Coursera – Deep Learning Speacialization

Course 1 : Deep Learning and Neural Networks

08 April 2020

23:33

|  |
| --- |
| Week 1: Introduction to deep learning |
| Welcome to the Deep Learning Specialization |
| Week 2: Neural Networks Basics |
| Week 3: Shallow neural networks |
| Week 4: Deep Neural Networks |
| Takeaways from discussion with Deep Learning Heros |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



|  |
| --- |
| **What is a neural network?**  Neural network can be best explained with an example:  Let say we want to predict the price of the house given size of the house.  We have the dataset with relation between price and size follows:  **House Price Prediction Example:**  Here in this graph we can see the blue line is the best  fit function which can represent the size to price. This  Can be assumed as the simplest form of the neural network with one neuron.  This can be scaled to number of input parameters for more complex scenario. Then we might need to stack more neurons on one another and connect to get a network. This network of neurons then would be called as **Neural Network.** As depicted here. The nice thing about the neural network is we don’t need to think about how the network is finding relation among different inputs to predict the output.  We just need to train the network with input/output dataset (supervised learning) then it figures out the relations itself. |
| **Supervised learning with neural networks (Applications)**   |  |  |  |  | | --- | --- | --- | --- | | **Input (X)** | **Output (Y)** | **Application** | **NN Type** | | Home Features | Price | **Real Estate** | **Standard NN** | | Ad. User info | Click on Ad (0/1) ? | **Online Advertising** | **Standard NN** | | Image | Object (1….1000) | **Photo Tagging** | **CNN** | | Audio | Text Transcripts | **Speech Recognition** | **RNN** | | English | Chinese | **Machine Translation** | **RNN** | | Image, Radar info | Position of the car | **Autonomous Car** | **Hybrid** | |
| **Why is deep learning taking off?**   1. **Data** 2. **Computation** 3. **Algorithm** |
| **Binary Classification**  Logistic regression is used for Binary classification problems where we need to classify between 1/0.  Like, if we want to classify an image if it is a Cat (1) or non-Cat (0) this can be treated as a Binary classification problem.  Stacking features (X) and output (Y) as columns is better for implementation than in stacking in rows.  [Link to Notation](../References/Notations.pdf) |
| **Logistic Regression**  Given feature X we want to predict  So that,      We need to calculate parameters W, b so that correctly represents Y  In normal regression problem we can use W^TX + b as function for the algorithm. But in logistic regression it is not a good option because we are predicting a probability which would be between 0-1.  Hence, we use sigmoid function.   |  | | --- | | If z is very large then,  sigmoid(z) = 1  If z id very small (large negative) then, sigmoid(z) = 0 |   **Sigmoid function** |
| **Logistic regression cost function**  We know that where  Given , want  **Loss Function:**  In regression problem we generally use mean squared error function as loss function to know how good the model is in prediction.  But in case of logistic regression we cannot use this function, because it will not be convex function and might have many local minima.  **Intuitive Explanation:**  If y = 1 then , to have minimum loss we should have to be larger and hence large  If y = 0 then , to have minimum loss we should have to be larger and hence small  **Cost Function:**  Loss function is for one training example. If we want to average sum of all loss function, then it would be a cost function. We want the cost function J to be function of parameters so that updating the parameters we can minimize the cost function. |
| **Gradient Descent** |
| **Derivatives (Should be reviewed separately from khan Academy)** |
| **More Derivative Examples (Should be reviewed separately from khan Academy)** |
| **Derivative with a computation graph (Forward and Backward pass)**  With the help of computation graph, we can calculate the values for the forward pass and backward pass very easily. If we get from left to right, we can calculate the value for the forward pass and from right to left we can calculate the derivatives. |
| **Gradient descent with computation graph** |
| **Python & Vectorization**  **Vectorization**  Vectorization is computationally much more efficient than using traditional FOR loops. As in deep learning/AI we are dealing with large dataset, it becomes cardinal that we should use Vectorization.  **Example:**   |  |  | | --- | --- | | **FOR LOOP Implementation** | **Vector Implementation** | | Z= 0  For i in range () | **Z= np.dot(W,X) + b** | |
| **Vectorizing logistic regression**   |  |  | | --- | --- | | **FOR LOOP Implementation** | **Vector Implementation** | |  |  | |
| **Vectorizing logistic regression’s gradient descent**  Let say we have 2 features x\_1, x\_2. So, we would have 2 weight parameters (w\_1, w\_2) and bias b.  To derive the gradient descent, we need to calculate dw\_1, dw\_2, d\_b. Let’s see how we can implement this using FOR loop and Vectorization   |  |  | | --- | --- | | **For Loop Implementation** | **Vectorization** | |  | We can get rid out of the 2 for loops for forward and backward pass of the logistic gradient descent as follows.  Z = numpy.dot(W\*X) + b  A = sigmoid(Z) | |
| **Explanation of logistic regression cost function** |
|  |
| **What is a Neural Network?**  …. Shallow Network  **Neural Network Representation:**    In each neuron or computing node, we have 2 types of operations. Linear summation and passing through activation function.  Let’s understand the computation for blow case and how we can represent the same in vector format. |
| **Equations:** |
| **Vectorizing across multiple Training Examples:**  For multiple training example we need to stack the input features X and Values of Z and A in column metrix. |
| **Why we need a non-linear activation function for NN?**  If we don’t use any non-linear activation function then no matter how many layers we are using the end result will be a linear function. It appears that we can not represent real world problem with linear functions only. In some cases it can give good results like regression problem where the output value is a continueous real number. But in other cases we don’t have the desired result. |
| **Derivatives of Activation function commonly used**   |  |  | | --- | --- | | **Activation Function** | **Derivative** | | g(z) = Sigmoid(z) = 1/(1-e^-z) | g’(z) = (1/(1-e^-z))^2\*e^-z = g(z)\*(1-g(z)) | | g(z) = tanh(z) = (e^z-e^-z)/ (e^z+e^-z) | g’(z) = (1-tanh(z)^2) = (1-g(z)^2) | | g(z) = Relu(z) = {0 if z<0| z if z>0} | g’(z) = {0 if z<0 | 1 if z >=0} | | g(z) = LRelu(z) = {0.001z if z<0| z if z>0} | g’(z) = {0.001 if z<0 | 1 if z >=0} |   **Formulas for Gradient Descent:**    **Problem of Symmetry** |
| **Deep L-Layer Neural Network**  **Forward Propagation:**  In case of deep neural network forward prop we need to havee loop to calculate the activation at each layer to reach to the output.  **Getting Metrix dimensions correct**    **Why we need deep neural representation?**  As stated the case below (CNN) the initital layers are detecting low level features or simple functions and while we go into more deeper into the network it calculates surprisingly complex functions. So the problem like image recognition/ speech recognition or other in which we need to calculate very complex functions we need more deep network. While we traverse from first layer to deeper we can experience more complex functions or features are derived.    **The other intuition that people often cite is from circuit theory**    **Buiding Blocks for Deep neural network**      **Forward propagation and Backward propagation**          **Is there any Analogy between Deep neural network and human brain?**  No there is no analogy. People often cite a very loose analogy between a single unit of human brain neuron and neuron of the neural network. Because architecture seems similar. However, we can not say certainly how a single neuron of human brain functions. So, better we should restrict using this analogy s  **Key Takeaway from discussion with Geoffrey Hinton** |
| Derivation dl/dZ |