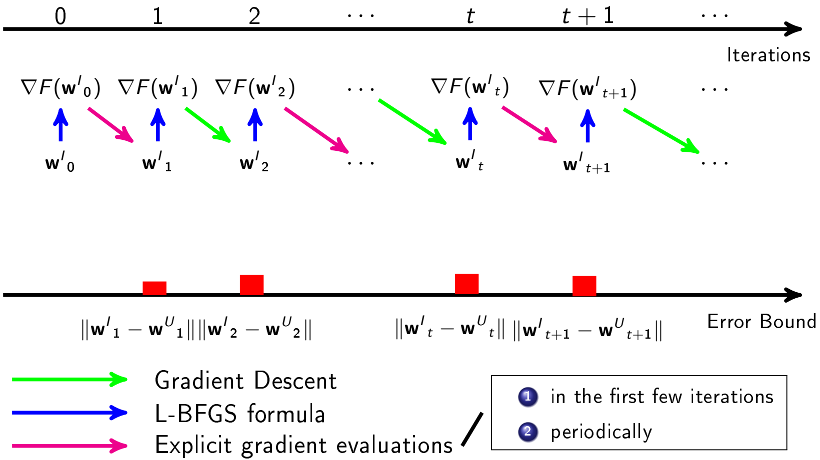
$${\mathbf{w}^I}_{t+1} = {\mathbf{w}^I}_t - \frac{\eta_t}{n-r}\left\{ n[\mathbf{B}_t({\mathbf{w}^I}_t - \mathbf{w}_t) + \nabla F(\mathbf{w}_t)] - \sum_{i \in R} \nabla F({\mathbf{w}^I}_t) \right\}$$

Problem with Error Bound



Controlling the Errors

- Do explicit evaluations for j_0 "burn-in" iterations and then periodically every T_0 iterations



Benefit of DeltaGrad

- DeltaGrad can be extended to when r samples are added rather than deleted
- Change the $+$ to minus in the update formula to get

$$\mathbf{w}^I{}_{t+1} = \mathbf{w}^I{}_t - \frac{\eta_t}{n+r}\left\{ n[\mathbf{B}_t(\mathbf{w}^I{}_t - \mathbf{w}_t) + \nabla F(\mathbf{w}_t)] + \sum_{i\in R} \nabla F(\mathbf{w}^I{}_t) \right\}$$

- Here $\sum_{i\in R} \nabla F(\mathbf{w}^I{}_t)$ is the gradient of the added r samples

Algorithm 1: DeltaGrad

Input : The full training set $(\mathbf{X}, \mathbf{Y})$, model parameters cached during the training phase over the full training samples $\{\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_t\}$ and corresponding gradients $\{\nabla F(\mathbf{w}_0), \nabla F(\mathbf{w}_1), \dots, \nabla F(\mathbf{w}_t)\}$, the indices of the removed training samples R, period T_0, total iteration number T, history size m, "burn-in" iteration number j_0, learning rate η_t

Output: Updated model parameter $\mathbf{w}^I{}_t$

```
Initialize w^I_0 ← w_0
Initialize an array ΔG = []
Initialize an array ΔW = []
for t = 0; t < T; t++ do
    if [((t − j_0) mod T_0) == 0] or t ≤ j_0 then
        compute ∇F(w^I_t) exactly
        compute ∇F(w^I_t) − ∇F(w_t) based on the cached gradient ∇F(w_t)
        set ΔG[k] = ∇F(w^I_t) − ∇F(w_t)
        set ΔW[k] = w^I_t − w_t, based on the cached parameters w_t
        k ← k + 1
        compute w^I_{t+1} by using exact GD update (equation (1))
    else
        Pass ΔW[−m :], ΔG[−m :], the last m elements in ΔW and ΔG,
          which are from the j_1^th, j_2^th, ..., j_m^th iterations where j_1 < j_2 < ··· < j_m
          depend on t, v = w^I_t − w_t, and the history size m, to the L-BFGFS
          Algorithm to get the approximation of H(w_t)v, i.e., B_{j_m} v
        Approximate ∇F(w^I_t) = ∇F(w_t) + B_{j_m}(w^I_t − w_t)
        Compute w^I_{t+1} by using the "leave-r-out" gradient formula, based on
          the approximated ∇F(w^I_t)
    end
end
return w^I_t
```

Theoretical Results

Results for GD

Theorem (Bound between true and incrementally updated iterates)

Assuming $F(\mathbf{w})$ is strongly convex, for large enough iterations t, the result ${\mathbf{w}^I}_t$ of *DeltaGrad* approximates the correct iteration values ${\mathbf{w}^U}_t$ at the rate of

$$\|{\mathbf{w}^U}_t - {\mathbf{w}^I}_t\| = o\left(\frac{r}{n}\right)$$

So $\|{\mathbf{w}^U}_t - {\mathbf{w}^I}_t\|$ is of a lower order than r/n. r/n is the "baseline error rate" of the original weights $\mathbf{w}_t$, i.e., $\|\mathbf{w}_t - {\mathbf{w}^I}_t\| = o(\frac{r}{n})$

▸ architecture of proof

Results for SGD

Theorem(Bound between true and incrementally updated iterates in SGD)

Assuming $F(\mathbf{w})$ is strongly convex, for large enough iterations t and mini-batch size B, the result ${\mathbf{w}^I}_t$ of *DeltaGrad* approximates the correct iteration values ${\mathbf{w}^U}_t$ at the rate of

$$\|{\mathbf{w}^U}_t - {\mathbf{w}^I}_t\| = o\left(\frac{r}{n} + \frac{1}{B^{\frac{1}{4}}}\right)$$

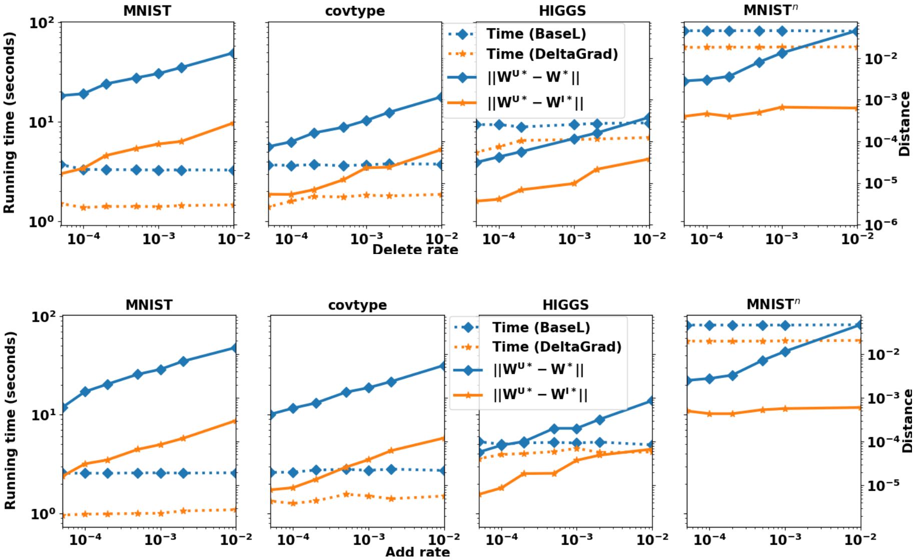
with high probability

Experimental Results

Setup

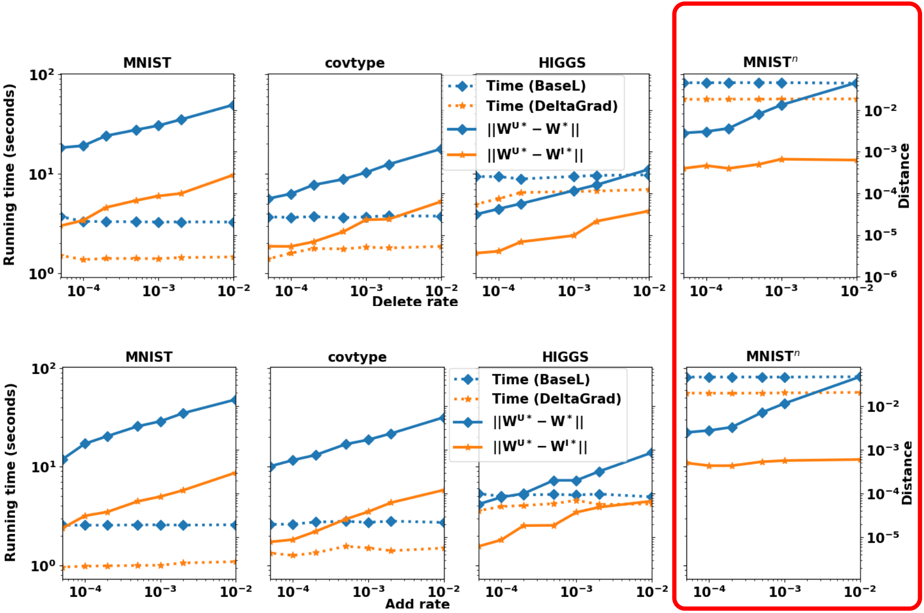
- **Datasets**: MNIST, RCV1, HIGGS
- **Model**: Logistic regression with L2 regularization
- **Baseline**: Naive retraining (BaseL)
- **Hyperparameters**: $j_0 = \{10, 10, 300\}$ and $T_0 = \{5, 10, 3\}$

Results



MNIST
covtype
HIGGS
$MNIST^n$
Running time (seconds)
Distance
Delete rate
Time (BaseL)
Time (DeltaGrad)
$||W^{U*} - W^*||$
$||W^{U*} - W^{I*}||$
10^2
10^1
10^0
10^{-4}
10^{-3}
10^{-2}
10^{-5}
10^{-6}
MNIST
covtype
HIGGS
$MNIST^n$
Running time (seconds)
Distance
Add rate
Time (BaseL)
Time (DeltaGrad)
$||W^{U*} - W^*||$
$||W^{U*} - W^{I*}||$

Results


MNIST
covtype
HIGGS
MNIST^n
Running time (seconds)
Distance
Delete rate
Add rate
Time (BaseL)
Time (DeltaGrad)
$||W^{U*} - W^*||$
$||W^{U*} - W^{I*}||$
10^{-4}
10^{-3}
10^{-2}
10^{0}
10^{1}
10^{2}
10^{-5}
10^{-6}

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten.
Certified Data Removal from Machine Learning Models.
arXiv:1911.03030 [cs, stat], August 2020.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou.
Making AI Forget You: Data Deletion in Machine Learning.
In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3518–3531. Curran Associates, Inc., 2019.

Yinjun Wu, Val Tannen, and Susan B. Davidson.
PrIU: A Provenance-Based Approach for Incrementally Updating Regression Models.
In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 447–462, Portland OR USA, June 2020. ACM.

Large Deletions

Large Deletions

Online Additions/Deletions

Architecture of Proof