Gradient Descent and Stochastic Gradient Descent

Jean-Michel Marin

University of Montpellier
Faculty of Sciences

HAX912X - 2025/2026

Context

The goal of the gradient descent method is to find a minimum of a multivariate function as efficiently as possible

The idea is simple : the negative gradient vector points toward lower values of the function, so one just needs to take a step in that direction and repeat the process

However, to accelerate convergence, additional parameters can be introduced (whose proper tuning often requires significant engineering effort)

Context

Imagine a drop of water at the top of a hill, the drop flows downward following the steepest slope and stops when it reaches a low point

This is exactly what gradient descent does : starting from a point on a surface, it computes the direction of the steepest slope using the gradient, takes a small step in that direction, and repeats the process until it reaches a local minimum

Gradient Descent Algorithm

Let us formalize the idea to highlight both the general principle and the technical difficulties that may arise

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function of several variables $\mathbf{x} = (x_1, \ldots, x_n)$ and assume we can compute its gradient $\nabla f(\mathbf{x})$

Given

- An initial point $\mathbf{x}^{(0)} \in \mathbb{R}^n$
- An error tolerance $\varepsilon > 0$

Gradient Descent Algorithm

Iteration
We compute a sequence of points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ recursively as follows (suppose we have already obtained the point $\mathbf{x}^{(k)}$)

1. Compute the gradient $\nabla f(\mathbf{x}^{(k)})$
2. Choose a step size δ_k and compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \delta_k \nabla f(\mathbf{x}^{(k)})$$

The next point $\mathbf{x}^{(k+1)}$ is obtained by moving from $\mathbf{x}^{(k)}$ in the direction opposite to the gradient, scaled by the step size δ_k

Stopping criterion $\nabla f(\mathbf{x}^{(k+1)}) \leqslant \varepsilon$

Gradient Descent Algorithm

Obviously, the closer the initial point is to a local minimum, the faster the algorithm will converge

However, since we do not know where this minimum lies (that is what we are trying to find), a simple strategy is to choose $\mathbf{x}^{(0)}$ at random

The choice of the step size δ_k is crucial ; one can always choose δ_k small enough to ensure that

$$f(\mathbf{x}^{(k+1)}) \leqslant f(\mathbf{x}^{(k)})$$

since the function decreases in the direction of $-\nabla f(\mathbf{x}^{(k)})$

Gradient Descent Algorithm

One may fix in advance a common step size δ for all iterations, for instance $\delta = 0.01$

The stopping criterion ensures that at $\mathbf{x}^{(k)}$, the gradient is very small

However, this does not guarantee that the point is close to a local minimum (and even less to a global one)

Recall that at a local minimum the gradient vanishes, but the converse is not necessarily true : a point with a zero gradient may also be a saddle point or even a local maximum

Gradient Descent Algorithm

In practice, instead of defining an error threshold ε, it is often more convenient to fix in advance a maximum number of iterations

It is important to compute $\nabla f(\mathbf{x})$ efficiently

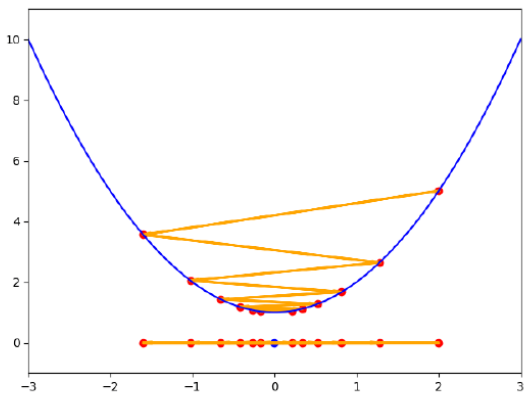
One could, of course, approximate each partial derivative using finite differences (however, for both speed and accuracy, it is preferable to use analytical expressions whenever possible)

Gradient Descent Algorithm
Examples

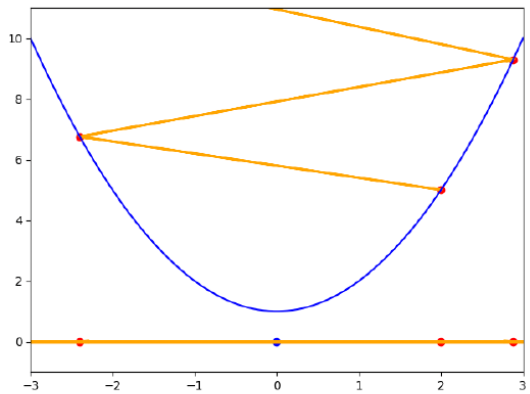
$$f(x) = x^2 + 1$$

$$x^{(0)} = 2$$

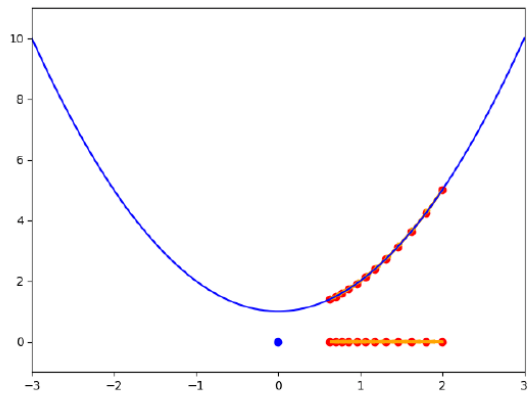
$\delta = 0.9$



$\delta = 1.1$



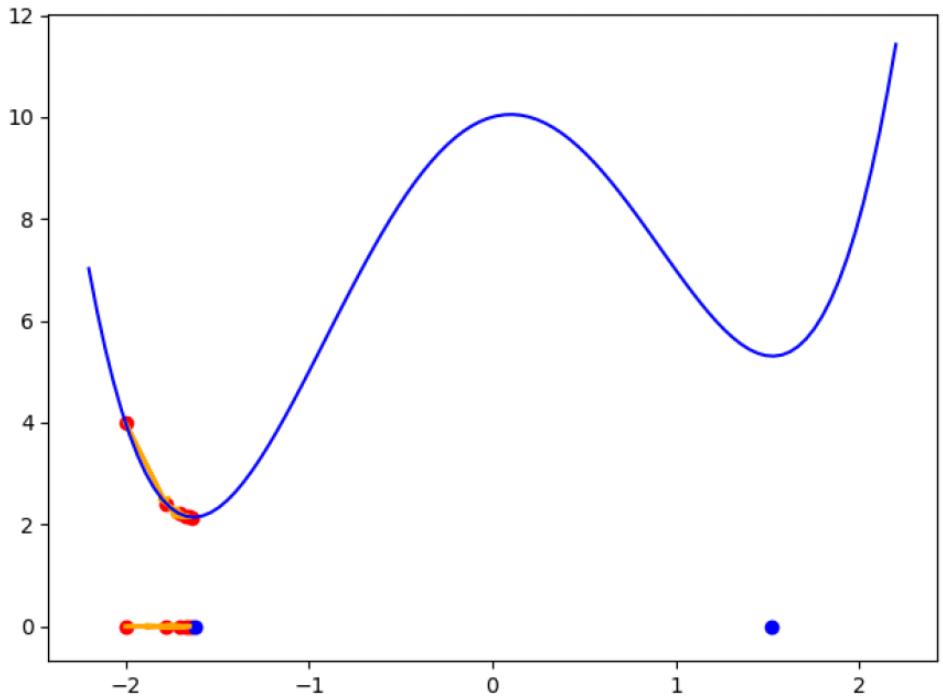
$\delta = 0.05$



Gradient Descent Algorithm
Examples

$$f(x) = x^4 - 5x^2 + x + 10$$

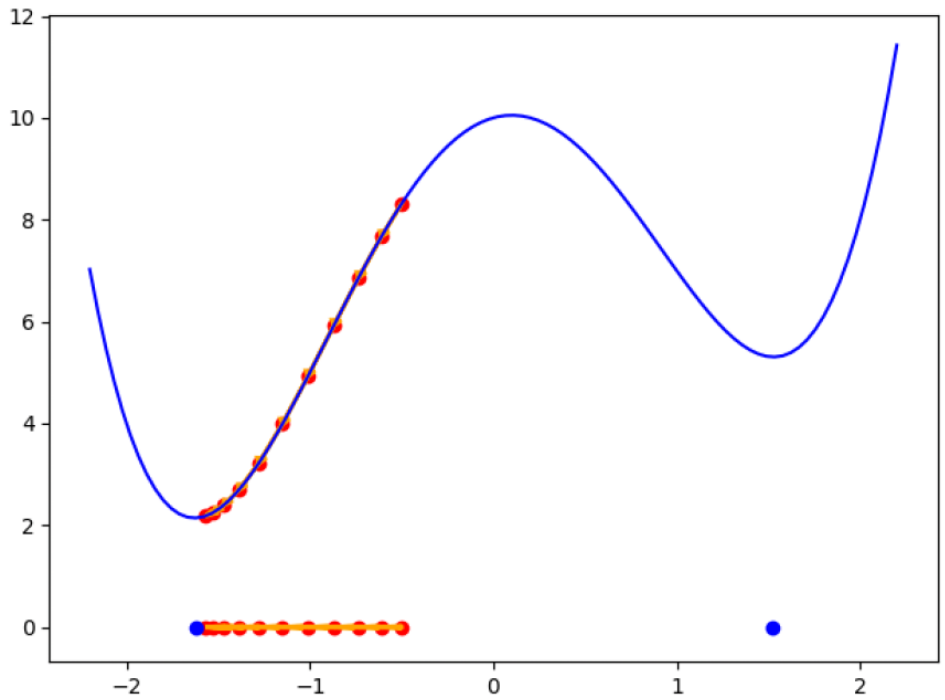
$$\delta = 0.02, \quad x^{(0)} = -2$$



Gradient Descent Algorithm
Examples

$$f(x) = x^4 - 5x^2 + x + 10$$

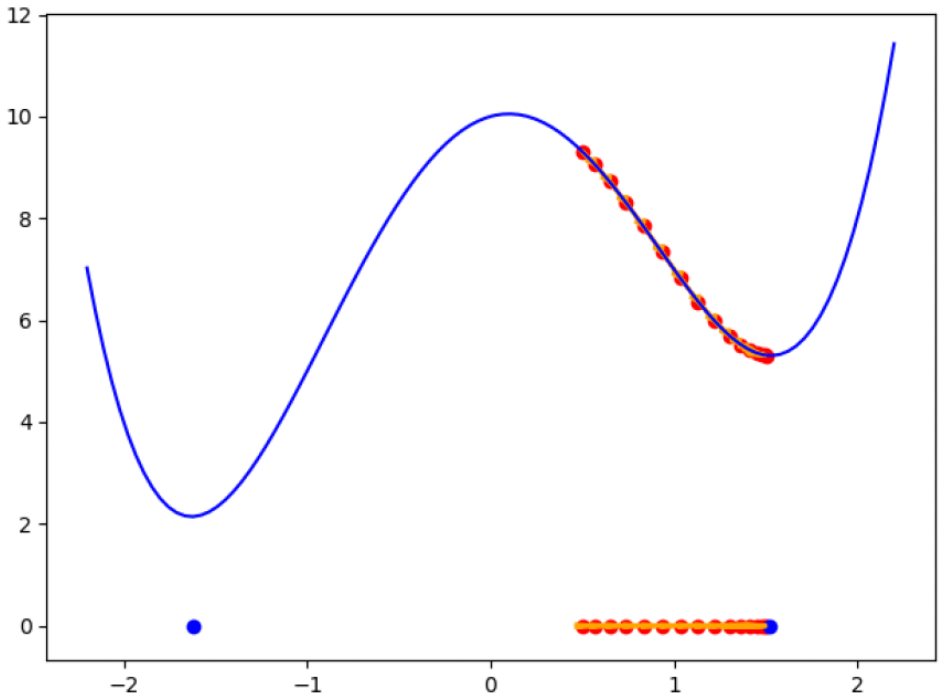
$$\delta = 0.02, \quad x^{(0)} = -0.5$$



Gradient Descent Algorithm
Examples

$$f(x) = x^4 - 5x^2 + x + 10$$

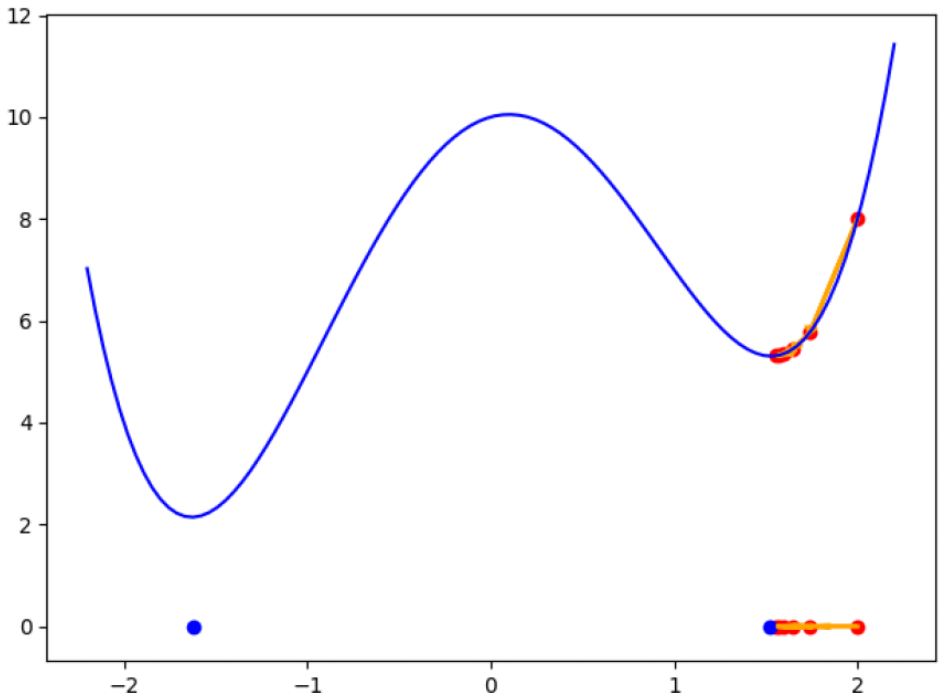
$$\delta = 0.02, \quad x^{(0)} = 0.5$$



Gradient Descent Algorithm
Examples

$$f(x) = x^4 - 5x^2 + x + 10$$

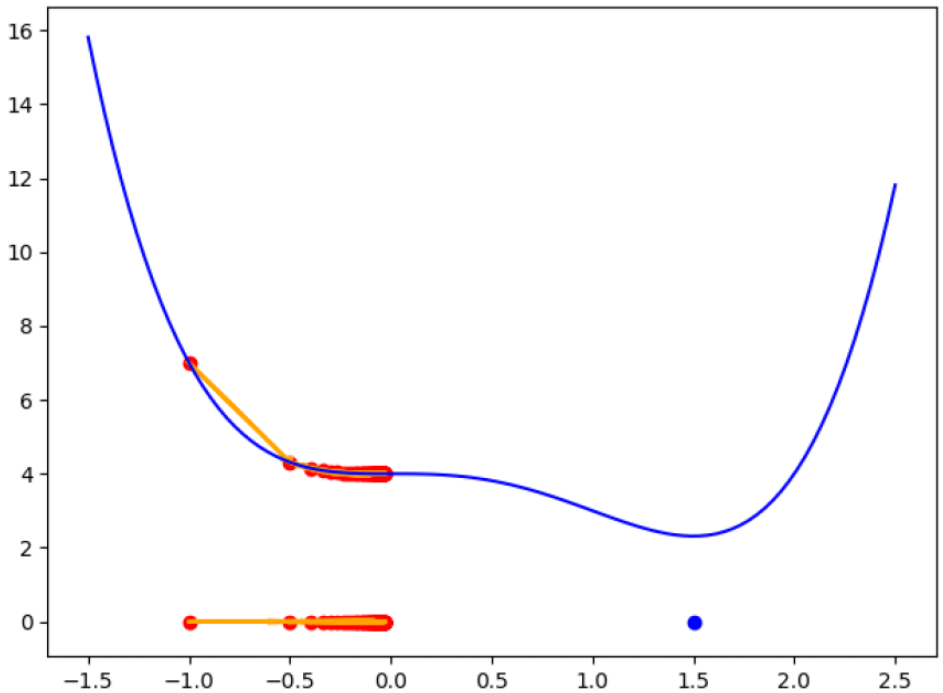
$$\delta = 0.02, \quad x^{(0)} = 2$$



Gradient Descent Algorithm
Examples

$$f(x) = x^4 - 2x^3 + x + 4$$

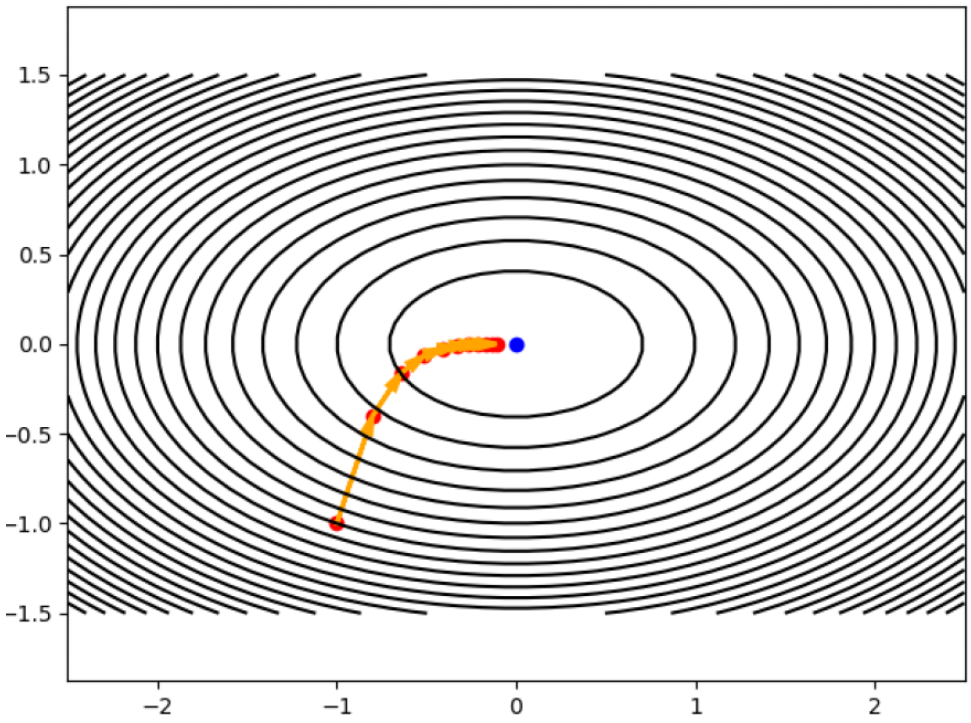
$$\delta = 0.05, \quad x^{(0)} = -1$$



Gradient Descent Algorithm
Examples

$$f(x_1, x_2) = x_1^2 + 3x_2^2$$

$$\delta = 0.1, \quad \mathbf{x}^{(0)} = (-1, -1)$$



Gradient Descent Algorithm
Varying the step size

We now focus on the choice of the step size δ, also called the *learning rate*

As we approach a minimum, the vector $\delta \nabla f(\mathbf{x}^{(k)})$ tends to zero even if δ remains constant

However, δ must be chosen neither too large nor too small : if δ is too large, the points $\mathbf{x}^{(k)}$ will oscillate around the minimum, whereas if δ is too small, convergence toward the minimum will be extremely slow

A common solution is to let δ vary during the iterations : starting with a relatively large δ_k for the first steps, and gradually decreasing it thereafter

Gradient Descent Algorithm
Varying the step size

Below are several possible update rules for the step size, where δ_0 is the initial step

Linear decay

$$\delta_k = \frac{\delta_0}{k+1}$$

Quadratic decay

$$\delta_k = \frac{\delta_0}{(k+1)^2}$$

Exponential decay

$$\delta_k = \delta_0 e^{-\beta k}$$

where β is a positive constant

Stochastic Gradient Descent (SGD)

In machine learning applications, the function $f(\theta)$ represents a *loss function* that depends on a large dataset

$$f(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(\theta)$$

where $\ell_i(\theta) = \ell(y_i, f_\theta(x_i))$ measures the error for observation i and θ the parameters of the model

Computing the full gradient $\nabla f(\theta)$ can therefore be very expensive when N is large

Stochastic Gradient Descent (SGD)

Idea of Stochastic Gradient Descent (SGD)

Instead of using the gradient over the entire dataset, we approximate it using only a single randomly selected data point (or a small subset, called a *mini-batch*) at each iteration

For iteration k :

1. Select a random index (or batch) i_k
2. Compute the stochastic gradient $\nabla \ell_{i_k}(\theta^{(k)})$
3. Update the parameters

$$\theta^{(k+1)} = \theta^{(k)} - \delta_k \nabla \ell_{i_k}(\theta^{(k)})$$

Stochastic Gradient Descent (SGD)
Remarks

- Each update uses only partial information about f, which introduces noise but greatly reduces computational cost
- The noise can help the algorithm escape shallow local minima or saddle points
- To ensure convergence, the step size δ_k is often decreased over time
- In practice, *mini-batch* SGD (using small batches of data) offers a good compromise between speed and stability

Stochastic Gradient Descent (SGD)
Gradient Descent Variants

Method	**Gradient Computation**
GD	Uses the full dataset (N samples)
SGD	Uses one random sample per iteration
Mini-Batch SGD	Uses a small subset (batch of b samples)

Stochastic Gradient Descent (SGD)
Gradient Descent Variants

Method	**Computation Cost**
GD	High (exact gradient)
SGD	Very low (fast updates)
Mini-Batch SGD	Moderate (parallelizable)

Stochastic Gradient Descent (SGD)
Gradient Descent Variants

Method	**Convergence Behavior**
GD	Smooth but potentially slow
SGD	Noisy trajectory, may oscillate
Mini-Batch SGD	Balances speed and stability

Stochastic Gradient Descent (SGD) Example

We illustrate the different variants of Gradient Descent on a simple *linear regression* problem with two parameters

$$y_i = \theta_1 x_{i1} + \theta_2 x_{i2} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

The objective is to minimize the *mean squared error* (MSE)

$$f(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^\top \theta)^2$$

with gradient

$$\nabla f(\theta) = \frac{2}{N} \mathbf{X}^\top (\mathbf{X}\theta - \mathbf{y})$$

Stochastic Gradient Descent (SGD)
Example

We generate a synthetic dataset with two predictors and compare three optimization strategies :

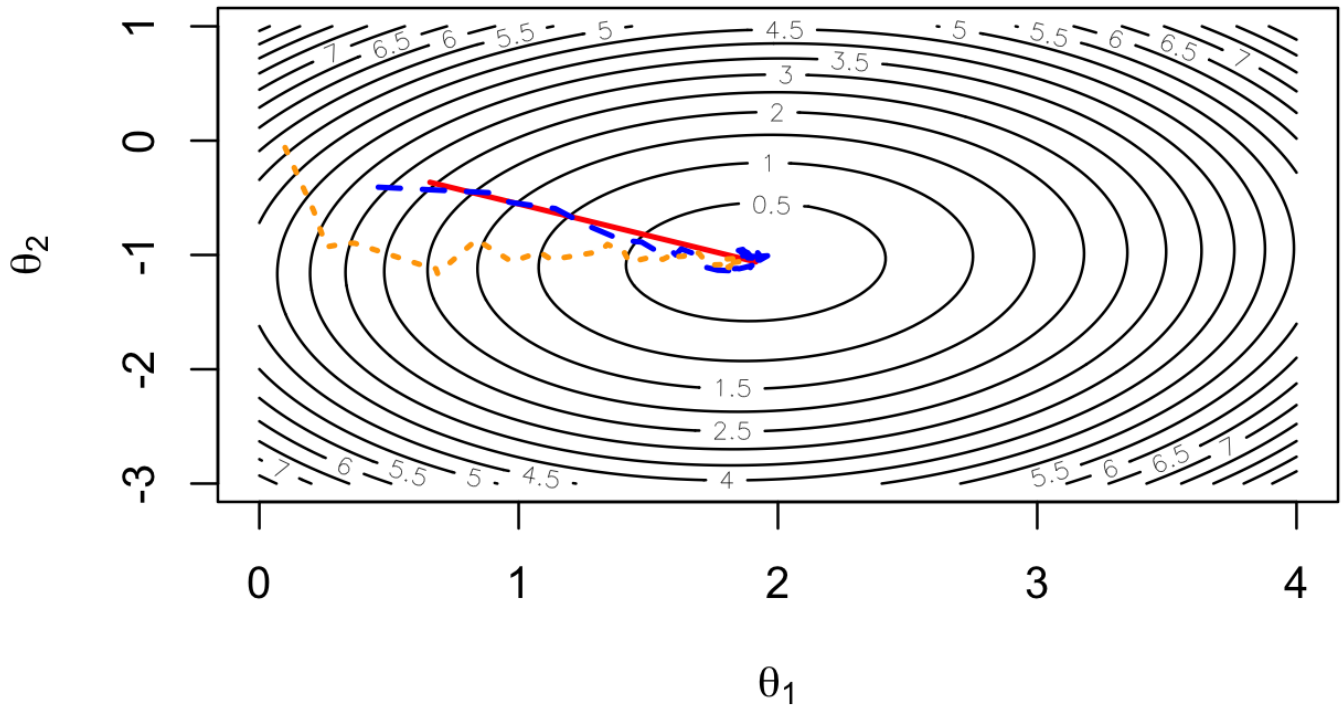
- GD uses the full dataset at each iteration
- SGD uses a single observation per iteration
- Mini-Batch SGD uses small random subsets of data

The updates are given by

$$\theta^{(k+1)} = \theta^{(k)} - \delta_k \nabla_{\text{sample}} f(\theta^{(k)})$$

where the gradient is computed from either the full dataset, a single data point ou a batch

Stochastic Gradient Descent (SGD) Example



Stochastic Gradient Descent (SGD) Example