Title

Subtitle

Author Name

A thesis presented for the degree of
Doctor of Philosophy

Department Name
University Name
Country
Date

Contents

Chapter 1

Gradient descent

1.1 Gradient descent 's algorithm

1.1.1 Introduction

Gradient descent is a general technique for minimizing a twice-differentiable functions through its slope. [1] It is used to find local minimums due to the facts that the start point it is crucial.

The basic idea is to update the weights using the gradients until it is not possible to continuous minimizing the error.

1.1.2 Math

We are going to define algorithm:

Let $w(0) \in \mathbb{R}^d$ be an arbitrary initial point, $E : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ a class C^2 function. The step size $\eta \in \mathbb{R}^+$ is a experimental coefficient about how much are we going to follow the slope to obtain the new weight. Let $w(t) \in \mathbb{R}^d \quad t \in \mathbb{N}$ be the weight for t iteration which is defined as

$$w(t+1) = w(t) - \eta \nabla E_{in}(w(t))$$

Properties

- This algorithm gives local minimums.
- Convergence it is not assured, so it would be necessary some stop criteria.
- For a convex function it would be a unique global minimum.
- η variable in time is importa: fixes learning gradient descent algorithm.

1.1.3 Algorithm

The following code snippet implement the algorithm, where $w(0)$ is the *initial_point*, E is the error, $\nabla E_{in}(w)$ is *gradient_function* and finally η is *eta*. The value $\eta = 0.1$ is a heuristic basic on purely practical observation [1].

This algorithm does not it is necessary to figurate a stop condition, otherwise the algorith could be iterating indefinetely.

```
def gradient_descent(inicial_point, E, gradient_function,  eta=0.1):
    '''
    initicial point: w_0
    E: error function
    gradient_function
    eta:  step size
    '''
    #### stop conditions   ######
    max_iter = 1000
    max_error =  0.0001
    ####

    iterations = 0
    current_error = E( intial_point[0], intial_point[1])
    w = initial_point

    while iterations < max_iter and error > max_error:

        w = w - eta * gradient_function(w[0], w[1])
        iterations += 1

    return w, iterations
```

1.1.4 Problem 2

We want to solve the following problem:

Run gradient descent's algorimth to find a minimun for function $E(u, v) = (u^3 e^{(v-s)} - 2 * v^2 e^{-u})^2$. Start with $(u, v) = (1, 1)$ and $\eta = 1.0$

Solve analytically the gradient of $E(u, v)$

$$\nabla E(u, v) = \left(\frac{\partial}{\partial u} (u^3 e^{(v-2)} - 2 * v^2 e^{-u})^2, \frac{\partial}{\partial v} (u^3 e^{(v-2)} - 2v^2 e^{-u})^2 \right) =$$

$$= \Big(2(3u^2 e^{(v-2)} + 2v^2 e^{-u}), 2(u^3 e^{(v-2)} - 4ve^{-u}) \Big)$$

Bibliography

[1] Hsuan-Tien Lin Yaser S. Abu-Mostafa, Malik Magdon-Ismail. *Learing From Data. A Short Course.* 2012.