Notes de cours CADL - session-2

Thibaut Marmey

October 26, 2018

- The basic components of a neural network
- How to use gradient descent to optimize parameters of a neural network
- How to create a neural network for performing regression

Contents

1 Introduction

1.1 Generalities

- Use data and gradient descent to teach the network what the values of the parameters of the network should be.
- Idea of machine learning : letting the machine learn from the data.
- We are interested in letting the computer figure out what representations it needs in order to better describe the data and some objective we've defined.

1.2 Gradient Descent

- Operation of the network are meant to transform the input data into something meaningful that we want the network to learn about.
- Parameters of the NW are random so output is random as well.
- If we need specific output, we can use "Gradient Descent" : way to optimize set of parameters.

1.3 Defining Cost

- Mesure of the "error"
- Exemple : recognize apple or orange. Random network spit ou random 0s or 1s for apples and oranges. We can define :
  - if the network predicts a 0 for an orange, then the error is 0. If the network predicts a 1 for an orange, then the error is 1.
  - And vice-versa for apples. If it spits out a 1 for an apple, then the error is 0. If it spits out a 0 for an apple, then the error is 1.
- Defining error in terms of our parameters :

$$error = network(image) - true\_label \tag{1}$$

$$network(image) = predicted\_label \tag{2}$$

$$E = f(X) - y \tag{3}$$

1.4 Minimizing Error

- Feed the network many images (100 for e.g) to see what the network is doing on average.
- Changing network's parameters can have effect on the error.
- The error provides a "training signal" or a measure of the "loss" or our network.
- Assumptions in assuming our funtion is continuous and differentiable.
- Gradiant descent in a nutshell : "Error", "Cost", "Loss", or "Training Signal"

1.5 Backpropagation

- The gradient is just saying, how does the error change at the current set of parameters.
- To figure out what is the gradiet we use backpropagation. Whatever differences that output has with the output we wanted it to have, gets *backpropagated* to every single param in our network.
- Backprop is an effective way to find the gradient. Uses the *chain rule* to find the gradian of the error.
- $y = mx + b$ linear function. The slope or gradian is m.
- The process described :

$$\theta = \theta - \eta * \nabla_\theta * J(\theta) \tag{4}$$

  - θ : parameters
  - η : gradient, with repect to parameters θ, ∇_θ
  - J : error
  - η : learning rate describes how far along this gradient we should travel, typically value between 0.01 to 0.00001

1.6 Local Minima/Optima

-