Introduction to Machine Learning

Stochastic Gradient Descent

Andres Mendez-Vazquez

January 28, 2023

Outline

1. Introduction
   - Review Gradient Descent
   - The Problems of Gradient Descent with Large Data Sets
   - Convergence of gradient descent with fixed step size
   - Convergence Rate
     - Convex Functions
     - Back to the Main Problem
   - Accelerating the Gradient Descent
   - Even with such Speeds

2. Accelerating Gradient Descent
   - Robbins-Monro Theorem
   - Robbins-Monro Scheme for Minimum-Square Error
   - Convergence

3. Improving and Measuring Stochastic Gradient Descent
   - Example of SGD Vs BGD
   - Using The Expected Value, The Mini-Batch
   - The Least-Mean Squares Adaptive Algorithm
   - Conclusions

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Gradient Descent [1, 2]

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}_0$.

Gradient Descent [1, 2]

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}_0$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}_0)$.

Gradient Descent [1, 2]

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}_0$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}_0)$.
3. Obtain value $\boldsymbol{w}_1$ by moving from $\boldsymbol{w}_0$ in the direction of the steepest descent:

Gradient Descent [1, 2]

The basic procedure is as follow

1. Start with a random weight vector $\boldsymbol{w}_0$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}_0)$.
3. Obtain value $\boldsymbol{w}_1$ by moving from $\boldsymbol{w}_0$ in the direction of the steepest descent:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta_n \nabla J(\boldsymbol{w}_n) \tag{1}$$

Gradient Descent [1, 2]

The basic procedure is as follow

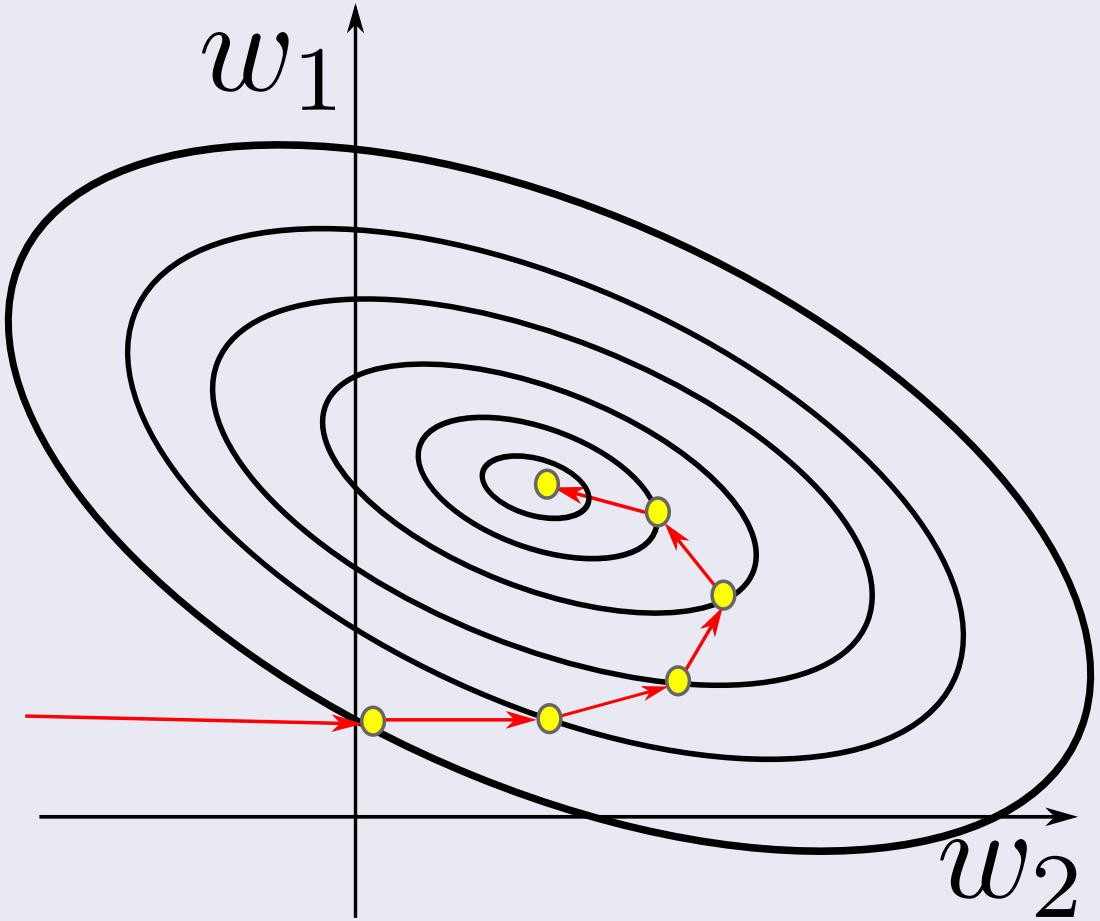
1. Start with a random weight vector $\boldsymbol{w}_0$.
2. Compute the gradient vector $\nabla J(\boldsymbol{w}_0)$.
3. Obtain value $\boldsymbol{w}_1$ by moving from $\boldsymbol{w}_0$ in the direction of the steepest descent:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta_n \nabla J(\boldsymbol{w}_n) \tag{1}$$

η_n is a positive scale factor or learning rate!!!

Geometrically

We have the following



$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k)\nabla J(\boldsymbol{w}(k))$$

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Although

It is possible to prove

- That the gradient direction gives the greatest increase direction!!!

Although

It is possible to prove

- That the gradient direction gives the greatest increase direction!!!

We have a problem in cost functions like in Deep Neural Networks

$$J(\boldsymbol{w}) = \sum_{i=1}^{N} \left(y_i - f(\boldsymbol{w}, \boldsymbol{x}_i)\right)^2$$

- Where, we have that $f(\boldsymbol{w}, \boldsymbol{x}_i) = f_1 \circ f_2 \circ f_3 \circ \cdots \circ f_T(\boldsymbol{w}, \boldsymbol{x}_i)$

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Do you remember the problem of the η step size?

Gradient Descent with fixed step size

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla J\left(\boldsymbol{w}_n\right)$$

Do you remember the problem of the η step size?

Gradient Descent with fixed step size

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla J\left(\boldsymbol{w}_n\right)$$

Why to worry about this?

- Because, we want to know how fast Gradient Descent will find the answer...

We have

Lipschitz Continuous [3]

- Lipschitz continuity, named after Rudolf Lipschitz, is a strong form of uniform continuity for functions.

We have

Lipschitz Continuous [3]

- Lipschitz continuity, named after Rudolf Lipschitz, is a strong form of uniform continuity for functions.

Uniform continuity

- The function $f : A \to \mathbb{R}$ is said to be uniformly continuous on A iff for every $\epsilon > 0$, $\exists \delta > 0$ such that $|\boldsymbol{x} - \boldsymbol{y}| < \delta$ implies $|f(\boldsymbol{x}) - f(\boldsymbol{y})| < \epsilon$.

Lipschitz Continuous

Definition

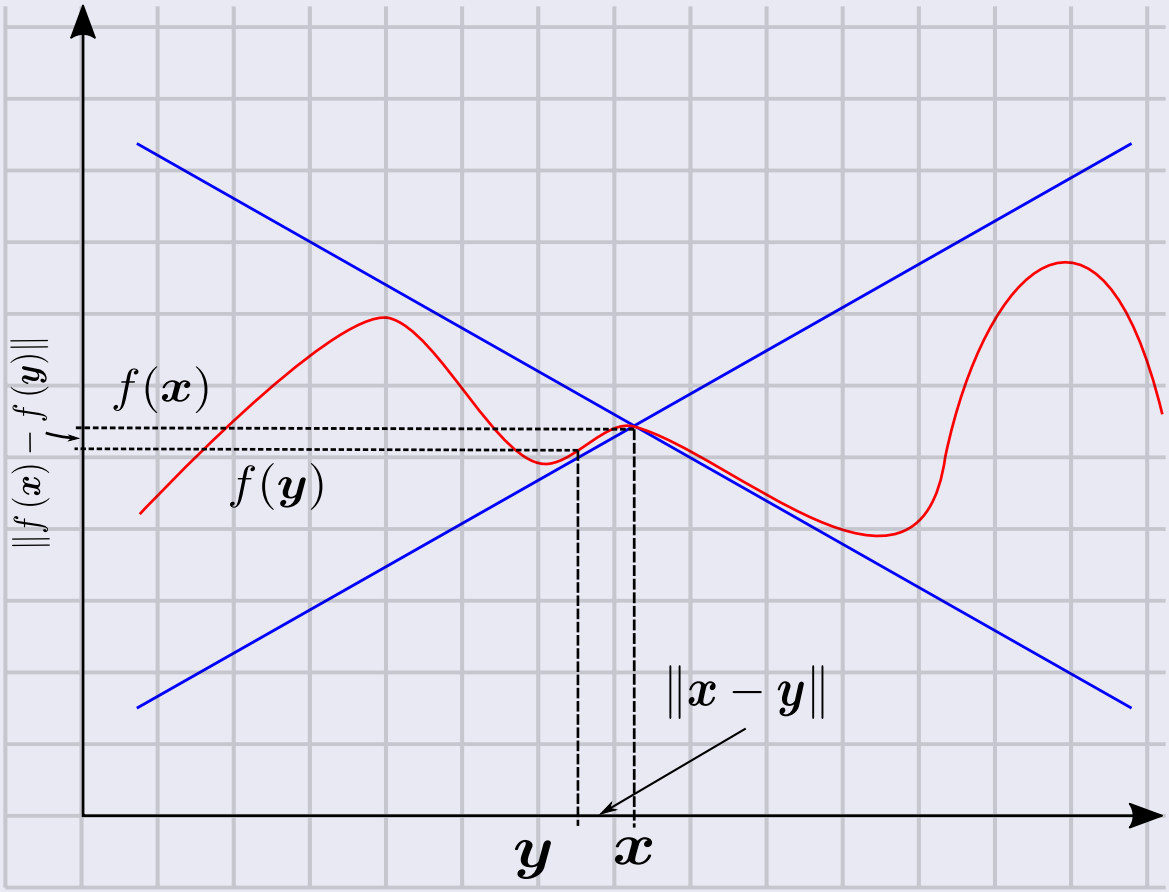
- A function $f : S \subset \mathbb{R}^n \to \mathbb{R}^n$ satisfies the Lipschitz Continuous at $\boldsymbol{x} \in S$, if there is a such constant $L > 0$ such that

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|$$

for all $\boldsymbol{y} \in S$ sufficiently near to $\boldsymbol{x}$. **Lipschitz continuity can be seen as a refinement of continuity.**

Example when you see L as the slope

Here the function $f : \mathbb{R} \to \mathbb{R}$



An interesting property of such setup

The derivative of the function cannot exceed L (Example, $f : \mathbb{R} \to \mathbb{R}$)

$$f'(x) = \lim_{\delta \to \infty} \frac{f(x+\delta) - f(x)}{\delta}$$

An interesting property of such setup

The derivative of the function cannot exceed L (Example, $f : \mathbb{R} \to \mathbb{R}$)

$$f'(x) = \lim_{\delta \to \infty} \frac{f(x+\delta) - f(x)}{\delta}$$

Then, we have that

$$f'(x) = \lim_{\delta \to \infty} \frac{f(x) - f(y)}{x - y} \leq \lim_{\delta \to \infty} \frac{|f(x) - f(y)|}{|x - y|} \leq L$$

Therefore

Lipschitz Continuity implies

$$\left|f'(x)\right| < L$$

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Convergence idea

Definition (Big O - Upper Bound) [4]

For a given function $g(n)$:

$$O(g(n)) = \{f(n) | \text{ There exists } c > 0 \text{ and } n_0 > 0 \text{ s.t. } 0 \leq f(n) \leq cg(n) \ \forall n \geq n_0\}$$

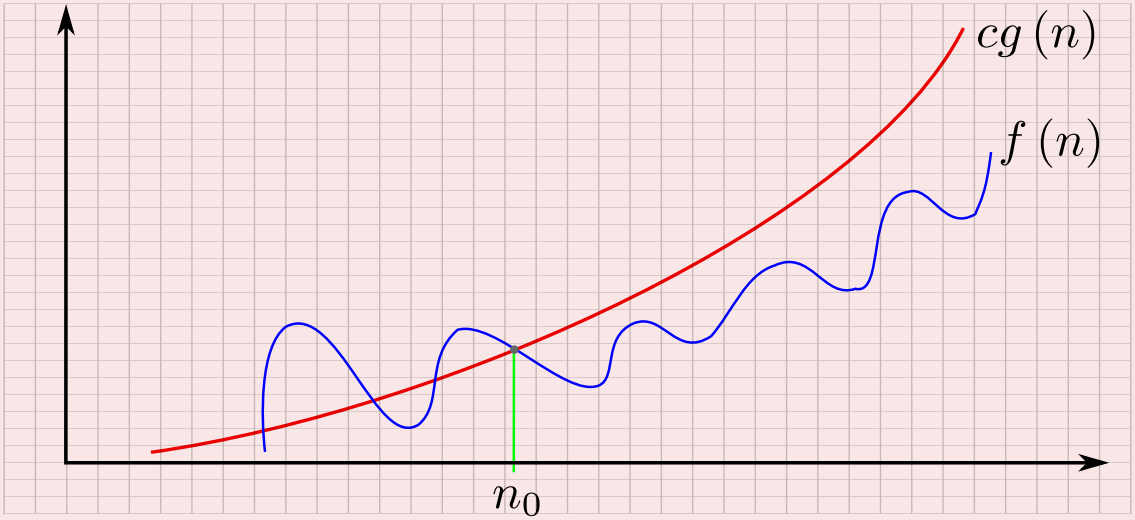
Convergence idea

Definition (Big O - Upper Bound) [4]

For a given function $g(n)$:

$$O(g(n)) = \{f(n) | \text{ There exists } c > 0 \text{ and } n_0 > 0 \text{ s.t. } 0 \leq f(n) \leq cg(n) \ \forall n \geq n_0\}$$

Example



What are the implications?

Definition [3]

- Suppose that the sequence $\{x_k\}$ converges to the number L:

What are the implications?

Definition [3]

- Suppose that the sequence $\{x_k\}$ converges to the number L:

We say that this sequence converges linearly to L, if there exists a number $\frac{1}{n} \in (0,1)$ such that

$$\lim_{k \longrightarrow \infty} \frac{|x_{n+1} - L|}{|x_n - L|} = \frac{1}{n}$$

What are the implications?

Definition [3]

- Suppose that the sequence $\{x_k\}$ converges to the number L:

We say that this sequence converges linearly to L, if there exists a number $\frac{1}{n} \in (0,1)$ such that

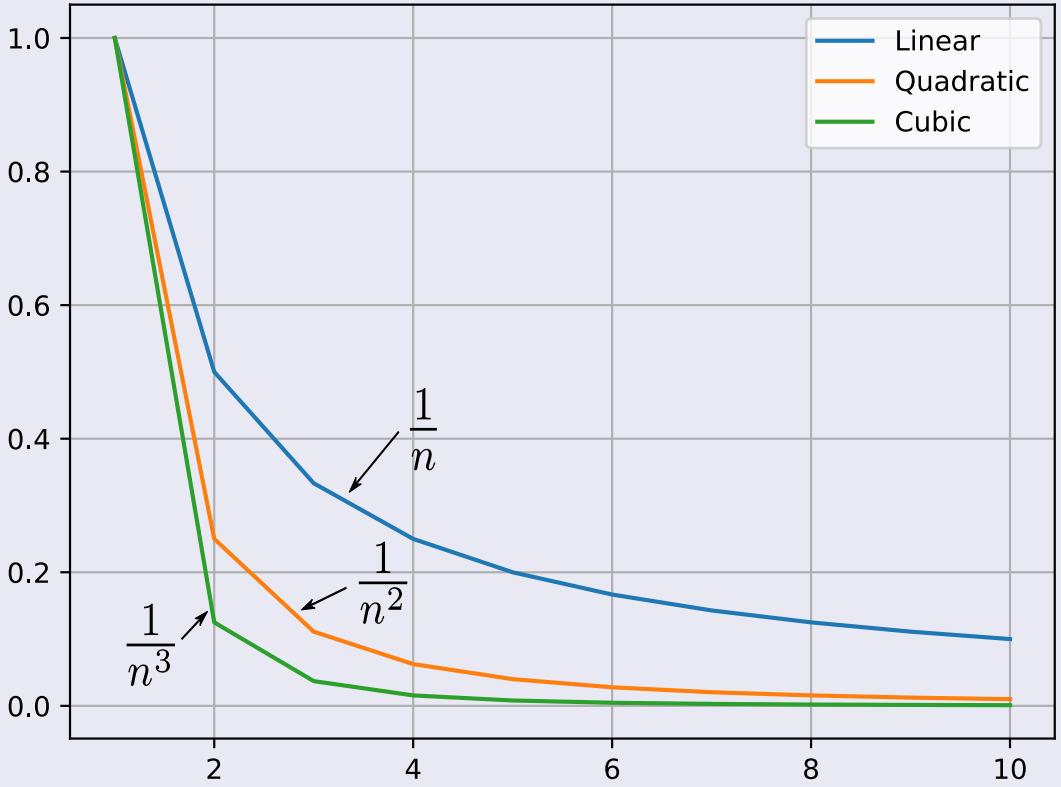
$$\lim_{k \longrightarrow \infty} \frac{|x_{n+1} - L|}{|x_n - L|} = \frac{1}{n}$$

Thus, Gradient Descent has a linear convergence speed

- If you do a comparison with quadratic convergence...

Example

As you can see the quadratic is faster than linear in convergence



Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Why the importance of Convex Functions?

There is an interest on the rates of convergence for many optimization algorithms

- And they are affected by the different cost function that can be used:
  - Lipschitz-continuity, convexity, strong convexity, and smoothness

Why the importance of Convex Functions?

There is an interest on the rates of convergence for many optimization algorithms

- And they are affected by the different cost function that can be used:
  - Lipschitz-continuity, convexity, strong convexity, and smoothness

There are different rates of convergence for the Gradient Descent

- For example when a function is strongly convex

$$\nabla^2 f(x) \succeq \alpha I \Longleftrightarrow \nabla^2 f(x) - \alpha I \succeq 0 \text{ (Matrix greater of equal)}$$

Why the importance of Convex Functions?

There is an interest on the rates of convergence for many optimization algorithms

- And they are affected by the different cost function that can be used:
  - Lipschitz-continuity, convexity, strong convexity, and smoothness

There are different rates of convergence for the Gradient Descent

- For example when a function is strongly convex

$$\nabla^2 f(x) \succeq \alpha I \Longleftrightarrow \nabla^2 f(x) - \alpha I \succeq 0 \text{ (Matrix greater of equal)}$$

This means that

- The curvature of $f(\boldsymbol{x})$ is not very close to zero, making possible to accelerate the convergence

Convex Sets

Definition

- For a convex set X, for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\boldsymbol{x}, \boldsymbol{y} \in X$, the line between them lies within the set

$$\boldsymbol{z} = \lambda \boldsymbol{x} + (1 - \lambda)\, \boldsymbol{y},\ \forall \theta \in (0, 1) \text{ then } \boldsymbol{z} \in X$$

  - The sum $\lambda \boldsymbol{x} + (1 - \lambda)\, \boldsymbol{y}$ is termed as convex linear combination.

Convex Functions

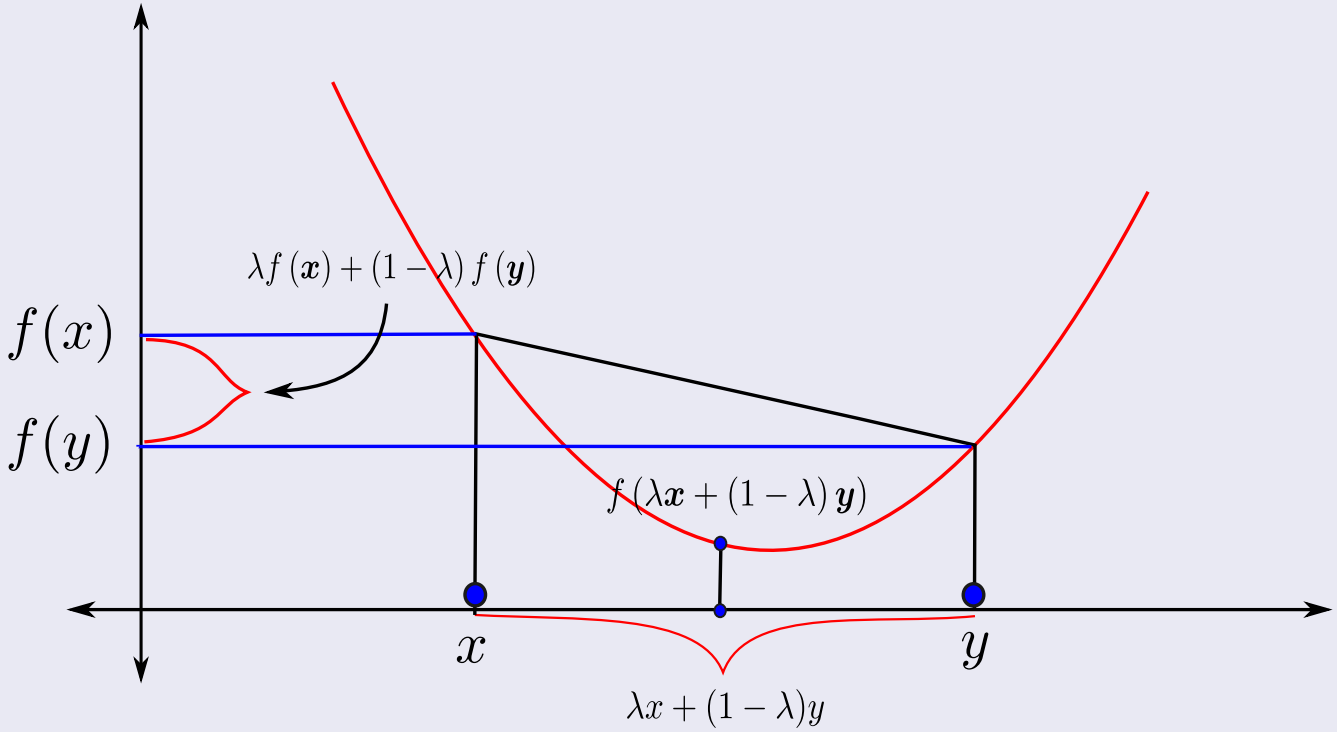
Definition

- . A function $f(\boldsymbol{x})$ is convex if the following holds:
  1. The Domain of f is convex
  2. $\forall \boldsymbol{x}, \boldsymbol{y}$ in the Domain of f and $\lambda \in (0, 1)$

$$f\left(\lambda \boldsymbol{x} + (1-\lambda)\,\boldsymbol{y}\right) \leq \lambda f\left(\boldsymbol{x}\right) + (1-\lambda)\, f\left(\boldsymbol{y}\right)$$

Graphically

This can further expanded to functions



Outline

1. Introduction
   - Review Gradient Descent
   - The Problems of Gradient Descent with Large Data Sets
   - Convergence of gradient descent with fixed step size
   - Convergence Rate
     - Convex Functions
     - Back to the Main Problem
   - Accelerating the Gradient Descent
   - Even with such Speeds

2. Accelerating Gradient Descent
   - Robbins-Monro Theorem
   - Robbins-Monro Scheme for Minimum-Square Error
   - Convergence

3. Improving and Measuring Stochastic Gradient Descent
   - Example of SGD Vs BGD
   - Using The Expected Value, The Mini-Batch
   - The Least-Mean Squares Adaptive Algorithm
   - Conclusions

Convergence of gradient descent with **fixed step size**

Theorem

- Suppose the function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable, and we have that $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{y}\|$ (Lipschitz Continuous Gradient) for any $\boldsymbol{x}, \boldsymbol{y}$ and $L > 0$.

Convergence of gradient descent with **fixed step size**

Theorem

- Suppose the function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and differentiable, and we have that $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{y}\|$ (Lipschitz Continuous Gradient) for any $\boldsymbol{x}, \boldsymbol{y}$ and $L > 0$.

We have that

- Then, if we run the **gradient descent** for k iterations with a fixed step size $\eta \leq \frac{1}{L}$, it will yield a solution f_n which satisfies

$$f(x_n) - f(x^*) \leq \frac{\left\|x_{(0)} - x^*\right\|_2^2}{2\eta n}$$

where $f(x^*)$ is the optimal value.

Proof

$f(\boldsymbol{x})$ is Lipschitz continuous with constant L implies ($\|\boldsymbol{y}-\boldsymbol{x}\|^2 = \|\boldsymbol{y}-\boldsymbol{x}\|_2^2$)

$$\nabla^2 f(x) - LI \text{ as semi-definite matrix}$$

Proof

$f(\boldsymbol{x})$ is Lipschitz continuous with constant L implies ($\|\boldsymbol{y}-\boldsymbol{x}\|^2 = \|\boldsymbol{y}-\boldsymbol{x}\|_2^2$)

$$\nabla^2 f(x) - LI \text{ as semi-definite matrix}$$

We have the following inequality

$$\begin{aligned} f(\boldsymbol{y}) &= f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y}-\boldsymbol{x}) + \frac{1}{2}\nabla^2 f(\boldsymbol{x}) \|\boldsymbol{y}-\boldsymbol{x}\|^2 \\ &\leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y}-\boldsymbol{x}) + \frac{1}{2} L \|\boldsymbol{y}-\boldsymbol{x}\|^2 \end{aligned}$$

Proof

Now, if we apply the Gradient update $\boldsymbol{y} = \boldsymbol{x}^+ = \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$

$$
\begin{aligned}
f\left(\boldsymbol{x}^+\right) &\leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \left(\boldsymbol{x}^+ - \boldsymbol{x}\right) + \frac{1}{2} L \left\|\boldsymbol{x}^+ - \boldsymbol{x}\right\|^2 \\
&= f(\boldsymbol{x}) - \eta \|\nabla f(\boldsymbol{x})\|^2 + \frac{1}{2} L \eta^2 \|\nabla f(\boldsymbol{x})\|^2 \\
&= f(\boldsymbol{x}) - \left(1 - \frac{1}{2} L \eta\right) \eta \|\nabla f(\boldsymbol{x})\|^2
\end{aligned}
$$

Proof

Now, if we apply the Gradient update $\boldsymbol{y} = \boldsymbol{x}^+ = \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$

$$\begin{aligned} f\left(\boldsymbol{x}^+\right) &\leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \left(\boldsymbol{x}^+ - \boldsymbol{x}\right) + \frac{1}{2} L \left\|\boldsymbol{x}^+ - \boldsymbol{x}\right\|^2 \\ &= f(\boldsymbol{x}) - \eta \|\nabla f(\boldsymbol{x})\|^2 + \frac{1}{2} L \eta^2 \|\nabla f(\boldsymbol{x})\|^2 \\ &= f(\boldsymbol{x}) - \left(1 - \frac{1}{2} L \eta\right) \eta \|\nabla f(\boldsymbol{x})\|^2 \end{aligned}$$

Using $\eta \leq \frac{1}{L}$

$$-\left(1 - \frac{1}{2} L \eta\right) \leq -\frac{1}{2}$$

Therefore

We have that

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}\right) - \frac{1}{2}\eta \left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2} \tag{2}$$

Therefore

We have that

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}\right) - \frac{1}{2}\eta \left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2} \tag{2}$$

Implying that

- This inequality implies that the objective function value strictly decreases until it reaches the optimal value

Therefore

We have that

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}\right) - \frac{1}{2}\eta \left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2} \tag{2}$$

Implying that

- This inequality implies that the objective function value strictly decreases until it reaches the optimal value

This only holds when η is small enough

- This explains why we observe in practice that gradient descent diverges when the step size is too large.

Since f is convex

We can write

$$f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{x}^* - \boldsymbol{x})$$
$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x})^T (\boldsymbol{x} - \boldsymbol{x}^*)$$

Since f is convex

We can write

$$f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{x}^* - \boldsymbol{x})$$
$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x})^T (\boldsymbol{x} - \boldsymbol{x}^*)$$

This comes from the "First order condition for convexity"

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x})$$

Then

Plugging this in to (Equation 2)

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}^{*}\right) + \nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{x} - \boldsymbol{x}^{*}\right) - \frac{1}{2}\eta\left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2}$$

Then

Plugging this in to (Equation 2)

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}^{*}\right) + \nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{x} - \boldsymbol{x}^{*}\right) - \frac{1}{2}\eta\left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2}$$

Therefore

$$f\left(\boldsymbol{x}^{+}\right) - f\left(\boldsymbol{x}^{*}\right) \leq \frac{1}{2\eta}\left[\left\|\boldsymbol{x} - \boldsymbol{x}^{*}\right\|^{2} - \left\|\boldsymbol{x} - \eta\nabla f\left(\boldsymbol{x}\right) - \boldsymbol{x}^{*}\right\|^{2}\right]$$

Then

Plugging this in to (Equation 2)

$$f\left(\boldsymbol{x}^{+}\right) \leq f\left(\boldsymbol{x}^{*}\right)+\nabla f\left(\boldsymbol{x}\right)^{T}\left(\boldsymbol{x}-\boldsymbol{x}^{*}\right)-\frac{1}{2}\eta\left\|\nabla f\left(\boldsymbol{x}\right)\right\|^{2}$$

Therefore

$$f\left(\boldsymbol{x}^{+}\right)-f\left(\boldsymbol{x}^{*}\right) \leq \frac{1}{2\eta}\left[\left\|\boldsymbol{x}-\boldsymbol{x}^{*}\right\|^{2}-\left\|\boldsymbol{x}-\eta\nabla f\left(\boldsymbol{x}\right)-\boldsymbol{x}^{*}\right\|^{2}\right]$$

Then plugging this $\boldsymbol{x}^{+}=\boldsymbol{x}-\eta\nabla f\left(\boldsymbol{x}\right)$ into

$$f\left(\boldsymbol{x}^{+}\right)-f\left(\boldsymbol{x}^{*}\right) \leq \frac{1}{2\eta}\left[\left\|\boldsymbol{x}-\boldsymbol{x}^{*}\right\|^{2}-\left\|\boldsymbol{x}^{+}-\boldsymbol{x}^{*}\right\|^{2}\right]$$

Then

Summing over all iterations and the telescopic sum in the right side

$$\sum_{i=1}^{n}\left[f\left(\boldsymbol{x}^{(i)}\right)-f\left(\boldsymbol{x}^{*}\right)\right]\leq\frac{1}{2\eta}\left[\left\|\boldsymbol{x}^{(0)}-\boldsymbol{x}^{*}\right\|^{2}\right]$$

Then

Summing over all iterations and the telescopic sum in the right side

$$\sum_{i=1}^{n}\left[f\left(\boldsymbol{x}^{(i)}\right)-f\left(\boldsymbol{x}^{*}\right)\right]\leq\frac{1}{2\eta}\left[\left\|\boldsymbol{x}^{(0)}-\boldsymbol{x}^{*}\right\|^{2}\right]$$

Finally, using the fact that f decreasing on every iteration

$$f\left(\boldsymbol{x}^{(n)}\right)-f\left(\boldsymbol{x}^{*}\right)\leq\frac{1}{n}\sum_{i=1}^{n}\left[f\left(\boldsymbol{x}^{(i)}\right)-f\left(\boldsymbol{x}^{*}\right)\right]\leq\frac{1}{2\eta n}\left[\left\|\boldsymbol{x}^{(0)}-\boldsymbol{x}^{*}\right\|^{2}\right]=C\times\frac{1}{n}$$

Therefore

It converges with rate

$$O\left(\frac{1}{n}\right)$$

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Accelerating the Gradient Descent

It is possible to modify the Batch Gradient Descent

- In order to accelerate it several modifications have been proposed

Accelerating the Gradient Descent

It is possible to modify the Batch Gradient Descent

- In order to accelerate it several modifications have been proposed

Possible Methods

- Polyak's Momentum Method or Heavy-Ball Method (1964)
- Nesterov's Proposal (1983)
- Stochastic Gradient Descent (1951)

Polyak's Momentum Method

Polyak's Step Size

- He Proposed that the step size could be modified to

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \nabla f\left(\boldsymbol{w}_n\right) + \mu\left(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}\right) \text{ with } \mu \in [0,1], \alpha > 0$$

Polyak's Momentum Method

Polyak's Step Size

- He Proposed that the step size could be modified to

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \nabla f\left(\boldsymbol{w}_n\right) + \mu\left(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}\right) \text{ with } \mu \in [0,1], \alpha > 0$$

Basically, the method uses the previous gradient information through the step difference $(\boldsymbol{w}_n - \boldsymbol{w}_{n-1})$

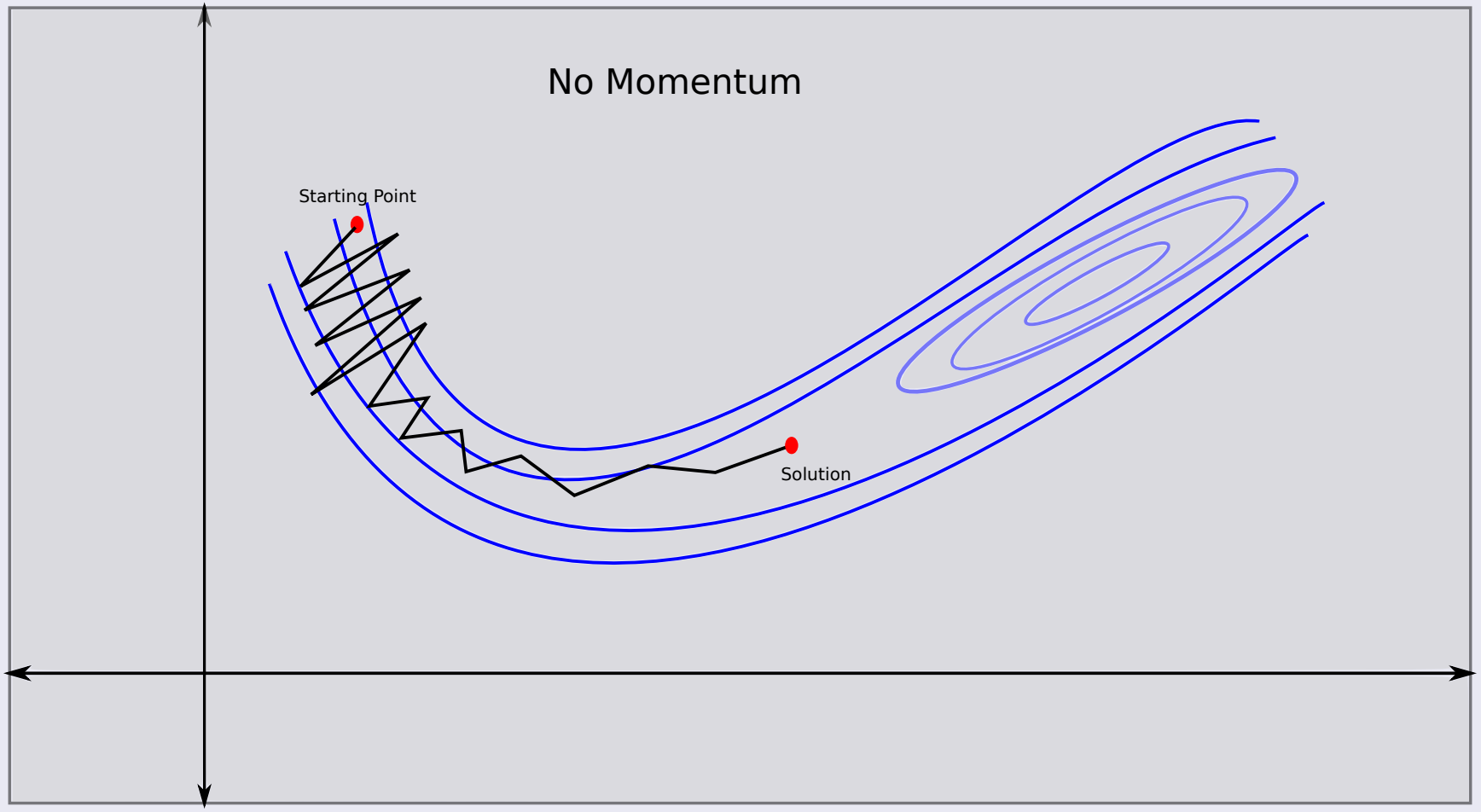
- By the discretization of the second order ODE

$$\ddot{\boldsymbol{w}} + a\dot{\boldsymbol{w}} + b\nabla f\left(\boldsymbol{w}\right) = 0$$

  - **which models the motion of a body in a potential field given by f with friction.**

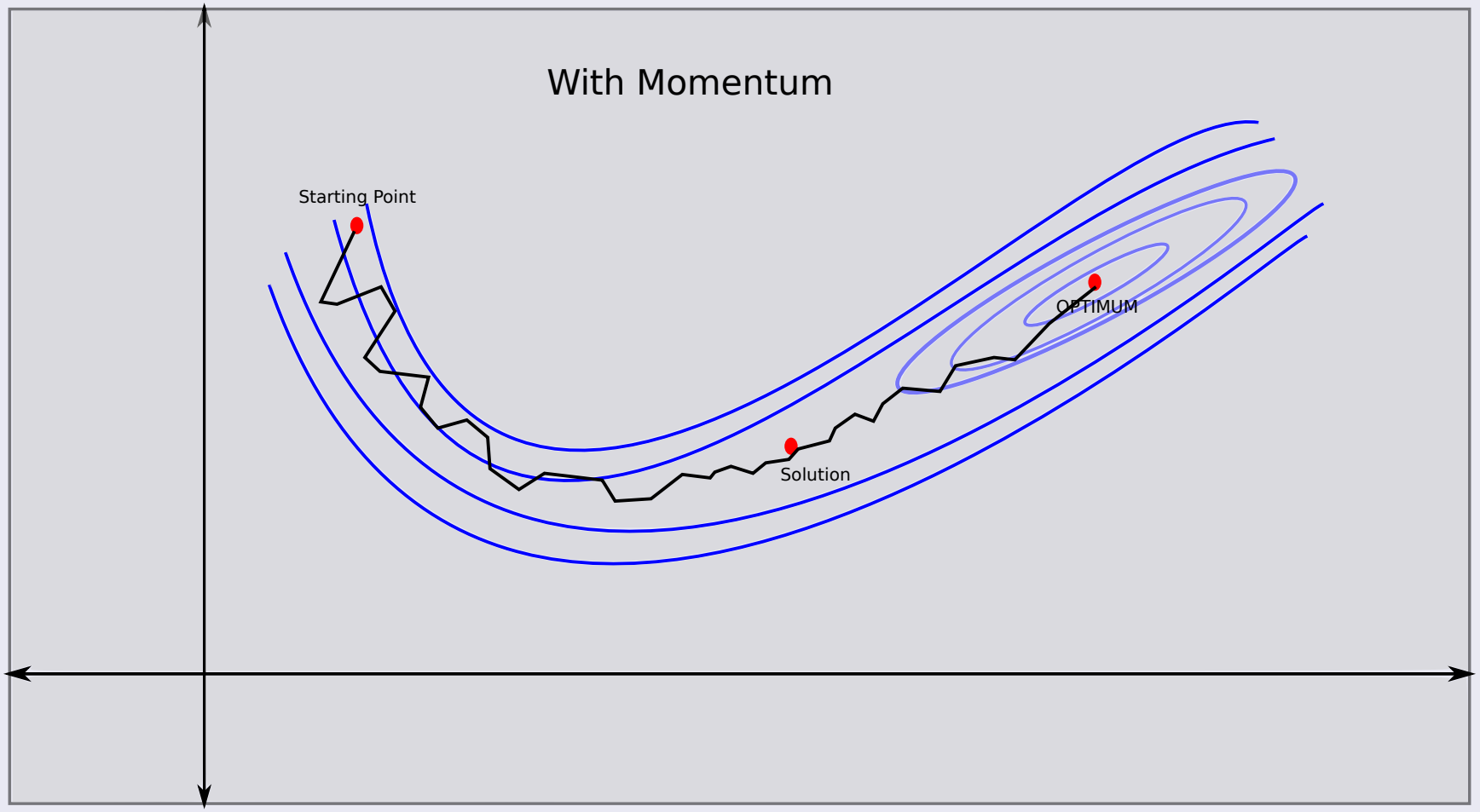
The Momentum helps to stabilize the GD

If we do not have Momentum



Then, with Momentum

If we have Momentum



Problem

It has been proved that the method has problems

- L. Lessard, B. Recht, and A. Packard. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. ArXiv e-prints, Aug. 2014.

Problem

It has been proved that the method has problems

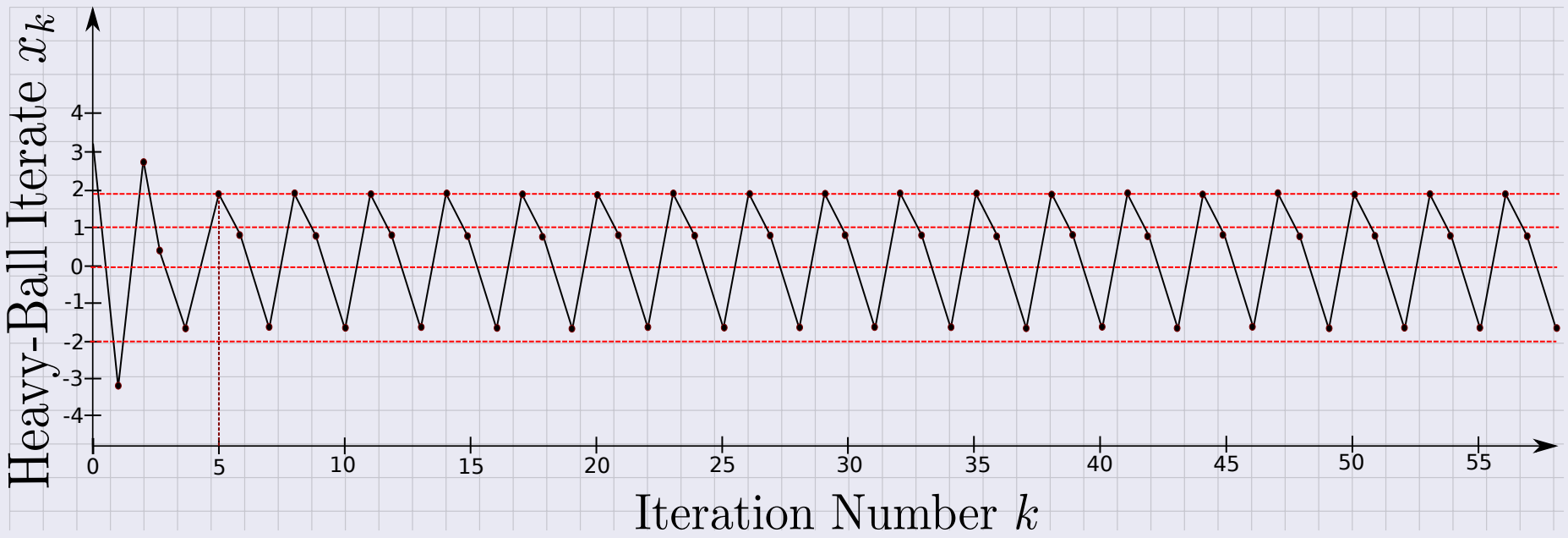
- L. Lessard, B. Recht, and A. Packard. Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints. ArXiv e-prints, Aug. 2014.

Under the function

$$\nabla f(x) = \begin{cases} 25x & \text{if } x < 1 \\ x + 24 & \text{if } 1 \leq x \leq 2 \\ 25x - 24 & \text{if otherwise} \end{cases}$$

In Lessard et al.

We have a non-convergence (Original Lessard et al.) [5]



Nesterov's Proposal

He proposed a Quasi-Convex Combination

- Instead to use

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \nabla f\left(\boldsymbol{w}_n\right) + \mu\left(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}\right)$$

Nesterov's Proposal

He proposed a Quasi-Convex Combination

- Instead to use

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \nabla f\left(\boldsymbol{w}_n\right) + \mu\left(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}\right)$$

Have an intermediate step to update $\boldsymbol{w}_{n+1}$

$$\boldsymbol{w}_{n+1} = \left(1 - \gamma_n\right) \boldsymbol{y}_{n+1} + \gamma_n \boldsymbol{y}_n$$

Nesterov's Proposal

He proposed a Quasi-Convex Combination

- Instead to use

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \alpha \nabla f\left(\boldsymbol{w}_n\right) + \mu\left(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}\right)$$

Have an intermediate step to update $\boldsymbol{w}_{n+1}$

$$\boldsymbol{w}_{n+1} = \left(1 - \gamma_n\right) \boldsymbol{y}_{n+1} + \gamma_n \boldsymbol{y}_n$$

This allow to weight the actual original gradient change

- with the previous gradient change... making possible to avoid the original problem by Polyak... Which is based in Lyapunov Analysis

Nesterov's Proposal [6]

Nesterov's Accelerated Gradient Descent (A Quasi-Convex Modification)

$$\begin{aligned}\boldsymbol{y}_{n+1} =& \boldsymbol{w}_n - \frac{1}{\beta}\nabla J\left(\boldsymbol{w}_n\right)\\ \boldsymbol{w}_{n+1} =& \left(1-\gamma_n\right)\boldsymbol{y}_{n+1} + \gamma_n \boldsymbol{y}_n\end{aligned}$$

Nesterov's Proposal [6]

Nesterov's Accelerated Gradient Descent (A Quasi-Convex Modification)

$$\begin{aligned}\boldsymbol{y}_{n+1} =& \boldsymbol{w}_n - \frac{1}{\beta}\nabla J\left(\boldsymbol{w}_n\right)\\ \boldsymbol{w}_{n+1} =& \left(1-\gamma_n\right)\boldsymbol{y}_{n+1} + \gamma_n \boldsymbol{y}_n\end{aligned}$$

Where, we use the following constants

$$\begin{aligned}\lambda_0 =& 0\\ \lambda_n =& \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2}\\ \gamma_n =& \frac{1-\lambda_n}{\lambda_{n+1}}\end{aligned}$$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**
4. $\quad \boldsymbol{y}_{n+1} = \boldsymbol{w}_n - \frac{1}{\beta} \nabla J\left(\boldsymbol{w}_n\right)$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**
4. $\quad \boldsymbol{y}_{n+1} = \boldsymbol{w}_n - \frac{1}{\beta} \nabla J\left(\boldsymbol{w}_n\right)$
5. $\quad \lambda_n = \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2}$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**
4. $\quad \boldsymbol{y}_{n+1} = \boldsymbol{w}_n - \frac{1}{\beta} \nabla J\left(\boldsymbol{w}_n\right)$
5. $\quad \lambda_n = \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2}$
6. $\quad \lambda_{n+1} = \frac{1+\sqrt{1+4\lambda_n^2}}{2}$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**
4. $\quad \boldsymbol{y}_{n+1} = \boldsymbol{w}_n - \frac{1}{\beta} \nabla J\left(\boldsymbol{w}_n\right)$
5. $\quad \lambda_n = \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2}$
6. $\quad \lambda_{n+1} = \frac{1+\sqrt{1+4\lambda_n^2}}{2}$
7. $\quad \gamma_n = \frac{1-\lambda_n}{\lambda_{n+1}}$

Nesterov's Algorithm

Nesterov Accelerated Gradient

Input: Training Time T, Learning Rate β, an initialization $\boldsymbol{w}_0$

1. $y_0 \leftarrow \boldsymbol{w}_0$
2. $\lambda_0 \leftarrow 0$
3. **for** $t = 0$ **to** $T - 1$ **do**
4. $\quad \boldsymbol{y}_{n+1} = \boldsymbol{w}_n - \frac{1}{\beta} \nabla J\left(\boldsymbol{w}_n\right)$
5. $\quad \lambda_n = \frac{1+\sqrt{1+4\lambda_{n-1}^2}}{2}$
6. $\quad \lambda_{n+1} = \frac{1+\sqrt{1+4\lambda_n^2}}{2}$
7. $\quad \gamma_n = \frac{1-\lambda_n}{\lambda_{n+1}}$
8. $\quad \boldsymbol{w}_{n+1} = \left(1 - \gamma_n\right) \boldsymbol{y}_{n+1} + \gamma_n \boldsymbol{y}_n$

With the following complexity

Theorem (Nesterov 1983)

- Let f be a convex and β-smooth function (∇f is β-Lipschitz continous), then Nesterov's Accelerated Gradient Descent satisfies:

$$f\left(\boldsymbol{y}_{n+1}\right) - f\left(\boldsymbol{w}^*\right) \leq \frac{2\beta \left\|\boldsymbol{w}_1 - \boldsymbol{w}^*\right\|^2}{n^2}$$

With the following complexity

Theorem (Nesterov 1983)

- Let f be a convex and β-smooth function (∇f is β-Lipschitz continous), then Nesterov's Accelerated Gradient Descent satisfies:

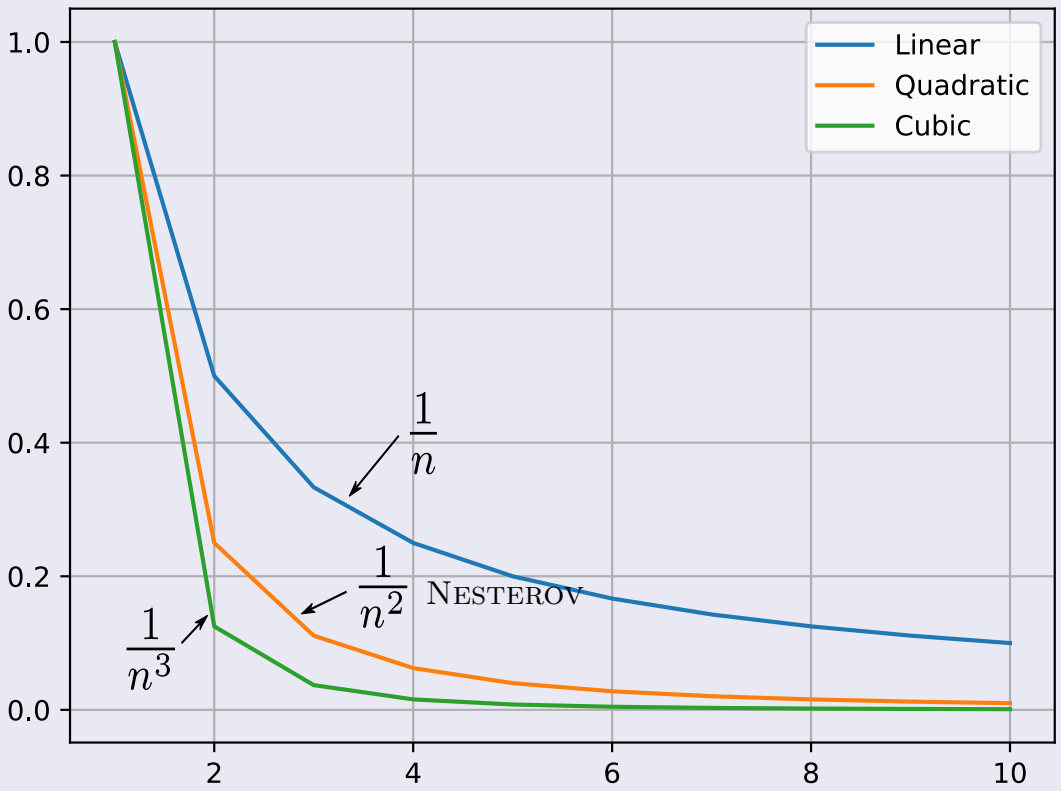
$$f\left(\boldsymbol{y}_{n+1}\right) - f\left(\boldsymbol{w}^*\right) \leq \frac{2\beta \left\|\boldsymbol{w}_1 - \boldsymbol{w}^*\right\|^2}{n^2}$$

It converges with rate

$$O\left(\frac{1}{n^2}\right)$$

Example

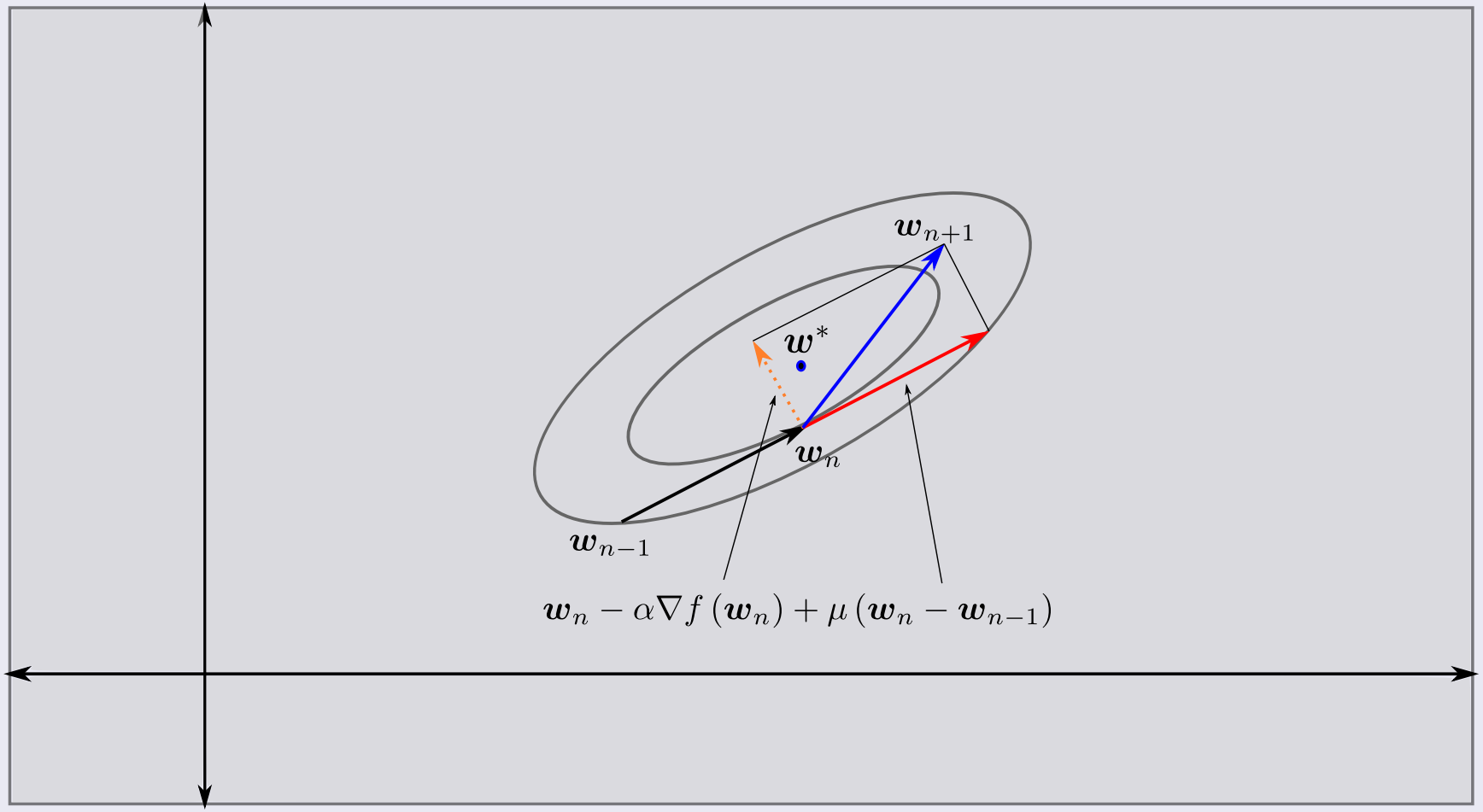
As you can see Nesterov is faster...



Remark, Polyak vs Nesterov

We have a remarkable difference

- The gradient descent step (orange arrow) is perpendicular to the level set before applying momentum to $\boldsymbol{w}_1$ (red arrow) in Polyak's algorithm



In the case of Nesterov

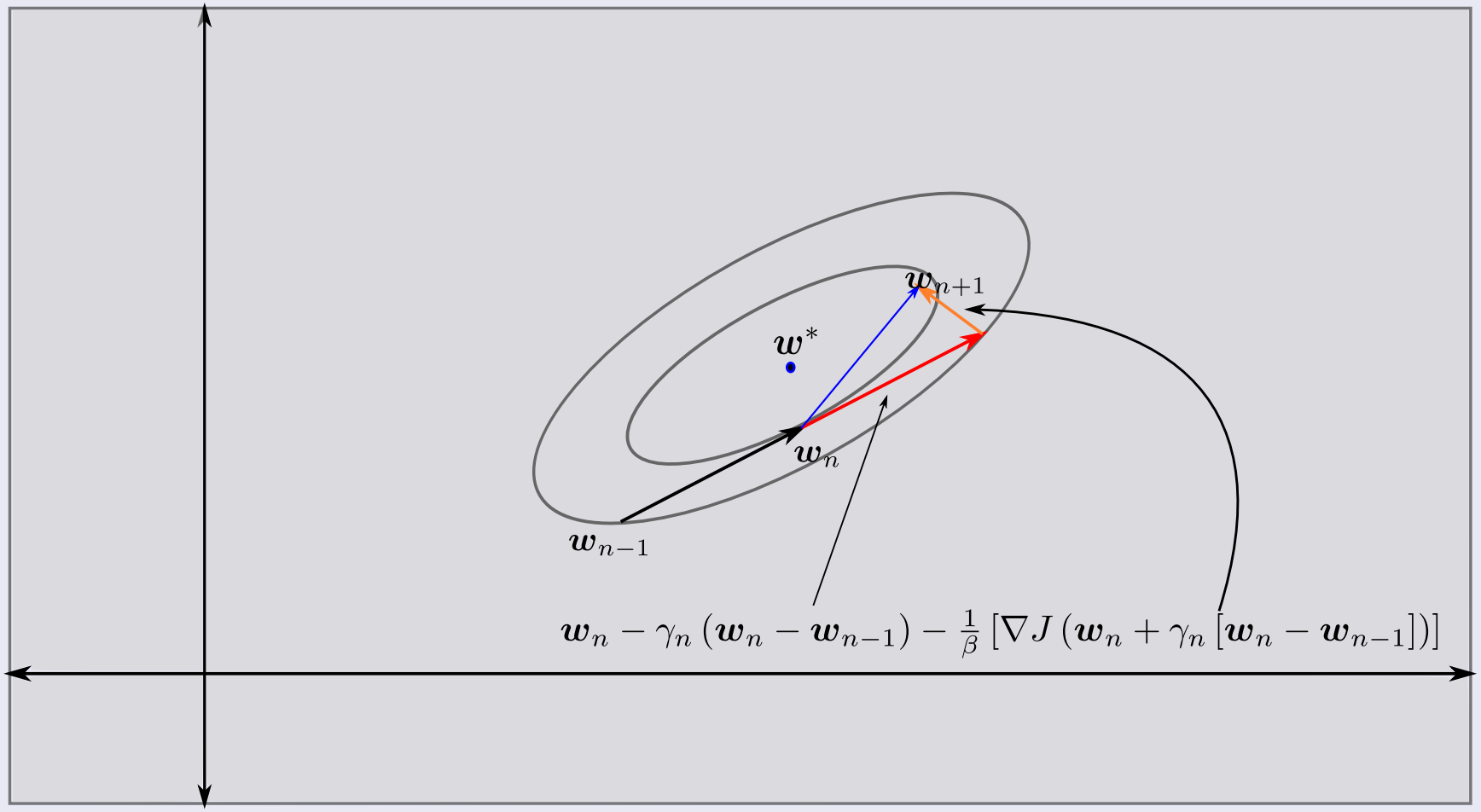
If we rewrite the equations

$$
\begin{aligned}
\boldsymbol{w}_{n+1} =& (1-\gamma_n)\left[\boldsymbol{w}_n - \frac{1}{\beta}\nabla J(\boldsymbol{w}_n)\right] + \gamma_n \boldsymbol{y}_n \\
=& \boldsymbol{w}_n - \gamma_n \boldsymbol{w}_n - \frac{1}{\beta}\nabla J(\boldsymbol{w}_n) + \frac{\gamma_n}{\beta}\nabla J(\boldsymbol{w}_n) + \gamma_n \boldsymbol{w}_{n-1} - \frac{\gamma_n}{\beta}\nabla J(\boldsymbol{w}_{n-1}) \\
=& \boldsymbol{w}_n - \gamma_n(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}) - \frac{1}{\beta}\left[\nabla J(\boldsymbol{w}_n) + \gamma_n \nabla J(\boldsymbol{w}_n) - \gamma_n \nabla J(\boldsymbol{w}_{n-1})\right] \\
=& \boldsymbol{w}_n - \gamma_n(\boldsymbol{w}_n - \boldsymbol{w}_{n-1}) - \frac{1}{\beta}\left[\nabla J(\boldsymbol{w}_n + \gamma_n[\boldsymbol{w}_n - \boldsymbol{w}_{n-1}])\right]
\end{aligned}
$$

In Nesterov

We have a remarkable difference

- it is perpendicular to the level set after applying momentum to $\boldsymbol{w}_1$ in Nesterov's algorithm.



There is a dependence with respect with different properties of f

In this table, we can see upper bounds for the convergences $D = \|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2$ and λ regularization term [7]

Properties of the Objective Function	Upper Bound for Gradient Descent
convex and L-Lipschitz	$\frac{D_1 L}{\sqrt{n}}$
convex and β-smooth	$\frac{\beta D_1^2}{n}$
α-strongly convex and L-Lipschitz	$\frac{L^2}{\alpha n}$
α-strongly convex and β-smooth	$\beta D_1^2 \exp\left(-\frac{4n}{\beta/\lambda}\right)$

A Hierarchy can be established (Black Box Model)

Based on the following idea

- A black box model assumes that the algorithm does not know the objective function f being minimized.

A Hierarchy can be established (Black Box Model)

Based on the following idea

- A black box model assumes that the algorithm does not know the objective function f being minimized.

Not only that

- Information about the objective function can only be accessed by querying an oracle.

A Hierarchy can be established (Black Box Model)

Based on the following idea

- A black box model assumes that the algorithm does not know the objective function f being minimized.

Not only that

- Information about the objective function can only be accessed by querying an oracle.

Remarks

- The oracle serves as a bridge between the unknown objective function and the optimizer.

Furthermore

At any given step, the optimizer queries the oracle with a guess x

- The oracle responds with information about the function around x

Furthermore

At any given step, the optimizer queries the oracle with a guess x

- The oracle responds with information about the function around x

For Example

- Value of the Cost function, Gradient, Hessian, etc.

Then, we have

Zeroth Order Methods

- These methods only require the value of function f at the current guess x.
  - The Bisection, Genetic Algorithms, Simulated Annealing and Metropolis-Hastings methods

Then, we have

Zeroth Order Methods

- These methods only require the value of function f at the current guess x.
  - The Bisection, Genetic Algorithms, Simulated Annealing and Metropolis-Hastings methods

First Order Methods

- These methods can inquire the value of the function f and its first derivative.
  - Gradient descent, Nesterov's and and Polyak's

Then, we have

Zeroth Order Methods

- These methods only require the value of function f at the current guess x.
  - The Bisection, Genetic Algorithms, Simulated Annealing and Metropolis-Hastings methods

First Order Methods

- These methods can inquire the value of the function f and its first derivative.
  - Gradient descent, Nesterov's and and Polyak's

Second Order Methods

- These methods require the value of the function f, its first derivative (gradient), and its second derivative (Hessian).
  - Newton's method. Improving the efficiency of these algorithms is an active area of research.

One of the Last Hierarchy

Adaptive Methods and Conjugate Gradients

- The methods we mentioned until this point assume that all dimensions of a vector-valued variable have a common set of hyperparameters.

One of the Last Hierarchy

Adaptive Methods and Conjugate Gradients

- The methods we mentioned until this point assume that all dimensions of a vector-valued variable have a common set of hyperparameters.

Adaptive methods relax this assumption

- They allow for every variable to have its own set of hyperparameters.

One of the Last Hierarchy

Adaptive Methods and Conjugate Gradients

- The methods we mentioned until this point assume that all dimensions of a vector-valued variable have a common set of hyperparameters.

Adaptive methods relax this assumption

- They allow for every variable to have its own set of hyperparameters.

Some popular methods under this paradigm

- AdaGrad and ADAM

Finally, but not less important

Lower Bounds

- Lower bounds are useful because they tell us what's the best possible rate of convergence we can have given a category of optimizer.

Finally, but not less important

Lower Bounds

- Lower bounds are useful because they tell us what's the best possible rate of convergence we can have given a category of optimizer.

Something Notable

- Without lower bounds, an unnecessary amount of research energy would be spent in designing better optimizers
  - Even if convergence rate improvement is impossible within this category of algorithm

Finally, but not less important

Lower Bounds

- Lower bounds are useful because they tell us what's the best possible rate of convergence we can have given a category of optimizer.

Something Notable

- Without lower bounds, an unnecessary amount of research energy would be spent in designing better optimizers
  - Even if convergence rate improvement is impossible within this category of algorithm

However, if we prove that each procedure has a lower bounded rate of convergence

- We do not know if a specific method reaches this bound.

However

Please, take a look

- Convex Optimization: Algorithms and Complexity by Sébastien Bubeck - Theory Group, Microsoft Research [7]

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

In our classic Convex Scenario [2]

Least Square Problem locking to minimize the average of the LSE

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2M} \sum_{m=1}^{M} \left(\boldsymbol{w}^T \boldsymbol{x}_m - y_m\right)^2 = \min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2M} \|X\boldsymbol{w} - Y\|^2$$

Therefore

Calculating the Gradient

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} \left(\boldsymbol{w}^T \boldsymbol{x}_m - y_m \right) \boldsymbol{x}_m$$

Observations

It is easy to verify that the complexity per iteration is $O(dM)$

- With M is for the sum and d is for $\boldsymbol{w}^T \boldsymbol{x}_m$.

Drawbacks

When the number of samples M is Large

- Even with a rate of linear convergence, Gradient Descent

Drawbacks

When the number of samples M is Large

- Even with a rate of linear convergence, Gradient Descent

Not only that but in the On-line Learning scenario

- The data $(\boldsymbol{x}_i, y_i)$ is coming one by one making the gradient not computable.

Therefore

Thus, the need to look for something faster

- Two possibilities:

Therefore

Thus, the need to look for something faster

- Two possibilities:
  - Accelerating Gradient Decent Using Stochastic Gradient Descent!!!

Therefore

Thus, the need to look for something faster

- Two possibilities:
  - Accelerating Gradient Decent Using Stochastic Gradient Descent!!!
  - Accelerating Gradient Descent Using The Best of Both World, Min-Batch!!!

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

We have that the Robbins-Monro Theorem[8]

The origins of such techniques are traced back to 1951

- When Robbins and Monro introduced the method of stochastic approximation
  - DARPA project!!!

We have that the Robbins-Monro Theorem[8]

The origins of such techniques are traced back to 1951

- When Robbins and Monro introduced the method of stochastic approximation
  - DARPA project!!!

Setup, given a function $M(\boldsymbol{w})$ and a constant α such that the equation

$$M(\boldsymbol{w}) = \alpha$$

- It has a unique root $\boldsymbol{w} = \boldsymbol{w}^*$

Goal

We want to compute the root, $\boldsymbol{w}$, of such equation

$$M\left(\boldsymbol{w}^*\right) = \alpha$$

Goal

We want to compute the root, $\boldsymbol{w}$, of such equation

$$M\left(\boldsymbol{w}^{*}\right)=\alpha$$

Then, we want to generate values $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{n-1}$ thus, we generate $\boldsymbol{w}_n$ from

1. $M\left(\boldsymbol{w}_1\right), M\left(\boldsymbol{w}_2\right), ..., M\left(\boldsymbol{w}_{n-1}\right)$
2. and the possible derivatives $M'\left(\boldsymbol{w}_1\right), M'\left(\boldsymbol{w}_2\right), ..., M'\left(\boldsymbol{w}_{n-1}\right)$

Goal

We want to compute the root, $\boldsymbol{w}$, of such equation

$$M\left(\boldsymbol{w}^{*}\right)=\alpha$$

Then, we want to generate values $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{n-1}$ thus, we generate $\boldsymbol{w}_n$ from

1. $M\left(\boldsymbol{w}_1\right), M\left(\boldsymbol{w}_2\right), ..., M\left(\boldsymbol{w}_{n-1}\right)$
2. and the possible derivatives $M'\left(\boldsymbol{w}_1\right), M'\left(\boldsymbol{w}_2\right), ..., M'\left(\boldsymbol{w}_{n-1}\right)$

Thus, we would love that

$$\lim_{n\to\infty} \boldsymbol{w}_n = \boldsymbol{w}^{*}$$

Instead, we suppose that for each $\boldsymbol{w}$ corresponds a Random Variable $Y = Y(\boldsymbol{w})$

This Random Variable has a distribution function

$$Pr\left[Y(\boldsymbol{w}) \leq y\right] = H(y|\boldsymbol{w})$$

Instead, we suppose that for each $\boldsymbol{w}$ corresponds a Random Variable $Y = Y(\boldsymbol{w})$

This Random Variable has a distribution function

$$Pr\left[Y(\boldsymbol{w}) \leq y\right] = H(y|\boldsymbol{w})$$

Such that

$$M(\boldsymbol{w}) = \int_{-\infty}^{\infty} y dH(y|\boldsymbol{w})$$

We Postulate

First a bound to the $M(\boldsymbol{w})$

$$|M(\boldsymbol{w})| \leq C < \infty, \quad \int_{-\infty}^{\infty} (y - M(\boldsymbol{w}))^2 \, dH(y|\boldsymbol{w}) \leq \sigma^2 < \infty$$

IMPORTANT

Neither the exact nature of $H(y|\boldsymbol{w})$ nor that of $M(\boldsymbol{w})$ is known

- But an important assumption is that

$$M(\boldsymbol{w}) - \alpha = 0$$

It has only one root

IMPORTANT

Neither the exact nature of $H(y|\boldsymbol{w})$ nor that of $M(\boldsymbol{w})$ is known

- But an important assumption is that

$$M(\boldsymbol{w}) - \alpha = 0$$

It has only one root

Here is we use the α value to generate the root by assuming

- $M(\boldsymbol{w}) - \alpha \leq 0$ for $\boldsymbol{w} \leq \boldsymbol{w}^*$ and $M(\boldsymbol{w}) - \alpha \geq 0$ for $\boldsymbol{w} > \boldsymbol{w}^*$.

Now, For a positive δ

$M(\boldsymbol{w})$ is strictly increasing if

$$\|\boldsymbol{w}^* - \boldsymbol{w}\| < \delta$$

Now, For a positive δ

$M(\boldsymbol{w})$ is strictly increasing if

$$\|\boldsymbol{w}^* - \boldsymbol{w}\| < \delta$$

And Finally

$$\inf_{\|\boldsymbol{w}^* - \boldsymbol{w}\| \geq \delta} |M(\boldsymbol{w}) - \alpha| > 0$$

Now choose a sequence $\{\mu_i\}$

Such that

$$\sum_{i=1}^{\infty} \mu_i^2 = A < \infty \text{ and } \sum_{i=1}^{\infty} \mu_i = \infty$$

Now choose a sequence $\{\mu_i\}$

Such that

$$\sum_{i=1}^{\infty} \mu_i^2 = A < \infty \text{ and } \sum_{i=1}^{\infty} \mu_i = \infty$$

Now, we define a non-stationary Markov Chain $\{\boldsymbol{w}_n\}$

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = \mu_n \left(\alpha - y_n\right)$$

Now choose a sequence $\{\mu_i\}$

Such that

$$\sum_{i=1}^{\infty} \mu_i^2 = A < \infty \text{ and } \sum_{i=1}^{\infty} \mu_i = \infty$$

Now, we define a non-stationary Markov Chain $\{\boldsymbol{w}_n\}$

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = \mu_n \left(\alpha - y_n\right)$$

Where y_n is a random variable such that

$$Pr\left[y_n \leq y | \boldsymbol{w}_n\right] = H\left(y | \boldsymbol{w}_n\right)$$

Using the expected value!!!

Here, we define b_n

$$b_n = E\left[\boldsymbol{w}_n - \boldsymbol{w}^*\right]^2$$

Using the expected value!!!

Here, we define b_n

$$b_n = E\left[\boldsymbol{w}_n - \boldsymbol{w}^*\right]^2$$

We want conditions where this variance goes to zero

$$\lim_{n\to\infty} b_n = 0$$

- No matter what is the initial value $\boldsymbol{w}_0$.

We have then

Based on

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = \mu_n \left(\alpha - y_n \right)$$

We have then

Based on

$$\boldsymbol{w}_{n+1} - \boldsymbol{w}_n = \mu_n \left(\alpha - y_n\right)$$

We have then

$$\begin{aligned}
b_{n+1} =& E\left[\boldsymbol{w}_{n+1} - \boldsymbol{w}^*\right]^2 = E\left[E\left[\boldsymbol{w}_{n+1} - \boldsymbol{w}^*\right]^2 |\boldsymbol{w}_n\right] \\
=& E\left[\int_{-\infty}^{\infty} \left[\boldsymbol{w}_n - \boldsymbol{w}^* - \mu_n \left(y - \alpha\right)^2\right] dH\left(y|\boldsymbol{w}_n\right)\right] \\
=& b_n + \mu_n E\left[\int_{-\infty}^{\infty} \left(y - \alpha\right)^2 dH\left(y|\boldsymbol{w}_n\right)\right] - 2\mu_n E\left[\left(\boldsymbol{w}_n - \boldsymbol{w}^*\right)\left(M\left(\boldsymbol{w}_n\right) - \alpha\right)\right] \\
=& b_n + \mu_n^2 e_n - 2\mu_n d_n
\end{aligned}$$

With Values

We have

$$d_n = E\left[\left(\boldsymbol{w}_n - \boldsymbol{w}^*\right)\left(M\left(\boldsymbol{w}_n\right) - \alpha\right)\right]$$
$$e_n = E\left[\int_{-\infty}^{\infty} \left(y - \alpha\right)^2 dH\left(y|\boldsymbol{w}_n\right)\right]$$

With Values

We have

$$d_n = E\left[\left(\boldsymbol{w}_n - \boldsymbol{w}^*\right)\left(M\left(\boldsymbol{w}_n\right) - \alpha\right)\right]$$
$$e_n = E\left[\int_{-\infty}^{\infty} \left(y - \alpha\right)^2 dH\left(y|\boldsymbol{w}_n\right)\right]$$

From $M\left(\boldsymbol{w}\right) \leq \alpha$ for $\boldsymbol{w} \leq \boldsymbol{w}^*$ and $M\left(\boldsymbol{w}\right) \geq \alpha$ for $\boldsymbol{w} > \boldsymbol{w}^*$

$$d_n \geq 0$$

Additionally

Now, assuming that exist C such that

$$Pr\left[\left|Y\left(\boldsymbol{w}\right)\right| \leq C\right] = \int_{-C}^{C} dH\left(y|\boldsymbol{w}\right) = 1 \ \forall x$$

Additionally

Now, assuming that exist C such that

$$Pr\left[\left|Y\left(\boldsymbol{w}\right)\right| \leq C\right] = \int_{-C}^{C} dH\left(y|\boldsymbol{w}\right) = 1 \ \forall x$$

We can prove that

$$0 \leq e_n \leq \left[C + \left|\alpha\right|^2\right] < \infty$$

Additionally

Now, assuming that exist C such that

$$Pr\left[\left|Y\left(\boldsymbol{w}\right)\right| \leq C\right] = \int_{-C}^{C} dH\left(y|\boldsymbol{w}\right) = 1 \ \forall x$$

We can prove that

$$0 \leq e_n \leq \left[C + \left|\alpha\right|^2\right] < \infty$$

Now, given

$$\sum_{i=1}^{\infty} \mu_i^2 = A < \infty \text{ and } \sum_{i=1}^{\infty} \mu_i = \infty$$

Therefore $\sum_{i=1}^{\infty} \mu_i^2 e_i$ converges

Then, summing over i we obtain

$$b_{n+1} = b_1 + \sum_{i=1}^{n} \mu_i^2 e_i - 2 \sum_{i=1}^{n} \mu_i d_i$$

Therefore $\sum_{i=1}^{\infty} \mu_i^2 e_i$ converges

Then, summing over i we obtain

$$b_{n+1} = b_1 + \sum_{i=1}^{n} \mu_i^2 e_i - 2\sum_{i=1}^{n} \mu_i d_i$$

Since $b_{n+1} \geq 0$

$$\sum_{i=1}^{n} \mu_i d_i \leq \frac{1}{2}\left[b_1 + \sum_{i=1}^{n} \mu_i^2 e_i\right] < \infty$$

Then

Hence the positive-term series

$$\sum_{i=1}^{\infty} \mu_i d_i \text{ converges}$$

Then

Hence the positive-term series

$$\sum_{i=1}^{\infty} \mu_i d_i \text{ converges}$$

Then, $\lim_{n\to\infty} b_n$ exists and...

$$\lim_{n\to\infty} b_n = b_1 + \sum_{i=1}^{\infty} \mu_i^2 e_i - 2\sum_{i=1}^{\infty} \mu_i d_i = b$$

Therefore

If a sequence of $\{k_i\}$ of non-negative constants such that

$$d_i \geq k_i b_i, \quad \sum_{i=1}^{\infty} \mu_i k_i = \infty$$

Therefore

If a sequence of $\{k_i\}$ of non-negative constants such that

$$d_i \geq k_i b_i, \quad \sum_{i=1}^{\infty} \mu_i k_i = \infty$$

We want to prove that

$$\sum_{i=1}^{\infty} \mu_i k_i b_i < \infty$$

For this

We know that

$$\sum_{i=1}^{\infty} \mu_i d_i \text{ converges}$$

For this

We know that

$$\sum_{i=1}^{\infty} \mu_i d_i \text{ converges}$$

Therefore

$$k_i b_i \leq d_i \Rightarrow \mu_i k_i b_i \leq \mu_i d_i$$

Then

We have that

$$\sum_{i=1}^{\infty} \mu_i k_i b_i \leq \sum_{i=1}^{\infty} \mu_i d_i < \infty$$

Then

We have that

$$\sum_{i=1}^{\infty} \mu_i k_i b_i \leq \sum_{i=1}^{\infty} \mu_i d_i < \infty$$

Then, we have that

$$\sum_{i=1}^{\infty} \mu_i k_i b_i < \infty, \ \sum_{i=1}^{\infty} \mu_i k_i = \infty$$

Finally

For any $\epsilon > 0$ there must be infinitely values i such that $b_i < \epsilon$

- Therefore given that $\lim_{n\to\infty} b_n = b$ then $b = 0$.

Robbins and Monro Theorem (Original)

If $\{\mu_n\}$ is of type $\frac{1}{n}$

- Given a family of conditional probabilities

$$\{H(y|\boldsymbol{w}) = Pr(Y(\boldsymbol{w}) \leq y|\boldsymbol{w})\}$$

Robbins and Monro Theorem (Original)

If $\{\mu_n\}$ is of type $\frac{1}{n}$

- Given a family of conditional probabilities

$$\{H(y|\boldsymbol{w}) = Pr(Y(\boldsymbol{w}) \leq y|\boldsymbol{w})\}$$

We have the following Expected Risk

$$M(\boldsymbol{w}) = \int_{-\infty}^{\infty} y dH(y|\boldsymbol{w})$$

Now

If we additionally have that

$$Pr\left(\left|Y\left(\boldsymbol{w}\right)\right| \leq C\right) = \int_{-C}^{C} dH\left(y|\boldsymbol{w}\right) = 1 \tag{3}$$

Then under the following constraints

For some $\delta > 0$

$$\begin{aligned} M(\boldsymbol{w}) \leq & \alpha - \delta \text{ for } \boldsymbol{w} < \boldsymbol{w}^* \\ M(\boldsymbol{w}) \geq & \alpha + \delta \text{ for } \boldsymbol{w} > \boldsymbol{w}^* \end{aligned} \tag{4}$$

Then under the following constraints

For some $\delta > 0$

$$\begin{aligned} M(\boldsymbol{w}) \leq & \alpha - \delta \text{ for } \boldsymbol{w} < \boldsymbol{w}^* \\ M(\boldsymbol{w}) \geq & \alpha + \delta \text{ for } \boldsymbol{w} > \boldsymbol{w}^* \end{aligned} \tag{4}$$

Or Else

$$\begin{aligned} M(\boldsymbol{w}) < & \alpha \text{ for } \boldsymbol{w} < \boldsymbol{w}^* \\ M(\boldsymbol{w}^*) = & \alpha \\ M(\boldsymbol{w}) > & \alpha \text{ for } \boldsymbol{w} > \boldsymbol{w}^* \end{aligned} \tag{5}$$

Next

Furthermore

$$M(\boldsymbol{w}) \text{ is strictly increasing if } |\boldsymbol{w} - \boldsymbol{w}^*| < \delta \tag{6}$$

Next

Furthermore

$$M(\boldsymbol{w}) \text{ is strictly increasing if } |\boldsymbol{w} - \boldsymbol{w}^*| < \delta \tag{6}$$

And

$$\inf_{|\boldsymbol{w}-\boldsymbol{w}^*| \geq \delta} |M(\boldsymbol{w}) - \alpha| > 0 \tag{7}$$

Next

Furthermore

$$M(\boldsymbol{w}) \text{ is strictly increasing if } |\boldsymbol{w} - \boldsymbol{w}^*| < \delta \tag{6}$$

And

$$\inf_{|\boldsymbol{w}-\boldsymbol{w}^*|\geq\delta} |M(\boldsymbol{w}) - \alpha| > 0 \tag{7}$$

And Let $\{\mu_i\}$ be a sequence of positive numbers such that

$$\sum_{n=1}^{\infty} \mu_n = \infty \text{ and } \sum_{n=1}^{\infty} \mu_n^2 < \infty \tag{8}$$

Then

Let x_1 an arbitrary number, then under the recursion

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu_n \left(\alpha - y_n\right)$$

- Where $y_n \sim P\left(y|\boldsymbol{w}_n\right)$

Then

Let x_1 an arbitrary number, then under the recursion

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu_n \left(\alpha - y_n\right)$$

- Where $y_n \sim P\left(y|\boldsymbol{w}_n\right)$

Theorem

- If (3) and (8), either (4) or (5,6,7) hold, then $\boldsymbol{w}_n$ converges stochastically to $\boldsymbol{w}^*$ given that $b = 0$.

Recap of Robbins-Monro Proposal

Given the following function

$$f(\boldsymbol{w}) = E\left[\phi(\boldsymbol{w}, \eta)\right], \ \boldsymbol{w} \in \mathbb{R}^{d+1}$$

Recap of Robbins-Monro Proposal

Given the following function

$$f(\boldsymbol{w}) = E[\phi(\boldsymbol{w}, \eta)], \ \boldsymbol{w} \in \mathbb{R}^{d+1}$$

Given a series of i.i.d. observations $x_0, x_1, \cdots$

- The following iterative procedure (Robbins-Monro Scheme)

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu_n \phi(\boldsymbol{w}_{n-1}, \boldsymbol{x}_n)$$

Robbins-Monro Proposal

Starting from an arbitrary initial condition, $\boldsymbol{w}_0$

- It converges to a root of $M(\boldsymbol{w}) = \alpha$

Robbins-Monro Proposal

Starting from an arbitrary initial condition, $\boldsymbol{w}_0$

- It converges to a root of $M(\boldsymbol{w}) = \alpha$

Under some general conditions about the step size

$$\sum_{i=0}^{\infty} \mu_i^2 < \infty$$

$$\sum_{i=0}^{\infty} \mu_i \to \infty$$

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Mean-Square Error [2]

Cost function for MSE

$$J(\boldsymbol{w}) = E\left[\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right]$$

- Also known as the expected risk or the expected loss.

Mean-Square Error [2]

Cost function for MSE

$$J(\boldsymbol{w}) = E\left[\mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right]$$

- Also known as the expected risk or the expected loss.

Then, our objective is the reduction of the Expected Risk!!!

- Thus, the simple thing to do is to derive the function and make such gradient equal to zero.

Therefore

We can get the Gradient of the Expected Cost Function

$$\nabla J(\boldsymbol{w}) = E\left[\nabla \mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right]$$

- where the expectation is w.r.t. the pair $(\boldsymbol{x}, y)$

Therefore

We can get the Gradient of the Expected Cost Function

$$\nabla J(\boldsymbol{w}) = E\left[\nabla \mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right]$$

- where the expectation is w.r.t. the pair $(\boldsymbol{x}, y)$

Therefore, everything depends on the form of the Loss function

$$\mathcal{L}_1(\boldsymbol{w}, \boldsymbol{x}, y) = \frac{1}{2}\left\|\boldsymbol{w}^T\boldsymbol{x} - y\right\|_2^2 \text{ (Least Squared Loss)}$$

$$\mathcal{L}_2(\boldsymbol{w}, \boldsymbol{x}, y) = \left[\frac{1}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{1-y}\left[\frac{\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}{1+\exp\left\{\boldsymbol{w}^T\boldsymbol{x}\right\}}\right]^{y} \text{ (Logistic Loss)}$$

$$\mathcal{L}_3(\boldsymbol{w}, \boldsymbol{x}, y) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log\left(y_{nk}^{(l)}\right) \text{ (Cross-Entropy Loss)}$$

Therefore

We simply take $\alpha = 0$ then

$$\nabla J(\boldsymbol{w}) = E\left[\nabla \mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right] = 0$$

Therefore

We simply take $\alpha = 0$ then

$$\nabla J(\boldsymbol{w}) = E\left[\nabla \mathcal{L}(\boldsymbol{w}, \boldsymbol{x}, y)\right] = 0$$

Then, we apply the Robbins-Monroe Schema to the function

$$f(\boldsymbol{w}) = \nabla J(\boldsymbol{w}) = 0$$

Then

Given the sequence of observations $\{(\boldsymbol{x}_i, y_i)\}_{i=1,2,\ldots}$ and values $\{\mu_i\}_{i=1,2,\ldots}$

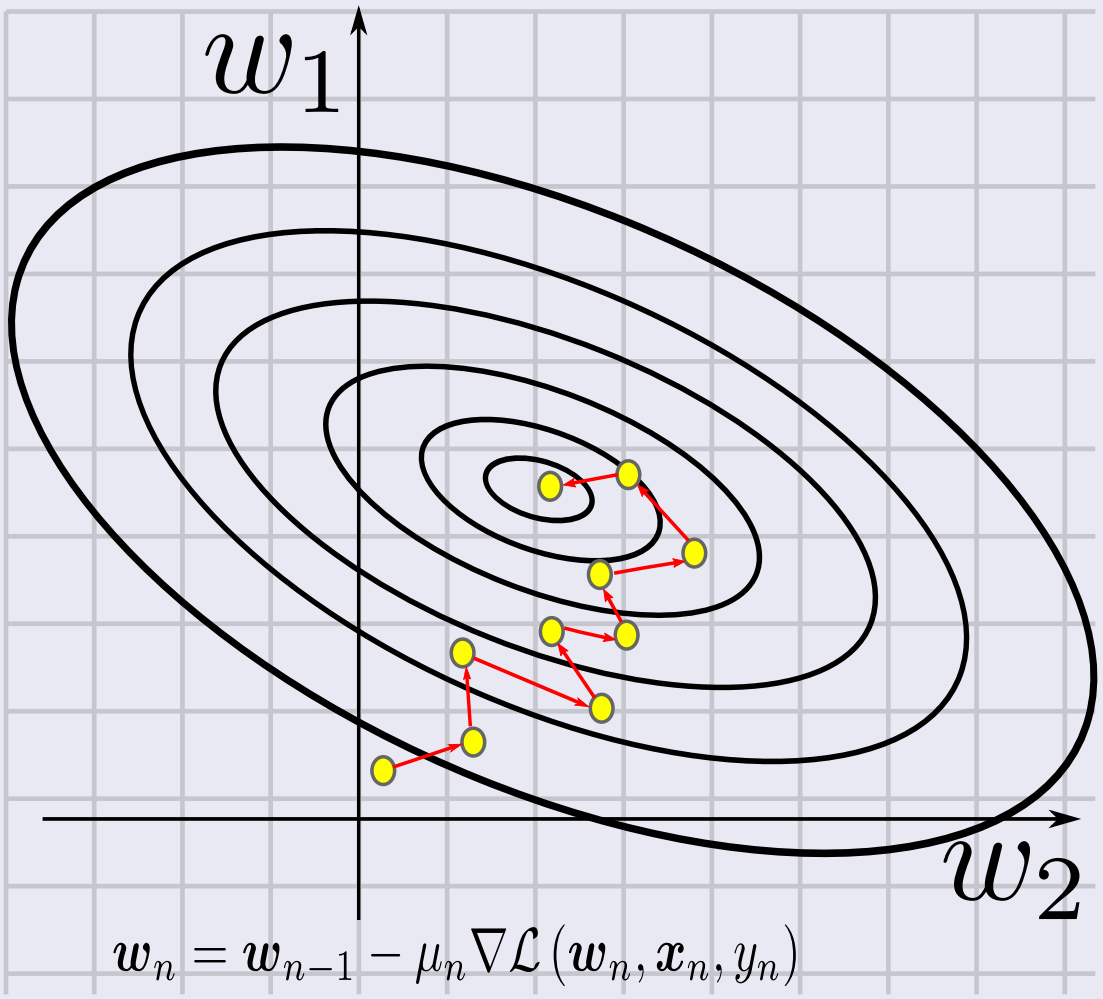
- We have that the iterative procedure becomes:

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu_n \nabla \mathcal{L}\left(\boldsymbol{w}_n, \boldsymbol{x}_n, y_n\right)$$

  - The Well known Vanilla Stochastic Gradient Descent (SGD)

Geometrically

We have the following



Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Therefore

However, although the theorem is important

- it is not by itself enough.

Therefore

However, although the theorem is important

- it is not by itself enough.

One has to know something more concerning

- The rate of convergence of such a scheme.

Therefore

However, although the theorem is important

- it is not by itself enough.

One has to know something more concerning

- The rate of convergence of such a scheme.

It has been shown that

$$\mu_n = O\left(\frac{1}{n}\right)$$

Additionally

Assuming that iterations have brought the estimate close to the optimal value

$$E\left(\boldsymbol{w}_n\right) = \boldsymbol{w}^* + \frac{1}{n}\boldsymbol{c}$$

Additionally

Assuming that iterations have brought the estimate close to the optimal value

$$E\left(\boldsymbol{w}_n\right) = \boldsymbol{w}^* + \frac{1}{n}\boldsymbol{c}$$

And

$$Cov\left(\boldsymbol{w}_n\right) = \frac{1}{n}V + O\left(\frac{1}{n^2}\right)$$

- Where $\boldsymbol{c}$ and V are constants that depend on the form of the expected risk.

Meaning

Therefore

- These formulas indicate that the parameter vector estimate fluctuates around the optimal value.

Meaning

Therefore

- These formulas indicate that the parameter vector estimate fluctuates around the optimal value.

However

- Low complexity requirements makes this algorithmic family to be the one that is selected in a number of practical applications.
  - Given the problem with Batch Gradient Descent (BGD)

Outline

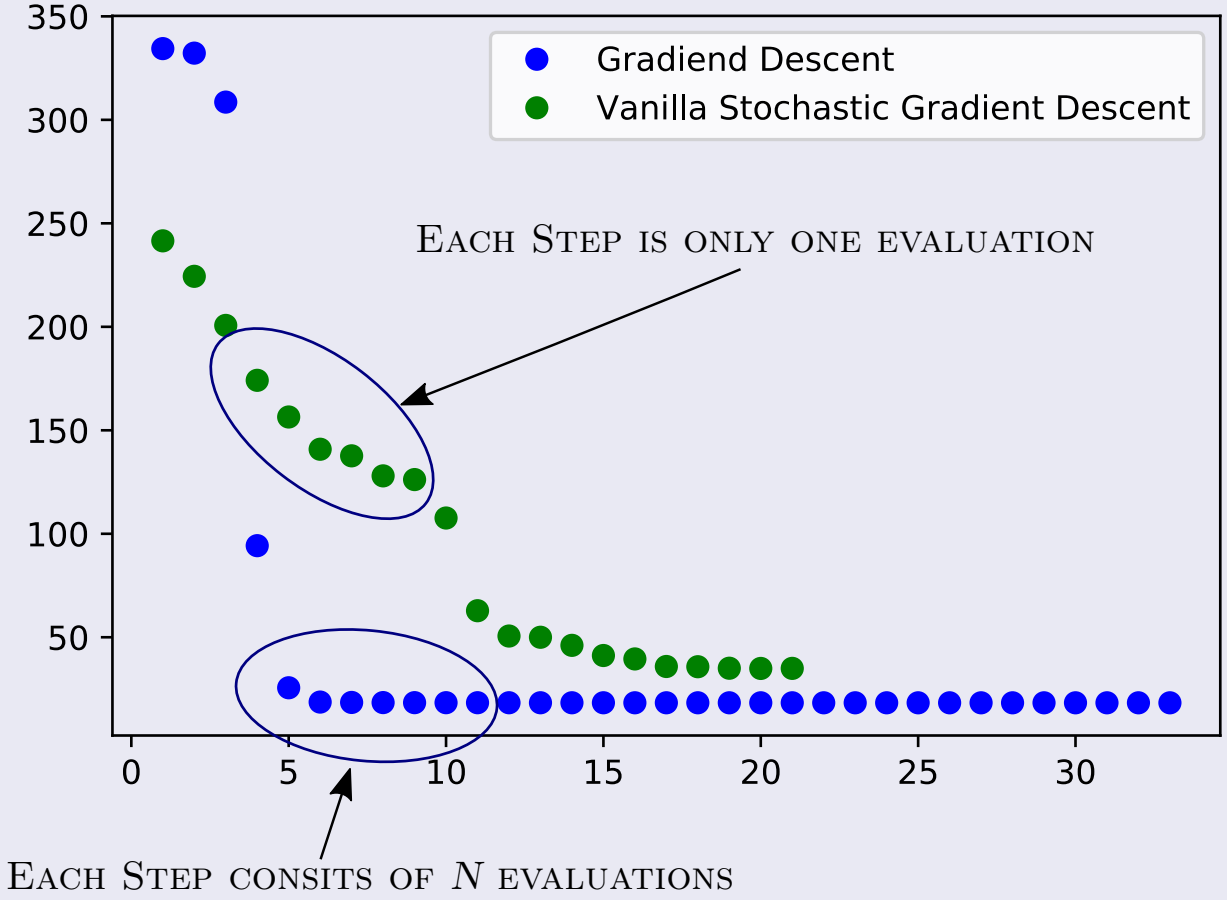
1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

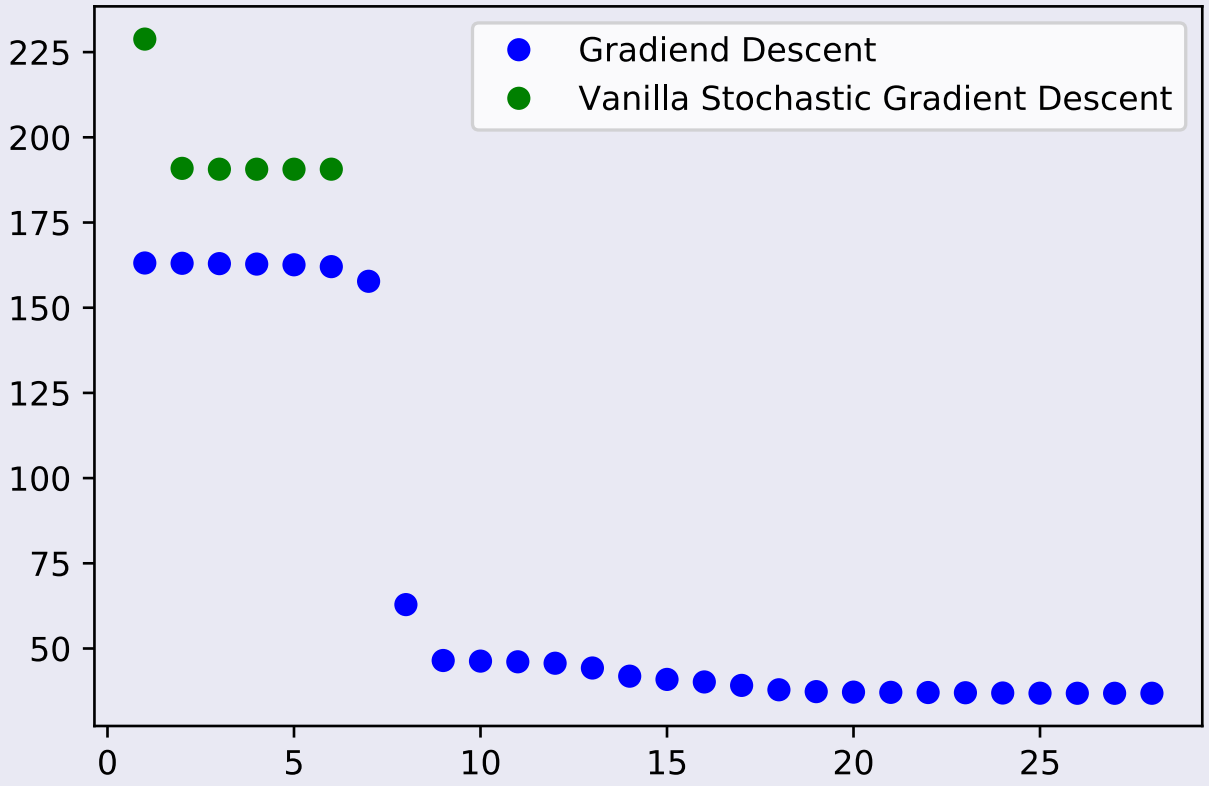
Example of SGD for, $\frac{1}{2}\sum_{i=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}-\boldsymbol{y}\right)^2$

We can see how from the Vanilla SGD improves over the Batch GD with respect to Speed of Evaluation



Problems

However, we need to improve such Vanilla Stochastic Gradient Descent



Outline

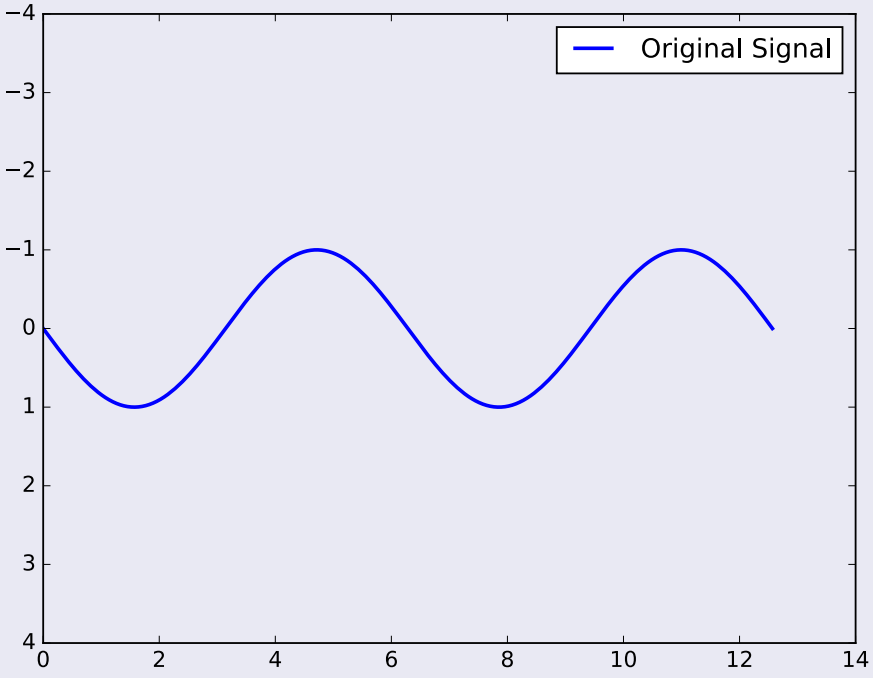
1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

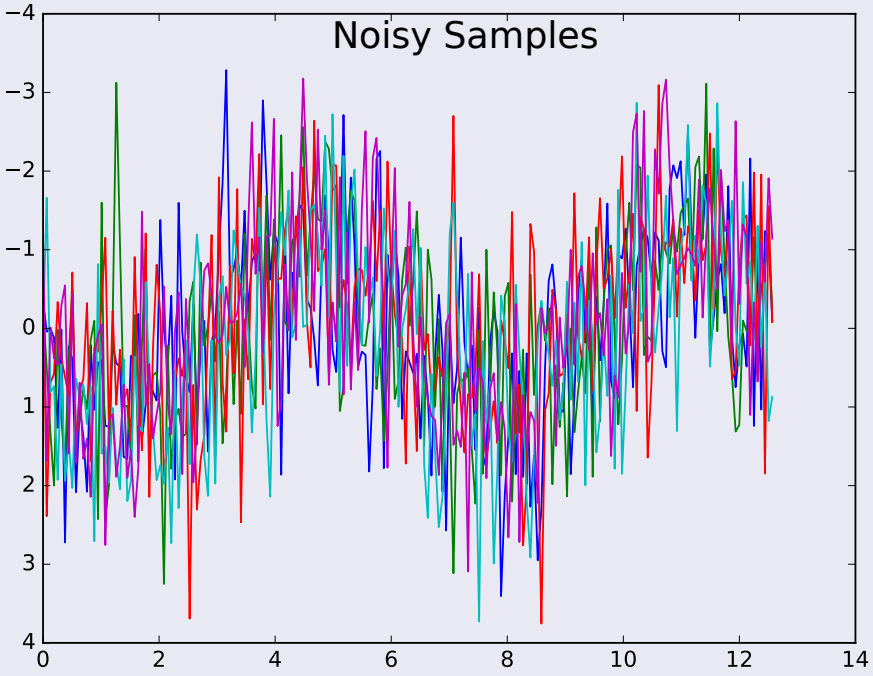
2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Do you Remember?

Imagine the following signal from $\sin(\theta)$





What if we know the noise?

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0, 1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

What if we know the noise?

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0,1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

We can use our knowledge of probability to remove such noise

$$E\left[\widehat{\boldsymbol{x}}_i\right] = E\left[\boldsymbol{x}_i + \epsilon\right] = E\left[\boldsymbol{x}_i\right] + E\left[\epsilon\right]$$

What if we know the noise?

Given a series of observed samples $\{\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, ..., \widehat{\boldsymbol{x}}_N\}$ with noise $\epsilon \sim N(0,1)$

We could use our knowledge on the noise, for example additive:

$$\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \epsilon$$

We can use our knowledge of probability to remove such noise

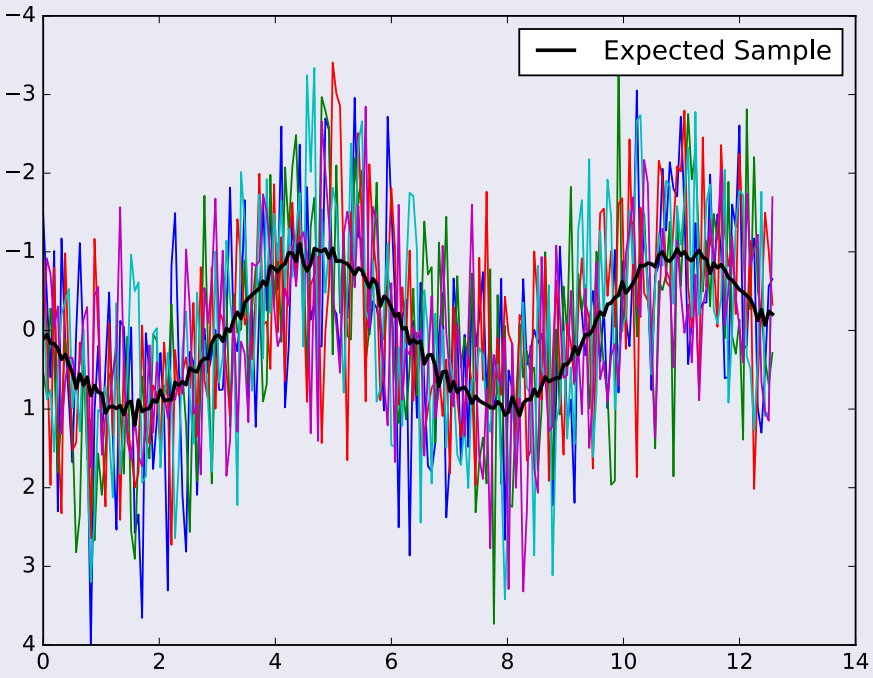
$$E\left[\widehat{\boldsymbol{x}}_i\right] = E\left[\boldsymbol{x}_i + \epsilon\right] = E\left[\boldsymbol{x}_i\right] + E\left[\epsilon\right]$$

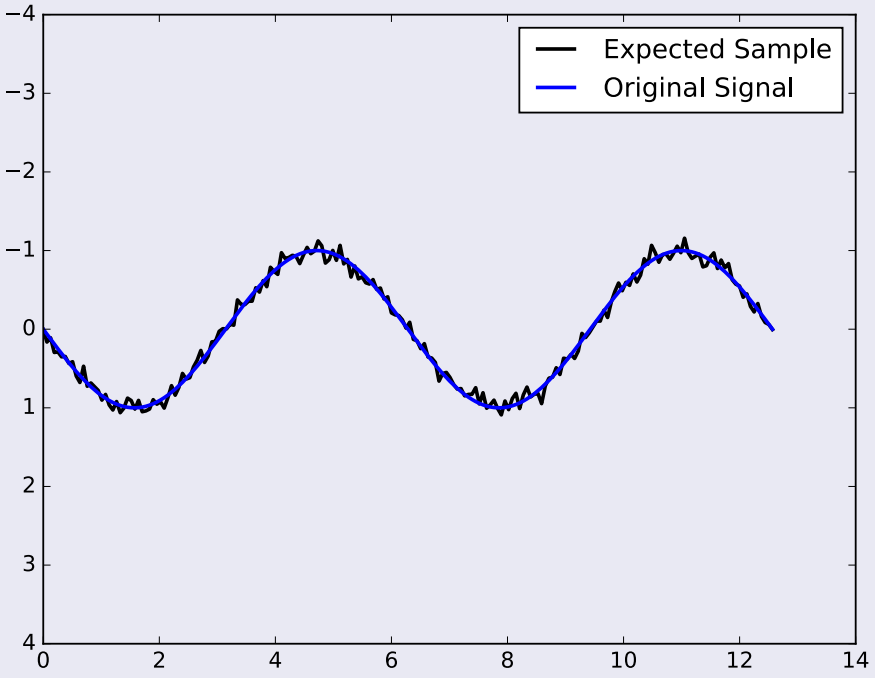
Then, because $E[\epsilon] = 0$

$$E\left[\boldsymbol{x}_i\right] = E\left[\widehat{\boldsymbol{x}}_i\right] \approx \frac{1}{N}\sum_{i=1}^{N} \widehat{\boldsymbol{x}}_i$$

In our example

We have a nice result





Thus

Using a similar idea, you could use an average [9]

$$\nabla J\left(\boldsymbol{w}_{k-1}|\boldsymbol{x}_{i:i+m}, y_{i:i+m}\right) = \ldots$$
$$\frac{1}{m}\sum_{i=1}^{m}\nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right)$$

Thus

Using a similar idea, you could use an average [9]

$$\nabla J\left(\boldsymbol{w}_{k-1}|\boldsymbol{x}_{i:i+m}, y_{i:i+m}\right) = \ldots$$
$$\frac{1}{m}\sum_{i=1}^{m}\nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right)$$

This allows to reduce the variance of the original Stochastic Gradient

- It reduces the variance of the parameter updates, which can lead to more stable convergence.

Thus

Using a similar idea, you could use an average [9]

$$\nabla J\left(\boldsymbol{w}_{k-1}|\boldsymbol{x}_{i:i+m}, y_{i:i+m}\right) = \ldots$$
$$\frac{1}{m}\sum_{i=1}^{m}\nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right)$$

This allows to reduce the variance of the original Stochastic Gradient

- It reduces the variance of the parameter updates, which can lead to more stable convergence.
- It can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

There are other more efficient options

We can update the $\boldsymbol{w}(k)$

- By Batches per epoch...

There are other more efficient options

We can update the $\boldsymbol{w}(k)$

- By Batches per epoch...

Therefore

1. for i in batch k

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \alpha \nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right)$$

Mini-batch gradient descent finally takes the best of both worlds

Min-Batch(X)

Input:

- Initialize $\boldsymbol{w}_0$, Set number of epochs, L, Set learning rate α

1. for $k = 1$ to L:
2. Randomly pick a mini batch of size m.
3. for $i = 1$ to m do:
4. Evaluate $g(k) = \nabla J(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i)$
5. $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \alpha g(k)$

Notes

Remark, for $\alpha = \frac{1}{m}$, the method is equivalent to average sample way

$$
\begin{aligned}
\boldsymbol{w}_k =& \boldsymbol{w}_{k-1} - \alpha \nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right) - \ldots \\
& \alpha \nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_{i+1}, y_{i+1}\right) - \ldots \\
& \alpha \nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_{i+m}, y_{i+m}\right) \\
=& \boldsymbol{w}_{k-1} - \frac{1}{m} \sum_{i=1}^{m} \nabla J\left(\boldsymbol{w}_{k-1}, \boldsymbol{x}_i, y_i\right)
\end{aligned}
$$

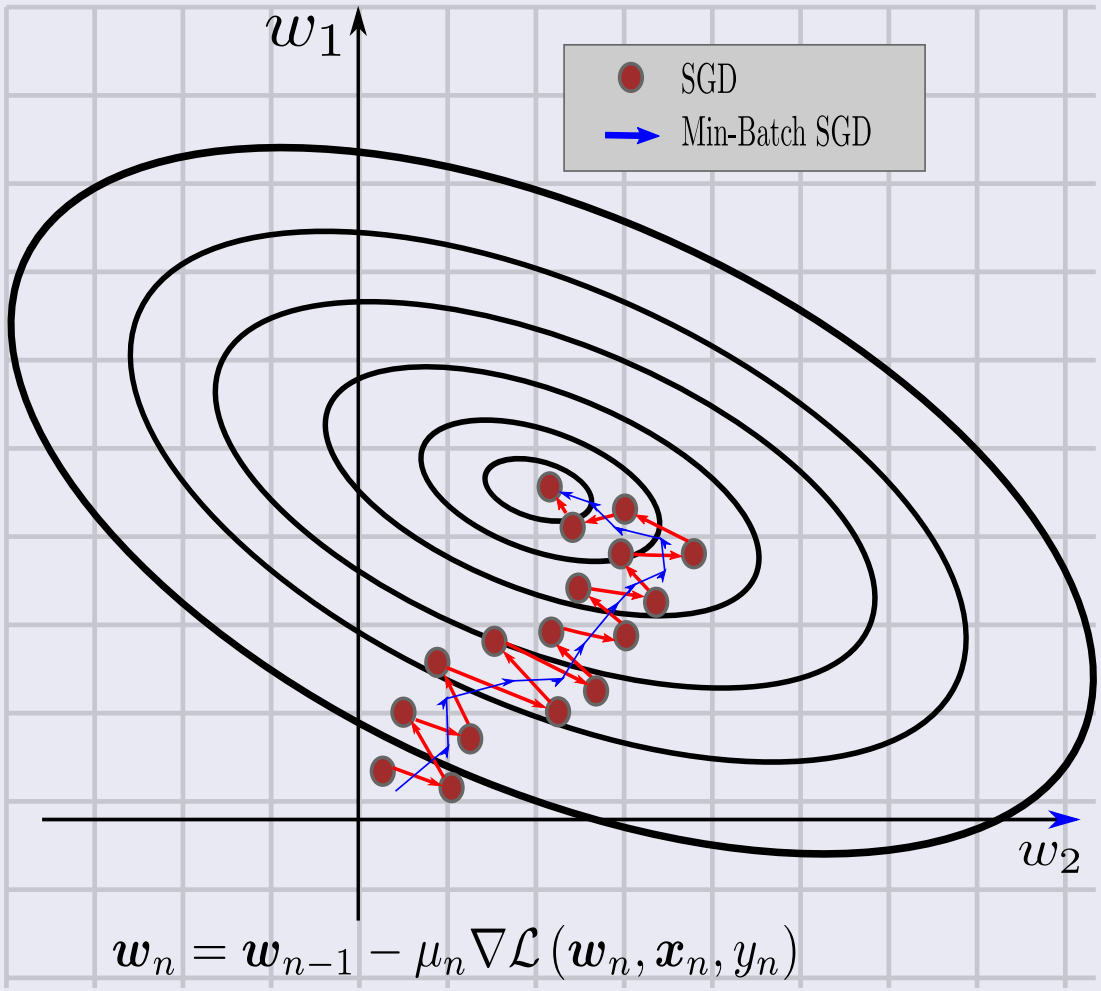
Notes

We have the following

- Common mini-batch sizes range between 50 and 256, but can vary for different applications.
- Mini-batch gradient descent is typically the algorithm of choice when training a neural network.

A Small Intuition

We have smoother version of the Stochastic Gradient Descent



$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} - \mu_n \nabla \mathcal{L}\left(\boldsymbol{w}_n, \boldsymbol{x}_n, y_n\right)$$

Drawbacks

Choosing a proper learning rate can be difficult

- A learning rate that is too small leads to painfully slow convergence,
- Too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge.

Drawbacks

Choosing a proper learning rate can be difficult

- A learning rate that is too small leads to painfully slow convergence,
- Too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge.

Learning Rate Schedules

- To adjust the learning rate during training by e.g. annealing
- These schedules and thresholds, however, have to be defined in advance not on-line

Drawbacks

Choosing a proper learning rate can be difficult

- A learning rate that is too small leads to painfully slow convergence,
- Too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge.

Learning Rate Schedules

- To adjust the learning rate during training by e.g. annealing
- These schedules and thresholds, however, have to be defined in advance not on-line

Another key challenge of minimizing highly non-convex error functions

- For example, neural networks, it is avoiding getting trapped in their numerous suboptimal local minima.

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

The MSE Linear Estimation, the Normal Equations

It was proved in slide set 2

- The optimal **Mean-Square Error estimate** of y given the value $X = \boldsymbol{x}$ is

$$E\left[y|\boldsymbol{x}\right] = \widehat{y}$$

  - In general, a nonlinear function.

The MSE Linear Estimation, the Normal Equations

It was proved in slide set 2

- The optimal **Mean-Square Error estimate** of y given the value $X = \boldsymbol{x}$ is

$$E\left[y|\boldsymbol{x}\right] = \widehat{y}$$

  - In general, a nonlinear function.

For Linear Estimators, in $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ joint distributed random variables of zero mean values

- Our goal is to obtain an estimate of $\boldsymbol{w} \in \mathbb{R}^d$ (Our Unknown θ) in the linear estimator model

$$\widehat{y} = \boldsymbol{w}^T \boldsymbol{x}$$

Thus, using MSE as the Cost Equation

Cost Function

$$J(\boldsymbol{w}) = E\left[(y - \widehat{y})^2\right]$$

Thus, using MSE as the Cost Equation

Cost Function

$$J(\boldsymbol{w}) = E\left[(y - \widehat{y})^2\right]$$

Thus, we are looking for an estimator that minimize the variance of the error

$$\epsilon = y - \widehat{y}$$

Thus, using MSE as the Cost Equation

Cost Function

$$J(\boldsymbol{w}) = E\left[(y - \widehat{y})^2\right]$$

Thus, we are looking for an estimator that minimize the variance of the error

$$\epsilon = y - \widehat{y}$$

We want to Minimize the cost function $J(\boldsymbol{w})$ by finding an optimal $\boldsymbol{w}^*$

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$$

Then, we can simply use $\nabla J(\boldsymbol{w}) = 0$

We have

$$\begin{aligned}
\nabla J(\boldsymbol{w}) =& \nabla E\left[\left(y - \boldsymbol{w}^T\boldsymbol{x}\right)^2\right] \\
=& \nabla E\left[\left(y - \boldsymbol{w}^T\boldsymbol{x}\right)\left(y - \boldsymbol{w}^T\boldsymbol{x}\right)\right] \\
=& \nabla\left\{E\left[y^2\right] - 2\boldsymbol{w}^T E\left[\boldsymbol{x}y\right] + \boldsymbol{w}^T E\left[\boldsymbol{x}\boldsymbol{x}^T\right]\boldsymbol{w}^T\right\} \\
=& -2\boldsymbol{p} + 2\Sigma_x\boldsymbol{w} = 0
\end{aligned}$$

Where, we have

$$\begin{aligned}
\boldsymbol{p} =& \left[E\left[yx_1\right], E\left[yx_2\right], ..., E\left[yx_d\right]\right] = E\left[\boldsymbol{x}y\right] \\
\Sigma_x =& E\left[\boldsymbol{x}\boldsymbol{x}^T\right]
\end{aligned}$$

This generates what is know as

Then, we get the Normal Equations

$$\Sigma_x \boldsymbol{w}^* = \boldsymbol{p}$$

The Stochastic Gradient Descent

Imagine the follow

- We assume that the covariance matrix and the cross-correlation vector are unknown.

The Stochastic Gradient Descent

Imagine the follow

- We assume that the covariance matrix and the cross-correlation vector are unknown.

We have that for a single sample

$$\mathcal{L}\left(\boldsymbol{w}, y, \boldsymbol{x}\right) = \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{x} - y\right)^2$$

Therefore

We know

- The solution corresponds to the root of the gradient of the cost function:

$$\Sigma_x \boldsymbol{w} - \boldsymbol{p} = E\left[\boldsymbol{x}\left(\boldsymbol{x}^T\boldsymbol{w} - y\right)\right] = 0$$

Therefore

We know

- The solution corresponds to the root of the gradient of the cost function:

$$\Sigma_x \boldsymbol{w} - \boldsymbol{p} = E\left[\boldsymbol{x}\left(\boldsymbol{x}^T\boldsymbol{w} - y\right)\right] = 0$$

We have

$$\nabla J(\boldsymbol{w}) = \Sigma_x \boldsymbol{w} - \boldsymbol{p} = E\left[\boldsymbol{x}\left(\boldsymbol{x}^T\boldsymbol{w} - y\right)\right] = 0$$

Therefore

We know

- The solution corresponds to the root of the gradient of the cost function:

$$\Sigma_x \boldsymbol{w} - \boldsymbol{p} = E\left[\boldsymbol{x}\left(\boldsymbol{x}^T\boldsymbol{w} - y\right)\right] = 0$$

We have

$$\nabla J(\boldsymbol{w}) = \Sigma_x \boldsymbol{w} - \boldsymbol{p} = E\left[\boldsymbol{x}\left(\boldsymbol{x}^T\boldsymbol{w} - y\right)\right] = 0$$

Then

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu_n \boldsymbol{x}_n \left(\boldsymbol{x}_n^T \boldsymbol{w}_{n-1} - y_n\right)$$

The Least-Mean Squares Adaptive Algorithm

The stochastic gradient algorithm for MSE

- It converges to the optimal mean-square error solution provided that μ_n satisfies the two convergence conditions.

The Least-Mean Squares Adaptive Algorithm

The stochastic gradient algorithm for MSE

- It converges to the optimal mean-square error solution provided that μ_n satisfies the two convergence conditions.

Once the algorithm has converged

- It "locks" at the obtained solution.

The Least-Mean Squares Adaptive Algorithm

The stochastic gradient algorithm for MSE

- It converges to the optimal mean-square error solution provided that μ_n satisfies the two convergence conditions.

Once the algorithm has converged

- It "locks" at the obtained solution.

In a case where the statistics of the involved process changes

- The algorithm cannot track the changes.

Therefore

if such changes occur, the error term

$$e_n = y_n - \boldsymbol{x}_n^T \boldsymbol{w}_{n-1}$$

- It will get larger values.

Therefore

if such changes occur, the error term

$$e_n = y_n - \boldsymbol{x}_n^T \boldsymbol{w}_{n-1}$$

- It will get larger values.

However

- Because μ_n is very small, the increased value of the error will not lead to corresponding changes of the estimate at time n.

Solution

This can be overcome if one sets the value of μ_n

- To a preselected fixed value, μ.

Solution

This can be overcome if one sets the value of μ_n

- To a preselected fixed value, μ.

The celebrated Least-Mean-Squares Algorithms

- Algorithm LMS
  1. $\boldsymbol{w}_{-1} = 0 \in \mathbb{R}^d$
  2. Select a value μ
  3. **for** $n = 0, 1, \ldots$ **do**
  4. $\quad e_n = y_n - \boldsymbol{x}_n^T \boldsymbol{w}_{n-1}$
  5. $\quad \boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu e_n \boldsymbol{x}_n$

Complexity

Something Notable

- The complexity of the algorithm amounts to $2d$ multiplications/additions (MADs) per time update.

Complexity

Something Notable

- The complexity of the algorithm amounts to $2d$ multiplications/additions (MADs) per time update.

However

- As the algorithm converges close the solution

Complexity

Something Notable

- The complexity of the algorithm amounts to $2d$ multiplications/additions (MADs) per time update.

However

- As the algorithm converges close the solution

Thus

- The error term is expected to take small values making the updates to remain close the solution

Important

Given that μ has a constant value

- The algorithm has now the "agility" to update the estimates
  - In an attempt to "push" the error to lower values.

Important

Given that μ has a constant value

- The algorithm has now the "agility" to update the estimates
  - In an attempt to "push" the error to lower values.

Something Notable

- This small variation of the iterative scheme has important implications.

Important

Given that μ has a constant value

- The algorithm has now the "agility" to update the estimates
  - In an attempt to "push" the error to lower values.

Something Notable

- This small variation of the iterative scheme has important implications.

No More a Robbins-Monro stochastic family

- The resulting algorithm is no more a member of the Robbins-Monro stochastic approximation family.

Outline

1. Introduction
- Review Gradient Descent
- The Problems of Gradient Descent with Large Data Sets
- Convergence of gradient descent with fixed step size
- Convergence Rate
  - Convex Functions
  - Back to the Main Problem
- Accelerating the Gradient Descent
- Even with such Speeds

2. Accelerating Gradient Descent
- Robbins-Monro Theorem
- Robbins-Monro Scheme for Minimum-Square Error
- Convergence

3. Improving and Measuring Stochastic Gradient Descent
- Example of SGD Vs BGD
- Using The Expected Value, The Mini-Batch
- The Least-Mean Squares Adaptive Algorithm
- Conclusions

Conclusions

In Machine Learning

- We need to have the best speedups to handle the problem dealing with Big Data...

Conclusions

In Machine Learning

- We need to have the best speedups to handle the problem dealing with Big Data...

As we get more and more algorithms

- It is clear that optimization for Big Data is one of the hottest trends in Machine Learning

Conclusions

In Machine Learning

- We need to have the best speedups to handle the problem dealing with Big Data...

As we get more and more algorithms

- It is clear that optimization for Big Data is one of the hottest trends in Machine Learning

Take a look to

- Leon Bottou, Frank E. Curtis, Jorge Nocedal: **Optimization Methods for Large-Scale Machine Learning**. SIAM Review 60(2): 223-311 (2018)

C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*.
Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*.
Academic Press, 1st ed., 2015.

W. Rudin, *Real and Complex Analysis, 3rd Ed.*
New York, NY, USA: McGraw-Hill, Inc., 1987.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*.
The MIT Press, 3rd ed., 2009.

L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.

Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*.
Springer Publishing Company, Incorporated, 1 ed., 2014.

S. Bubeck, "Convex optimization: Algorithms and complexity," *arXiv preprint arXiv:1405.4980*, 2014.

H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 09 1951.

M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), pp. 661–670, ACM, 2014.