Stochastic Gradient Descent with Differential Privacy

Shuang Song[†], Kamalika Chaudhuri[†], Anand D. Sarwate[‡]

† University of California, San Diego
‡ Toyota Technological Institute at Chicago

December 21, 2013

Efficient methods for large-scale learning

Large-scale inference problems face several challenges

- Hard to fit a single data set on one machine.
- Data acquired online, or sequentially.
- Data may come with constraints (privacy, duh).

Optimization over a data set

Consider the problem of solving a minimization problem over a data set $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ of n data points in $\mathbb{R}^d$:

$$\operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{z}_i),$$

Minimizing this directly using, e.g. gradient descent, involves computing n gradients for each data point.

Optimization over a data set

Consider the problem of solving a minimization problem over a data set $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ of n data points in $\mathbb{R}^d$:

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}, \mathbf{z}_i),$$

Minimizing this directly using, e.g. gradient descent, involves computing n gradients for each data point.
As an example for this talk, $\mathbf{z}_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x} \in \mathbb{R}^d$ is data for individual i and $y_i \in \{-1, 1\}$ is a label:

$$\mathbf{w}^* = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_t \mathbf{w}^\top \mathbf{x}_t})$$

this is *regularized logistic regression*.

Computational issues

Regularized logistic regression:

$$\mathbf{w}^* = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_t \mathbf{w}^\top \mathbf{x}_t})$$

Computational issues

Regularized logistic regression:

$$\mathbf{w}^* = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_t \mathbf{w}^\top \mathbf{x}_t})$$

We can interpret $\mathrm{sign}(\mathbf{w}^{*\top}\mathbf{x})$ as a decision rule of predicting a label for a new data point $\mathbf{x}$.

Computational issues

Regularized logistic regression:

$$\mathbf{w}^* = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_t \mathbf{w}^\top \mathbf{x}_t})$$

We can interpret $\mathrm{sign}(\mathbf{w}^{*\top}\mathbf{x})$ as a decision rule of predicting a label for a new data point $\mathbf{x}$.

Often solve this problem using *gradient descent*. When n is large we have to compute n gradients for each data point.

Stochastic gradients

A popular method for optimization in this setting is *stochastic gradient descent* (SGD). At each time step $t = 1, 2, \ldots$, sample a point $(\mathbf{x}_t, y_t)$ uniformly from the data set:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

Stochastic gradients

A popular method for optimization in this setting is *stochastic gradient descent* (SGD). At each time step $t = 1, 2, \ldots$, sample a point $(\mathbf{x}_t, y_t)$ uniformly from the data set:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

where η_t is the *learning rate* or *step size* – often $1/t$ or $1/\sqrt{t}$.

Stochastic gradients

A popular method for optimization in this setting is *stochastic gradient descent* (SGD). At each time step $t = 1, 2, \ldots$, sample a point $(\mathbf{x}_t, y_t)$ uniformly from the data set:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

where η_t is the *learning rate* or *step size* – often $1/t$ or $1/\sqrt{t}$.

- The expected gradient is the true gradient: "stochastic approximation."

Stochastic gradients

A popular method for optimization in this setting is *stochastic gradient descent* (SGD). At each time step $t = 1, 2, \ldots$, sample a point $(\mathbf{x}_t, y_t)$ uniformly from the data set:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

where η_t is the *learning rate* or *step size* – often $1/t$ or $1/\sqrt{t}$.

- The expected gradient is the true gradient: "stochastic approximation."
- In practice, just sample without replacement.

Stochastic gradients

A popular method for optimization in this setting is *stochastic gradient descent* (SGD). At each time step $t = 1, 2, \ldots$, sample a point $(\mathbf{x}_t, y_t)$ uniformly from the data set:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

where η_t is the *learning rate* or *step size* – often $1/t$ or $1/\sqrt{t}$.

- The expected gradient is the true gradient: "stochastic approximation."
- In practice, just sample without replacement.
- Taking a few passes through the data typically works well.

Privacy?

$$\mathbf{w}_{t+1} = w_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

What happens when the data is private? E.g. $\mathbf{x}$ are lab measurements, y is a disease state.

Privacy?

$$\mathbf{w}_{t+1} = w_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

What happens when the data is private? E.g. $\mathbf{x}$ are lab measurements, y is a disease state.

- Data is still a *limited resource*.

Privacy?

$$\mathbf{w}_{t+1} = w_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

What happens when the data is private? E.g. $\mathbf{x}$ are lab measurements, y is a disease state.

- Data is still a *limited resource*.
- Now we want to balance privacy, utility, and efficiency.

Privacy?

$$\mathbf{w}_{t+1} = w_t - \eta_t(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t))$$

What happens when the data is private? E.g. $\mathbf{x}$ are lab measurements, y is a disease state.

- Data is still a *limited resource*.
- Now we want to balance privacy, utility, and efficiency.
- Maybe easy? Iterations are already random.

Privacy!

It is easy to guarantee differential privacy in this setting:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t) + \mathbf{Z}_t \right),$$

Privacy!

It is easy to guarantee differential privacy in this setting:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t) + \mathbf{Z}_t \right),$$

where each $\mathbf{Z}_t$ is a random noise vector in $\mathbb{R}^d$ drawn independently from the density:

$$\rho(\mathbf{z}) \propto e^{-(\varepsilon/2)\|\mathbf{z}\|}$$

Privacy!

It is easy to guarantee differential privacy in this setting:

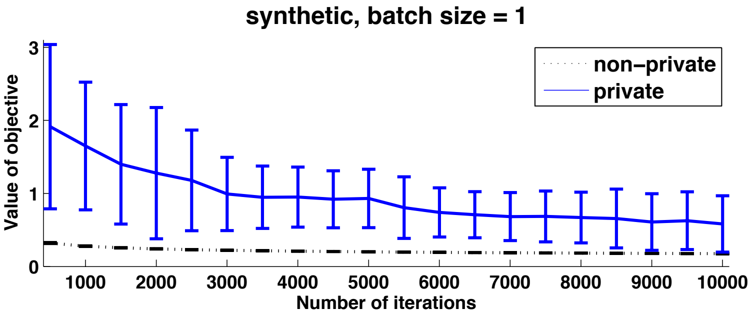
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t) + \mathbf{Z}_t \right),$$

where each $\mathbf{Z}_t$ is a random noise vector in $\mathbb{R}^d$ drawn independently from the density:

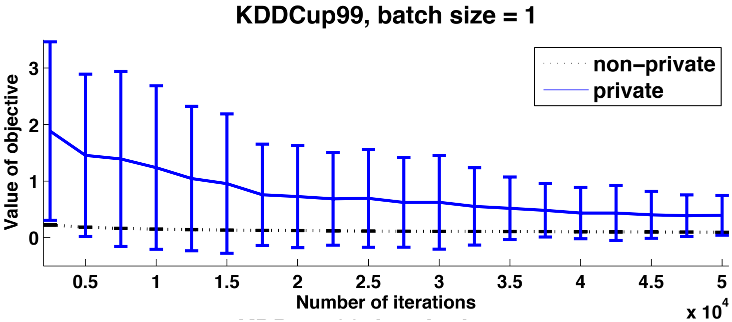
$$\rho(\mathbf{z}) \propto e^{-(\varepsilon/2)\|\mathbf{z}\|}$$

This guarantees differentially privacy in the *continuous observation setting* (Dwork, Naor, Pitassi, and Rothblum '10) or the *local privacy model* (c.f. Duchi, Jordan, and Wainwright '12).

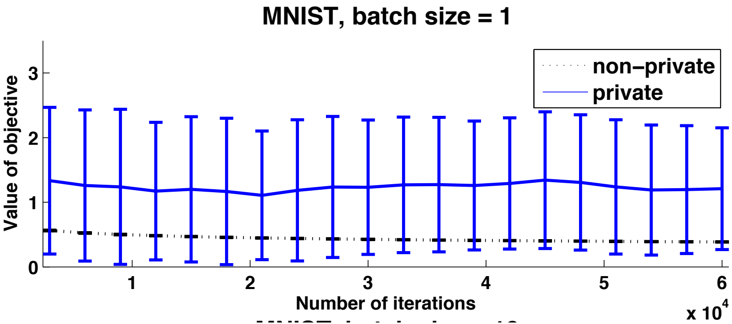
... but not off the shelf



… but not off the shelf



... but not off the shelf



Lots of knobs to tweak

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t) + \mathbf{Z}_t \right),$$

There are lots of things to play with here:

- Step size: choosing it is sort of an art
- Noise level vs. multiple passes
- Polyak averaging
- **Minibatching**: processing multiple points at once

Minibatching

Process *a batch of* b *points* chosen uniformly at random. At each t select a set B_t:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \frac{1}{b} \sum_{(\mathbf{x}_i, y_i) \in B_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i, y_i) \right)$$

With privacy:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \frac{1}{b} \sum_{(\mathbf{x}_i, y_i) \in B_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i, y_i) + \frac{1}{b} \mathbf{Z}_t \right).$$

Note: we've given up on local privacy here.

The minibatching tradeoff

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \frac{1}{b} \sum_{(\mathbf{x}_i, y_i) \in B_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i, y_i) + \frac{1}{b} \mathbf{Z}_t \right).$$

If we make 1 pass through the data, we now have a new tradeoff:

The minibatching tradeoff

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \frac{1}{b} \sum_{(\mathbf{x}_i, y_i) \in B_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i, y_i) + \frac{1}{b} \mathbf{Z}_t \right).$$

If we make 1 pass through the data, we now have a new tradeoff:

- Many gradient steps that are very noisy.

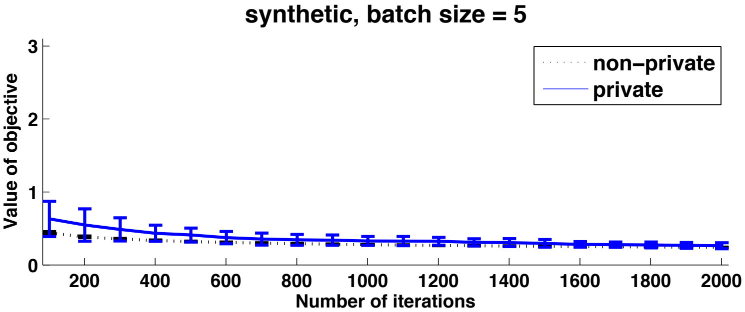
The minibatching tradeoff

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \left(\lambda \mathbf{w}_t + \frac{1}{b} \sum_{(\mathbf{x}_i, y_i) \in B_t} \nabla \ell(\mathbf{w}_t, \mathbf{x}_i, y_i) + \frac{1}{b} \mathbf{Z}_t \right).$$

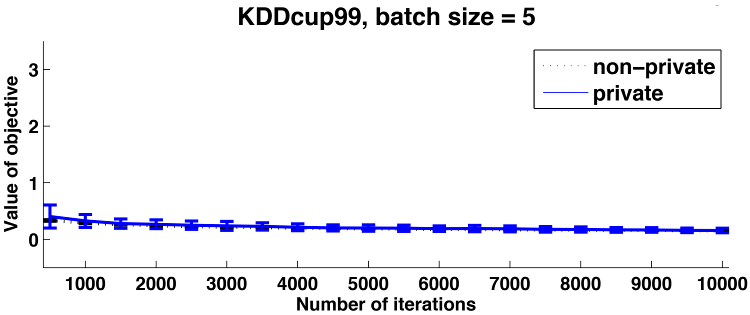
If we make 1 pass through the data, we now have a new tradeoff:

- Many gradient steps that are very noisy.
- Fewer gradient steps that are less noisy.

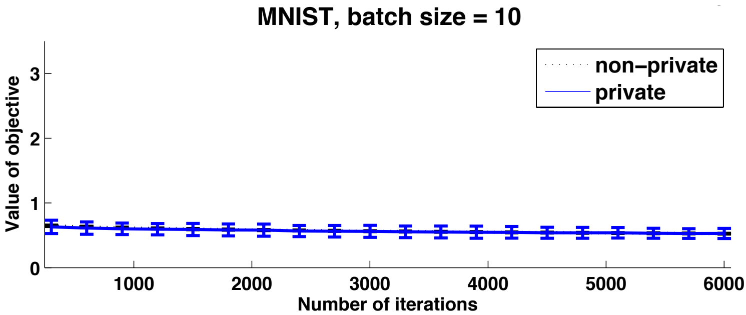
Empirical performance of minibatching



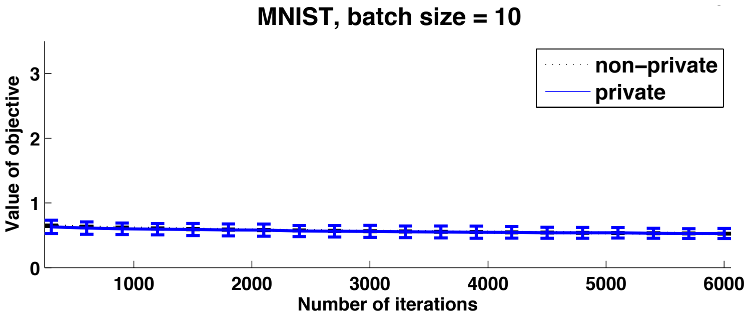
Empirical performance of minibatching



Empirical performance of minibatching

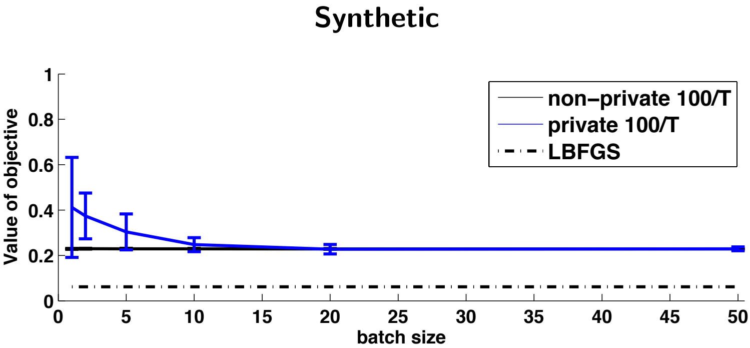


Empirical performance of minibatching

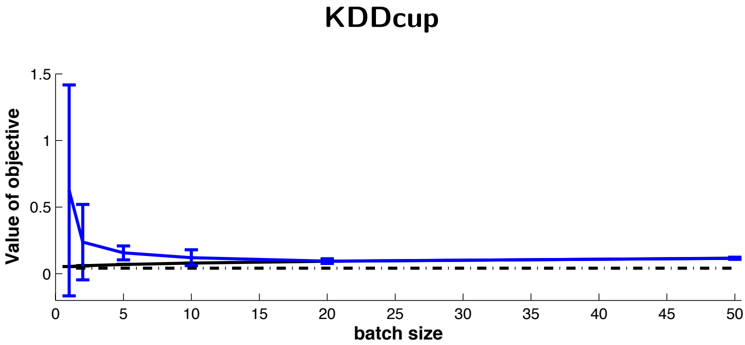


Minibatching helps a lot!

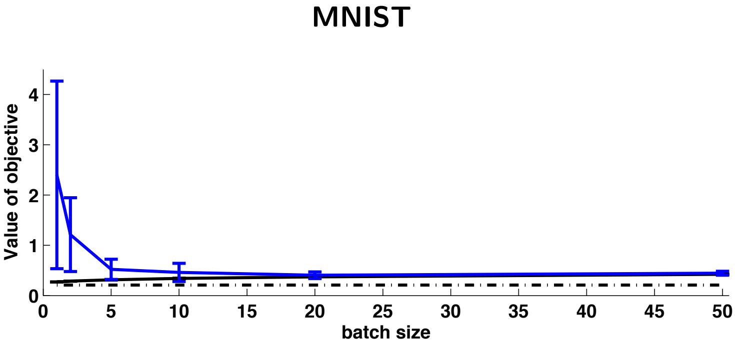
What's the best batch size?



What's the best batch size?

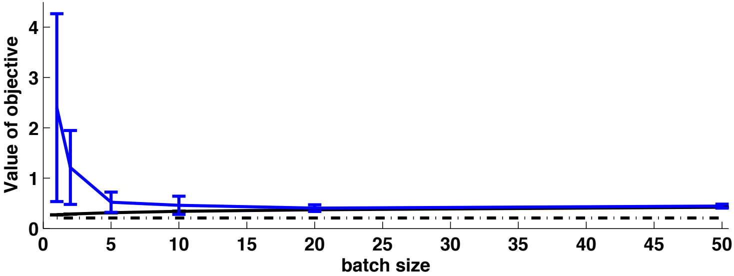


What's the best batch size?



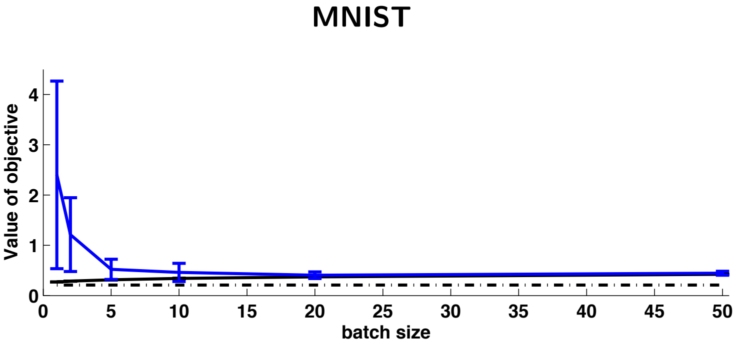
What's the best batch size?

MNIST



Bigger batches are good!

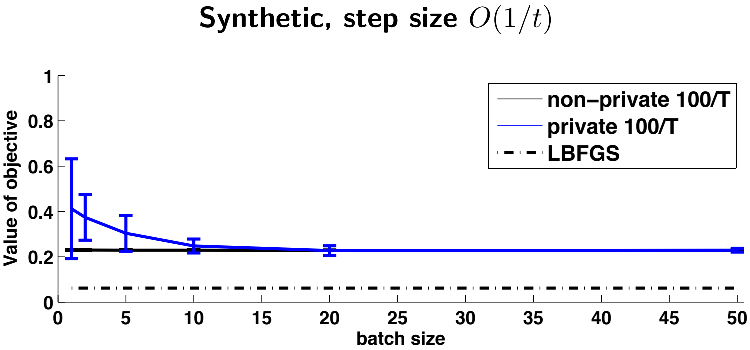
What's the best batch size?



Bigger batches are good!

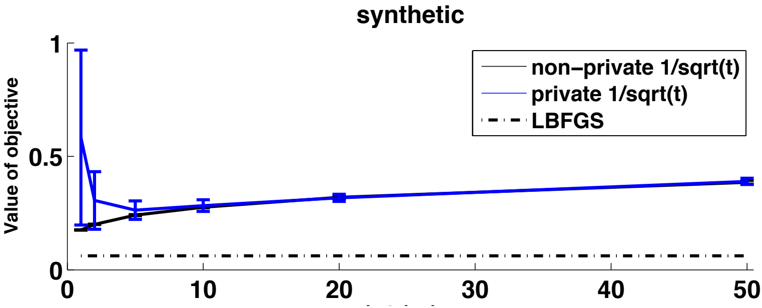
(that seems suspicious...)

The role of step size



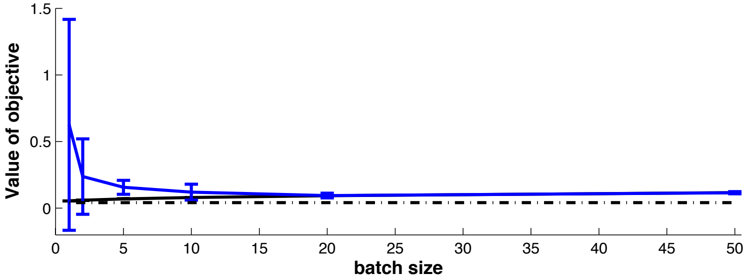
The role of step size

Synthetic, step size $O(1/\sqrt{t})$



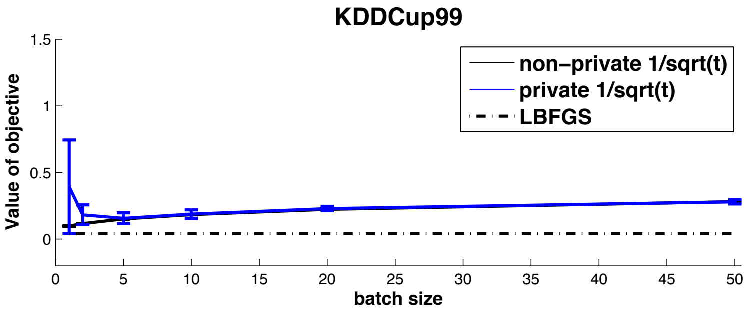
The role of step size

KDDcup, step size $O(1/t)$



The role of step size

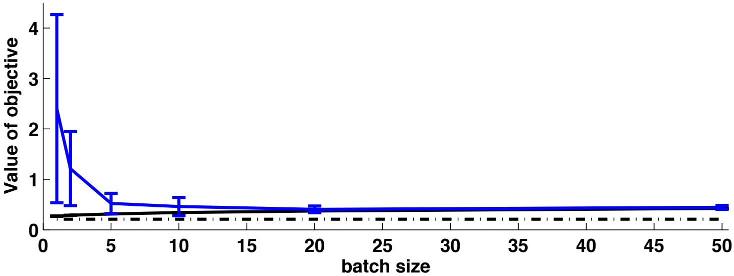
KDDcup, step size $O(1/\sqrt{t})$



Bigger batches are good for aggressive step sizes.

The role of step size

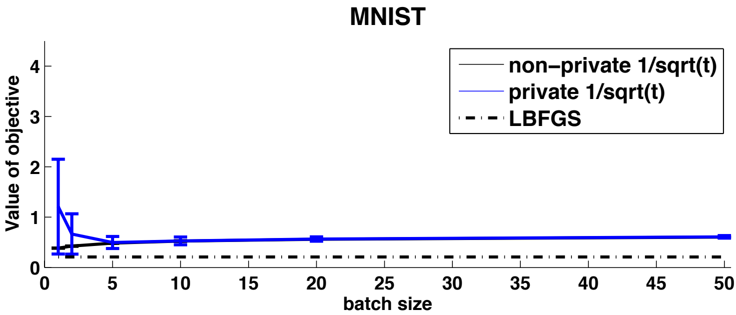
MNIST, step size $O(1/t)$



Bigger batches are good for aggressive step sizes.
Optimal batch sizes for more conservative step sizes.

The role of step size

MNIST, step size $O(1/\sqrt{t})$



Bigger batches are good for aggressive step sizes.
Optimal batch sizes for more conservative step sizes.

Many (many!) interesting questions

- How should we set all of these knobs in the presence of privacy noise?
- What gains are there without continuous observation?
- How should we incorporate multiple sources of data with different privacy constraints?
- Connections to online learning (Jain, Kothari, and Thakurta '11)?
- Other optimization algorithms like mirror descent (Smith and Thakurta '13)?
- Stochastic optimization versions of other algorithms (Hardt and Roth '13, Hardt '13)?

Thank you!

(now you can go drink)