DeltaGrad: Rapid retraining of machine learning models

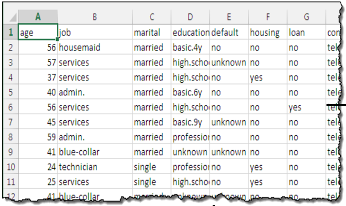
Yinjun Wu[1] Edgar Dobriban[2] Susan B. Davidson [1]

[1]Department of Computer and Information Science
University of Pennsylvania

[2]Department of Statistics
University of Pennsylvania

ICML, 2020

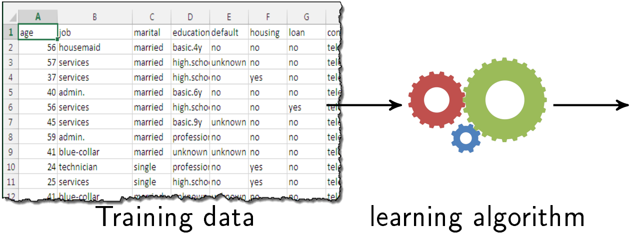
What we are trying to solve



	A	B	C	D	E	F	G	
1	age	job	marital	education	default	housing	loan	con
2	56	housemaid	married	basic.4y	no	no	no	tel
3	57	services	married	high.scho	unknown	no	no	tel
4	37	services	married	high.scho	no	yes	no	tel
5	40	admin.	married	basic.6y	no	no	no	tel
6	56	services	married	high.scho	no	no	yes	tel
7	45	services	married	basic.9y	unknown	no	no	tel
8	59	admin.	married	professio	no	no	no	tel
9	41	blue-collar	married	unknown	unknown	no	no	tel
10	24	technician	single	professio	no	yes	no	tel
11	25	services	single	high.scho	no	yes	no	tel
12	41	blue-collar	[illegible]	[illegible]	[illegible]	no	no	tel

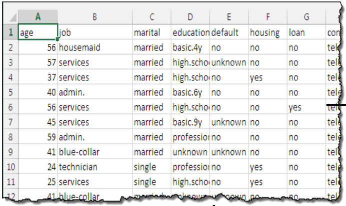
Training data

What we are trying to solve



	A	B	C	D	E	F	G	
1	age	job	marital	education	default	housing	loan	con
2	56	housemaid	married	basic.4y	no	no	no	tel
3	57	services	married	high.scho	unknown	no	no	tel
4	37	services	married	high.scho	no	yes	no	tel
5	40	admin.	married	basic.6y	no	no	no	tel
6	56	services	married	high.scho	no	no	yes	tel
7	45	services	married	basic.9y	unknown	no	no	tel
8	59	admin.	married	professio	no	no	no	tel
9	41	blue-collar	married	unknown	unknown	no	no	tel
10	24	technician	single	professio	no	yes	no	tel
11	25	services	single	high.scho	no	yes	no	tel
12	41	blue-collar						tel

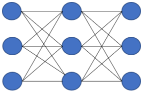
What we are trying to solve



	A	B	C	D	E	F	G	
1	age	job	marital	education	default	housing	loan	con
2	56	housemaid	married	basic.4y	no	no	no	tel
3	57	services	married	high.scho	unknown	no	no	tel
4	37	services	married	high.scho	no	yes	no	tel
5	40	admin.	married	basic.6y	no	no	no	tel
6	56	services	married	high.scho	no	no	yes	tel
7	45	services	married	basic.9y	unknown	no	no	tel
8	59	admin.	married	professio	no	no	no	tel
9	41	blue-collar	married	unknown	unknown	no	no	tel
10	24	technician	single	professio	no	yes	no	tel
11	25	services	single	high.scho	no	yes	no	tel
12	41	blue-collar	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	tel

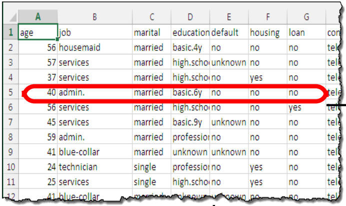
Training data

learning algorithm

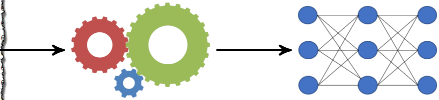


ML models

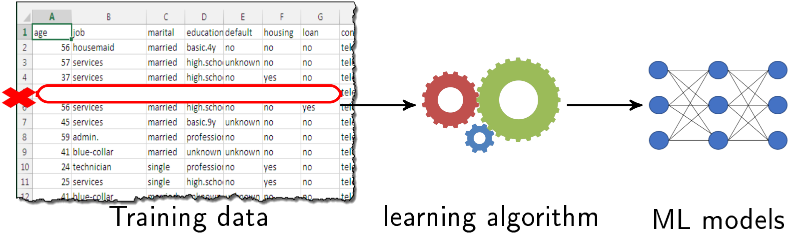
What we are trying to solve



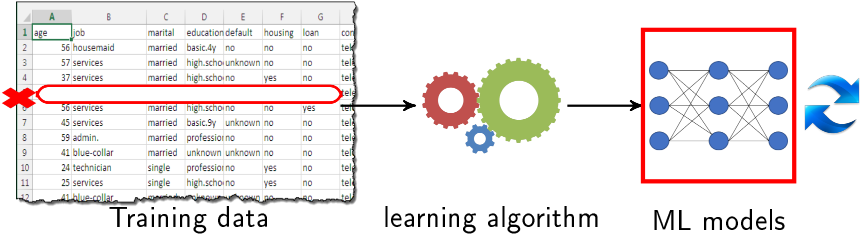
	A	B	C	D	E	F	G	
1	age	job	marital	education	default	housing	loan	con
2	56	housemaid	married	basic.4y	no	no	no	tel
3	57	services	married	high.scho	unknown	no	no	tel
4	37	services	married	high.scho	no	yes	no	tel
5	40	admin.	married	basic.6y	no	no	no	tele
6	56	services	married	high.scho	no	no	yes	tel
7	45	services	married	basic.9y	unknown	no	no	tel
8	59	admin.	married	professio	no	no	no	tel
9	41	blue-collar	married	unknown	unknown	no	no	tel
10	24	technician	single	professio	no	yes	no	tel
11	25	services	single	high.scho	no	yes	no	tel
12	41	blue-collar	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	tel



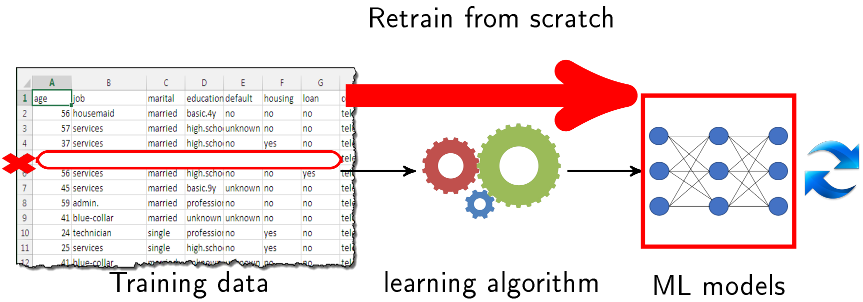
What we are trying to solve



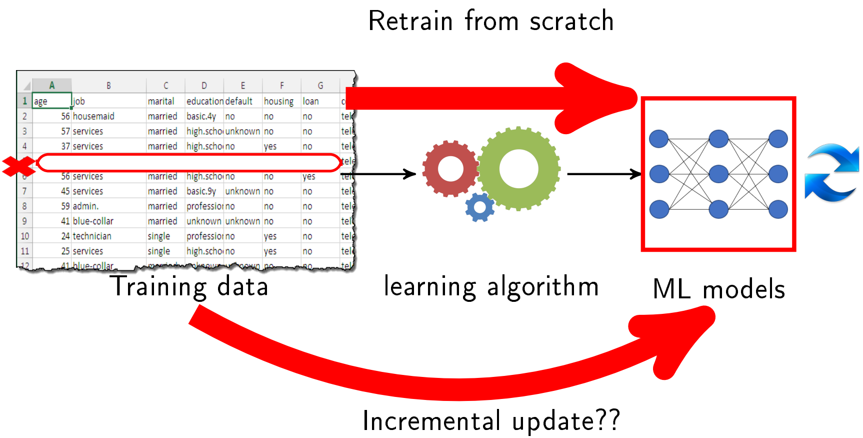
What we are trying to solve



What we are trying to solve



What we are trying to solve

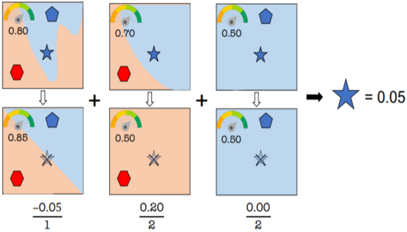


Applications

GPDR issues, Privacy

Applications

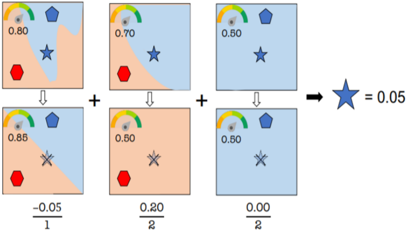
GPDR issues, Privacy



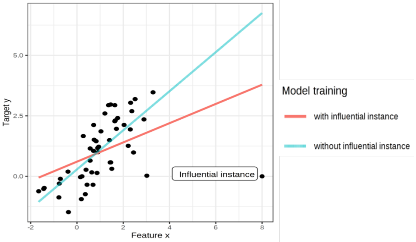
Data valuation, Shapley value [GZ19]

Applications

GPDR issues, Privacy



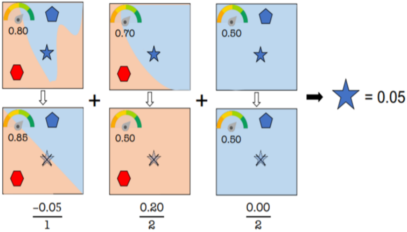
Data valuation, Shapley value [GZ19]



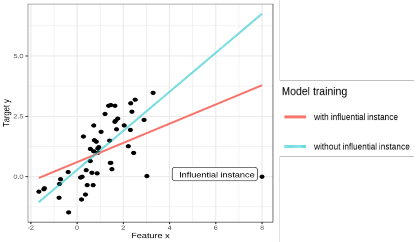
Deletion diagnostics, Robustness

Applications

GPDR issues, Privacy



Data valuation, Shapley value [GZ19]

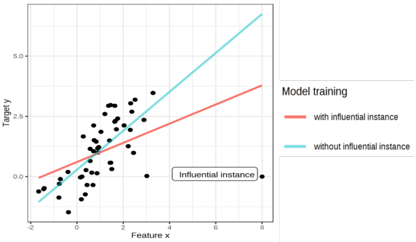


Deletion diagnostics, Robustness

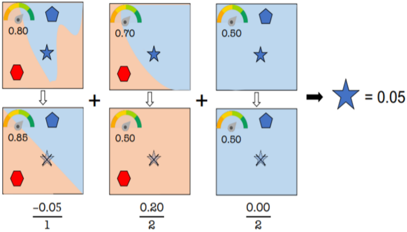
$$\hat{b}(\hat{f}_n) = (n-1)\left(n^{-1}\sum_{i=1}^{n} \hat{f}_{-i} - \hat{f}_n\right)$$

Bias reduction

GPDR issues, Privacy



Deletion diagnostics, Robustness



Data valuation, Shapley value [GZ19]

$$\hat{b}(\hat{f}_n) = (n-1)\left(n^{-1}\sum_{i=1}^{n}\hat{f}_{-i} - \hat{f}_n\right)$$

Bias reduction

Uncertainty quantification, etc ...

Outline

Outline

State of the art

- Most prior works target incrementally updating some specific ML models after the deletion of a small number of training samples:

State of the art

- Most prior works target incrementally updating some specific ML models after the deletion of a small number of training samples:
  - Linear regression and Logistic regression [WTD20][GGHvdM19]
  - K-means [GGVZ19]
  - etc..

Challenges

- Can we incrementally update general ML models trained by GD/SGD?

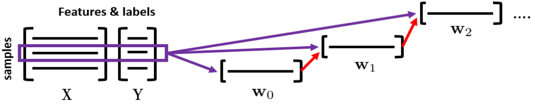
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{\eta_t}{B} \sum_{i \in \mathcal{B}_t} \nabla F_i(\mathbf{w}_t)$$

Challenges

- Can we incrementally update general ML models trained by GD/SGD?

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

- This is difficult due to “dense computational dependencies” [Sch]

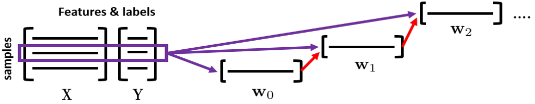


Challenges

- Can we incrementally update general ML models trained by GD/SGD?

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

- This is difficult due to “dense computational dependencies” [Sch]



- Approximation may be essential for incremental updates [GGVZ19]

Outline

Starting from Gradient Descent (GD)

- Given the following objective function:

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

Starting from Gradient Descent (GD)

- Given the following objective function:

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

- GD update rules before and after deletion (R: a set of deleted training samples, $|R| \ll n$):

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

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

Starting from Gradient Descent (GD)

- Given the following objective function:

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

- GD update rules before and after deletion (R: a set of deleted training samples, $|R| \ll n$):

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

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

Starting from Gradient Descent (GD)

- Given the following objective function:

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

- GD update rules before and after deletion (R: a set of deleted training samples, $|R| \ll n$):

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

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

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

Starting from Gradient Descent (GD)

- Given the following objective function:

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

- GD update rules before and after deletion (R: a set of deleted training samples, $|R| \ll n$):

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

Δ

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

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

Starting from Gradient Descent (GD)

- Given the following objective function:

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

- GD update rules before and after deletion (R: a set of deleted training samples, $|R| \ll n$):

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

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

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

Δ

Thoughts

- Effectively, we need to compute the GD/SGD path after a small perturbation of the data
- We can think of this as taking a small change "delta" of Gradient Descent, hence the name *DeltaGrad*

Some observations

- By denoting $\frac{1}{n}\sum_{i=1}^{n}\nabla F_i\left(\mathbf{w}\right)=\nabla F\left(\mathbf{w}\right)$, according to the Cauchy mean value theorem ($\mathbf{H}(\mathbf{w})$ is the Hessian matrix at $\mathbf{w}$):

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

where $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left({\mathbf{w}^U}_t - \mathbf{w}_t\right)\right)dx$

Some observations

- By denoting $\frac{1}{n}\sum_{i=1}^{n} \nabla F_i(\mathbf{w}) = \nabla F(\mathbf{w})$, according to the Cauchy mean value theorem ($\mathbf{H}(\mathbf{w})$ is the Hessian matrix at $\mathbf{w}$):

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

where $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left({\mathbf{w}^U}_t - \mathbf{w}_t\right)\right) dx$

- However, explicitly maintaining and evaluating the Hessian matrix is expensive!

Some observations

- By denoting $\frac{1}{n}\sum_{i=1}^{n} \nabla F_i(\mathbf{w}) = \nabla F(\mathbf{w})$, according to the Cauchy mean value theorem ($\mathbf{H}(\mathbf{w})$ is the Hessian matrix at $\mathbf{w}$):

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

where $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left({\mathbf{w}^U}_t - \mathbf{w}_t\right)\right) dx$

- However, explicitly maintaining and evaluating the Hessian matrix is expensive!
  - Classical optimization methods for efficiently approximating $\mathbf{H}_t$, e.g. L-BFGS algorithm [MS79, Noc80, BNS94, BLNZ95, ZBLN97, NW06, MR15]

Brief review of the L-BFGS algorithm

- In the L-BFGS algorithm, the gradients are incrementally updated at each step:

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

where $\mathbf{B}_t$ approximates $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx$

Brief review of the L-BFGS algorithm

- In the L-BFGS algorithm, the gradients are incrementally updated at each step:

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

  where $\mathbf{B}_t$ approximates $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx$

- By denoting $\mathbf{s}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$ and $\nabla F(\mathbf{w}_{t+1}) - \nabla F(\mathbf{w}_t) = \mathbf{y}_t$:

$$(\mathbf{B}_t)\mathbf{v} = g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-m}), \mathbf{v})$$

  where $\mathbf{v}$ is an arbitrary vector, m is a small integer and g is a function defined by the L-BFGS algorithm.

Brief review of the L-BFGS algorithm

- In the L-BFGS algorithm, the gradients are incrementally updated at each step:

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

  where $\mathbf{B}_t$ approximates $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx$
- By denoting $\mathbf{s}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$ and $\nabla F(\mathbf{w}_{t+1}) - \nabla F(\mathbf{w}_t) = \mathbf{y}_t$:

$$(\mathbf{B}_t)\mathbf{v} = g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-m}), \mathbf{v})$$

  where $\mathbf{v}$ is an arbitrary vector, m is a small integer and g is a function defined by the L-BFGS algorithm.
- Then:

$$\begin{aligned}&\nabla F(\mathbf{w}_{t+1}) - \nabla F(\mathbf{w}_t) \approx \mathbf{B}_t(\mathbf{w}_{t+1} - \mathbf{w}_t)\\ &= g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-m}), \mathbf{w}_{t+1} - \mathbf{w}_t)\end{aligned}$$

From L-BFGS algorithm to our case

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

$$\mathbf{B}_t \approx \mathbf{H}_t$$

$$= \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx$$

$$\mathbf{s}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$$

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

From L-BFGS algorithm to our case

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

$$\mathbf{B}_t \approx \mathbf{H}_t$$

$$= \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right)\right) dx$$

$$\mathbf{s}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$$

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

→

$$\nabla F(\mathbf{w}^U{}_t) - \nabla F(\mathbf{w}_t) \approx \mathbf{B}_t(\mathbf{w}^U{}_t - \mathbf{w}_t)$$

$$\mathbf{B}_t \approx \mathbf{H}_t$$

$$= \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}^U{}_t - \mathbf{w}_t\right)\right) dx$$

$$\mathbf{s}_t = \mathbf{w}^U{}_t - \mathbf{w}_t$$

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

From the L-BFGS algorithm to our case - cont.

- By utilizing the L-BFGS algorithm:

$$\mathbf{B}_t({\mathbf{w}^U}_t - \mathbf{w}_t) = g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-m}), {\mathbf{w}^U}_t - \mathbf{w}_t)$$
$$\Rightarrow \nabla F({\mathbf{w}^U}_t) \approx \nabla F(\mathbf{w}_t) + \mathbf{B}_t({\mathbf{w}^U}_t - \mathbf{w}_t)$$

From the L-BFGS algorithm to our case - cont.

- By utilizing the L-BFGS algorithm:

$$\mathbf{B}_t(\mathbf{w}^U{}_t - \mathbf{w}_t) = g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \dots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \dots, \mathbf{s}_{t-m}), \mathbf{w}^U{}_t - \mathbf{w}_t)$$
$$\Rightarrow \nabla F(\mathbf{w}^U{}_t) \approx \nabla F(\mathbf{w}_t) + \mathbf{B}_t(\mathbf{w}^U{}_t - \mathbf{w}_t)$$

- By using $\mathbf{w}^I$ as approximated $\mathbf{w}^U$:

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

From the L-BFGS algorithm to our case - cont.

- By utilizing the L-BFGS algorithm:

$$\mathbf{B}_t(\mathbf{w}^U{}_t - \mathbf{w}_t) = g((\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-m}), (\mathbf{s}_{t-1}, \mathbf{s}_{t-2}, \ldots, \mathbf{s}_{t-m}), \mathbf{w}^U{}_t - \mathbf{w}_t)$$
$$\Rightarrow \nabla F(\mathbf{w}^U{}_t) \approx \nabla F(\mathbf{w}_t) + \mathbf{B}_t(\mathbf{w}^U{}_t - \mathbf{w}_t)$$

- By using $\mathbf{w}^I$ as approximated $\mathbf{w}^U$:

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

- Go back to the Gradient Descent update rule:

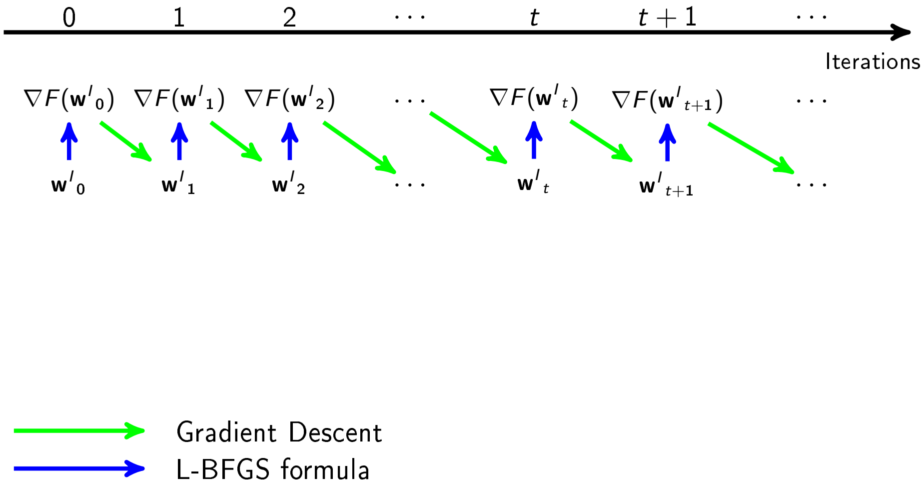
$$\begin{aligned}
\mathbf{w}^I{}_{t+1} &\approx \mathbf{w}^I{}_t - \frac{\eta_t}{n - |R|}[\sum_{i=1}^{n} \nabla F_i\left(\mathbf{w}^I{}_t\right) - \sum_{i \in R} \nabla F_i\left(\mathbf{w}^I{}_t\right)] \\
&= \mathbf{w}^I{}_t - \frac{\eta_t}{n - |R|}[n \nabla F(\mathbf{w}^I{}_t) - \sum_{i \in R} \nabla F_i(\mathbf{w}^I{}_t)] \\
&= \mathbf{w}^I{}_t - \frac{\eta_t}{n - |R|}\left\{n[\nabla F(\mathbf{w}_t) + \mathbf{B}_t(\mathbf{w}^I{}_t - \mathbf{w}_t)] - \sum_{i \in R} \nabla F_i(\mathbf{w}^I{}_t)\right\}
\end{aligned}$$

A remaining problem

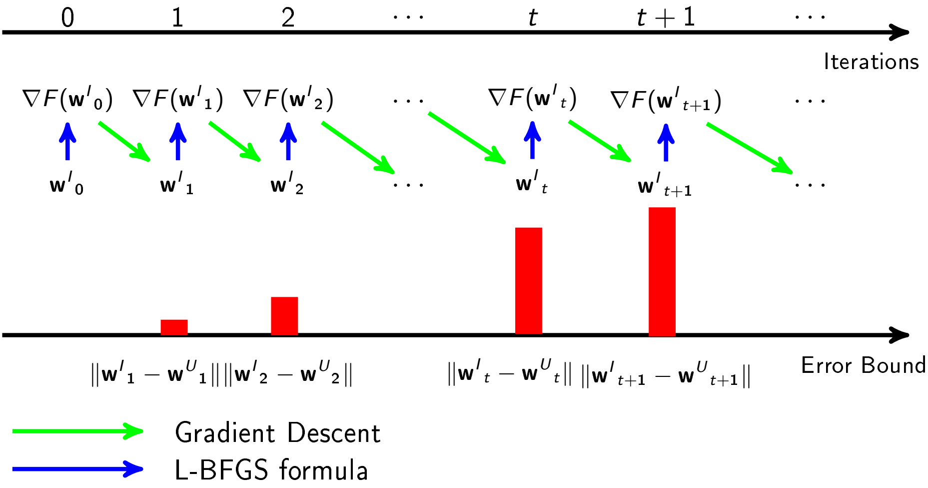
	0	1	2	$\cdots$	t	$t+1$	$\cdots$
	$\nabla F(\mathbf{w}'_0)$	$\nabla F(\mathbf{w}'_1)$	$\nabla F(\mathbf{w}'_2)$	$\cdots$	$\nabla F(\mathbf{w}'_t)$	$\nabla F(\mathbf{w}'_{t+1})$	$\cdots$
	$\mathbf{w}'_0$	$\mathbf{w}'_1$	$\mathbf{w}'_2$	$\cdots$	$\mathbf{w}'_t$	$\mathbf{w}'_{t+1}$	$\cdots$

Iterations

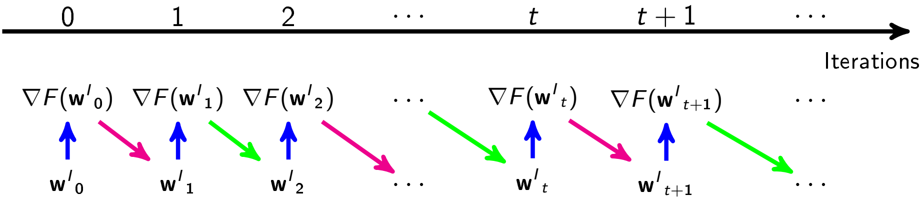
A remaining problem



A remaining problem



To control the errors

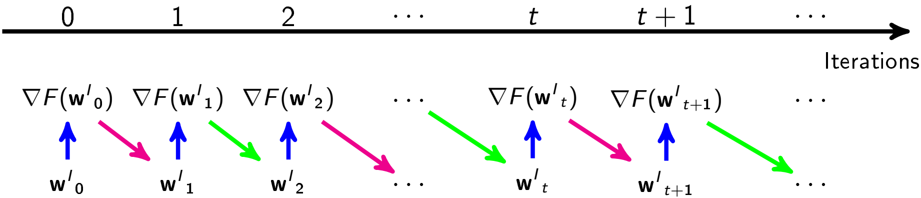


Gradient Descent

L-BFGS formula

Explicit gradient evaluations

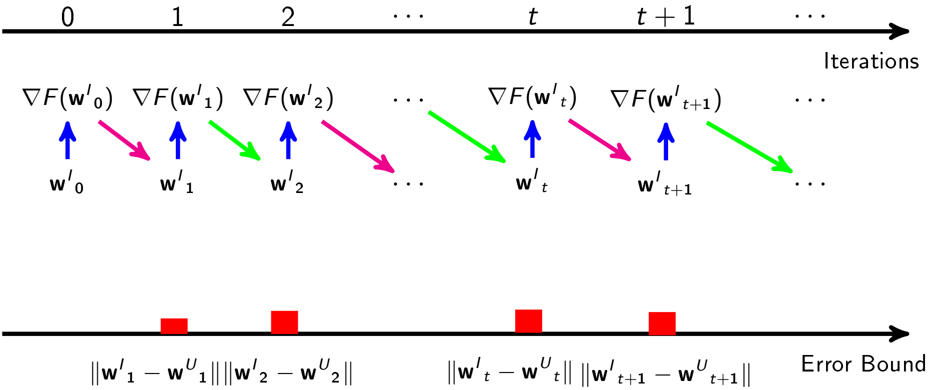
To control the errors



- Gradient Descent
- L-BFGS formula
- Explicit gradient evaluations /

1. in the first few iterations
2. periodically

To control the errors



- Green arrow: Gradient Descent
- Blue arrow: L-BFGS formula
- Pink arrow: Explicit gradient evaluations /

1. in the first few iterations
2. periodically

Extra benefit of DeltaGrad

- Gradient Descent update rule after minor deletions:

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

$$= \mathbf{w}^{U}{}_{t} - \frac{\eta_t}{n - |R|} [n \nabla F(\mathbf{w}^{U}{}_{t}) - \sum_{i \in R} \nabla F_i(\mathbf{w}^{U}{}_{t})]$$

Extra benefit of DeltaGrad

- Gradient Descent update rule after minor deletions:

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

$$= \mathbf{w}^{U}{}_{t} - \frac{\eta_t}{n - |R|} [n \nabla F(\mathbf{w}^{U}{}_{t}) - \sum_{i \in R} \nabla F_i(\mathbf{w}^{U}{}_{t})]$$

Can enable minor additions on training dataset by replacing - with +

Extra benefit of DeltaGrad

- Gradient Descent update rule after minor deletions:

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

$$= \mathbf{w}^{U}{}_{t} - \frac{\eta_t}{n-|R|} [n \nabla F(\mathbf{w}^{U}{}_{t}) - \sum_{i \in R} \nabla F_i(\mathbf{w}^{U}{}_{t})]$$

Evaluate the gradients on the added samples

Can enable minor additions on training dataset by replacing - with +

Outline

Theoretical results on gradient descent

Theorem (Bound between true and incrementally updated iterates)

By assuming that the objective function $F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\mathbf{w})$ is strongly convex, for a large enough iteration counter t, the result ${\mathbf{w}^I}_t$ of DeltaGrad approximates the correct iteration values ${\mathbf{w}^U}_t$ at the rate

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

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

Theoretical results on stochastic gradient descent

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

By assuming that the objective function $F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} F_i(\mathbf{w})$ is strongly convex, for a large enough iteration counter t and a mini-batch size B, the result ${\mathbf{w}^I}_t$ of DeltaGrad approximates the correct iteration values ${\mathbf{w}^U}_t$ at the rate

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

with high probability.

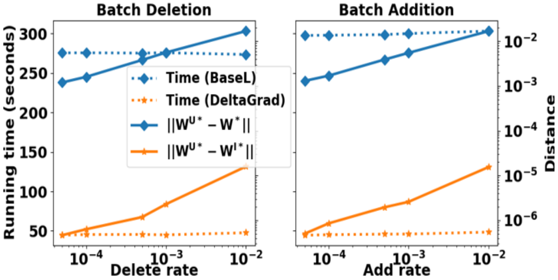
Outline

Experimental setup

- Datasets: Various standard benchmark datasets
- Using logistic regression model with L2 regression
- Compare DeltaGrad with the baseline approach, i.e. the approach of retraining from scratch (BaseL)

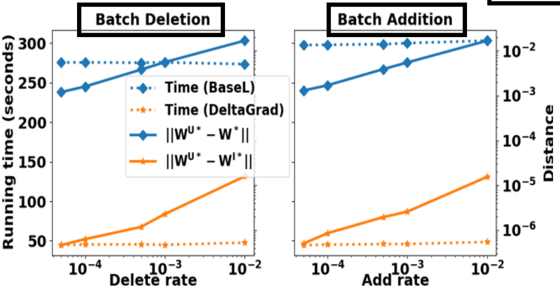
Experimental results

RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)



RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)

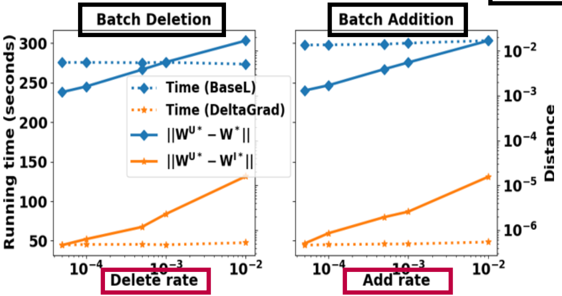
One deletion or addition with a subset of samples once



Experimental results

RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)

One deletion or addition with a subset of samples once



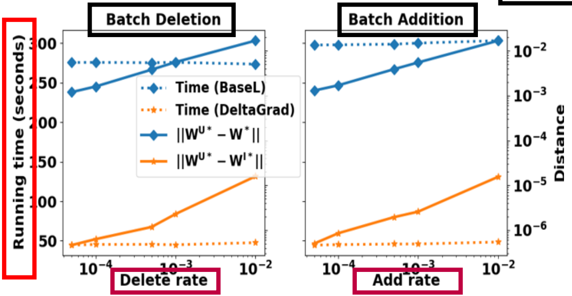
Varying the number of removed/added samples
Delete/Add rate: the number of removed/added samples VS the entire training dataset size

Experimental results

RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)

One deletion or addition with a subset of samples once

Time to update the model



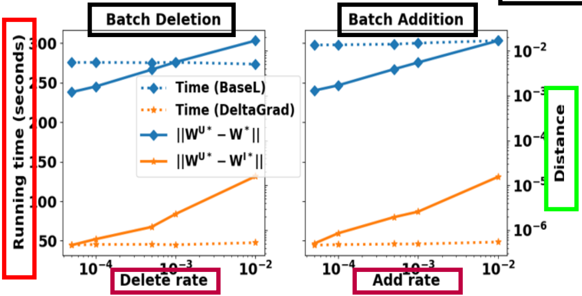
Varying the number of removed/added samples
Delete/Add rate: the number of removed/added samples VS the entire training dataset size

Experimental results

RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)

One deletion or addition with a subset of samples once

Time to update the model



Error bounds: $\|\mathbf{w}^{U*} - \mathbf{w}^{I*}\|$: Difference of updated parameters with and without approximation

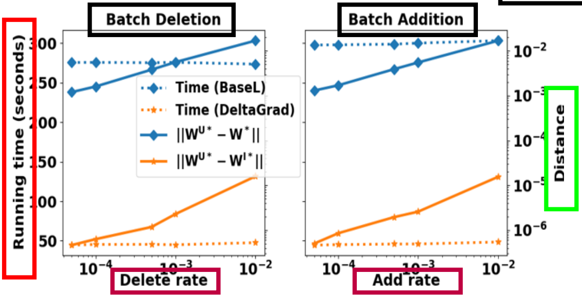
Varying the number of removed/added samples
Delete/Add rate: the number of removed/added samples VS the entire training dataset size

Experimental results

RCV1 (number of features = 47k, minibatch = 16k, Iterations = 400)

One deletion or addition with a subset of samples once

Time to update the model



Error bounds: $\|\mathbf{w}^{U*} - \mathbf{w}^{I*}\|$: Difference of updated parameters with and without approximation

Varying the number of removed/added samples
Delete/Add rate: the number of removed/added samples VS the entire training dataset size

Observations:

1. Up to 6x speed-ups relative to BaseL
2. Error bound is negligible

Wrap up and future work

- We proposed a method DeltaGrad which can incrementally update general strongly convex ML models.
  - Our code: `https://github.com/thuwuyinjun/DeltaGrad`
- Future work: Relax the strong convexity assumption

References I

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu.
A limited memory algorithm for bound constrained optimization.
SIAM Journal on scientific computing, 16(5):1190–1208, 1995.

Richard H Byrd, Jorge Nocedal, and Robert B Schnabel.
Representations of quasi-newton matrices and their use in limited memory methods.
Mathematical Programming, 63(1-3):129–156, 1994.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten.
Certified data removal from machine learning models.
arXiv preprint arXiv:1911.03030, 2019.

References II

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou.
Making ai forget you: Data deletion in machine learning.
In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

Amirata Ghorbani and James Zou.
Data shapley: Equitable valuation of data for machine learning.
In *International Conference on Machine Learning*, pages 2242–2251, 2019.

Aryan Mokhtari and Alejandro Ribeiro.
Global convergence of online limited memory bfgs.
The Journal of Machine Learning Research, 16(1):3151–3181, 2015.

Hermann Matthies and Gilbert Strang.
The solution of nonlinear finite element equations.
International journal for numerical methods in engineering, 14(11):1613–1626, 1979.

Jorge Nocedal.
Updating quasi-newton matrices with limited storage.
Mathematics of computation, 35(151):773–782, 1980.

Jorge Nocedal and Stephen Wright.
Numerical optimization.
Springer Science & Business Media, 2006.

Sebastian Schelter.
âĂIJamnesiaâĂİ-a selection of machine learning models that can forget user data very fast.
suicide, 8364(44035):46992.

References IV

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, 2020.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal.
Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization.
ACM Transactions on Mathematical Software (TOMS), 23(4):550–560, 1997.