SUMMARY OF COURSERA COURSE
NEURAL NETWORKS AND DEEP LEARNING (4 WEEKS)
RATINGS: 5/5

WEEK 1 – INTRODUCTION TO DEEP LEARNING

UNIT 1: Welcome

- After this specialization we should be able to put deep learning on our resume with confidence
- AI is the new Electricity, deep learning is one of the most sort after skill
- Overview of Deep Learning Specialization
- CNN are often applied to images

UNIT 2: What is a neural network

- Deep learning refers to training neural networks
- “ReLU”: Rectified Linear Unit
- Deeper neural network is achieved by stacking together more neurons
- You only to provide a neural network with the input and output, it figures out the neuron (hidden units) by itself
- Neural networks are very useful in supervised learning applications

UNIT 3: Supervised Learning with Neural Networks

- Almost all the economic value created by neural networks has been through one type of machine learning called Supervised Learning
- Neural networks applied for: “Real estate”, “Online Advertising”, “Photo tagging”, “speech recognition”, “Machine translation”, “Autonomous driving”
- CNN for image recognitions, RNN for sequence data like audio, speech, language, Custom/hybrid neural network for autonomous driving
- Structured data: database of data (rows and columns)
- Unstructured data: audio, image, text
- We will cover both structured and unstructured data

UNIT 4: Why is Deep Learning taking off?

- We generate a lot of data now because of digitalization of the world
- (Scale)Size of the neural network and data drives deep learning progress
- In small training sets performance is sometimes determined by your skill in feature engineering, but in large training sets large NN perform well
- Computational Ability and Algorithmic innovations (sigmoid to ReLU function has helped with computation time) drive deep learning progress

- Process of training neural networks is iterative: “idea” > “code” > “experiment”
- Faster computation helps to iterative much faster
- Deep learning will keep on getting better for years to come

UNIT 5: About this course

- Specialization comprises of 5 courses
- Overview of course: “Introduction”, “Basics of NN programming” (start programming exercise), “One Hidden layer NN”, “Deep Neural Networks”

UNIT 5: Frequently Asked Questions (FAQ)

- Screenshotted

UNIT 6: Course Resources

- Discussion Forum

UNIT 7: How to use Discussion Forum

- Screenshotted

UNIT 8: Quiz

UNIT 9: Geoffrey Hinton Interview

- To introduce us to some of the heroes in deep learning
- Dropping some big deep learning grammars
- He believes unsupervised learning will be crucial in the future
- Read a little bit of the literature and notice something they are doing wrong and try to fix it
- Never stop programming, trust your intuitions
- Our relationship with computers have changed, we now show the computer what to do and not just hard code it

WEEK 2 – LOGISTIC REGRESSION AS A NEURAL NETWORK

UNIT 1: Binary Classification

- Basics of neural networks
- Learn about forward and back propagation
- Logistic regression an algorithm for binary classification
- Your computer stores images in RGB
- “(x, y)” = single training example
- “m training examples”
- “X” : stacking our training examples in columns (nx * m)
- “Y”: stacking our labels in columns (1*m)

UNIT 2: Logistic Regression

- Sigmoid function = 1/(1+e^{-z}), so the larger z is the closer to zero the sigmoid function is

UNIT 3: Clarification about upcoming logistic regression cost function video

- We want –ylog(y_hat) to be "as big as possible" not "as small as possible"

UNIT 4: Logistic Regression Cost Function ()

- We have to define a "cost function" in order to measure model performance
- Logistic regression can be viewed as a very small neural net

UNIT 5: Clarification about upcoming gradient descent video

- The negative sign should apply to the entire cost function (both terms in the summation)

UNIT 6: Gradient Descent

- We want to find the optimal parameters to minimize the cost function "J"
- Our cost function "J" has to be convex to allow for gradient descent
- Whichever direction we start from "gradient descent" will still approach the global optimal

UNIT 7: Derivatives

- You don't need a deep understanding of calculus to implement deep learning, all you need is an intuitive understanding of calculus
- In linear function the slope is always the same (the function derivative is the same)

UNIT 8: More Derivative Examples

- Derivatives are defined by "infinitesimal differences"
- Derivative/ slope: ratio of change of f(x) to change in x
- The slope can be difference for other non-linear function at different point on the graph

UNIT 9: Computation Graph

- Computations of a neural network are organized in terms of a forward pass (forward propagation) where we compute the output of the neural network, followed by the backward pass (back propagation) where we compute gradients/ derivatives
- Computation graph comes in handy when there is some special output variable you want to optimize
- Left to right (forward pass), right to left (backward pass)

UNIT 10: Derivatives with a Computation Graph

- The chain rule
- When doing backpropagation there is always one output variable we want to optimize

UNIT 11: Logistic Regression Gradient Descent

- “input x, w” > “z = w1x1 + w2x2 + b” > “a=sigmoid(z)” > “Loss function”

UNIT 12: Gradient Descent on m Examples

- Cost function is the average of the individual loss functions, also the derivative of the cost function is simply the average of the derivatives of the individual loss functions
- You have to write for loop for examples and also for loop for features
- Vectorization technique help us to eliminate the use of explicit for loops

UNIT 13: Derivation of DL/dz (optional reading)

UNIT 14: Vectorization

- Vectorization is simply the act of getting rid of explicit for loops in your code
- It helps to reduce computation time
- Python tries to uses parallelization to speed up runtime, GPUs are better than CPUs in this

UNIT 15: More Vectorization Examples

- Whenever possible, avoid explicit for-loop, always try built-in function where possible

UNIT 16: Clarification of “dz”

UNIT 17: Vectorising Logistic Regression

- Instead of looping we use numpy built-in functions to vectorize the calculations for the logistic regression prediction

UNIT 18: Vectorizing Logistic Regression’s Gradient Output

- For multiple iteration of gradient descent, we still need “for loop” for the number of iterations

UNIT 19: Broadcasting in Python

- Broadcasting is a technique used in python to make our code to run faster
- Checkout the numpy broadcasting documentation

UNIT 20: A note on python/numpy vectors

- “rank 1 arrays” are neither row nor column vectors
- Rank 1 arrays don’t behave like row or columns, so try to avoid it
- When coding do not use data structures that are “rank 1”
- “assert(a.shape == (5,1))”
- You can always reshape “rank 1 arrays”: a.reshape(5,1)

UNIT 21: Quick tour of Jupyter/iPython Notebooks

- Ready to tackle my first programming exercise

UNIT 22: Explanation of logistic regression cost function (optional)

- Maximizing the log of the probability means minimizing the loss function

UNIT 23: Quiz

UNIT 24: Deep Learning Honor Code

UNIT 25: Programming Assignment FAQ

UNIT 26: Python Basics with Numpy (optional)

UNIT 27: Practice Programming Assignment

- Python_Basics_With_Numpy_v3a.ipynb

UNIT 28: Logistic Regression with a neural network mindset

UNIT 29: Practice Programming Assignment

- Logistic_Regression_with_a_Neural_Network_mindset_v6a.ipynb

UNIT 30: Pieter Abbeel Interview

- He started as an Engineering Graduate
- He thinks Machine Learning is the core of everything
- He worked a lot with “Reinforced Learning”
- You also need “domain expertise” and Machine Learning techniques for most projects
- You really need “negative examples” in Reinforced Learning
- Always have the impact of your Artificial Intelligence project
- Lot of online resources to get started, and try things out, don’t just watch videos
- You learn a lot faster if you have a mentor
- “Deep Reinforcement Learning”

WEEK 3 – NEURAL NETWORKS OVERVIEW

UNIT 1: Neural Networks Overview

- We will learn to implement a neural network
- You can create a neural network by stacking together sigmoid unit
- In neural network we run the sigmoid unit multiple times

UNIT 2: Neural Network Representation

- Input Layer > Hidden Layer > Output Layer

- You don’t see the values of the “Hidden Layer”
- We don’t count the “output layer” in conventional naming

UNIT 3: Computing a Neural Network’s Output

- In neural network we run the sigmoid unit multiple times
- We will stack our nodes vertically (rule of thumb for vectorization)

UNIT 4: Vectorizing across multiple examples

- We stack the examples up in columns (stack our examples horizontally)

UNIT 5: Explanation for Vectorized Implementation

UNIT 6: Clarification – Activation Function

- “z” as the horizontal axis

UNIT 7: Activation Functions

- When building your neural network one of the choices you get to make is what “activation function” to use in the hidden layer, as well as what the “output units” of your neural network
- Activation Functions: “Sigmoid Function”, “Hyperbolic Tangent Function”, “ReLU Function”, “Leaking ReLU Function”
- Hyperbolic function almost always works better the sigmoid function
- The activation function can be different for each layer
- If you are doing binary classification the “sigmoid function” is good for the output layer
- “ReLU Functions” is what most people use for the hidden layers, your neural networks work faster with “ReLU”
- Never use “sigmoid function” for hidden layers, use for output layer of binary classification
- It is difficult to know in advance which function will works best, so sample them all and see which works best

UNIT 8: Why do you need non-linear activation functions

- For your neural network to compute interesting functions you need to pick a non-linear function
- A linear hidden layer is in itself useless because decomposition of linear function is still a linear function
- The one place we might use linear function is in the “output layer”

UNIT 9: Derivatives of activation functions

- When implementing back propagation, we need to compute the slope/derivative of the activation functions

UNIT 10: Gradient Descent for Neural Network

- Backpropagation in 6 six equations for our 2 layer Neural Network

UNIT 11: Clarification about upcoming Backpropagation Intuition (Optional)

- The text should be “dw[1]” instead of “dw[2]”

UNIT 12: Backpropagation Intuition (optional)

- Chain rule in calculus
- Make sure the dimensions of your matrix mash up when doing back prop
- The derivation of the back prop algorithm is one of the most complicated pieces of math in Machine Learning, requires both linear algebra and derivative of matrixes

UNIT 13: Random Initialization

- It is important to initialize weights randomly in neural networks, it is not okay to initialize the weights to zero
- It you initialize the weights (especially W) to zero the hidden units will all be the same (symmetric) no matter how long you iterate (gradient descent)
- W = np.random.randn((2,2))*0.01, we usually initialize the weights to small values

UNIT 14: Quiz

UNIT 15: Programming Assignment

UNIT 16: Ian Goodfellow Interview

- He had the intuition that deep learning is the future and started focusing on it
- “Generative Models”, he developed GANS (Generative Advisory Networks)
- He hopes to make GANS as stable as Deep Learning
- You don’t really need to get a Phd to get into a AI, you can get a foothold by writing good code and putting on GitHub, creating/working interesting projects allows to apply machine learning, writing articles/papers on “Archive”

WEEK 4 – DEEP NEURAL NETWORK

UNIT 1: Deep L-layer Neural Network

- We have seen all the ideas we already need to build our deep neural network
- When counting layers, we only count the hidden and output layers
- Use “L” to denote number of layers
- You can check out the notation guide

UNIT 2: Forward Propagation in a Deep Network

- It is like a repetition of activation functions multiple times
- There is an unavoidable for loop that helps us to loop over each of the layers of the deep neural network

UNIT 3: Clarification about getting your matrix dimensions' right video

- Correct formula should be a = g(z), a and g have dimensions ($n^{[1]}$, 1)

UNIT 4: Getting your matrix dimensions right

- One of the debugging tools he uses is to get a piece of paper and work through the dimensions and matrix he is working with
- Neural networks can have multiple output units
- Keeping track of the dimensions of our matrix help us to avoid some bugs

UNIT 5: Why Deep Representations

- Neural network with deep hidden layers perform very well in general
- Using deep hidden layers helps the neural network to go from simple to complex (by building on earlier layers) as it progresses through the deep hidden layers
- People make comparison between neural networks and how we think the human brain works
- There are functions you can compute with a "small" (number of layer nodes) L-layer deep neural network that shallower networks require exponentially more hidden units to compute
- Some circuit theory stuff
- "Deep Learning", results of branding

UNIT 6: Building blocks of deep neural networks

- One iteration of gradient descent for neural network starts with forward prop then corresponding back prop to get the derivatives

UNIT 7: Clarification about Upcoming Forward and Backward Propagation Video

- "dw_l = dz_l * a_l-1.T"
- "a" should be transposed

UNIT 8: Forward and Backward Propagation

- Similar forward prop and back prop for the shallow layer neural network
- You understand concepts better when you code it up
- He still gets surprise by the working of his machine learning implementations, because lot of complexity of ML comes from the data not the code

UNIT 9: Parameters vs Hyperparameters

- Being effective in neural networks requires you to organize your parameters and your hyperparameters
- Parameters: W, b
- Hyperparameters: Learning rate, no of iterations, no of hidden layers, no of hidden units, choice of activation function
- Hyperparameters are parameters that control the ultimate parameters (W, b)
- Deep learning neural networks has a lot of hyper parameters, we will discuss them in the 2nd course
- Applied Deep Learning is a very empirical process (try lot of different ideas): Idea > Code > Experiment
- It difficult to know in advance the best hyperparameters for a particular deep learning project
- Every few months double check the hyperparameters of your deployed Deep learning model

UNIT 10: Clarification about What does this have to do with the brain video

UNIT 11: What does this have to do with the brain

- “not a whole lot”
- They use the analogy of deep learning neuron to the brain neurons
- The working of the brain is still at large mysterious to neuro scientist
- The analogy is not very effective (not useful) today

UNIT 12: Quiz

UNIT 13: Programming Assignment Part 1

UNIT 14: Programming Assignment Part 2