Stochastic Gradient Descent

George Michailidis

gmichail@ucla.edu

STAT 102B

The Stochastic Gradient Descent (SGD) Algorithm

It has become the workhorse algorithm for training machine learning and deep neural network models

The key motivation is that the data set under consideration is extremely large, both in terms of number of observations n and variables p

We will illustrate the main idea in a regression model.

All the issues discussed thus far on how to select the step size and the use of momentum methods are also applicable

Problem Setup

Objective: Up to this point, the problem of interest has been formulated as

$f : \mathbb{R}^n \to \mathbb{R}$

$$\min_{x \in \mathbb{R}^n} f(x)$$

However, in statistics and machine learning, the objective function of interest takes the form

$$f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x), \tag{1}$$

since we have access to m observations in the data set

Remark:
We introduce the scaling $1/m$, to account for the sample size in the data set

Illustrative Example: Linear Regression Model - I

Recap from Lecture 1.1

A linear regression model with p predictor variables $x_1, \cdots, x_p$ and a response variable y is given by

$$y_i = \beta_1 x_{1i} + \cdots + \beta_p x_{pi} + \epsilon_i, \quad i = 1, \cdots, m, \tag{2}$$

where $\beta_j, j = 1, \cdots, p$ the regression coefficients for the p predictors and ϵ_i is a random noise term

We assume that the intercept term $\beta_0 = 0$

The random noise satisfies $\mathbb{E}(\epsilon_i) = 0,\ \mathrm{Var}(\epsilon_i) = \sigma_\epsilon^2$

Illustrative Example: the Linear Regression Model - II

The regression model in matrix form can be written as:

$$y = X\beta + \epsilon,$$

where

- y is a $m \times 1$ vector containing the observed responses,
- X an $m \times p$ matrix containing the observed values of the p predictors plus a column of ones for the intercept term,
- β is a $p \times 1$ vector containing the regression coefficients, and
- ϵ is a $m \times 1$ vector containing the unobserved error terms

and the objective function we are interested in minimizing:
the Sum of Squared Errors

$$\text{SSE}(\beta) = \frac{1}{2m}||y - X\beta||_2^2 = \frac{1}{2m}(y - X\beta)^\top(y - X\beta)$$

Illustrative Example: the Linear Regression Model - III

Let us rewrite the regression model in the following form:

$$y_i = x_i^T \beta + \epsilon_i, \quad i = 1, \cdots, m$$

where

- x_i is a p column vector containing the values of the p predictors, and
- β is a p column vector containing the p regression coefficients

Then, the SSE(β) function can be written as

$$\text{SSE}(\beta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \text{SSE}_i(\beta) = \frac{1}{2m} \sum_{i=1}^{m} \frac{1}{2} (y_i - x_i^\top \beta)^2$$

Illustrative Example: the Linear Regression Model - IV

Then, the gradient for a single function $\text{SSE}_i(\beta)$ (i.e., for a single observation) is given by

$$\nabla \text{SSE}_i(\beta) = (y_i - x_i^\top \beta)(-x_i) = -x_i y_i + x_i^\top x_i \beta$$

By taking the average across across all m observations, we obtain

$$\begin{aligned} \frac{1}{m} \sum_{i=1}^{m} \nabla \text{SSE}_i(\beta) &= \frac{1}{m} \left(- \sum_{i=1}^{m} x_i y_i + \sum_{i=1}^{m} x_i^\top x_i \beta \right) \\ &= \frac{1}{m} \left(X^\top X \beta - X^\top y \right) = \nabla \text{SSE}(\beta) \end{aligned}$$

Illustrative Example: Logistic Regression Model - I

The logistic model (or logit model) is a statistical model that models the log-odds of an event as a linear combination of p predictor variables.

The task is to predict a categorical label y given data on predictors $x \in \mathbb{R}^p$

Let $y_i = 1$, if an event occurs (e.g., an image corresponds to a cat) and $y_i = 0$ (e.g., an image corresponds to a dog), otherwise, $i = 1, \cdots, m$

Further,

$$\pi_i(x_i, \beta) = \mathbb{P}(y_i = 1 | x_i, \beta) = \frac{1}{1 + \exp(-x_i^\top \beta)}, \tag{3}$$

i.e., the probability of the i-th event occurring is a function of predictor variables x_i, given in (3)

Illustrative Example: Logistic Regression Model - II

The likelihood function of this model is given by

$$L(\beta) = \prod_{i=1}^{m} \left[\pi_i(x_i, \beta)\right]^{y_i} \left[1 - \pi_i(x_i, \beta)\right]^{1-y_i}, \tag{4}$$

and the log-likelihood by

$$\ell(\beta) = \sum_{i=1}^{m} \left[y_i \log\left(\pi_i(x_i, \beta)\right) + (1 - y_i) \log\left[1 - \pi_i(x_i, \beta)\right] \right] \tag{5}$$

Substituting the form of the probability function given in (3) in the log-likelihood function (5) and after some algebra, we obtain the following form of the log-likelihood function

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{m} \left[y_i (x_i^\top \beta) + \log\left(\frac{1}{1 + \exp(x_i^\top \beta)} \right) \right] \\ &= \sum_{i=1}^{m} \left[y_i (x_i^\top \beta) - \log\left(1 + \exp(x_i^\top \beta)\right) \right] \end{aligned} \tag{6}$$

Example: Logistic Regression Model - III

So, the optimization problem for logistic regression becomes

$$\min_{\beta} -\ell(\beta) = \frac{1}{m}\sum_{i=1}^{m} f_i(\beta) \tag{7}$$

with

$$f_i(\beta) \equiv -y_i(x_i^\top \beta) + \log\left(1 + \exp(x_i^\top \beta)\right) \tag{8}$$

Remarks:

- Note that rescaling the log-likelihood function by a factor of $1/m$ does not alter the nature of the optimization problem
- Due to the presence of the nonlinear term in (6), the standard algorithm to estimate the parameters of the logistic regression model is the Gauss-Newton algorithm (that will be presented in Week 6)

The gradient of $f_i(\beta)$ - I

Gradient of first term in (8)

$$\nabla_\beta \left[-y_i (x_i^\top \beta) \right] = -y_i x_i \tag{9}$$

Gradient of second term in (8)

We apply the chain rule:

$$\begin{aligned} \nabla_\beta \left[\log\left(1 + \exp(x_i^\top \beta)\right) \right] &= \frac{1}{1 + \exp(x_i^\top \beta)} \cdot \exp(x_i^\top \beta) \cdot \nabla_\beta (x_i^\top \beta) \\ &= \frac{\exp(x_i^\top \beta)}{1 + \exp(x_i^\top \beta)} \cdot x_i \\ &= \frac{1}{1 + \exp(-x_i^\top \beta)} \cdot x_i \\ &= \sigma(x_i^\top \beta) \cdot x_i \end{aligned} \tag{10}$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the logistic sigmoid function

The gradient of $f_i(\beta)$ - II

Combining both terms:

$$\nabla_\beta f_i(\beta) = -y_i x_i + \sigma(x_i^\top \beta) \cdot x_i = \Big[\sigma(x_i^\top \beta) - y_i\Big] \cdot x_i \tag{11}$$

Therefore, the gradient of $f_i(\beta)$ is given by:

$$\nabla_\beta f_i(\beta) = \Big[\sigma(x_i^\top \beta) - y_i\Big] \cdot x_i \tag{12}$$

The gradient descent algorithm for the function $f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$

$f_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \cdots, m$

Since

$$\nabla f(x) = \nabla\left(\frac{1}{m}\sum_{i=1}^{m} f_i(x)\right) = \frac{1}{m}\sum_{m=1}^{m} \nabla f_i(x) \tag{13}$$

an update of the GD algorithm is given by

$$x_{k+1} = x_k - \frac{\eta_k}{m}\left(\sum_{m=1}^{m} \nabla f_i(x)\right) \tag{14}$$

The Stochastic Gradient Descent (SGD) Algorithm

Let I_k denote a subset of the index set $\{1, \cdots, m\}$ at iteration k

The cardinality of the index set I_k (the number of elements of I_k) can vary from 1 to m

The Stochastic Gradient Descent (SGD) Algorithm

Let I_k denote a subset of the index set $\{1, \cdots, m\}$ at iteration k

The cardinality of the index set I_k (the number of elements of I_k) can vary from 1 to m

Then, the main steps of the stochastic gradient descent algorithm are

1. Select the cardinality s of index set I_k
2. Select $x_0 \in \mathbb{R}^n$
3. While stopping criterion $>$ tolerance do:
   - $x_{k+1} = x_k - \eta_k \left[\nabla f_{I_k}(x_k)\right]$
   - Calculate the value of the stopping criterion

where

$$f_{I_k}(x_k) = \frac{1}{s} \sum_{i \in I_k} f_i(x_k) \quad \text{and thus} \quad \nabla \left[f_{I_k}(x_k)\right] = \frac{1}{s} \sum_{i \in I_k} \nabla f_i(x_k) \tag{15}$$

Remarks on the Index Set I_k

1. The value of the parameter s that determines the size of the index set I_k is called the batch size
   If $s = m$, then SGD=GD
   Using $s = 1$ works, but progress of SGD may be very slow
2. The selection of the index set I_k should be random
   Essentially, draw s indices from the set $\{1, \cdots, m\}$ and recall that each index has probability of been drawn equal to $1/m$

Rationale for using SGD

- One rationale behind using SGD is to reduce computational requirements, both in terms of memory and calculations
  - For example, for the logistic regression problem, each GD update requires $\mathcal{O}(mp)$ evaluations, whereas an SGD update with a small batch size s requires $\mathcal{O}(sp)$
  - If m is very large (e.g., $m \geq 10^6$ and p is also large, for $s = 1000$ you can do 1000 SGD updates for the cost of a single GD update
- Another rationale that has become very important when training ML or deep learning models is that SGD due to its built-in randomness can "escape" local minima and saddle points, objective functions $f(\cdot)$ that have both local minima and saddle points (non-convex functions)

Visualizing the SGD Setting - I

Let $f(x) : \mathbb{R} \to \mathbb{R}$ with

$$f(x) = 1.32x^2 = \frac{1}{100}\sum_{i=1}^{100} \gamma_i x^2, \quad \text{with } \gamma_i \sim \mathrm{N}(1.2, 1) \tag{16}$$

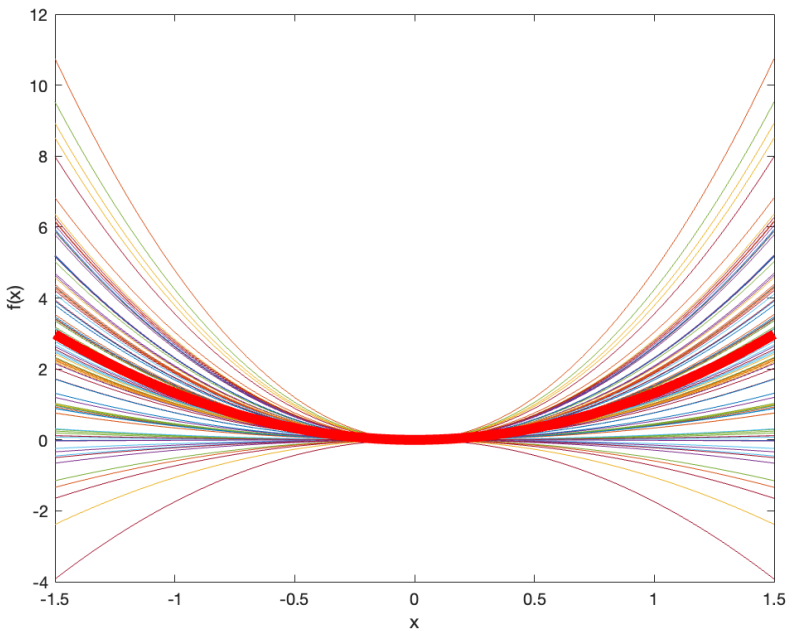

Figure 1: Plot of the $100 f_i(x) = \gamma_i x^2$ together with $f(x)$ (in bold red)

Visualizing the SGD Setting - II

It can be seen that $f''(x) = 2.64 > 0$ and hence the function's global minimum is at $x^\star = 0$

However, $f_i''(x) = \frac{2}{100}\gamma_i$ and hence for certain i, $f_i(x)$ is increasing throughout its domain

This simple example also illustrates that

$$d_k = -\nabla f_{I_k}(x) \tag{17}$$

may not be a descent direction for all I_k

In the illustrative example, take $I_k = \{i_\star : \gamma_{i_\star} < 0\}$
Then,

$$\left[\nabla f(x)\right]^\top d_k = \left[\nabla f(x)\right]^\top \nabla f_{I_k} f(x_k) = (2.64x)(-\frac{2}{100}\gamma_{i_\star} x) = -\frac{5.128}{100}\gamma_{i_\star} x^2 > 0 \tag{18}$$

since $\gamma_{i_\star} < 0$

Is d_k in SGD a descent direction? - I

The previous calculation showed that

$$\exists\ I_k \subseteq \{1, \cdots, m\} : \ \left[\nabla f(x_k)\right]^\top \nabla f_{I_k}(x_k) > 0, \tag{19}$$

unless $I_k = \{1, \cdots, m\}$

Hence, unless SGD=GD, we can not guarantee that the direction selected by SGD is necessarily a decent one!

Is d_k in SGD a descent direction? - I

The previous calculation showed that

$$\exists\ I_k \subseteq \{1, \cdots, m\} : \ \left[\nabla f(x_k)\right]^\top \nabla f_{I_k}(x_k) > 0, \tag{19}$$

unless $I_k = \{1, \cdots, m\}$

Hence, unless SGD=GD, we can not guarantee that the direction selected by SGD is necessarily a decent one!

Why is then SGD considered a good algorithm?

Is d_k in SGD a descent direction? - II

Let $f(x) : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable with $H \equiv \nabla^2 f(x)$ positive definite and recall that I_k is selected at random

Consider the direction used by SGD, $d_k = -\nabla f_{I_k}(x_k)$

$$\mathbb{E}\left(\left[\nabla f(x_k)\right]^\top d_k \right) = \left[\nabla f(x_k)\right]^\top \mathbb{E}\left(d_k \right) \tag{20}$$

where the expectation is taken over the random index set I_k

The following fact (established in the Appendix) holds for any mini-batch of arbitrary size $s = 1, \cdots, m$

$$\mathbb{E}\left(d_k \right) = -\nabla f(x_k) \tag{21}$$

Hence, d_k is a descent direction in expectation

A Note on Terminology

- In many papers, blogs and textbooks, authors make a distinction between SGD and batch GD
  Specifically, they reserve the term SGD for $s = 1$ and refer to batch GD, otherwise
- Other authors use the terms interchangeably, as is the case in my lecture notes
- The batch is usually referred to as the mini-batch

Selection of step size η_k

- Fixed Step Size:
  - As is the case for full GD, it is simple to implement but requires careful selection
  - For "nice" functions (i.e., convex, that possess a unique global minimum), any $\eta \in (0, \frac{1}{L})$ would work, where L is the Lipschitz constant of $f(\cdot)$
    As discussed in Lecture 2.1, finding the Lipschitz constant is a challenging task
- Decaying Step Size:
  - Common schedules:
    - Inverse scaling: $\eta_k = \frac{\eta_0}{1+\lambda k}$
    - Step decay: reduce η every few iterations
    - Exponential decay: $\eta_k = \eta_0 \cdot \gamma^k$, $\gamma < 1$
- Adaptive Methods:
  - AdaGrad, ADAM, ADAM-W
  - They have become very popular and widely used for SGD
- Backtracking Line Search:
  - Rarely used with SGD due to computational cost
  - More suitable for full (or large mini-batch) gradient descent

Selection of the batch size s

There are no "hard" guidelines to select the batch size s

In practice, especially when training deep neural networks, it is common to select a small mini-batch size (e.g., $s \in (16 - 128)$)

The mini-batch size influences the required step size and the convergence behavior of SGD, as illustrated next

Illustration of SGD - I

Consider a linear regression problem with $m = 50,000$ observations and $p = 20$ uncorrelated predictors

We track the number of iterations for different mini-batch sizes and also the relative ℓ_2 distance of the SGD estimate of the regression coefficient from the least squares solution (the gold standard), given by

$$\text{rel} - \ell_2 = \frac{\|\hat{\beta}_{\text{SGD}} - \hat{\beta}_{\text{LS}}\|_2}{\|\hat{\beta}_{\text{LS}}\|_2}$$

Illustration of SGD - II

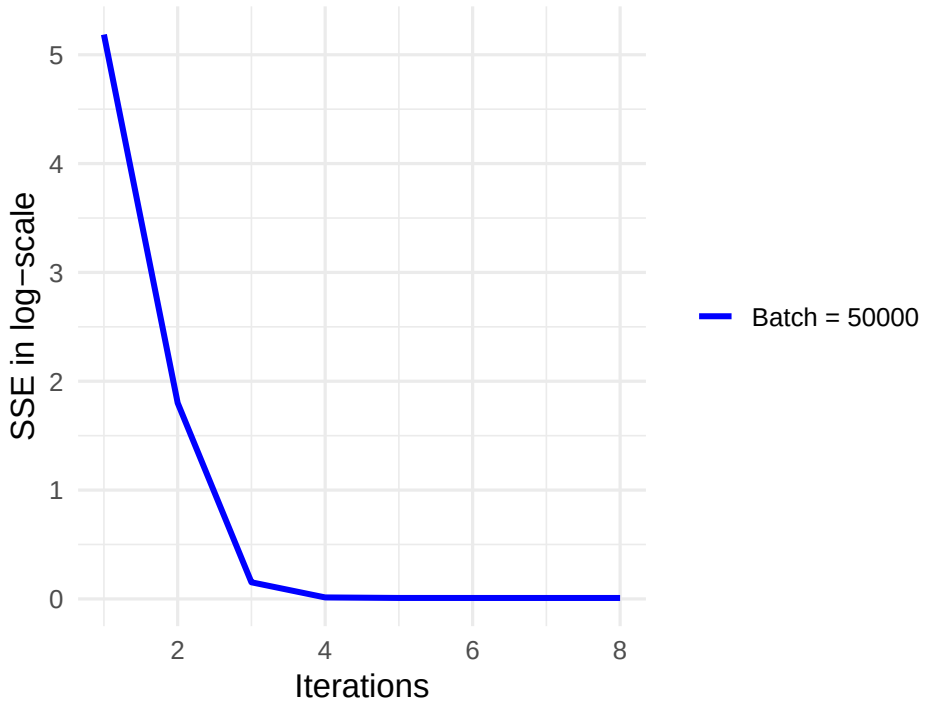

Figure 2: rel $-\ \ell_2 = 1.04 \times 10^{-6}$

A constant step $\eta = \frac{1}{L}$ is used; recall that the SSE(β) is a quadratic function with Lipschitz constant $L = \lambda_{\max}(\frac{1}{m}(X^\top X))$

Illustration of SGD - III

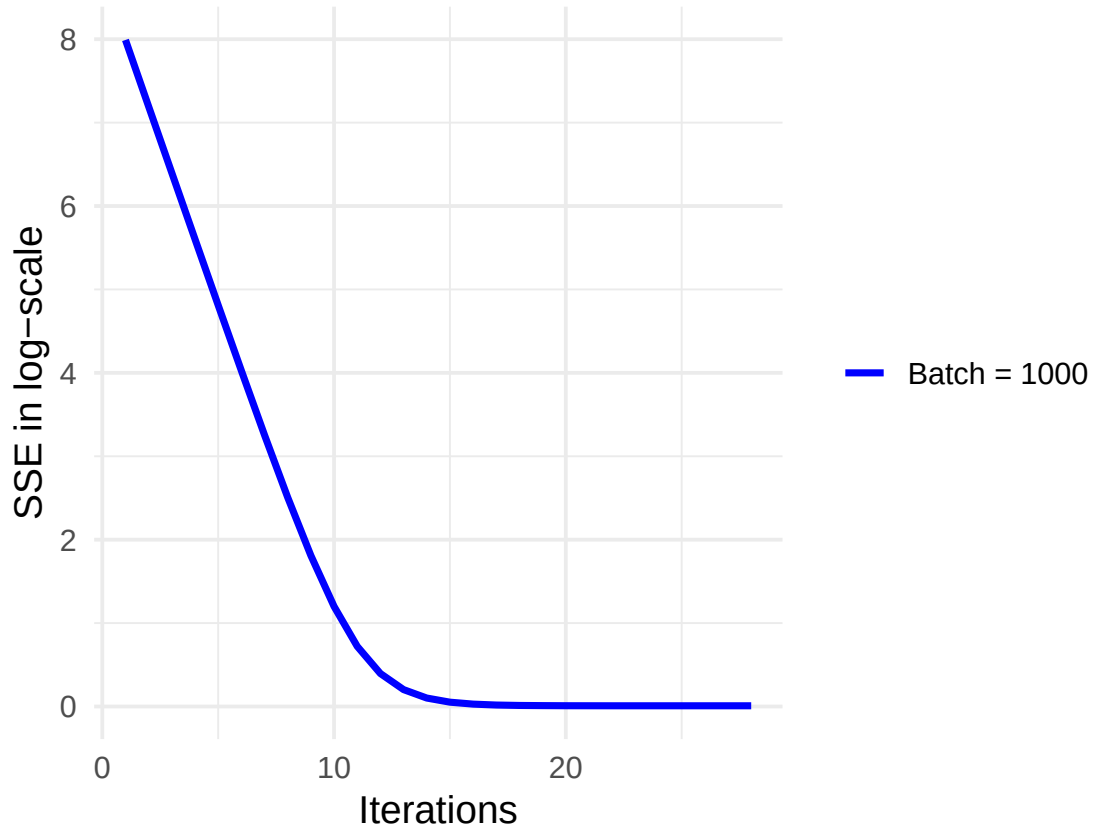

Figure 3: $\eta = \frac{1}{L} \times 10^{-2}, \text{rel} - \ell_2 = 2.65 \times 10^{-5}$

Illustration of SGD - III

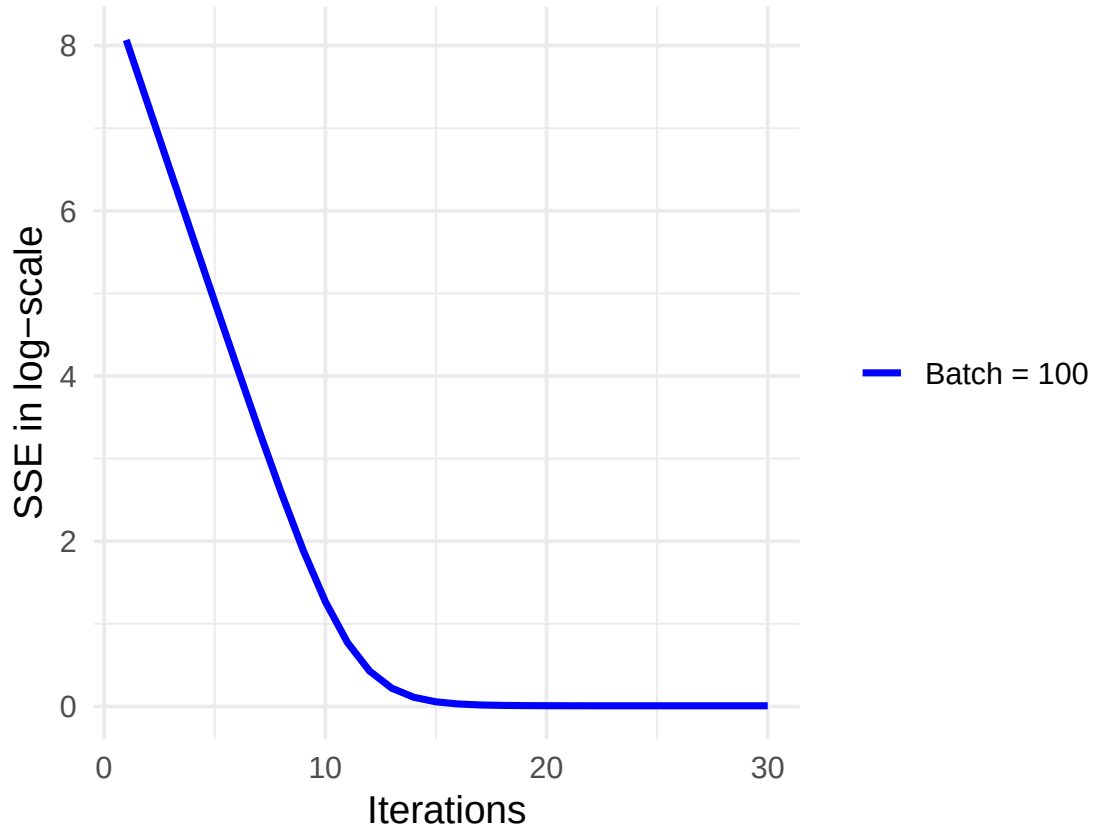

Figure 4: $\eta = \frac{1}{L} \times 10^{-3}, \text{rel} - \ell_2 = 2.95 \times 10^{-5}$

Illustration of SGD - IV

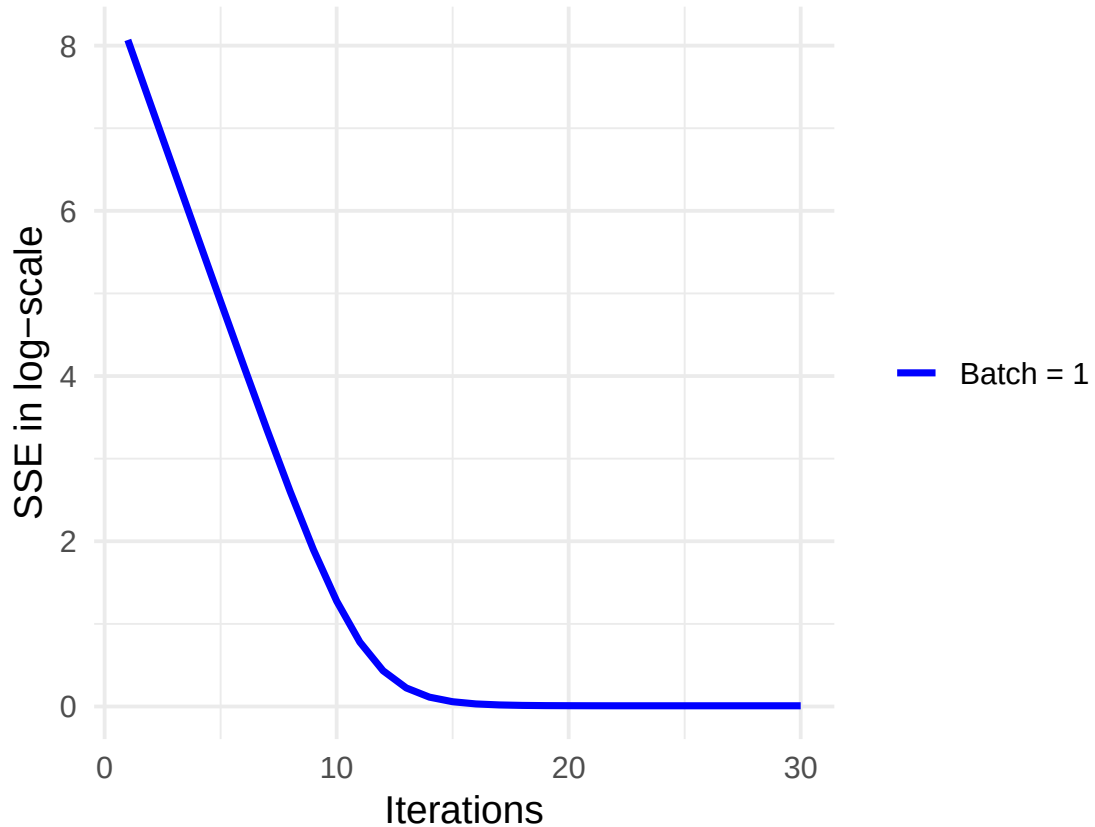

Figure 5: $\eta = \frac{1}{L} \times 10^{-5}, \text{rel} - \ell_2 = 1.81 \times 10^{-4}$

Illustration of SGD - V

Next, we consider a linear regression problem with $m = 50,000$ observations and $p = 20$ very correlated predictors

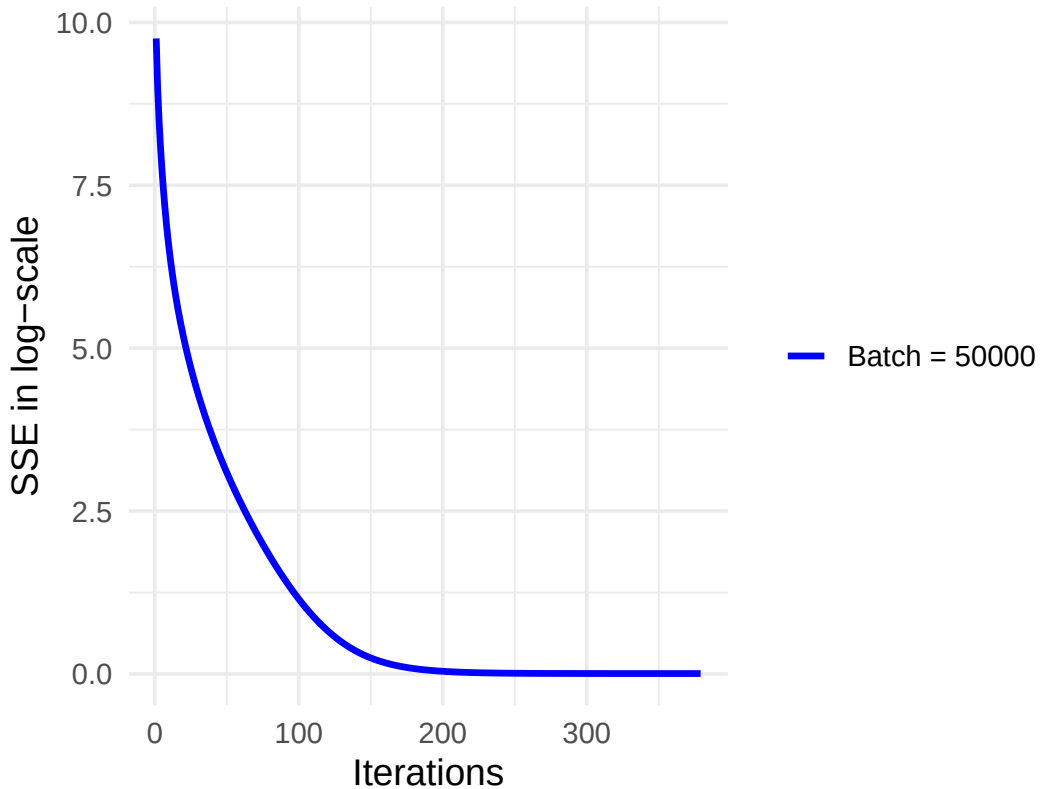

Figure 6: $\eta = \frac{1}{L}, \text{rel} - \ell_2 = 7.65 \times 10^{-5}$

Illustration of SGD - VI

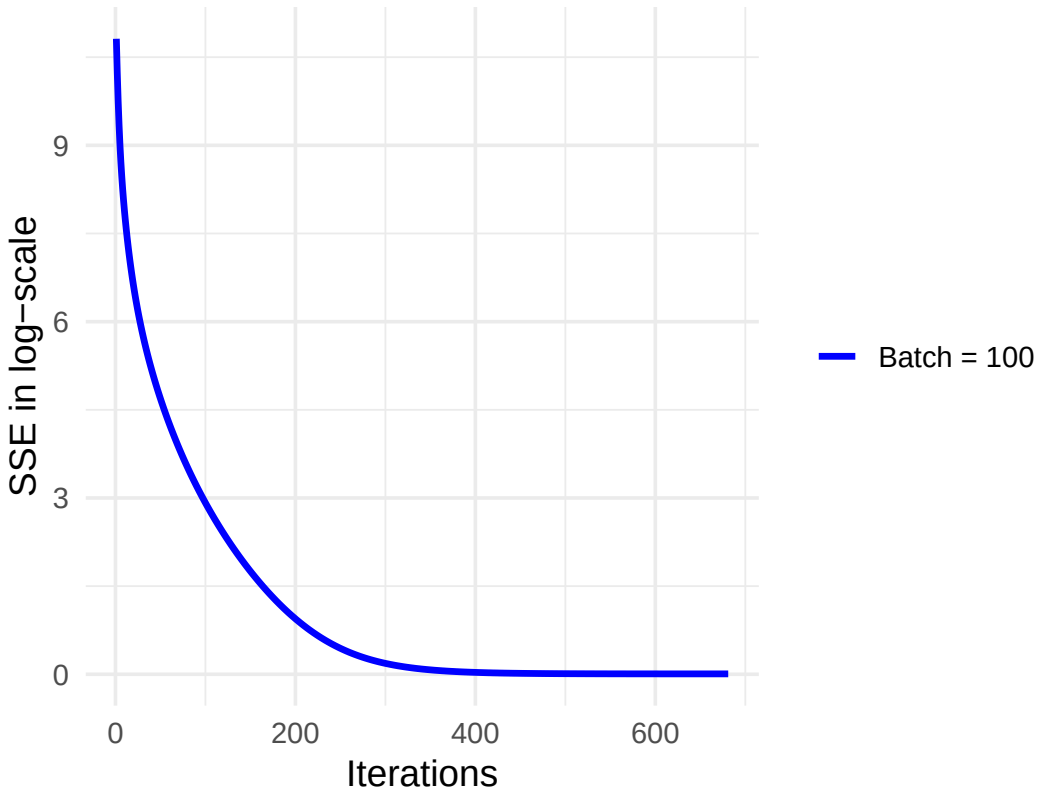

Figure 7: $\eta = \frac{1}{L} \times 10^{-3}, \text{rel} - \ell_2 = 1.49 \times 10^{-4}$

Illustration of SGD - VII

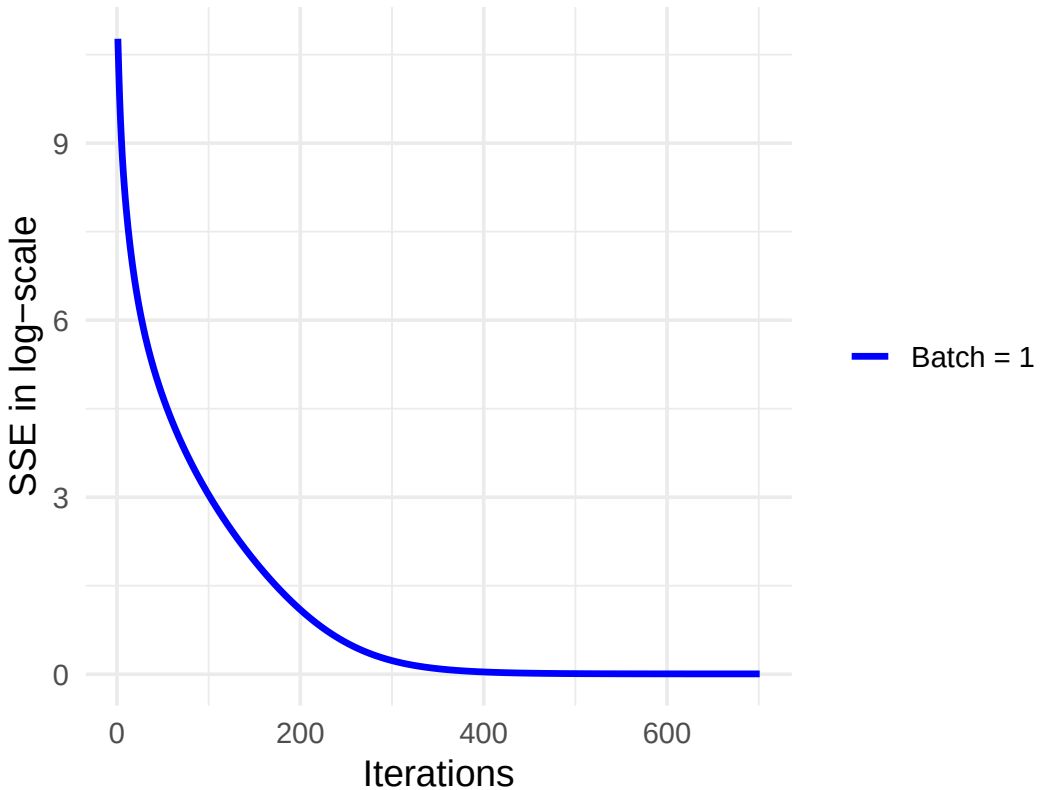

Figure 8: $\eta = \frac{1}{L} \times 10^{-5}, \text{rel} - \ell_2 = 1.82 \times 10^{-4}$

Remarks - I

The selection of the step size is critical

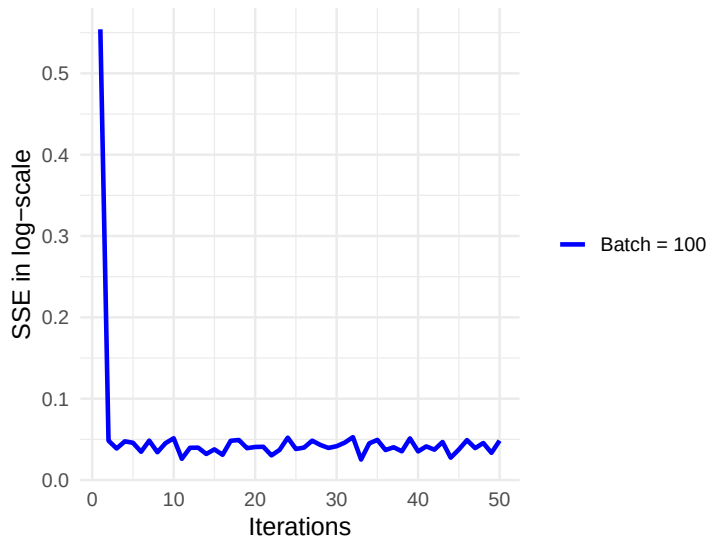

Figure 9: $\eta = \frac{1}{L} \times 10^{-2}$, rel $- \ell_2 = 5.7 \times 10^{-4}$

However, SGD never converges; it always exceeds the tolerance $= 10^{-6}$ for the convergence criterion

The plot shown above is truncated; in practice, an upper bound on the number of iterations is used in the code to avoid such phenomena

Remarks - II

To avoid such behavior and to also speed up convergence, the use of adaptive momentum methods is very popular

A Subtle Issue when Using SGD for Training Machine Learning Models - I

Recall the setting from slide 31

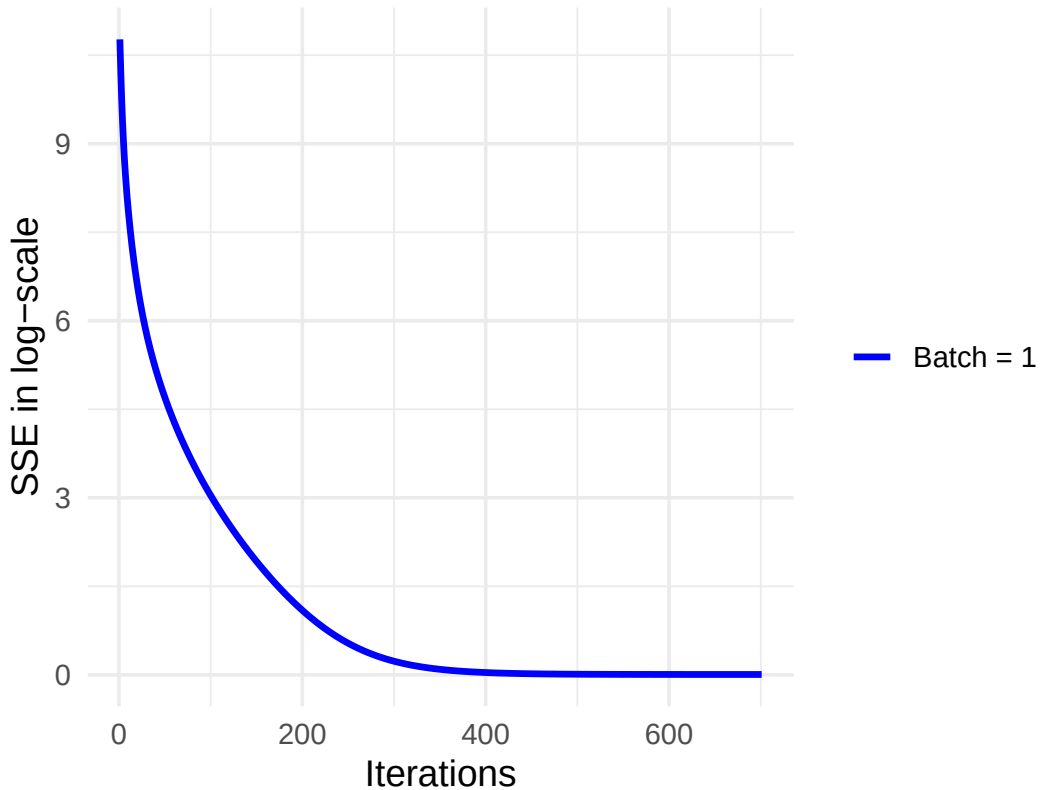

Figure 10: $\eta = \frac{1}{L} \times 10^{-5}, \text{rel} - \ell_2 = 1.82 \times 10^{-4}$

A Subtle Issue when Using SGD for Training Machine Learning Models - II

Based on the results, SGD required ~ 700 iterations to satisfy a tolerance of 10^{-6} for the stopping criterion

Given a mini-batch size of $s = 1$, SGD used information in the best case scenraio from 600 observations (out of the 50000 available in the data set) to estimate the regression coefficients β

The rel-ℓ_2 metric shows that the estimate $\hat{\beta}$ is accurate; i.e, very close to the least squares solution that used all 50K observations!

A Subtle Issue when Using SGD for Training Machine Learning Models - III

However, for more complex models—whose objective functions are not as "nice" as the sum of squared errors (SSE(β)), which is quadratic—one typically prefers to leverage the entire data set, rather than just a subset of it

This naturally raises the distinction between an epoch and a mini-batch SGD iteration

This distinction will be discussed and illustrated in the next lecture, where we will delve into the mechanism of training a feed-forward neural network using SGD, along with all its technical nuances

Appendix

Establishing that d_k in SGD is a descent direction in expectation

Then,

$$\begin{aligned}
\mathbb{E}\left(\nabla f_{I_k}(x_k)\right) &= \mathbb{E}\left(\frac{1}{s}\sum_{i\in I_k}\nabla f_i(x_k)\right) \qquad (22)\\
&= \mathbb{E}\left(\frac{1}{s}\sum_{i=1}^{m}\nabla f_i(x_k)\mathbf{1}_{\{1\in I_k\}}\right)\\
&= \frac{1}{s}\sum_{i=1}^{m}\nabla f_i(x_k)\mathbb{E}\left(\mathbf{1}_{\{1\in I_k\}}\right)\\
&= \frac{1}{s}\sum_{i=1}^{m}\nabla f_i(x_k)\frac{s}{m} = \frac{1}{m}\sum_{i=1}^{m}\nabla f_i(x_k) = \nabla f(x_k)
\end{aligned}$$

Hence, d_k is indeed a descent direction in expectation (over the random selection of the mini-batch)