Introduction to Deep Learning

Seungyeop Hyun

Department of Statistics
Seoul National University

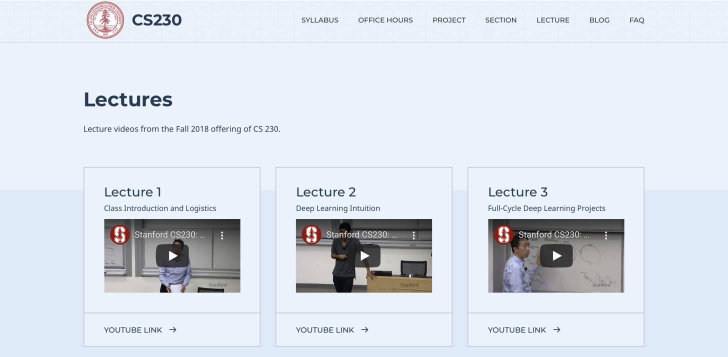
Jan 21st, 2022

Outline

Introduction to Deep Learning

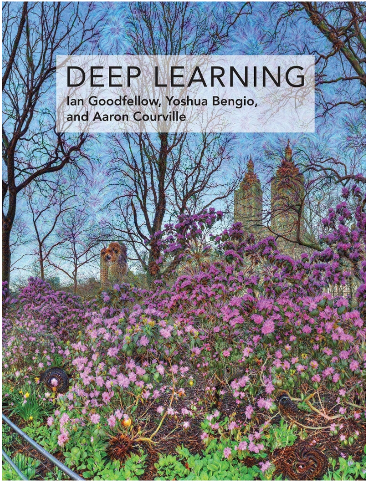
- Materials
- Logistic Regression as a Neural Network
- 2-layer Neural Network
- Elements of Deep Learning
- Exercise

CS230



- Website : cs230.stanford.edu
- Coursera : coursera.org

Deep Learning - Ian Goodfellow



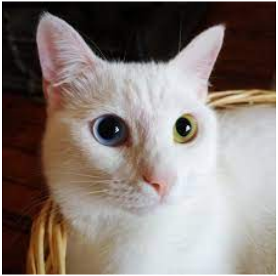
- By Ian Ian Goodfellow, Yoshua Bengio, Aaron Courville.
- Considered "Bible" of deep learning.

Outline

Introduction to Deep Learning

- Materials
- Logistic Regression as a Neural Network
- 2-layer Neural Network
- Elements of Deep Learning
- Exercise

Binary Classification



- Is it cat(1) or not(0)?
- From image(data), we compute $\hat{y}^{cat} = 1, 0$

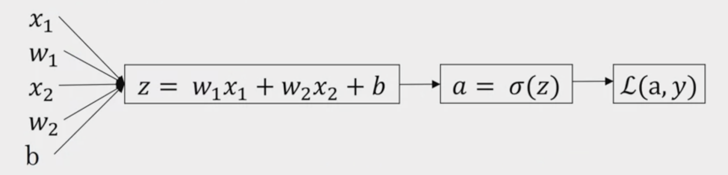
Logistic regression

- Given $x \in \mathbb{R}^{n_x}$, we want to calculate $\hat{y} = P(y = 1|x) \in [0, 1]$.
- We need the parameters $w \in \mathbb{R}^{n_x}, \;\; b \in \mathbb{R}$
- The output value is $\hat{y} = \sigma(w^T x + b)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$
- With a loss function, $L(\hat{y}, y)$, we can use cost function

$$
\begin{aligned}
J(w, b) &= \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y) \\
&= \frac{1}{m} \sum_{i=1}^{m} -(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}))
\end{aligned}
$$

- we estimate $w, \; b$ minimizing the cost function $J(w, b)$

Logistic regression as a Neural Network



- The procedure of logistic regression when $n_x = 2$, $x = (x1, x2)'$.
- We find $\hat{w}$, $\hat{b}$ with a **gradient descent**, so we need the derivatives of $L(a, y)$
- It can be summarized by the repeating
  1. compute $L(a, y)$ from x with $w, b \cdots$ forward propagation
  2. compute the derivative $\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b} \quad \cdots$ backward propagation

Logistic regression as a Neural Network

Assume there is a single data $x = (x1, x2)'$. Then the cost function is $L(a, y) = -(y \log a + (1 - y) \log (1 - a))$

1. First, compute

$$\frac{dL}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

2. Using *chain rule*,

$$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz} = a - y$$

where $\frac{da}{dz} = \sigma'(z) = a(1-a)$

Logistic regression as a Neural Network

3. Finally, we can take $\frac{dL}{dw_1}, \frac{dL}{db_1}, \frac{dL}{dw_2}, \frac{dL}{db_2}$. For example,

$$\frac{dL}{dw_1} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_1} = x_1\frac{dL}{dz}$$

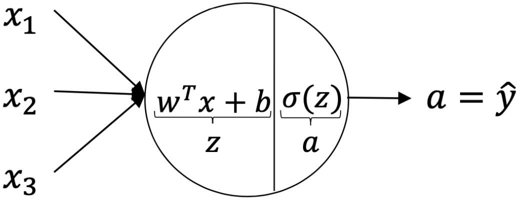
- We can compute the derivatives with reverse direction of calculating a.

Outline

Introduction to Deep Learning

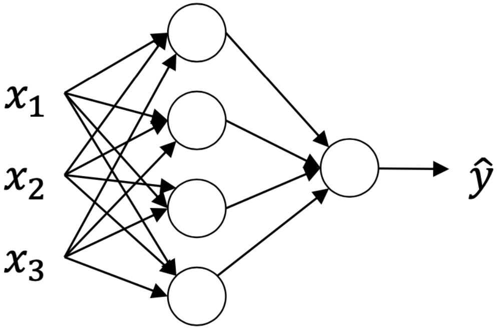
- Materials
- Logistic Regression as a Neural Network
- 2-layer Neural Network
- Elements of Deep Learning
- Exercise

Representation



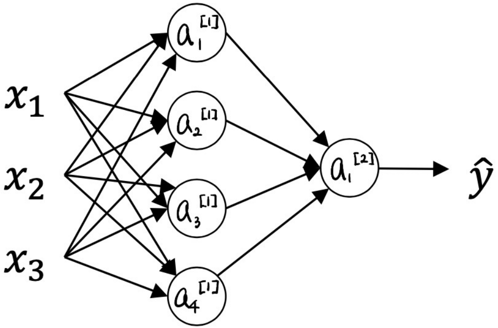
- Multiply weights to x, and throw the values to the sigmoid function. It is the calculation that takes place in the circle called *node*.
- The collection of nodes forms a *layer*. And the neural network is made up of stacked layers.

Representation



- Example of 2-layer neural network.
- $x = (x_1, x_2, x_3)'$ called **input layer**
- 4 circles in the middle form a layer called **hidden layer**. We can stack more layers if we need.
- A circle before $\hat{y}$ called **output layer**, there can be 2 or more nodes.

Representation



- We want the value of $a_1^{[2]} = \sum_{i=1}^{4} a_i^{[1]}$. Each $a_i^{[1]}$ is computed,

$$z_i^{[1]} = {w_i^{[1]}}^T x + b_i^{[1]} \longrightarrow a_i^{[1]} = \sigma(z_i^{[1]})$$

where $w_i^{[1]}$ is the weight vector for $a_i^{[1]}$

Representation

- Weights w can be expressed by matrix.

$$W^{[1]} = \begin{pmatrix} {w_1^{[1]}}^T \\ {w_2^{[1]}}^T \\ {w_3^{[1]}}^T \\ {w_4^{[1]}}^T \end{pmatrix}, \quad W^{[2]} = ({w_1^{[2]}}^T)$$

- For m data, data matrix is $X = (x^{(1)}, \cdots, x^{(m)}) \in \mathbb{R}^{n_x \times m}$ where $x^{(i)} \in \mathbb{R}^{n_x}, \; n_x = 3$
- $Z^{[1]}, A^{[1]}$ are calculated by

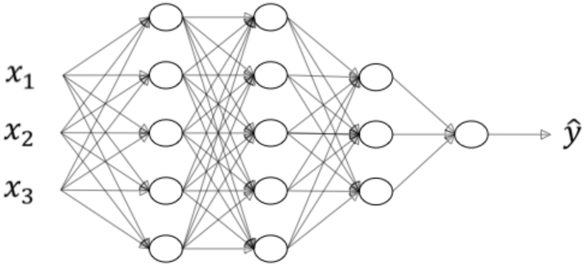
$$Z^{[1]} = W^{[1]}X + b^{[1]} \longrightarrow A^{[1]} = \sigma(Z^{[1]})$$

Outline

Introduction to Deep Learning

- Materials
- Logistic Regression as a Neural Network
- 2-layer Neural Network
- Elements of Deep Learning
- Exercise

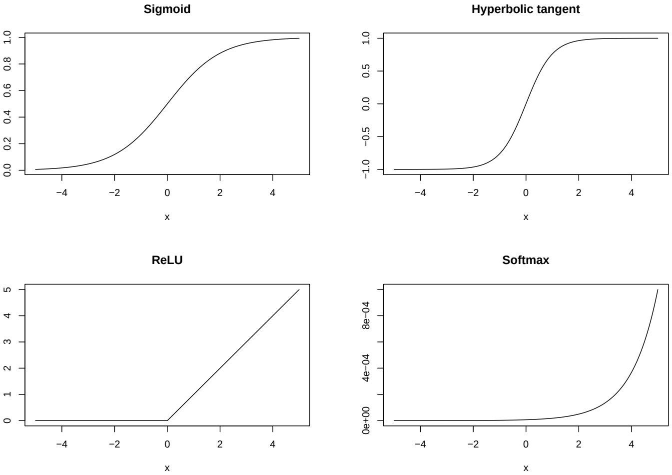
Example: 4-array Neural Network



- $L = 4$: number of layers
- Data matrix $X = (x^{(1)}, \cdots, x^{(m)}) \in \mathbb{R}^{n_x \times m}, \ n_x = 3.$
- $W^{[l]}$: weights for $Z^{[l]}$
- $A^{[l]}, \ n^{[l]}$: activations and number of nodes in layer l

Activation function

- There are various types of Activation function.



Activation function

- Choice of activation function is very important when we use neural network. It depends on the type of expected values of $\hat{y}$
- Sigmoid, Hyperbolic tangent have a *vanishing gradient* problem.
- Using ReLU(Rectified Linear Unit) it the default recommendation.

Activation function

- Activation function should be non-linear.
- If there are linear activation function like $g(z) = z$, the calculating through the hidden layer is meaningless.
- Assume that $Z_1 = W_1 A_0 + b_1$ pass the activation function so the output is $A_1 = Z_1$. And it is used as input of next layer. Then the output of the next layer is

$$\begin{aligned} A_2 = W_2 A_1 + b_2 &= W_2(W_1 A_0 + b_1) + b_2 \\ &= (W_2 W_1) A_0 + (W_2 b_1 + b_2) = W' A_0 + b' \end{aligned}$$

- No matter how many layers in the neural network, it has the same effect with the value pass through just one layer with linear activation function.

Forward propagation

- At layer l, we use $A^{[l-1]} \in \mathbb{R}^{n^{[l-1]} \times m}$ as input and return the output $A^{[l]}$
- Parameters : $W^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}},\ b^{[l]} \in \mathbb{R}^{n^{[l]}}$
- Calculation.

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(Z^{[l]})$$

  where $g^{[l]}$ is an activation function at layer l.
- Direction from input layer to output layer.

Backward propagation

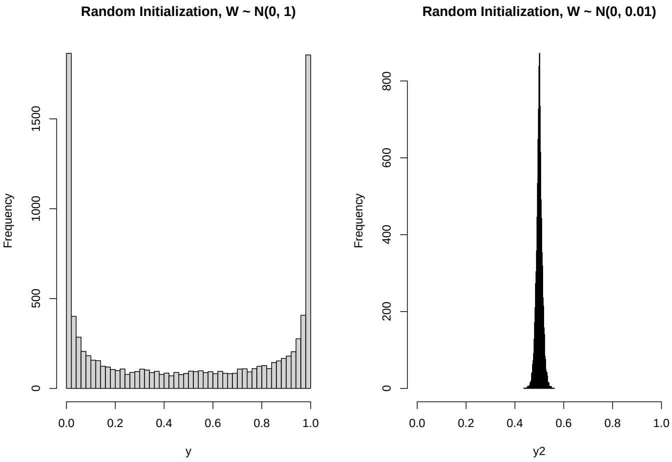
- Procedure with reverse direction from forward propagation.
- At layer l, we use the derivative $dA^{[l]} := \frac{\partial J(W,b)}{\partial A^{[l]}}$ as input and return $dA^{[l-1]}, dW^{[l]}, db^{[l]}$
- We have to use $W^{[l]}, b^{[l]}, Z^{[l]}$ cached from forward propagation.

$$
\begin{aligned}
dZ^{[l]} &= dA^{[l]} \frac{\partial A^{[l]}}{\partial Z^{[l]}} = dA^{[l]} * g^{[l]\prime}(Z^{[l]}) \\
dW^{[l]} &= \frac{1}{m} dZ^{[l]} {A^{[l-1]}}^T \\
db^{[l]} &= \frac{1}{m} \text{rowsum}(dZ^{[l]}) \\
dA^{[l-1]} &= {W^{[l]}}^T dZ^{[l]}
\end{aligned}
$$

Random initialization

- If we give a nice initial value to the parameter, we are more likely to find the proper estimate.
- If we set all the values of weights 0, the output value from the node at each layer have same value, and it means the derivatives for updating parameters are same.
- The method of random initialization depends on the activation function. For example, assume that we use sigmoid as the activation function.

Random initialization



Random initialization

- The derivative of sigmoid function is

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

  if $\sigma(x) = 0, 1$, the gradient equals to zero.

- We can use *Xavier initialization* for the randomize the value of initial weights.

$$W \sim N\left(0, \sqrt{\frac{2}{n_{in} + n_{out}}}\right)$$

- It is known that the method gives the value small enough to train the model.

Summary

Outline

Introduction to Deep Learning

- Materials
- Logistic Regression as a Neural Network
- 2-layer Neural Network
- Elements of Deep Learning
- Exercise