Lab 1: activities around gradient descent

Silèye Ba

Lecturer, Neoma Business School, Paris
Machine Learning Scientist, L'Oreal Research & Innovation, Paris

12.01.2023

Overview

Function minimization with analytic gradient descent

Objective

Given function defined by $f(x^1, x^2) = (x^1 - 1)^2 + 2(x^2 - 2)^2$:

- Derive $\nabla f(x^1, x^2)$ the gradient of function $f(x^1, x^2)$
- Write a Python function based on gradient descent to find the point where function $f(x^1, x^2)$ reaches it's minimum value
- Display the sequence of points traversed during gradient descent
- Analyse the behaviour of the algorithm with respect to values of λ

Reminder Gradient descent algorithm

1. Select a descent step $\lambda > 0$ and number of descent steps T
2. Initialize $\mathbf{x}_0 = (x_0^1, x_0^2)$ to some values
3. Iterate for $t = 1$ to T :
    - $\mathbf{x}_t = \mathbf{x}_{t-1} - \lambda \nabla f(\mathbf{x}_{t-1})$

Function minimization with Tensorflow numeric gradient

Objective

Given function defined by $f(x^1, x^2) = (x^1 - 1)^2 + 2(x^2 - 2)^2$:

- Compute function's Tensorflow numeric gradient
- Write a Python function to find the point where function $f(x^1, x^2)$ reachs it's minimum value using numeric gradient
- Analyse the behaviour of the algorithm with respect to values of λ
- What are the advantages of using numeric gradient ?

Training a linear regression model with gradient descent

Objective

- We are given a dataset of samples and labels $(\mathbf{x}_n, y_n)$, $n = 1, ..., N$
- Samples $\mathbf{x}_n \in \mathbb{R}^D$ and labels $y_n \in \mathbb{R}$
- Our aim is to build a model $f_{\mathbf{w},b}(\mathbf{x}) = \sum_{d=1}^{D} w^d x^d + b$ such that $f_{w,b}(\mathbf{x}_n) \approx y_n$
- Implement Python/Tensorflow functions to find optimal parameters $\mathbf{w}$ and b by minimizing the square loss function :

$$L(\mathbf{w}, b) = \frac{1}{N} \sum_{n=1}^{N} (f_{w,b}(\mathbf{x}_n) - y_n)^2$$

Training a logistic regression model with gradient descent

Objective

- We are given a dataset of samples and labels $(\mathbf{x}_n, y_n)$, $n = 1, ..., N$
- Samples $\mathbf{x}_n \in \mathbb{R}^D$ and labels $y_n \in \{0, 1\}$
- Our aim is to build a model

$$f_{\mathbf{w},b}(\mathbf{x}) = \frac{1}{1 + \exp -(\sum_{d=1}^{D} w^d x^d + b)}$$

  such that $f_{w,b}(\mathbf{x}_n) \approx y_n$
- Implement Python/Tensorflow functions that find optimal parameters $\mathbf{w}$ and b by minimizing the cross entropy loss (negative log-likelihood) :

$$L(\mathbf{w}, b) = -\frac{1}{N} \sum_{n=1}^{N} (y_n \log f_{w,b}(\mathbf{x}_n) + (1 - y_n) \log(1 - f_{w,b}(\mathbf{x}_n))$$