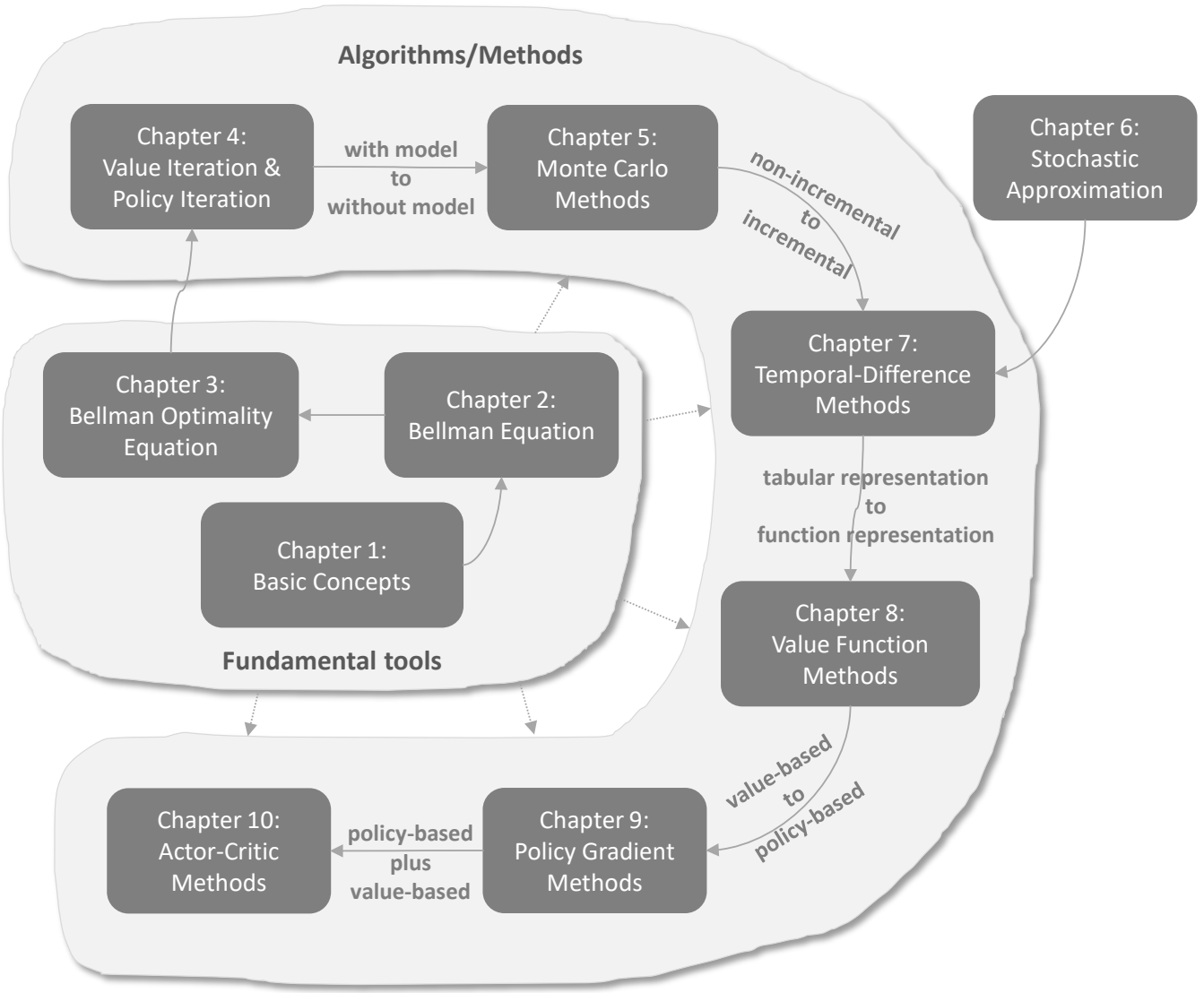
Lecture 6:
Stochastic Approximation
and
Stochastic Gradient Descent

Shiyu Zhao

Department of Artificial Intelligence

Westlake University

Outline


Algorithms/Methods
Chapter 4: Value Iteration & Policy Iteration
with model to without model
Chapter 5: Monte Carlo Methods
non-incremental to incremental
Chapter 6: Stochastic Approximation
Chapter 7: Temporal-Difference Methods
Chapter 3: Bellman Optimality Equation
Chapter 2: Bellman Equation
Chapter 1: Basic Concepts
Fundamental tools
tabular representation to function representation
Chapter 8: Value Function Methods
value-based to policy-based
Chapter 10: Actor-Critic Methods
policy-based plus value-based
Chapter 9: Policy Gradient Methods

Introduction

- In the last lecture, we introduced Monte-Carlo learning.
- In the next lecture, we will introduce temporal-difference (TD) learning.
- In this lecture, we press the pause button to get us better prepared. Why?
  - There is a knowledge gap!
  - The ideas and expressions of TD algorithms are very different from the algorithms we studied so far.
  - Many students who see the TD algorithms the first time many wonder why these algorithms were designed in the first place and why they work effectively.

Introduction

- In the last lecture, we introduced Monte-Carlo learning.
- In the next lecture, we will introduce temporal-difference (TD) learning.
- In this lecture, we press the pause button to get us better prepared. Why?
  - There is a knowledge gap!
  - The ideas and expressions of TD algorithms are very different from the algorithms we studied so far.
  - Many students who see the TD algorithms the first time many wonder why these algorithms were designed in the first place and why they work effectively.

Introduction

In this lecture,

- We fill the knowledge gap between the previous and upcoming lectures by introducing basic stochastic approximation (SA) algorithms.
- We will see in the next lecture that the temporal-difference algorithms are special SA algorithms. As a result, it will be much easier to understand these algorithms.
- We will also study the important algorithm of stochastic gradient descent (SGD).

Outline

Outline

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation, again

Revisit the mean estimation problem:

- Consider a random variable X.
- Suppose that we collected a sequence of iid samples $\{x_i\}_{i=1}^N$.
- Our aim is to estimate $\mathbb{E}[X]$.
- The expectation of X can be approximated by

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- This approximation is the basic idea of Monte Carlo estimation.
- We know that $\bar{x} \to \mathbb{E}[X]$ as $N \to \infty$.

Why do we care about mean estimation so much?

- Many quantities in RL such as action values and gradients are defined as expectations!

Motivating example: mean estimation

New question: how to calculate the mean $\bar{x}$?

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

We have two ways.

- **The first way**, which is trivial, is to collect all the samples and then calculate the average.
  - The drawback of such way is that, if the samples are collected one by one over a period of time, we have to wait until all the samples to be collected.
- **The second way** can avoid this drawback because it calculates the average in an incremental manner.

Motivating example: mean estimation

New question: how to calculate the mean $\bar{x}$?

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

We have two ways.

- **The first way**, which is trivial, is to collect all the samples and then calculate the average.
  - The drawback of such way is that, if the samples are collected one by one over a period of time, we have to wait until all the samples to be collected.
- **The second way** can avoid this drawback because it calculates the average in an incremental manner.

Motivating example: mean estimation

New question: how to calculate the mean $\bar{x}$?

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

We have two ways.

- **The first way**, which is trivial, is to collect all the samples and then calculate the average.
  - The drawback of such way is that, if the samples are collected one by one over a period of time, we have to wait until all the samples to be collected.
- **The second way** can avoid this drawback because it calculates the average in an incremental manner.

Motivating example: mean estimation

New question: how to calculate the mean $\bar{x}$?

$$\mathbb{E}[X] \approx \bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i.$$

We have two ways.

- **The first way**, which is trivial, is to collect all the samples and then calculate the average.
  - The drawback of such way is that, if the samples are collected one by one over a period of time, we have to wait until all the samples to be collected.
- **The second way** can avoid this drawback because it calculates the average in an incremental manner.

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + x_k\right)$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$\begin{aligned} w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i &= \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + x_k\right) \\ &= \frac{1}{k}((k-1)w_k + x_k) \end{aligned}$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$\begin{aligned} w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i &= \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + x_k\right) \\ &= \frac{1}{k}((k-1)w_k + x_k) = w_k - \frac{1}{k}(w_k - x_k). \end{aligned}$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

In particular, suppose

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i \quad k = 1, 2, \ldots$$

and hence

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \ldots$$

Then, w_{k+1} can be expressed in terms of w_k as

$$\begin{aligned} w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i &= \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + x_k\right) \\ &= \frac{1}{k}((k-1)w_k + x_k) = w_k - \frac{1}{k}(w_k - x_k). \end{aligned}$$

Therefore, we obtain the following iterative algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$w_1 = x_1,$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$w_1 = x_1,$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$w_1 = x_1,$$
$$w_2 = w_1 - \frac{1}{1}(w_1 - x_1) = x_1,$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$\begin{aligned}
w_1 &= x_1, \\
w_2 &= w_1 - \frac{1}{1}(w_1 - x_1) = x_1, \\
w_3 &= w_2 - \frac{1}{2}(w_2 - x_2) = x_1 - \frac{1}{2}(x_1 - x_2) = \frac{1}{2}(x_1 + x_2),
\end{aligned}$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$\begin{aligned}
w_1 &= x_1, \\
w_2 &= w_1 - \frac{1}{1}(w_1 - x_1) = x_1, \\
w_3 &= w_2 - \frac{1}{2}(w_2 - x_2) = x_1 - \frac{1}{2}(x_1 - x_2) = \frac{1}{2}(x_1 + x_2), \\
w_4 &= w_3 - \frac{1}{3}(w_3 - x_3) = \frac{1}{3}(x_1 + x_2 + x_3),
\end{aligned}$$

Motivating example: mean estimation

Verification: we can use

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k)$$

to calculate the mean $\bar{x}$ incrementally:

$$\begin{aligned}
w_1 &= x_1, \\
w_2 &= w_1 - \frac{1}{1}(w_1 - x_1) = x_1, \\
w_3 &= w_2 - \frac{1}{2}(w_2 - x_2) = x_1 - \frac{1}{2}(x_1 - x_2) = \frac{1}{2}(x_1 + x_2), \\
w_4 &= w_3 - \frac{1}{3}(w_3 - x_3) = \frac{1}{3}(x_1 + x_2 + x_3), \\
&\vdots \\
w_{k+1} &= \frac{1}{k}\sum_{i=1}^{k} x_i.
\end{aligned}$$

Remarks about this algorithm:

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

- An advantage of this algorithm is that it is incremental. A mean estimate can be obtained immediately once a sample is received. Then, the mean estimate can be used for other purposes immediately.
- The mean estimate is not accurate in the beginning due to insufficient samples. However, it is better than nothing. As more samples are obtained, the estimate can be improved gradually (that is $w_k \to \mathbb{E}[X]$ as $k \to \infty$).

Motivating example: mean estimation

Furthermore, consider an algorithm with a more general expression:

$$w_{k+1} = w_k - \alpha_k(w_k - x_k),$$

where $1/k$ is replaced by $\alpha_k > 0$.

- Does this algorithm still converge to the mean $\mathbb{E}[X]$? We will show that the answer is yes if $\{\alpha_k\}$ satisfy some mild conditions.
- We will also show that this algorithm is a special stochastic approximation algorithm and also a special stochastic gradient descent algorithm.
- In the next lecture, we will see that the temporal-difference algorithms have similar (but more complex) expressions.

Motivating example: mean estimation

Furthermore, consider an algorithm with a more general expression:

$$w_{k+1} = w_k - \alpha_k(w_k - x_k),$$

where $1/k$ is replaced by $\alpha_k > 0$.

- Does this algorithm still converge to the mean $\mathbb{E}[X]$? We will show that the answer is yes if $\{\alpha_k\}$ satisfy some mild conditions.
- We will also show that this algorithm is a special stochastic approximation algorithm and also a special stochastic gradient descent algorithm.
- In the next lecture, we will see that the temporal-difference algorithms have similar (but more complex) expressions.

Motivating example: mean estimation

Furthermore, consider an algorithm with a more general expression:

$$w_{k+1} = w_k - \alpha_k(w_k - x_k),$$

where $1/k$ is replaced by $\alpha_k > 0$.

- Does this algorithm still converge to the mean $\mathbb{E}[X]$? We will show that the answer is yes if $\{\alpha_k\}$ satisfy some mild conditions.
- We will also show that this algorithm is a special stochastic approximation algorithm and also a special stochastic gradient descent algorithm.
- In the next lecture, we will see that the temporal-difference algorithms have similar (but more complex) expressions.

Motivating example: mean estimation

Furthermore, consider an algorithm with a more general expression:

$$w_{k+1} = w_k - \alpha_k(w_k - x_k),$$

where $1/k$ is replaced by $\alpha_k > 0$.

- Does this algorithm still converge to the mean $\mathbb{E}[X]$? We will show that the answer is yes if $\{\alpha_k\}$ satisfy some mild conditions.
- We will also show that this algorithm is a special stochastic approximation algorithm and also a special stochastic gradient descent algorithm.
- In the next lecture, we will see that the temporal-difference algorithms have similar (but more complex) expressions.

Outline

Outline

Robbins-Monro algorithm

Stochastic approximation (SA):

- SA refers to a broad class of stochastic iterative algorithms solving root finding or optimization problems.
- Compared to many other root-finding algorithms such as gradient-based methods, SA is powerful in the sense that it does *not* require to know the expression of the objective function nor its derivative.

Robbins-Monro (RM) algorithm:

- The is a pioneering work in the field of stochastic approximation.
- The famous stochastic gradient descent algorithm is a special form of the RM algorithm.
- It can be used to analyze the mean estimation algorithms introduced in the beginning.

Robbins-Monro algorithm

Stochastic approximation (SA):

- SA refers to a broad class of stochastic iterative algorithms solving root finding or optimization problems.
- Compared to many other root-finding algorithms such as gradient-based methods, SA is powerful in the sense that it does *not* require to know the expression of the objective function nor its derivative.

Robbins-Monro (RM) algorithm:

- The is a pioneering work in the field of stochastic approximation.
- The famous stochastic gradient descent algorithm is a special form of the RM algorithm.
- It can be used to analyze the mean estimation algorithms introduced in the beginning.

Robbins-Monro algorithm

Stochastic approximation (SA):

- SA refers to a broad class of stochastic iterative algorithms solving root finding or optimization problems.
- Compared to many other root-finding algorithms such as gradient-based methods, SA is powerful in the sense that it does *not* require to know the expression of the objective function nor its derivative.

Robbins-Monro (RM) algorithm:

- The is a pioneering work in the field of stochastic approximation.
- The famous stochastic gradient descent algorithm is a special form of the RM algorithm.
- It can be used to analyze the mean estimation algorithms introduced in the beginning.

Robbins-Monro algorithm – Problem statement

Problem statement: Suppose we would like to find the root of the equation

$$g(w) = 0,$$

where $w \in \mathbb{R}$ is the variable to be solved and $g : \mathbb{R} \to \mathbb{R}$ is a function.

- Many problems can be eventually converted to this root finding problem. For example, suppose $J(w)$ is an objective function to be minimized. Then, the optimization problem can be converged to

$$g(w) = \nabla_w J(w) = 0$$

- Note that an equation like $g(w) = c$ with c as a constant can also be converted to the above equation by rewriting $g(w) - c$ as a new function.

Problem statement: Suppose we would like to find the root of the equation

$$g(w) = 0,$$

where $w \in \mathbb{R}$ is the variable to be solved and $g : \mathbb{R} \to \mathbb{R}$ is a function.

- Many problems can be eventually converted to this root finding problem. For example, suppose $J(w)$ is an objective function to be minimized. Then, the optimization problem can be converged to

$$g(w) = \nabla_w J(w) = 0$$

- Note that an equation like $g(w) = c$ with c as a constant can also be converted to the above equation by rewriting $g(w) - c$ as a new function.

Robbins-Monro algorithm – Problem statement

Problem statement: Suppose we would like to find the root of the equation

$$g(w) = 0,$$

where $w \in \mathbb{R}$ is the variable to be solved and $g : \mathbb{R} \to \mathbb{R}$ is a function.

- Many problems can be eventually converted to this root finding problem. For example, suppose $J(w)$ is an objective function to be minimized. Then, the optimization problem can be converged to

$$g(w) = \nabla_w J(w) = 0$$

- Note that an equation like $g(w) = c$ with c as a constant can also be converted to the above equation by rewriting $g(w) - c$ as a new function.

Robbins-Monro algorithm – Problem statement

How to calculate the root of $g(w) = 0$?

- Model-based: If the expression of g is known, there are many numerical algorithms that can solve this problem.
- Model-free: What if the expression of the function g is unknown? For example, the function is represented by an artificial neuron network.

Robbins-Monro algorithm – Problem statement

How to calculate the root of $g(w) = 0$?

- Model-based: If the expression of g is known, there are many numerical algorithms that can solve this problem.
- Model-free: What if the expression of the function g is unknown? For example, the function is represented by an artificial neuron network.

Robbins-Monro algorithm – Problem statement

How to calculate the root of $g(w) = 0$?

- Model-based: If the expression of g is known, there are many numerical algorithms that can solve this problem.
- Model-free: What if the expression of the function g is unknown? For example, the function is represented by an artificial neuron network.

Robbins-Monro algorithm – The algorithm

The Robbins-Monro (RM) algorithm that can solve this problem is as follows:

$$w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k), \qquad k = 1, 2, 3, \dots$$

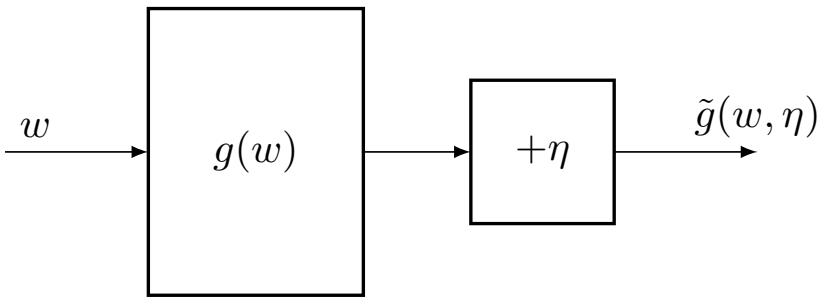
where

- w_k is the kth estimate of the root
- $\tilde{g}(w_k, \eta_k) = g(w_k) + \eta_k$ is the kth noisy observation
  - Why noise here? For example, consider a random sampling x of X.
- a_k is a positive coefficient.

Robbins-Monro algorithm – The algorithm

This algorithm relies on data instead of model:

- Input sequence: $\{w_k\}$
- Output sequence (noisy): $\{\tilde{g}(w_k, \eta_k)\}$



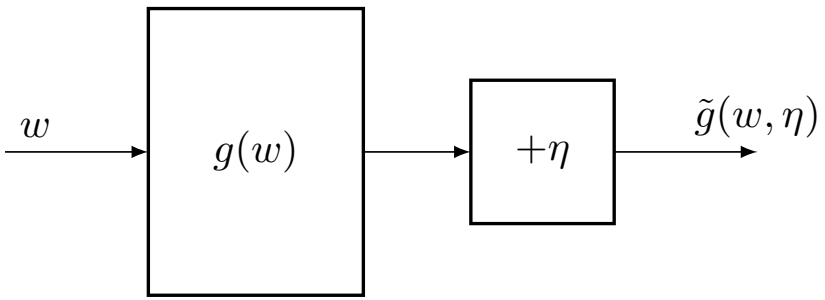
Philosophy: without model, we need data!

- The function $g(w)$ is viewed as a black box.
- The model here refers to the expression of the function.

Robbins-Monro algorithm – The algorithm

This algorithm relies on data instead of model:

- Input sequence: $\{w_k\}$
- Output sequence (noisy): $\{\tilde{g}(w_k, \eta_k)\}$



Philosophy: without model, we need data!

- The function $g(w)$ is viewed as a black box.
- The model here refers to the expression of the function.

Outline

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$

$$w_2 = w_1 - a_1 g(w_1) = 20 - 0.5 * 10 = 15$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$

$$w_2 = w_1 - a_1 g(w_1) = 20 - 0.5 * 10 = 15 \Longrightarrow g(w_2) = 5$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$
$$w_2 = w_1 - a_1 g(w_1) = 20 - 0.5 * 10 = 15 \Longrightarrow g(w_2) = 5$$
$$w_3 = w_2 - a_2 g(w_2) = 15 - 0.5 * 5 = 12.5$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$
$$w_2 = w_1 - a_1 g(w_1) = 20 - 0.5 * 10 = 15 \Longrightarrow g(w_2) = 5$$
$$w_3 = w_2 - a_2 g(w_2) = 15 - 0.5 * 5 = 12.5 \Longrightarrow g(w_3) = 2.5$$

Robbins-Monro algorithm – Illustrative examples

Toy example: manually solve $g(w) = w - 10$ using the RM algorithm.
Set: $w_1 = 20$, $a_k \equiv 0.5$, $\eta_k = 0$ (i.e., no observation error)

$$w_1 = 20 \Longrightarrow g(w_1) = 10$$

$$w_2 = w_1 - a_1 g(w_1) = 20 - 0.5 * 10 = 15 \Longrightarrow g(w_2) = 5$$

$$w_3 = w_2 - a_2 g(w_2) = 15 - 0.5 * 5 = 12.5 \Longrightarrow g(w_3) = 2.5$$

$$\vdots$$

$$w_k \to 10$$

Outline

Robbins-Monro algorithm – Convergence properties

Why can the RM algorithm find the root of $g(w) = 0$?

- First present an illustrative example.
- Second give the rigorous convergence analysis.

Robbins-Monro algorithm – Convergence properties

An illustrative example:

- $g(w) = \tanh(w - 1)$
- The true root of $g(w) = 0$ is $w^* = 1$.
- Parameters: $w_1 = 3$, $a_k = 1/k$, $\eta_k \equiv 0$ (no noise for the sake of simplicity)

The RM algorithm in this case is

$$w_{k+1} = w_k - a_k g(w_k)$$

since $\tilde{g}(w_k, \eta_k) = g(w_k)$ when $\eta_k = 0$.

Robbins-Monro algorithm – Convergence properties

An illustrative example:

- $g(w) = \tanh(w - 1)$
- The true root of $g(w) = 0$ is $w^* = 1$.
- Parameters: $w_1 = 3$, $a_k = 1/k$, $\eta_k \equiv 0$ (no noise for the sake of simplicity)

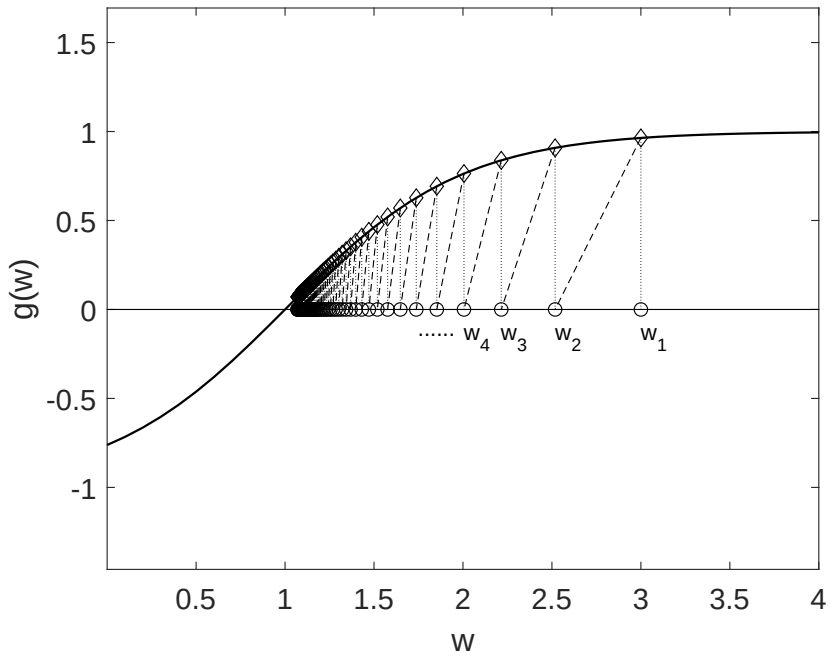
The RM algorithm in this case is

$$w_{k+1} = w_k - a_k g(w_k)$$

since $\tilde{g}(w_k, \eta_k) = g(w_k)$ when $\eta_k = 0$.

Robbins-Monro algorithm – Convergence properties

Simulation result: w_k converges to the true root $w^* = 1$.

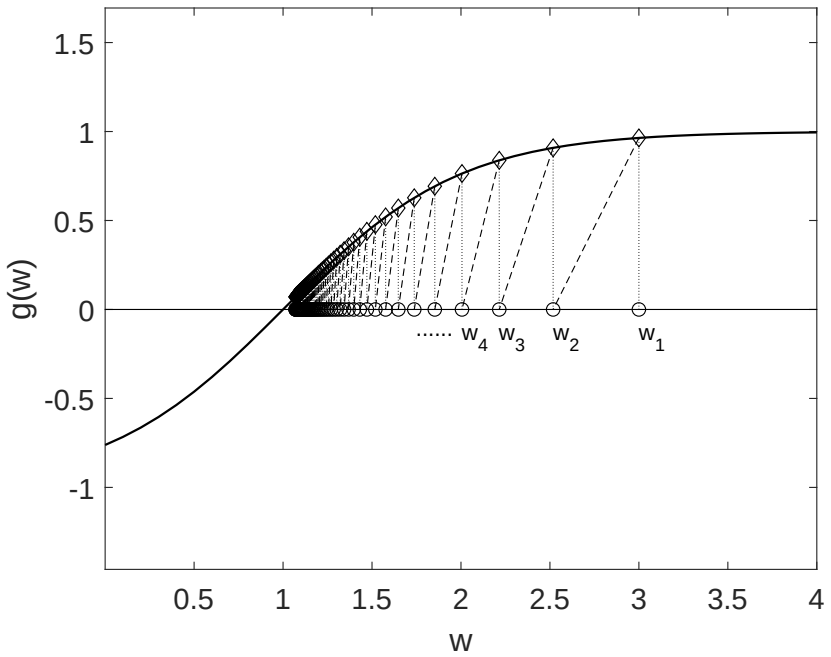


Intuition: w_{k+1} is closer to w^* than w_k.

- When $w_k > w^*$, we have $g(w_k) > 0$. Then, $w_{k+1} = w_k - a_k g(w_k) < w_k$ and hence w_{k+1} is closer to w^* than w_k.
- When $w_k < w^*$, we have $g(w_k) < 0$. Then, $w_{k+1} = w_k - a_k g(w_k) > w_k$ and w_{k+1} is closer to w^* than w_k.

Robbins-Monro algorithm – Convergence properties

Simulation result: w_k converges to the true root $w^* = 1$.

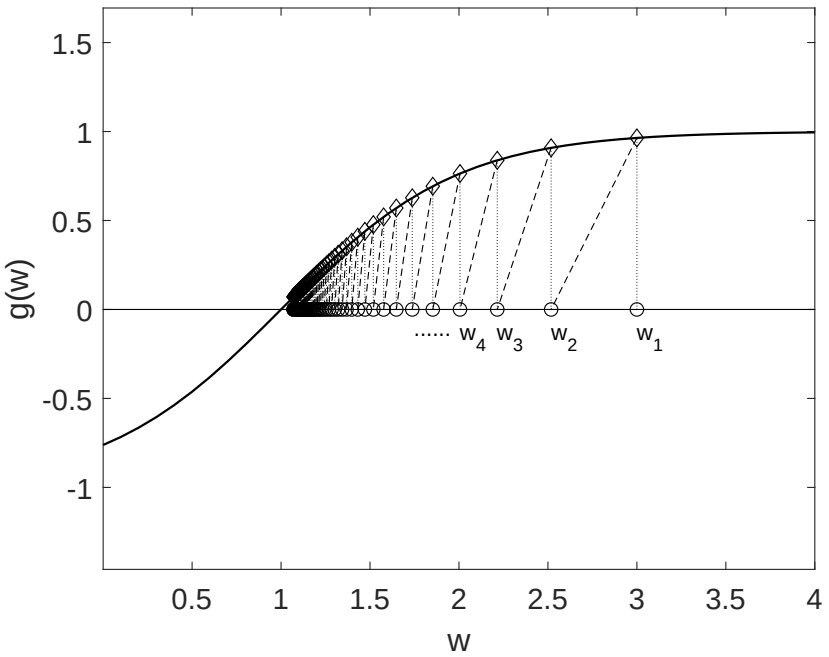


Intuition: w_{k+1} is closer to w^* than w_k.

- When $w_k > w^*$, we have $g(w_k) > 0$. Then, $w_{k+1} = w_k - a_k g(w_k) < w_k$ and hence w_{k+1} is closer to w^* than w_k.
- When $w_k < w^*$, we have $g(w_k) < 0$. Then, $w_{k+1} = w_k - a_k g(w_k) > w_k$ and w_{k+1} is closer to w^* than w_k.

Robbins-Monro algorithm – Convergence properties

Simulation result: w_k converges to the true root $w^* = 1$.

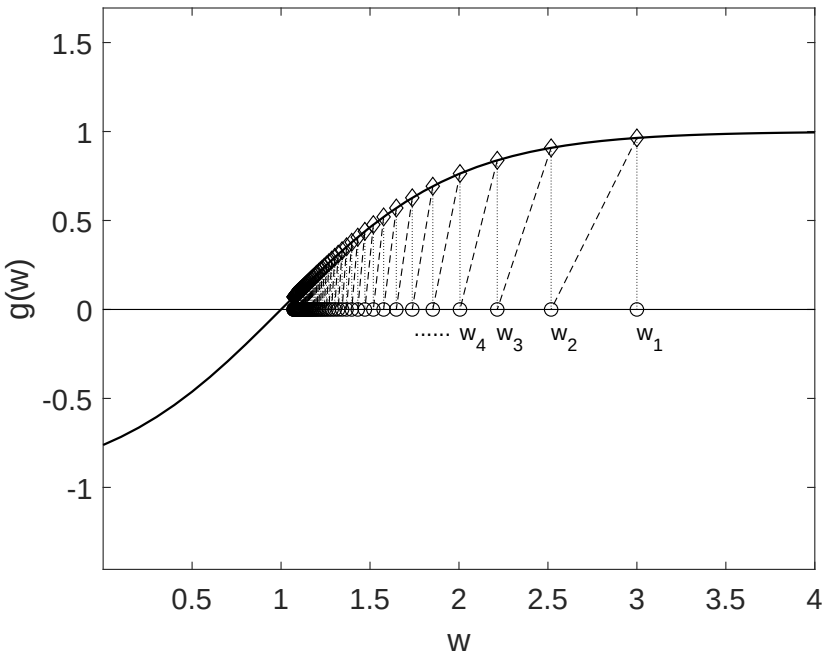


Intuition: w_{k+1} is closer to w^* than w_k.

- When $w_k > w^*$, we have $g(w_k) > 0$. Then, $w_{k+1} = w_k - a_k g(w_k) < w_k$ and hence w_{k+1} is closer to w^* than w_k.
- When $w_k < w^*$, we have $g(w_k) < 0$. Then, $w_{k+1} = w_k - a_k g(w_k) > w_k$ and w_{k+1} is closer to w^* than w_k.

Robbins-Monro algorithm – Convergence properties

Simulation result: w_k converges to the true root $w^* = 1$.



Intuition: w_{k+1} is closer to w^* than w_k.

- When $w_k > w^*$, we have $g(w_k) > 0$. Then, $w_{k+1} = w_k - a_k g(w_k) < w_k$ and hence w_{k+1} is closer to w^* than w_k.
- When $w_k < w^*$, we have $g(w_k) < 0$. Then, $w_{k+1} = w_k - a_k g(w_k) > w_k$ and w_{k+1} is closer to w^* than w_k.

Robbins-Monro algorithm – Convergence properties

The above analysis is intuitive, but not rigorous. A rigorous convergence result is given below.

Theorem (Robbins-Monro Theorem)

In the Robbins-Monro algorithm, if

1) $0 < c_1 \leq \nabla_w g(w) \leq c_2$ *for all* w*;*

2) $\sum_{k=1}^{\infty} a_k = \infty$ *and* $\sum_{k=1}^{\infty} a_k^2 < \infty$*;*

3) $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ *and* $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$*;*

where $\mathcal{H}_k = \{w_k, w_{k-1}, \dots\}$*, then* w_k *converges with probability 1 (w.p.1) to the root* w^* *satisfying* $g(w^*) = 0$*.*

Robbins-Monro algorithm – Convergence properties

Explanation of the three conditions:

- **Condition 1:** $0 < c_1 \leq \nabla_w g(w) \leq c_2$ for all w
  - g should be monotonically increasing,which ensures that the root of $g(w) = 0$ exists and is unique.
  - The gradient is bounded from the above.
  - This condition is not strict. Consider the example $g(w) = \nabla_w J(w) = 0$. This condition requires that $J(w)$ is convex.

- **Condition 2:** $\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$
  - $\sum_{k=1}^{\infty} a_k^2 < \infty$ ensures that a_k converges to zero as $k \to \infty$.
  - $\sum_{k=1}^{\infty} a_k = \infty$ ensures that a_k do not converge to zero too fast.

- **Condition 3:** $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$
  - A special yet common case is that $\{\eta_k\}$ is an iid stochastic sequence satisfying $\mathbb{E}[\eta_k] = 0$ and $\mathbb{E}[\eta_k^2] < \infty$. The observation error η_k is not required to be Gaussian.

Robbins-Monro algorithm – Convergence properties

Explanation of the three conditions:

- **Condition 1:** $0 < c_1 \leq \nabla_w g(w) \leq c_2$ for all w
  - g should be monotonically increasing,which ensures that the root of $g(w) = 0$ exists and is unique.
  - The gradient is bounded from the above.
  - This condition is not strict. Consider the example $g(w) = \nabla_w J(w) = 0$. This condition requires that $J(w)$ is convex.

- **Condition 2:** $\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$
  - $\sum_{k=1}^{\infty} a_k^2 < \infty$ ensures that a_k converges to zero as $k \to \infty$.
  - $\sum_{k=1}^{\infty} a_k = \infty$ ensures that a_k do not converge to zero too fast.

- **Condition 3:** $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$
  - A special yet common case is that $\{\eta_k\}$ is an iid stochastic sequence satisfying $\mathbb{E}[\eta_k] = 0$ and $\mathbb{E}[\eta_k^2] < \infty$. The observation error η_k is not required to be Gaussian.

Robbins-Monro algorithm – Convergence properties

Explanation of the three conditions:

- **Condition 1:** $0 < c_1 \leq \nabla_w g(w) \leq c_2$ for all w
  - g should be monotonically increasing,which ensures that the root of $g(w) = 0$ exists and is unique.
  - The gradient is bounded from the above.
  - This condition is not strict. Consider the example $g(w) = \nabla_w J(w) = 0$. This condition requires that $J(w)$ is convex.

- **Condition 2:** $\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$
  - $\sum_{k=1}^{\infty} a_k^2 < \infty$ ensures that a_k converges to zero as $k \to \infty$.
  - $\sum_{k=1}^{\infty} a_k = \infty$ ensures that a_k do not converge to zero too fast.

- **Condition 3:** $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$
  - A special yet common case is that $\{\eta_k\}$ is an iid stochastic sequence satisfying $\mathbb{E}[\eta_k] = 0$ and $\mathbb{E}[\eta_k^2] < \infty$. The observation error η_k is not required to be Gaussian.

Robbins-Monro algorithm – Convergence properties

Explanation of the three conditions:

- **Condition 1:** $0 < c_1 \leq \nabla_w g(w) \leq c_2$ for all w
  - g should be monotonically increasing,which ensures that the root of $g(w) = 0$ exists and is unique.
  - The gradient is bounded from the above.
  - This condition is not strict. Consider the example $g(w) = \nabla_w J(w) = 0$. This condition requires that $J(w)$ is convex.

- **Condition 2:** $\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$
  - $\sum_{k=1}^{\infty} a_k^2 < \infty$ ensures that a_k converges to zero as $k \to \infty$.
  - $\sum_{k=1}^{\infty} a_k = \infty$ ensures that a_k do not converge to zero too fast.

- **Condition 3:** $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$
  - A special yet common case is that $\{\eta_k\}$ is an iid stochastic sequence satisfying $\mathbb{E}[\eta_k] = 0$ and $\mathbb{E}[\eta_k^2] < \infty$. The observation error η_k is not required to be Gaussian.

Examine Condition 2 more closely:

$$\sum_{k=1}^{\infty} a_k^2 < \infty \qquad \sum_{k=1}^{\infty} a_k = \infty$$

- First, $\sum_{k=1}^{\infty} a_k^2 < \infty$ indicates that $a_k \to 0$ as $k \to \infty$.

Why is this condition important?

Since

$$w_{k+1} - w_k = -a_k \tilde{g}(w_k, \eta_k),$$

- If $a_k \to 0$, then $a_k \tilde{g}(w_k, \eta_k) \to 0$ and hence $w_{k+1} - w_k \to 0$.
- We need the fact that $w_{k+1} - w_k \to 0$ if w_k converges eventually.
- If $w_k \to w^*$, $g(w_k) \to 0$ and $\tilde{g}(w_k, \eta_k)$ is dominant by η_k.

Examine Condition 2 more closely:

$$\sum_{k=1}^{\infty} a_k^2 < \infty \qquad \sum_{k=1}^{\infty} a_k = \infty$$

- First, $\sum_{k=1}^{\infty} a_k^2 < \infty$ indicates that $a_k \to 0$ as $k \to \infty$.

  Why is this condition important?

  Since

  $$w_{k+1} - w_k = -a_k \tilde{g}(w_k, \eta_k),$$

  - If $a_k \to 0$, then $a_k \tilde{g}(w_k, \eta_k) \to 0$ and hence $w_{k+1} - w_k \to 0$.
  - We need the fact that $w_{k+1} - w_k \to 0$ if w_k converges eventually.
  - If $w_k \to w^*$, $g(w_k) \to 0$ and $\tilde{g}(w_k, \eta_k)$ is dominant by η_k.

Robbins-Monro algorithm – Convergence properties

Examine Condition 2 more closely:

$$\sum_{k=1}^{\infty} a_k^2 < \infty \qquad \sum_{k=1}^{\infty} a_k = \infty$$

- Second, $\sum_{k=1}^{\infty} a_k = \infty$ indicates that a_k should not converge to zero too fast.

Why is this condition important?

Summarizing $w_2 = w_1 - a_1\tilde{g}(w_1, \eta_1)$, $w_3 = w_2 - a_2\tilde{g}(w_2, \eta_2)$, $\ldots$, $w_{k+1} = w_k - a_k\tilde{g}(w_k, \eta_k)$ leads to

$$w_1 - w_\infty = \sum_{k=1}^{\infty} a_k\tilde{g}(w_k, \eta_k).$$

Suppose $w_\infty = w^*$. If $\sum_{k=1}^{\infty} a_k < \infty$, then $\sum_{k=1}^{\infty} a_k\tilde{g}(w_k, \eta_k)$ may be bounded. Then, if the initial guess w_1 is chosen arbitrarily far away from w^*, then the above equality would be invalid.

Examine Condition 2 more closely:

$$\sum_{k=1}^{\infty} a_k^2 < \infty \qquad \sum_{k=1}^{\infty} a_k = \infty$$

- Second, $\sum_{k=1}^{\infty} a_k = \infty$ indicates that a_k should not converge to zero too fast.

  Why is this condition important?
  Summarizing $w_2 = w_1 - a_1\tilde{g}(w_1, \eta_1)$, $w_3 = w_2 - a_2\tilde{g}(w_2, \eta_2)$, $\ldots$, $w_{k+1} = w_k - a_k\tilde{g}(w_k, \eta_k)$ leads to

$$w_1 - w_\infty = \sum_{k=1}^{\infty} a_k\tilde{g}(w_k, \eta_k).$$

  Suppose $w_\infty = w^*$. If $\sum_{k=1}^{\infty} a_k < \infty$, then $\sum_{k=1}^{\infty} a_k\tilde{g}(w_k, \eta_k)$ may be bounded. Then, if the initial guess w_1 is chosen arbitrarily far away from w^*, then the above equality would be invalid.

Robbins-Monro algorithm – Convergence properties

What $\{a_k\}$ satisfies the two conditions? $\sum_{k=1}^{\infty} a_k^2 < \infty, \sum_{k=1}^{\infty} a_k = \infty$

One typical sequence is

$$a_k = \frac{1}{k}$$

- It satisfies $\sum_{k=1}^{\infty} a_k = \infty$ since

$$\lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \ln n \right) = \kappa,$$

  where $\kappa \approx 0.577$ is called the Euler-Mascheroni constant (also called Euler's constant).

- It satisfies $\sum_{k=1}^{\infty} a_k^2 < \infty$ since

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} < \infty.$$

  The limit $\sum_{k=1}^{\infty} 1/k^2$ also has a specific name in the number theory: Basel problem.

Robbins-Monro algorithm – Convergence properties

What $\{a_k\}$ satisfies the two conditions? $\sum_{k=1}^{\infty} a_k^2 < \infty, \sum_{k=1}^{\infty} a_k = \infty$

One typical sequence is

$$a_k = \frac{1}{k}$$

- It satisfies $\sum_{k=1}^{\infty} a_k = \infty$ since

$$\lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \ln n \right) = \kappa,$$

where $\kappa \approx 0.577$ is called the Euler-Mascheroni constant (also called Euler's constant).

- It satisfies $\sum_{k=1}^{\infty} a_k^2 < \infty$ since

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} < \infty.$$

The limit $\sum_{k=1}^{\infty} 1/k^2$ also has a specific name in the number theory: Basel problem.

Robbins-Monro algorithm – Convergence properties

What $\{a_k\}$ satisfies the two conditions? $\sum_{k=1}^{\infty} a_k^2 < \infty, \sum_{k=1}^{\infty} a_k = \infty$

One typical sequence is

$$a_k = \frac{1}{k}$$

- It satisfies $\sum_{k=1}^{\infty} a_k = \infty$ since

$$\lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \ln n \right) = \kappa,$$

where $\kappa \approx 0.577$ is called the Euler-Mascheroni constant (also called Euler's constant).

- It satisfies $\sum_{k=1}^{\infty} a_k^2 < \infty$ since

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} < \infty.$$

The limit $\sum_{k=1}^{\infty} 1/k^2$ also has a specific name in the number theory: Basel problem.

What $\{a_k\}$ satisfies the two conditions? $\sum_{k=1}^{\infty} a_k^2 < \infty, \sum_{k=1}^{\infty} a_k = \infty$

One typical sequence is

$$a_k = \frac{1}{k}$$

- It satisfies $\sum_{k=1}^{\infty} a_k = \infty$ since

$$\lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \ln n \right) = \kappa,$$

where $\kappa \approx 0.577$ is called the Euler-Mascheroni constant (also called Euler's constant).

- It satisfies $\sum_{k=1}^{\infty} a_k^2 < \infty$ since

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} < \infty.$$

The limit $\sum_{k=1}^{\infty} 1/k^2$ also has a specific name in the number theory: Basel problem.

Outline

Robbins-Monro algorithm – Apply to mean estimation

Recall that

$$w_{k+1} = w_k + \alpha_k(x_k - w_k).$$

is the mean estimation algorithm introduced at the beginning of this lecture.

- If $\alpha_k = 1/k$, then $w_{k+1} = 1/k \sum_{i=1}^{k} x_i$.
- If α_k is not $1/k$, the convergence was not analyzed.

Next, we show that this algorithm is a special case of the RM algorithm. Then, its convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

Recall that

$$w_{k+1} = w_k + \alpha_k(x_k - w_k).$$

is the mean estimation algorithm introduced at the beginning of this lecture.

- If $\alpha_k = 1/k$, then $w_{k+1} = 1/k \sum_{i=1}^{k} x_i$.
- If α_k is not $1/k$, the convergence was not analyzed.

Next, we show that this algorithm is a special case of the RM algorithm. Then, its convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

Recall that

$$w_{k+1} = w_k + \alpha_k(x_k - w_k).$$

is the mean estimation algorithm introduced at the beginning of this lecture.

- If $\alpha_k = 1/k$, then $w_{k+1} = 1/k \sum_{i=1}^{k} x_i$.
- If α_k is not $1/k$, the convergence was not analyzed.

Next, we show that this algorithm is a special case of the RM algorithm. Then, its convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

1) Consider a function:

$$g(w) \doteq w - \mathbb{E}[X].$$

Our aim is to solve $g(w) = 0$. If we can do that, then we can obtain $\mathbb{E}[X]$.

- Mean estimation (i.e., finding $\mathbb{E}[X]$) is formulated as a root-finding problem (i.e., solving $g(w) = 0$).
- **Question:** Do we know the expression of $g(w)$ here?

2) The observation we can get is

$$\tilde{g}(w, x) \doteq w - x,$$

because we can only obtain samples of X. Note that

$$\begin{aligned}\tilde{g}(w, \eta) = w - x &= w - x + \mathbb{E}[X] - \mathbb{E}[X] \\ &= (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \doteq g(w) + \eta,\end{aligned}$$

3) The RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - \alpha_k \tilde{g}(w_k, \eta_k) = w_k - \alpha_k (w_k - x_k),$$

which is exactly the mean estimation algorithm.

The convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

1) Consider a function:

$$g(w) \doteq w - \mathbb{E}[X].$$

Our aim is to solve $g(w) = 0$. If we can do that, then we can obtain $\mathbb{E}[X]$.

- Mean estimation (i.e., finding $\mathbb{E}[X]$) is formulated as a root-finding problem (i.e., solving $g(w) = 0$).
- **Question:** Do we know the expression of $g(w)$ here?

2) The observation we can get is

$$\tilde{g}(w, x) \doteq w - x,$$

because we can only obtain samples of X. Note that

$$\begin{aligned}\tilde{g}(w, \eta) = w - x &= w - x + \mathbb{E}[X] - \mathbb{E}[X] \\ &= (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \doteq g(w) + \eta,\end{aligned}$$

3) The RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - \alpha_k \tilde{g}(w_k, \eta_k) = w_k - \alpha_k (w_k - x_k),$$

which is exactly the mean estimation algorithm.

The convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

1) Consider a function:

$$g(w) \doteq w - \mathbb{E}[X].$$

Our aim is to solve $g(w) = 0$. If we can do that, then we can obtain $\mathbb{E}[X]$.

- Mean estimation (i.e., finding $\mathbb{E}[X]$) is formulated as a root-finding problem (i.e., solving $g(w) = 0$).
- **Question:** Do we know the expression of $g(w)$ here?

2) The observation we can get is

$$\tilde{g}(w, x) \doteq w - x,$$

because we can only obtain samples of X. Note that

$$\begin{aligned}\tilde{g}(w, \eta) = w - x &= w - x + \mathbb{E}[X] - \mathbb{E}[X] \\ &= (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \doteq g(w) + \eta,\end{aligned}$$

3) The RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - \alpha_k \tilde{g}(w_k, \eta_k) = w_k - \alpha_k (w_k - x_k),$$

which is exactly the mean estimation algorithm.

The convergence naturally follows.

Robbins-Monro algorithm – Apply to mean estimation

1) Consider a function:

$$g(w) \doteq w - \mathbb{E}[X].$$

Our aim is to solve $g(w) = 0$. If we can do that, then we can obtain $\mathbb{E}[X]$.

- Mean estimation (i.e., finding $\mathbb{E}[X]$) is formulated as a root-finding problem (i.e., solving $g(w) = 0$).
- **Question:** Do we know the expression of $g(w)$ here?

2) The observation we can get is

$$\tilde{g}(w, x) \doteq w - x,$$

because we can only obtain samples of X. Note that

$$\begin{aligned}\tilde{g}(w, \eta) = w - x &= w - x + \mathbb{E}[X] - \mathbb{E}[X] \\ &= (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \doteq g(w) + \eta,\end{aligned}$$

3) The RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - \alpha_k \tilde{g}(w_k, \eta_k) = w_k - \alpha_k(w_k - x_k),$$

which is exactly the mean estimation algorithm.

The convergence naturally follows.

Dvoretzkys convergence theorem (optional)

Theorem (Dvoretzky's Theorem)

Consider a stochastic process

$$w_{k+1} = (1 - \alpha_k)w_k + \beta_k \eta_k,$$

where $\{\alpha_k\}_{k=1}^{\infty}, \{\beta_k\}_{k=1}^{\infty}, \{\eta_k\}_{k=1}^{\infty}$ *are stochastic sequences. Here* $\alpha_k \geq 0, \beta_k \geq 0$ *for all* k*. Then,* w_k *would converge to zero with probability 1 if the following conditions are satisfied:*

1) $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$; $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ *uniformly w.p.1;*

2) $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ *and* $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq C$ *w.p.1;*

where $\mathcal{H}_k = \{w_k, w_{k-1}, \ldots, \eta_{k-1}, \ldots, \alpha_{k-1}, \ldots, \beta_{k-1}, \ldots\}$.

- A more general result than the RM theorem.
  - It can be used to prove the RM theorem
  - It can be used to analyze the mean estimation problem.
  - An extension of it can be used to analyze Q-learning and TD learning algorithms.

Outline

Outline

Stochastic gradient descent

Next, we introduce stochastic gradient descent (SGD) algorithms:

- SGD is widely used in the field of machine learning and also in RL.
  - SGD is a special RM algorithm.
  - The mean estimation algorithm is a special SGD algorithm.

Problem setup: Suppose we aim to solve the following optimization problem:

$$\min_w \quad J(w) = \mathbb{E}[f(w, X)]$$

- w is the parameter to be optimized.
- X is a random variable. The expectation is with respect to X.
- w and X can be either scalars or vectors. The function $f(\cdot)$ is a scalar.

Stochastic gradient descent

Next, we introduce stochastic gradient descent (SGD) algorithms:

- SGD is widely used in the field of machine learning and also in RL.
  - SGD is a special RM algorithm.
  - The mean estimation algorithm is a special SGD algorithm.

Problem setup: Suppose we aim to solve the following optimization problem:

$$\min_w \quad J(w) = \mathbb{E}[f(w, X)]$$

- w is the parameter to be optimized.
- X is a random variable. The expectation is with respect to X.
- w and X can be either scalars or vectors. The function $f(\cdot)$ is a scalar.

Stochastic gradient descent

Method 1: gradient descent (GD)

$$w_{k+1} = w_k - \alpha_k \nabla_w \mathbb{E}[f(w_k, X)] = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

Drawback: Calculating the expectation requires the distribution of X.

Method 2: batch gradient descent (BGD)

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Hence

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Drawback: it requires many samples in each iteration for each w_k.

Stochastic gradient descent

Method 1: gradient descent (GD)

$$w_{k+1} = w_k - \alpha_k \nabla_w \mathbb{E}[f(w_k, X)] = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

Drawback: Calculating the expectation requires the distribution of X.

Method 2: batch gradient descent (BGD)

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Hence

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Drawback: it requires many samples in each iteration for each w_k.

Method 1: gradient descent (GD)

$$w_{k+1} = w_k - \alpha_k \nabla_w \mathbb{E}[f(w_k, X)] = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

Drawback: Calculating the expectation requires the distribution of X.

Method 2: batch gradient descent (BGD)

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Hence

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Drawback: it requires many samples in each iteration for each w_k.

Method 1: gradient descent (GD)

$$w_{k+1} = w_k - \alpha_k \nabla_w \mathbb{E}[f(w_k, X)] = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

Drawback: Calculating the expectation requires the distribution of X.

Method 2: batch gradient descent (BGD)

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Hence

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Drawback: it requires many samples in each iteration for each w_k.

Method 1: gradient descent (GD)

$$w_{k+1} = w_k - \alpha_k \nabla_w \mathbb{E}[f(w_k, X)] = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

Drawback: Calculating the expectation requires the distribution of X.

Method 2: batch gradient descent (BGD)

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Hence

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i)$$

Drawback: it requires many samples in each iteration for each w_k.

Method 3: stochastic gradient descent (SGD)

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k),$$

- Compared to the gradient descent method:
  - Replace the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$ by the stochastic gradient $\nabla_w f(w_k, x_k)$.
- Compared to the batch gradient descent method:
  - let $n = 1$.

Outline

Stochastic gradient descent – Example and application

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

Exercises:

- Exercise 1: Show that the optimal solution is $w^* = \mathbb{E}[X]$.
- Exercise 2: Write out the GD algorithm for solving this problem.
- Exercise 3: Write out the SGD algorithm for solving this problem.

Stochastic gradient descent – Example and application

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

Exercises:

- Exercise 1: Show that the optimal solution is $w^* = \mathbb{E}[X]$.
- Exercise 2: Write out the GD algorithm for solving this problem.
- Exercise 3: Write out the SGD algorithm for solving this problem.

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

Exercises:

- Exercise 1: Show that the optimal solution is $w^* = \mathbb{E}[X]$.
- Exercise 2: Write out the GD algorithm for solving this problem.
- Exercise 3: Write out the SGD algorithm for solving this problem.

Stochastic gradient descent – Example and application

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

- **Exercise 1:** Show that the optimal solution is $w^* = \mathbb{E}[X]$.
- **Answer to exercise 1:** The optimal solution w^* must satisfy

$$\nabla_w J(w) = 0$$

which is

$$\nabla_w \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right] = \mathbb{E}\left[\nabla_w \frac{1}{2}\|w - X\|^2\right] = \mathbb{E}\left[w - X\right] = 0$$

Therefore, we formulate the mean estimation problem (i.e., finding $\mathbb{E}[X]$) as an optimization problem (i.e., optimizing $J(w)$).

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

- **Exercise 2:** Write out the GD algorithm for solving this problem.
- **Answer to exercise 2:** The GD algorithm for solving the above problem is

$$\begin{aligned} w_{k+1} &= w_k - \alpha_k \nabla_w J(w_k) \\ &= w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)] \\ &= w_k - \alpha_k \mathbb{E}[w_k - X]. \end{aligned}$$

We next consider an example:

$$\min_{w} \quad J(w) = \mathbb{E}[f(w, X)] = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right],$$

where

$$f(w, X) = \|w - X\|^2/2 \qquad \nabla_w f(w, X) = w - X$$

- **Exercise 3:** Write out the SGD algorithm for solving this problem.
- **Answer to exercise 3:** The SGD algorithm for solving the above problem is

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k) = w_k - \alpha_k (w_k - x_k)$$

  - It is the same as the mean estimation algorithm we presented before.
  - Therefore, that mean estimation algorithm is a special SGD algorithm.

Outline

Stochastic gradient descent – Convergence

Idea of SGD:

$$w_{k+1} = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

$$\Downarrow$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k)$$

where the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$ is replaced by the stochastic gradient $\nabla_w f(w_k, X)$.

Question: Since

$$\nabla_w f(w_k, x_k) \neq \mathbb{E}[\nabla_w f(w, X)]$$

whether $w_k \to w^*$ as $k \to \infty$ by SGD?

Observation: The stochastic gradient is a noisy measurement of the true gradient:

$$\nabla_w f(w_k, x_k) = \mathbb{E}[\nabla_w f(w, X)] + \underbrace{\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w, X)]}_{\eta}$$

where η is the noise.

Stochastic gradient descent – Convergence

Idea of SGD:

$$w_{k+1} = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

$$\Downarrow$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k)$$

where the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$ is replaced by the stochastic gradient $\nabla_w f(w_k, X)$.

Question: Since

$$\nabla_w f(w_k, x_k) \neq \mathbb{E}[\nabla_w f(w, X)]$$

whether $w_k \to w^*$ as $k \to \infty$ by SGD?

Observation: The stochastic gradient is a noisy measurement of the true gradient:

$$\nabla_w f(w_k, x_k) = \mathbb{E}[\nabla_w f(w, X)] + \underbrace{\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w, X)]}_{\eta}$$

where η is the noise.

Idea of SGD:

$$w_{k+1} = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

$$\Downarrow$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k)$$

where the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$ is replaced by the stochastic gradient $\nabla_w f(w_k, X)$.

Question: Since

$$\nabla_w f(w_k, x_k) \neq \mathbb{E}[\nabla_w f(w, X)]$$

whether $w_k \to w^*$ as $k \to \infty$ by SGD?

Observation: The stochastic gradient is a noisy measurement of the true gradient:

$$\nabla_w f(w_k, x_k) = \mathbb{E}[\nabla_w f(w, X)] + \underbrace{\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w, X)]}_{\eta}$$

where η is the noise.

Stochastic gradient descent – Convergence

We next show that SGD is a special RM algorithm. Then, the convergence naturally follows.

The aim of SGD is to minimize

$$J(w) = \mathbb{E}[f(w, X)]$$

This problem can be converted to a root-finding problem:

$$\nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)] = 0$$

Let

$$g(w) = \nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)]$$

Then, the aim of SGD is to find the root of $g(w) = 0$.

Stochastic gradient descent – Convergence

We next show that SGD is a special RM algorithm. Then, the convergence naturally follows.

The aim of SGD is to minimize

$$J(w) = \mathbb{E}[f(w, X)]$$

This problem can be converted to a root-finding problem:

$$\nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)] = 0$$

Let

$$g(w) = \nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)]$$

Then, the aim of SGD is to find the root of $g(w) = 0$.

Stochastic gradient descent – Convergence

We next show that SGD is a special RM algorithm. Then, the convergence naturally follows.

The aim of SGD is to minimize

$$J(w) = \mathbb{E}[f(w, X)]$$

This problem can be converted to a root-finding problem:

$$\nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)] = 0$$

Let

$$g(w) = \nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)]$$

Then, the aim of SGD is to find the root of $g(w) = 0$.

Stochastic gradient descent – Convergence

We next show that SGD is a special RM algorithm. Then, the convergence naturally follows.

The aim of SGD is to minimize

$$J(w) = \mathbb{E}[f(w, X)]$$

This problem can be converted to a root-finding problem:

$$\nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)] = 0$$

Let

$$g(w) = \nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)]$$

Then, the aim of SGD is to find the root of $g(w) = 0$.

Stochastic gradient descent – Convergence

What we can measure is

$$\begin{aligned}\tilde{g}(w,\eta) &= \nabla_w f(w,x) \\ &= \underbrace{\mathbb{E}[\nabla_w f(w,X)]}_{g(w)} + \underbrace{\nabla_w f(w,x) - \mathbb{E}[\nabla_w f(w,X)]}_{\eta}.\end{aligned}$$

Then, the RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k) = w_k - a_k \nabla_w f(w_k, x_k).$$

- It is exactly the SGD algorithm.
- Therefore, SGD is a special RM algorithm.

What we can measure is

$$\begin{aligned}\tilde{g}(w,\eta) &= \nabla_w f(w,x) \\ &= \underbrace{\mathbb{E}[\nabla_w f(w,X)]}_{g(w)} + \underbrace{\nabla_w f(w,x) - \mathbb{E}[\nabla_w f(w,X)]}_{\eta}.\end{aligned}$$

Then, the RM algorithm for solving $g(w) = 0$ is

$$w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k) = w_k - a_k \nabla_w f(w_k, x_k).$$

- It is exactly the SGD algorithm.
- Therefore, SGD is a special RM algorithm.

Since SGD is a special RM algorithm, its convergence naturally follows.

Theorem (Convergence of SGD)

In the SGD algorithm, if

1) $0 < c_1 \leq \nabla_w^2 f(w, X) \leq c_2$*;*

2) $\sum_{k=1}^{\infty} a_k = \infty$ *and* $\sum_{k=1}^{\infty} a_k^2 < \infty$*;*

3) $\{x_k\}_{k=1}^{\infty}$ *is iid;*

then w_k *converges to the root of* $\nabla_w \mathbb{E}[f(w, X)] = 0$ *with probability 1.*

For the proof see the book.

Outline

Stochastic gradient descent – Convergence pattern

Idea of SGD:

$$w_{k+1} = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]$$

$$\Downarrow$$

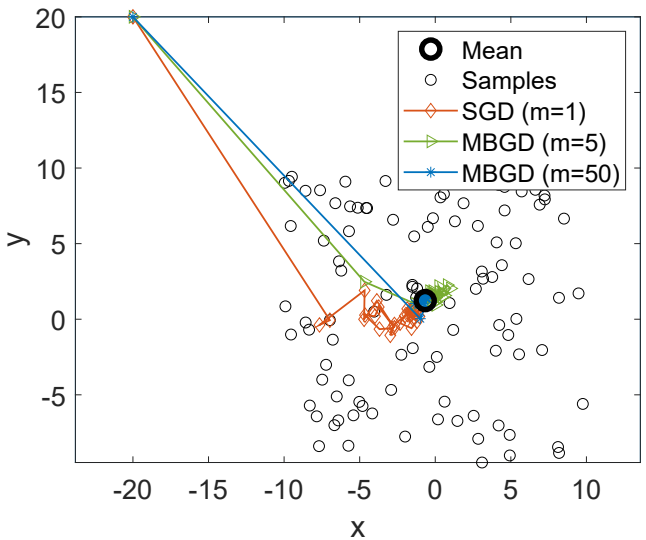
$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k)$$

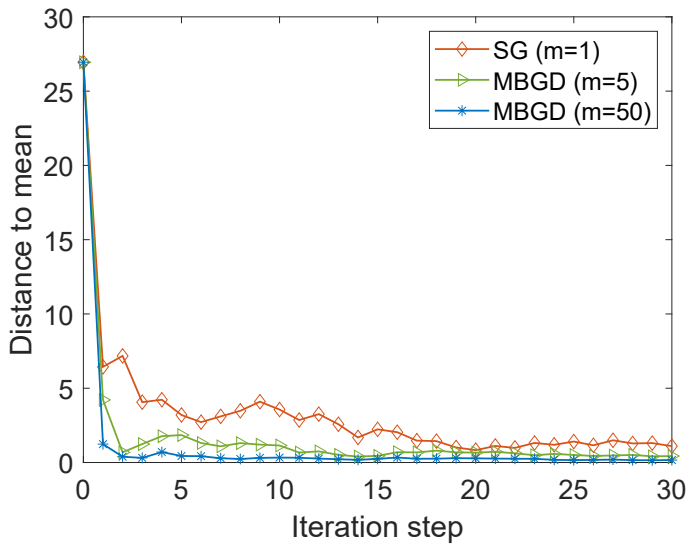
where the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$ is replaced by the stochastic gradient $\nabla_w f(w_k, X)$.

Question: Since the stochastic gradient is random, whether the convergence of SGD is slow or random?

Stochastic gradient descent – Convergence pattern

Example: $X \in \mathbb{R}^2$ represents a random position in the plane. Its distribution is uniform in the square area centered at the origin with the side length as 20. The true mean is $\mathbb{E}[X] = 0$. The mean estimation is based on 100 iid samples $\{x_i\}_{i=1}^{100}$.



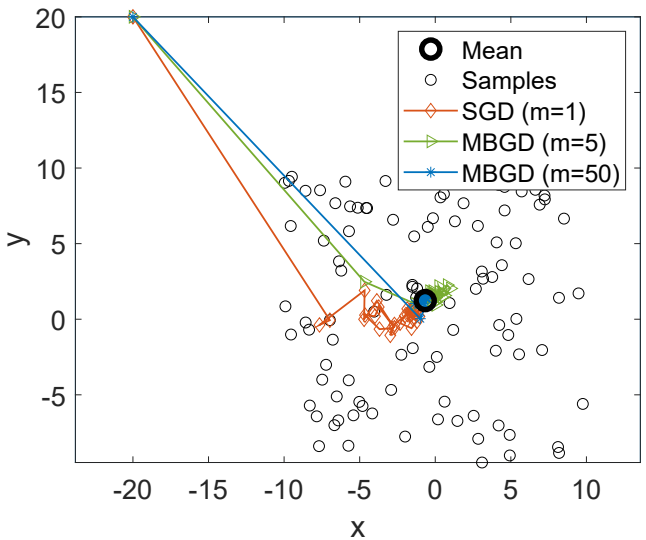


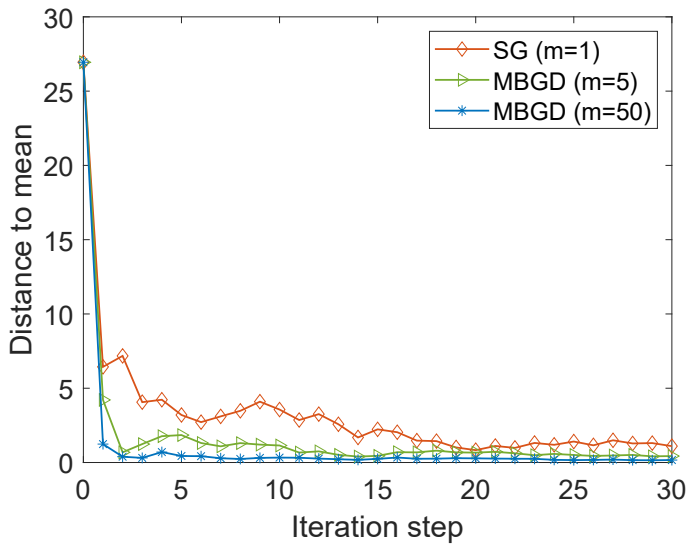
Observations:

- When the estimate (e.g., the initial guess) is far away from the true value, the SGD estimate can approach the neighborhood of the true value fast.
- When the estimate is close to the true value, it exhibits certain randomness but still approaches the true value gradually.

Stochastic gradient descent – Convergence pattern

Example: $X \in \mathbb{R}^2$ represents a random position in the plane. Its distribution is uniform in the square area centered at the origin with the side length as 20. The true mean is $\mathbb{E}[X] = 0$. The mean estimation is based on 100 iid samples $\{x_i\}_{i=1}^{100}$.





Observations:

- When the estimate (e.g., the initial guess) is far away from the true value, the SGD estimate can approach the neighborhood of the true value fast.
- When the estimate is close to the true value, it exhibits certain randomness but still approaches the true value gradually.

Question: Why such a pattern?

Answer: We answer this question by considering the relative error between the stochastic and batch gradients:

$$\delta_k \doteq \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)]|}.$$

It can be proven that

$$\delta_k \leq \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{c|w_k - w^*|}.$$

The proof is given in the next slide. The proof is optional.

Question: Why such a pattern?

Answer: We answer this question by considering the relative error between the stochastic and batch gradients:

$$\delta_k \doteq \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)]|}.$$

It can be proven that

$$\delta_k \leq \frac{\left|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]\right|}{c|w_k - w^*|}.$$

The proof is given in the next slide. The proof is optional.

Stochastic gradient descent – Convergence pattern (optional)

Since $\mathbb{E}[\nabla_w f(w^*, X)] = 0$, we have

$$\delta_k = \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)] - \mathbb{E}[\nabla_w f(w^*, X)]|} = \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)(w_k - w^*)]|}$$

where the last equality is due to the mean value theorem and $\tilde{w}_k \in [w_k, w^*]$.

Suppose f is strictly convex such that

$$\nabla_w^2 f \geq c > 0$$

for all w, X, where c is a positive bound.

Then, the denominator of δ_k becomes

$$\begin{aligned}\left|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)(w_k - w^*)]\right| &= \left|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)](w_k - w^*)\right| \\ &= \left|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)]\right|\left|(w_k - w^*)\right| \geq c|w_k - w^*|.\end{aligned}$$

Substituting the above inequality to δ_k gives

$$\delta_k \leq \frac{\left|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]\right|}{c|w_k - w^*|}.$$

Stochastic gradient descent – Convergence pattern

Note that

$$\delta_k \leq \frac{\left| \overbrace{\nabla_w f(w_k, x_k)}^{\text{stochastic gradient}} - \overbrace{\mathbb{E}[\nabla_w f(w_k, X)]}^{\text{true gradient}} \right|}{\underbrace{c|w_k - w^*|}_{\text{distance to the optimal solution}}}.$$

The above equation suggests an interesting convergence pattern of SGD.

- The upper bound is inversely proportional to $|w_k - w^*|$.
  - When $|w_k - w^*|$ is large, the relative error δ_k is small and SGD behaves like GD.
  - When $|w_k - w^*|$ is small, the relative error δ_k may be large (the upper bound may not be tight). Then, SGD exhibits more randomness in the neighborhood of w^*.

Stochastic gradient descent – Convergence pattern

Note that

$$\delta_k \leq \frac{\left| \overbrace{\nabla_w f(w_k, x_k)}^{\text{stochastic gradient}} - \overbrace{\mathbb{E}[\nabla_w f(w_k, X)]}^{\text{true gradient}} \right|}{\underbrace{c|w_k - w^*|}_{\text{distance to the optimal solution}}}.$$

The above equation suggests an interesting convergence pattern of SGD.

- The upper bound is inversely proportional to $|w_k - w^*|$.
  - When $|w_k - w^*|$ is large, the relative error δ_k is small and SGD behaves like GD.
  - When $|w_k - w^*|$ is small, the relative error δ_k may be large (the upper bound may not be tight). Then, SGD exhibits more randomness in the neighborhood of w^*.

Outline

BGD, MBGD, and SGD

Suppose we would like to minimize $J(w) = \mathbb{E}[f(w, X)]$ given a set of random samples $\{x_i\}_{i=1}^n$ of X.

The BGD, SGD, MBGD algorithms solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \qquad \text{(SGD)}$$

- **In the BGD algorithm**, all the samples are used in every iteration. When n is large, $(1/n) \sum_{i=1}^n \nabla_w f(w_k, x_i)$ is close to the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$.
- **In the MBGD algorithm**, $\mathcal{I}_k$ is a subset of $\{1, \ldots, n\}$ with the size as $|\mathcal{I}_k| = m$. The set $\mathcal{I}_k$ is obtained by m times iid samplings.
- **In the SGD algorithm**, x_k is randomly sampled from $\{x_i\}_{i=1}^n$ at time k.

BGD, MBGD, and SGD

Suppose we would like to minimize $J(w) = \mathbb{E}[f(w, X)]$ given a set of random samples $\{x_i\}_{i=1}^n$ of X.

The BGD, SGD, MBGD algorithms solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \qquad \text{(SGD)}$$

- **In the BGD algorithm**, all the samples are used in every iteration. When n is large, $(1/n)\sum_{i=1}^n \nabla_w f(w_k, x_i)$ is close to the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$.
- **In the MBGD algorithm**, $\mathcal{I}_k$ is a subset of $\{1, \ldots, n\}$ with the size as $|\mathcal{I}_k| = m$. The set $\mathcal{I}_k$ is obtained by m times iid samplings.
- **In the SGD algorithm**, x_k is randomly sampled from $\{x_i\}_{i=1}^n$ at time k.

BGD, MBGD, and SGD

Suppose we would like to minimize $J(w) = \mathbb{E}[f(w, X)]$ given a set of random samples $\{x_i\}_{i=1}^n$ of X.

The BGD, SGD, MBGD algorithms solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \qquad \text{(SGD)}$$

- **In the BGD algorithm**, all the samples are used in every iteration. When n is large, $(1/n) \sum_{i=1}^n \nabla_w f(w_k, x_i)$ is close to the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$.
- **In the MBGD algorithm**, $\mathcal{I}_k$ is a subset of $\{1, \ldots, n\}$ with the size as $|\mathcal{I}_k| = m$. The set $\mathcal{I}_k$ is obtained by m times iid samplings.
- **In the SGD algorithm**, x_k is randomly sampled from $\{x_i\}_{i=1}^n$ at time k.

BGD, MBGD, and SGD

Suppose we would like to minimize $J(w) = \mathbb{E}[f(w, X)]$ given a set of random samples $\{x_i\}_{i=1}^n$ of X.

The BGD, SGD, MBGD algorithms solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \qquad \text{(SGD)}$$

- **In the BGD algorithm**, all the samples are used in every iteration. When n is large, $(1/n) \sum_{i=1}^n \nabla_w f(w_k, x_i)$ is close to the true gradient $\mathbb{E}[\nabla_w f(w_k, X)]$.
- **In the MBGD algorithm**, $\mathcal{I}_k$ is a subset of $\{1, \ldots, n\}$ with the size as $|\mathcal{I}_k| = m$. The set $\mathcal{I}_k$ is obtained by m times iid samplings.
- **In the SGD algorithm**, x_k is randomly sampled from $\{x_i\}_{i=1}^n$ at time k.

BGD, MBGD, and SGD

Compare MBGD with BGD and SGD:

- Compared to SGD, MBGD has less randomness because it uses more samples instead of just one as in SGD.
- Compared to BGD, MBGD does not require to use all the samples in every iteration, making it more flexible and efficient.
- If $m = 1$, MBGD becomes SGD.
- If $m = n$, MBGD does NOT become BGD strictly speaking because MBGD uses randomly fetched n samples whereas BGD uses all n numbers. In particular, MBGD may use a value in $\{x_i\}_{i=1}^n$ multiple times whereas BGD uses each number once.

BGD, MBGD, and SGD

Compare MBGD with BGD and SGD:

- Compared to SGD, MBGD has less randomness because it uses more samples instead of just one as in SGD.
- Compared to BGD, MBGD does not require to use all the samples in every iteration, making it more flexible and efficient.

- If $m = 1$, MBGD becomes SGD.
- If $m = n$, MBGD does NOT become BGD strictly speaking because MBGD uses randomly fetched n samples whereas BGD uses all n numbers. In particular, MBGD may use a value in $\{x_i\}_{i=1}^n$ multiple times whereas BGD uses each number once.

BGD, MBGD, and SGD

Compare MBGD with BGD and SGD:

- Compared to SGD, MBGD has less randomness because it uses more samples instead of just one as in SGD.
- Compared to BGD, MBGD does not require to use all the samples in every iteration, making it more flexible and efficient.

- If $m = 1$, MBGD becomes SGD.
- If $m = n$, MBGD does NOT become BGD strictly speaking because MBGD uses randomly fetched n samples whereas BGD uses all n numbers. In particular, MBGD may use a value in $\{x_i\}_{i=1}^n$ multiple times whereas BGD uses each number once.

BGD, MBGD, and SGD

Compare MBGD with BGD and SGD:

- Compared to SGD, MBGD has less randomness because it uses more samples instead of just one as in SGD.
- Compared to BGD, MBGD does not require to use all the samples in every iteration, making it more flexible and efficient.

- If $m = 1$, MBGD becomes SGD.
- If $m = n$, MBGD does NOT become BGD strictly speaking because MBGD uses randomly fetched n samples whereas BGD uses all n numbers. In particular, MBGD may use a value in $\{x_i\}_{i=1}^n$ multiple times whereas BGD uses each number once.

BGD, MBGD, and SGD – Illustrative examples

Given some numbers $\{x_i\}_{i=1}^n$, our aim is to calculate the mean $\bar{x} = \sum_{i=1}^n x_i/n$. This problem can be equivalently stated as the following optimization problem:

$$\min_w \quad J(w) = \frac{1}{2n} \sum_{i=1}^n \|w - x_i\|^2$$

The three algorithms for solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^n (w_k - x_i) = w_k - \alpha_k (w_k - \bar{x}), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} (w_k - x_j) = w_k - \alpha_k \left(w_k - \bar{x}_k^{(m)}\right), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k (w_k - x_k), \qquad \text{(SGD)}$$

where $\bar{x}_k^{(m)} = \sum_{j \in \mathcal{I}_k} x_j/m$.

BGD, MBGD, and SGD – Illustrative examples

Given some numbers $\{x_i\}_{i=1}^n$, our aim is to calculate the mean $\bar{x} = \sum_{i=1}^n x_i/n$. This problem can be equivalently stated as the following optimization problem:

$$\min_w \quad J(w) = \frac{1}{2n}\sum_{i=1}^n \|w - x_i\|^2$$

The three algorithms for solving this problem are, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{n}\sum_{i=1}^n (w_k - x_i) = w_k - \alpha_k(w_k - \bar{x}), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m}\sum_{j\in\mathcal{I}_k} (w_k - x_j) = w_k - \alpha_k\left(w_k - \bar{x}_k^{(m)}\right), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k(w_k - x_k), \qquad \text{(SGD)}$$

where $\bar{x}_k^{(m)} = \sum_{j\in\mathcal{I}_k} x_j/m$.

BGD, MBGD, and SGD

Let $\alpha_k = 1/k$. Given 100 points, using different mini-batch sizes leads to different convergence speed.

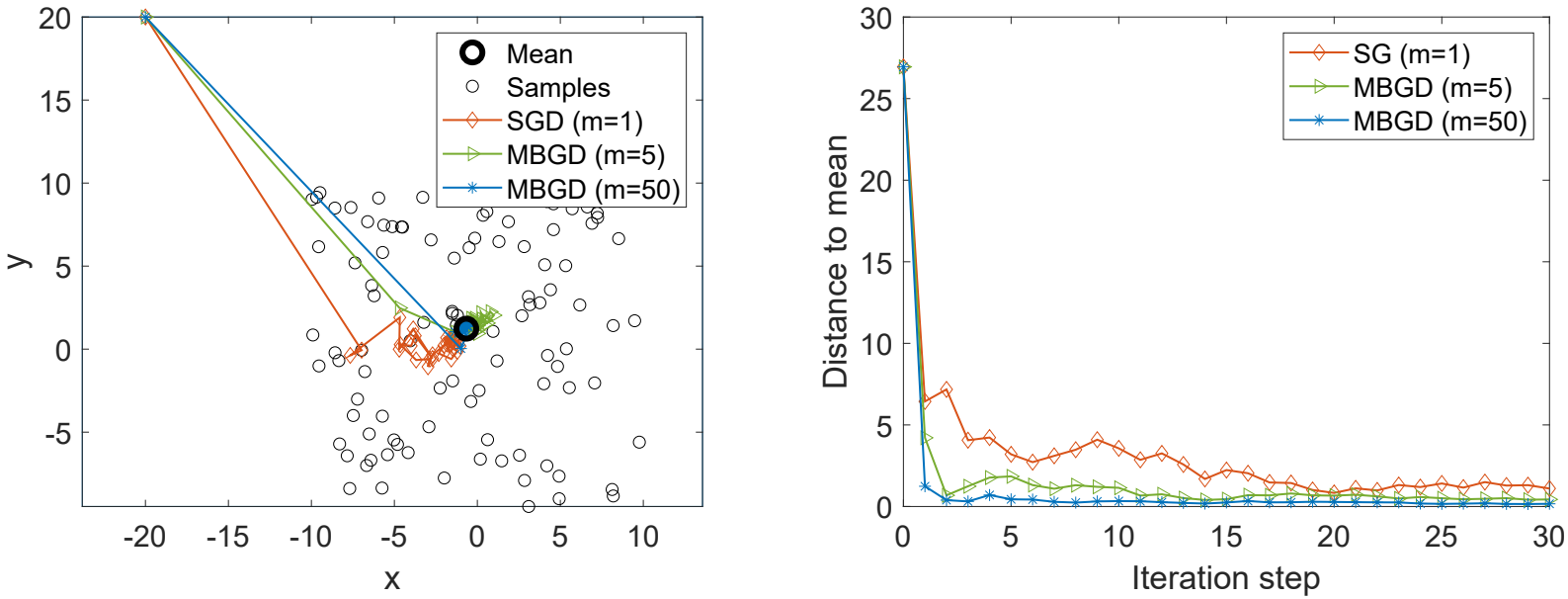


Figure: An illustrative example for mean estimation by different GD algorithms.

Outline

Summary

- Mean estimation: compute $\mathbb{E}[X]$ using $\{x_k\}$

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k).$$

- RM algorithm: solve $g(w) = 0$ using $\{\tilde{g}(w_k, \eta_k)\}$

$$w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k)$$

- SGD algorithm: minimize $J(w) = \mathbb{E}[f(w, X)]$ using $\{\nabla_w f(w_k, x_k)\}$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k),$$

These results are useful:

- We will see in the next chapter that the temporal-difference learning algorithms can be viewed as stochastic approximation algorithms and hence have similar expressions.
- They are important optimization techniques that can be applied to many other fields.