1 Week1

Introduction to Deep Learning.

1.1 What is a Neural Network?

1.2 Supervised Learning with Neural Networks

1.3 Why is Deep Learning taking off?

1.4 About this Course

1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring your Machine Learning project
4. Convolutional NeuralNetworks
5. Natural Language Processing: Building sequence models

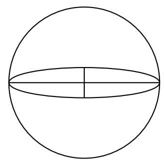
1.5 Outline of this Course

- Week1. Introduction
- Week2. Basics of Neural Network programming
- Week3. One hidden layer Neural Networks
- Week4. Deep Neural Networks

2 Week2

Basics of Neural Network Programming

How do I write an equation in LaTeX?



In 1902, Einstein created this equation: $E = mc^2$

And Newton came up with this one: $\sum F = ma$

$$5 + 5 = 10 \tag{1}$$

$$\begin{gathered} A = \frac{5\pi r^2}{2} \\ A = \frac{1}{2}\pi r^2 \end{gathered} \tag{2}$$

2.1 Neural Network Notations

General comments:
superscript (i) will denote the i^{th} training example.
superscript $[l]$ will denote the l^{th} layer.

Sizes:

- m: number of examples in the dataset
- n_x: input size
- n_y: output size
- $n_h^{[l]}$: number of hidden units of the l^th layer.
  In a for loop, it is possible to denote $n_x = n_h^{[0]}$ and $n_y = n_h^{[number of layer+1]}$
- L: number of layers in the network

Objects:

- $X \in \mathbb{R}^{n_x \times m}$ is the input matrix
- $x^{(i)} \in \mathbb{R}^{n_x} is the i^{th} example represented as a column vector$
- $Y \in \mathbb{R}^{n_y \times m}$ is the label matrix
- $y^{(i)} \in \mathbb{R}^{n_y}$ is the output label for the i^{th} example
- $W^{[l]} \in \mathbb{R}^{number of units in next layer \times number of units in the previous layer}$ is the weight matrix, superscript $[l]$ indicates the layer
-

Common forward propagation equation examples:

-
-

Examples of cost functions:

-
-

2.2 Binary Classification

Use matrix without using for loops.
Computation using Forward propagation and Backward propagation.
Logistic regression is an algorithm for binary classification.

m training examples $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ where $x^{(i)} \in \mathbb{R}^{n_x}$ and $y^{(i)} \in \{0, 1\}$ for $i \in [1, m]$

$$X = \begin{bmatrix} \vdots & \vdots & \vdots \\ X^{(1)} & X^{(1)} & X^{(m)} \\ \vdots & \vdots & \vdots \end{bmatrix} \in \mathbb{R}^{n_x \times m}$$

$$X.shape = (n_x, m)$$

$$Y = [Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}] \in \mathbb{R}^{1 \times m}$$

$$Y.shape = (1, m)$$

2.3 Logistic Regression

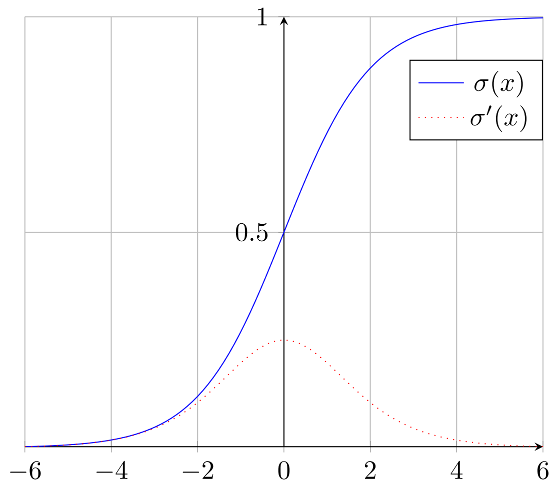
Given X, want $\hat{Y} = P(Y = 1|X)$ where $X \in \mathbb{R}^{n_x}$

Parameters: $\omega \in \mathbb{R}^{n_x}$ a n_x dimensional vector, $b \in \mathbb{R}$ a real number.

Output $\hat{y} = \sigma(\omega^T X + b) = \sigma(z)$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Drawing a sigmoid function and its derivative in tikz



2.4 Logistic Regression Cost Function

To train the parameter ω and b of a Logistic Regression Model, we need a cost function.

$\hat{y} = \sigma(\omega^T X + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$

$\hat{y}^{(i)} = \sigma(\omega^T X^{(i)} + b)$ where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$

Given $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss(error) function (for a single training Example):
$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$

Cost function (for the entire training Examples):
$$\mathcal{J}(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$$

The loss function computes the error for a single training example; the cost function is the average of the loss functions of the entire training set.

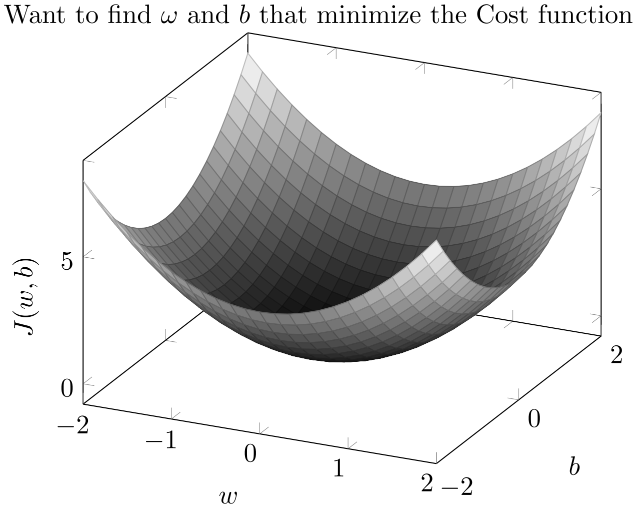
In training logistic regression model, we will try to find ω and b such that they minimize the Cost function $\mathcal{J}(\omega, b)$.

Logistic Regression can be seen as a very small Neural Network.

2.5 Gradient Descent

$$\mathcal{J}(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

Want to find ω and b that minimize the Cost function $\mathcal{J}(\omega, b)$.



$\mathcal{J}(\omega, b)$ is a convex function with a single local optimum.
No matter where you initialize the point, you should get to the same point (Global optimum).

Repeat:

$\omega := \omega - \alpha\frac{\partial\mathcal{J}(\omega,b)}{\partial\omega}$

$\omega := \omega - \alpha d\omega$

$b := b - \alpha\frac{\partial\mathcal{J}(\omega,b)}{\partial b}$

$b := b - \alpha db$

where α is the learning rate.