# Weekly Presentation
# DeltaGrad: Rapid retraining of machine learning models

Yinjun Wu Edgar Dobriban Susan B Davidson

September 29, 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 [**?**]
  - Newton step and noise for *certified data removal* [**?**]
  - K-means clustering [**?**]

# 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

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

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

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

- $\sum_{i \in R} \nabla F_i \left( \mathbf{w}^U{}_t \right)$ 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 $\mathbf{Y} = (\mathbf{y}_t, \mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m})$ and $\mathbf{S} = (\mathbf{s}_t, \mathbf{s}_{t-1}, \ldots, \mathbf{s}_{t-m})$
- 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

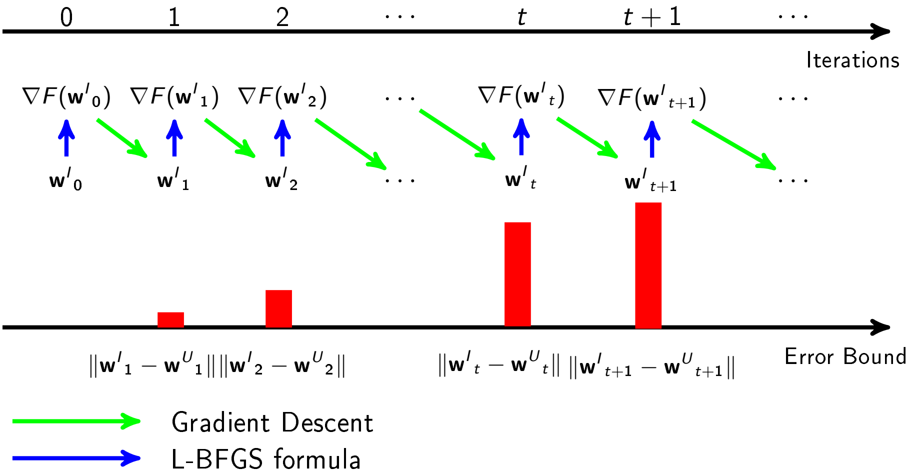
- Denoting $\mathbf{w}^I$ as the approximate $\mathbf{w}^U$ we have

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

- replacing in (**??**)

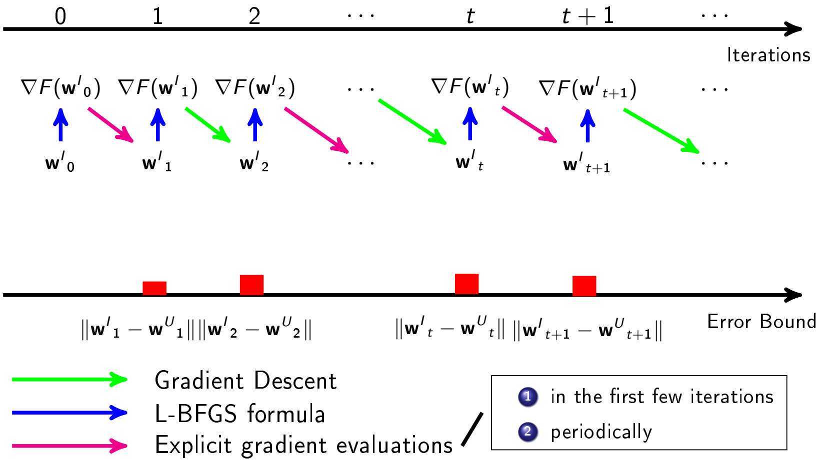
$${\mathbf{w}^I}_{t+1} = {\mathbf{w}^I}_t - \frac{\eta_t}{n-r}\left\{ n[\mathbf{B}_t({\mathbf{w}^I}_t - \mathbf{w}_t) + \nabla F(\mathbf{w}_t)] - \sum_{i \in R} \nabla F({\mathbf{w}^I}_t) \right\}$$

# Problem with Error Bound



# Controlling the Errors

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




/

1. in the first few iterations
2. periodically

# 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

[H]

**return** $\mathbf{w}^I{}_t$

# Theoretical Results

# Experimental Results

# Large Deletions

# Large Deletions