

# Stochastic Gradient Descent

He He

CDS, NYU

February 6, 2021

## Gradient Descent for Empirical Risk - Scaling Issues

# Gradient Descent for Empirical Risk and Averages

- Suppose we have a hypothesis space of functions  $\mathcal{F} = \{f_w : \mathcal{X} \rightarrow \mathcal{A} \mid w \in \mathbb{R}^d\}$ 
  - Parameterized by  $w \in \mathbb{R}^d$ .
- ERM is to find  $w$  minimizing

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \ell(f_w(x_i), y_i)$$

- Suppose  $\ell(f_w(x_i), y_i)$  is differentiable as a function of  $w$ .
- Then we can do gradient descent on  $\hat{R}_n(w)$ ...

## Gradient Descent: How does it scale with $n$ ?

- At every iteration, we compute the gradient at current  $w$ :

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- We have to touch all  $n$  training points to take a single step. [ $O(n)$ ]
- Will this scale to “big data”?
- Can we make progress without looking at all the data?

# Stochastic Gradient Descent

# “Noisy” Gradient Descent

- We know gradient descent works.
- But the gradient may be slow to compute.
- What if we just use an estimate of the gradient?
- Turns out that can work fine.
- **Intuition:**
  - Gradient descent is an iterative procedure anyway.
  - At every step, we have a chance to recover from previous missteps.

# Minibatch Gradient

- The **full gradient** is

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

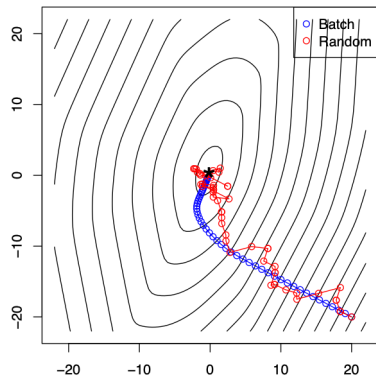
- It's an average over the **full batch** of data  $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ .
- Let's take a random subsample of size  $N$  (called a **minibatch**):

$$(x_{m_1}, y_{m_1}), \dots, (x_{m_N}, y_{m_N})$$

- The **minibatch gradient** is

$$\nabla \hat{R}_N(w) = \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_{m_i}), y_{m_i})$$

# Batch vs Stochastic Methods



Rule-of-thumb for stochastic methods:

- Stochastic methods work well far from the optimum
- But struggles close the the optimum

(Slides adapted from Ryan Tibshirani)



- What can we say about the minibatch gradient? It's random. What's its expectation?

$$\begin{aligned}\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] &= \frac{1}{N} \sum_{i=1}^N \mathbb{E} [\nabla_w \ell(f_w(x_{m_i}), y_{m_i})] \\ &= \mathbb{E} [\nabla_w \ell(f_w(x_{m_1}), y_{m_1})] \\ &= \sum_{i=1}^n \mathbb{P}(m_1 = i) \nabla_w \ell(f_w(x_i), y_i) \\ &= \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i) \\ &= \nabla \hat{R}_n(w)\end{aligned}$$

# Minibatch Gradient Properties

- Minibatch gradient is an **unbiased estimator** for the [full] batch gradient:

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

$$\frac{1}{N} \text{Var} \left[ \nabla \hat{R}_1(w) \right] = \text{Var} \left[ \nabla \hat{R}_N(w) \right]$$

- Tradeoffs of minibatch size:
  - Bigger  $N \implies$  Better estimate of gradient, but slower (more data to touch)
  - Smaller  $N \implies$  Worse estimate of gradient, but can be quite fast

# Convergence of SGD

- For convergence guarantee, use **diminishing step sizes**, e.g.  $\eta_k = 1/k$  (dampens noise in step direction)
- Theoretically, GD is much faster than SGD in terms of convergence rate
  - much faster to add a digit of accuracy on the minimum
  - but most of that benefit happens once you're already pretty close
- However, in many ML problems we don't care about optimizing to high accuracy

# Step Sizes in Minibatch Gradient Descent

## Minibatch Gradient Descent (minibatch size $N$ )

- initialize  $w = 0$
- repeat
  - randomly choose  $N$  points  $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{D}_n$
  - $w \leftarrow w - \eta \left[ \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
- For SGD, fixed step size can work well in practice.
- Typical approach: Fixed step size reduced by constant factor whenever validation performance stops improving.
- Other tricks: Bottou (2012), “Stochastic gradient descent tricks”

- **Gradient descent** or “full-batch” gradient descent
  - Use full data set of size  $n$  to determine step direction
- **Minibatch gradient descent**
  - Use a random subset of size  $N$  to determine step direction
- **Stochastic gradient descent**
  - Minibatch with  $N = 1$ .
  - Use a single randomly chosen point to determine step direction.

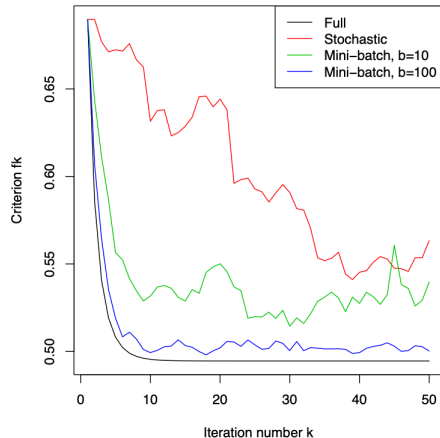
These days terminology isn't used so consistently, so always clarify the [mini]batch size.

SGD is much more efficient in time and memory cost and has been quite successful in large-scale ML.

## Practical Comparison of GD vs SGD

# Logistic regression with $\ell_2$ regularization

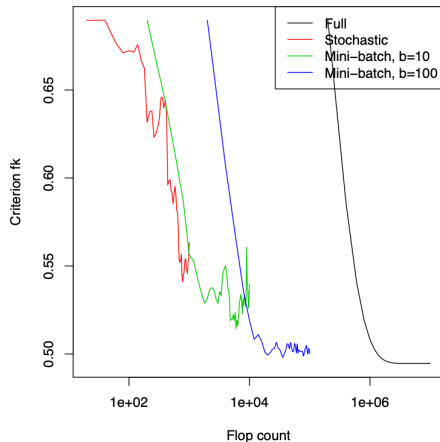
Batch methods converge faster



(Example from Ryan Tibshirani)

# Logistic regression with $\ell_2$ regularization

Stochastic methods are computationally more efficient

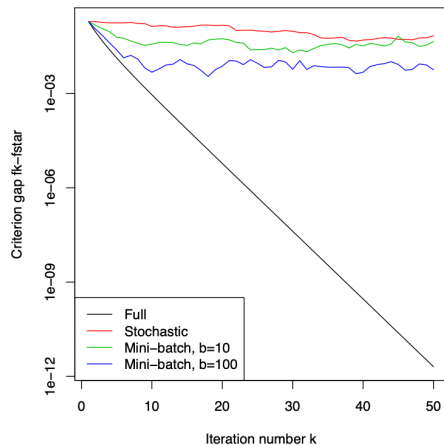


(Example from Ryan Tibshirani)



# Logistic regression with $\ell_2$ regularization

Batch methods are much faster close to the optimum



(Example from Ryan Tibshirani)