Machine Learning: Introduction to Linear Regression, Logistic Regression, and Neural Networks

Machine Learning:

- Chapter 1: Introduction
- Chapter 2: Python Demos
- Chapter 3: Mathematical Concepts
- Chapter 4: Linear and Logistic Regression
- Chapter 5: Neural Networks
- Chapter 6: Optimization, Validation, and Regularization
- Chapter 7: Case Studies
- Chapter 8: Introduction to Tensorflow
- Chapter 9: Summary and Thank You

1.1 What is Machine Learning?

Identifying Cats and Dogs

Pictures source: pixabay.com

How do we as humans learn to identify cats and dogs?

- At an early age, parents/siblings/teachers point to the animals or to pictures and say that this is a cat or this is a dog.
- As this happens 10s or 100s of times during our early years, “rules” to identify cats and dogs are encoded in our brains.
- Using these “rules”, children become able to classify whether an animal is a cat or a dog without help from others
- This is referred to as Concept Learning in human psychology

Machine Learning – Definition

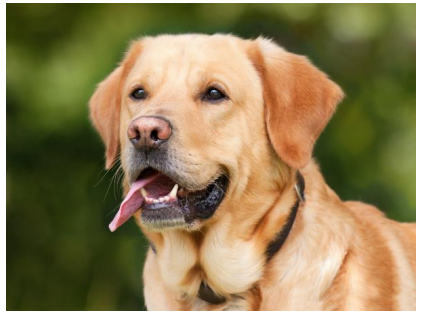
- Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.
- Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.
- Source: https://en.wikipedia.org/wiki/Machine_learning
- How does relate to identifying cats and dogs?
  - The examples we are shown in early childhood constitute the training data
  - Rules for identifying cats and dogs are encoded in our brains
  - In later years, we are able to perform the identification using the rules without using any further explicit instructions

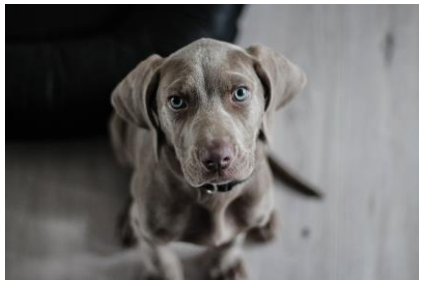
Machine Learning:

Three areas of application:

- Classification
- Clustering
- Game playing

Machine Learning: Classification



Pictures source: pixabay.com

For classification (cats and dogs) using machine learning:

- Training Data consists of 100s or 1000s of images each labelled as cat or dog
- Based on training data, Machine Learning system develops model/rules to perform classification
- When new image (without label) is shown, system uses the model/rules to make a prediction of cat or dog

Machine Learning: Clustering



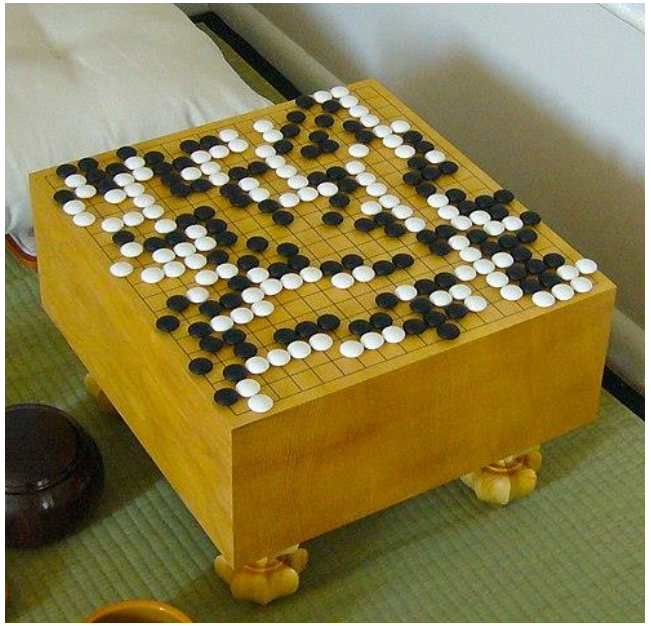



Pictures source: pixabay.com

For clustering using machine learning:

- Input consists of images (cars and bicycles) without labels
- Machine Learning system is used to find patterns/“clusters” in the data
- In this case, expect 2 broad clusters (cars and bicycles)
- If sufficient data, system may also find sub-clusters (convertibles, 2-doors, 4-doors, road bikes, mountain bikes, blue objects, etc)

Machine Learning: Game Playing



By Goban1 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=15223468

For game playing (such as Go above), Machine Learning system:

- Is provided (encoded) with the rules of game (how to move pieces)
- Through self-play or provided with database of games, Machine Learning system learns what moves to make at each turn to ultimately win game
- AlphaZero program that plays Go, Chess, and Shogi was trained solely using self-play in a short period of time and was able to beat other systems convincingly
  - See https://en.wikipedia.org/wiki/AlphaZero for more details

Types of Machine Learning

Type	Description (Adapted from https://en.wikipedia.org/wiki/Machine_learning)
Supervised Learning	Process of learning a function that maps input information to labelled output information. The input/output information is called the training data. The learned function is then used to predict output labels when new input information is provided. Applications: regression, image classification, language translation, spam classification, auto-completion, etc
Unsupervised Learning	Process of learning previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision Applications: clustering, data mining
Reinforcement Learning	Process of learning what actions to take given the situation so as to maximize a reward Applications: Game playing (Atari, Chinese checkers, Chess, Go, etc), Industrial Control, etc

Supervised Learning

Application	Input Information	Output Information/Label
Regression	• House features: lot area, floor area, # of bathrooms, # of floors, size of garage, etc	• Price
Image Classification	• Images of cats and dogs • Bone x-rays • Images of animals	• cat/dog • normal/broken • Animal name
Language Translation	• English words and phrases	• French translations
Spam Filter	• Messages	• spam/not spam
Auto-Completion	• (2,3,4,5 …) word phrases	• Next word

Supervised Learning: Three Approaches

Course presents three approaches for Supervised Learning:

- Linear Regression
  - Used for prediction of real values
  - Simple approach, which you have probably seen in your courses
  - Useful for introducing concepts
- Logistic Regression
  - Used for binary classification
  - Mathematics and coding are extensions of those for Linear Regression
- Neural Networks (Multi-Layer Perceptrons)
  - Used for binary and multi-class classification (or regression)
  - Mathematics and coding build on concepts for Linear and Logistic Regression

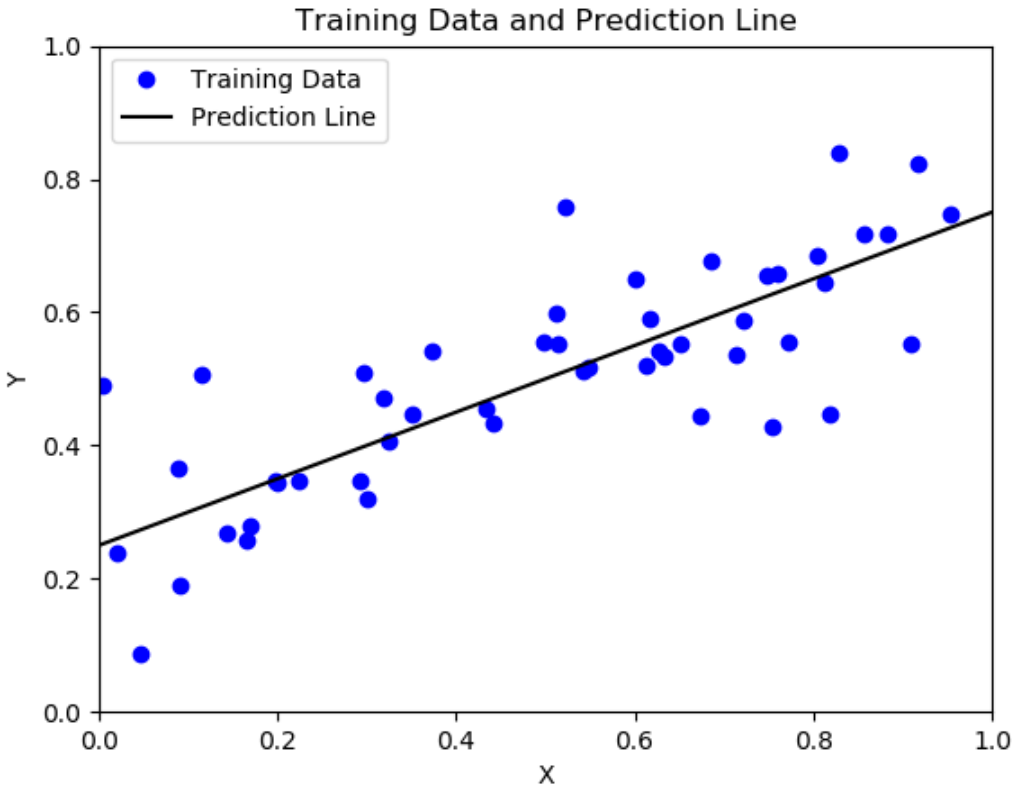
Linear Regression – Line Fitting

Training Data:
- Input information: X values
- Output information: Y values

Linear Regression Goal:
- Find straight line that best fits the training data

Prediction:
- Use line to predict Y values given new input X values

Why start with Linear Regression?
- Simple problem with well known solution
- This course will present a general approach that can also be applied to Logistic Regression and Neural Networks



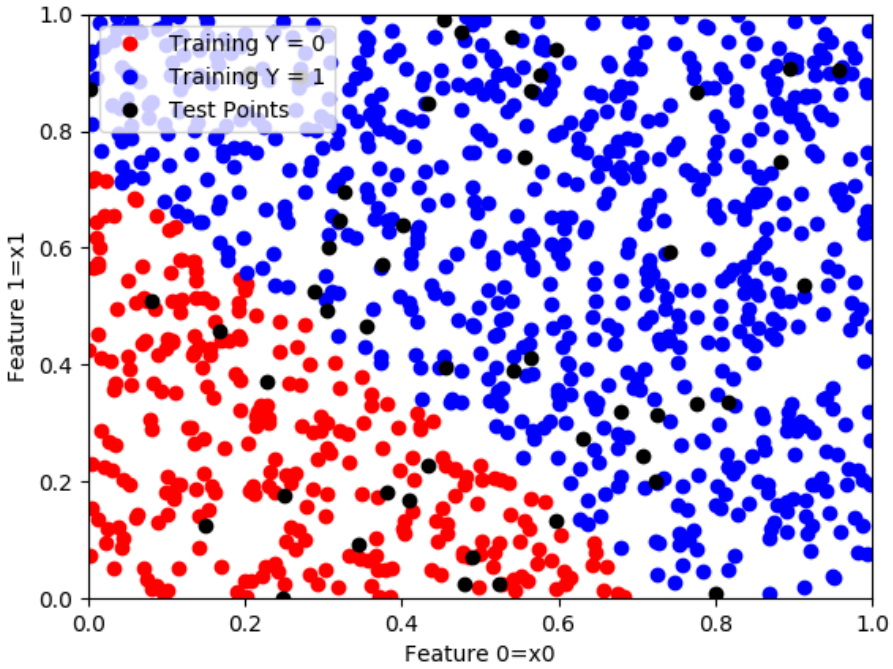
Logistic Regression – Binary Classification

Training Data:
- Input Information: points in (x0,x1) plane
- Output Information: label 0 (red) or 1 (blue) for each point

Logistic Regression Goal:
- Find function that best fits 0 and 1 labels in training data

Prediction:
- Using function, determine labels for new input test points (black points in picture)

Logistic Regression:
- Approach builds on that for Linear Regression
- Not suitable when boundary between 0 and 1 regions is not straight line



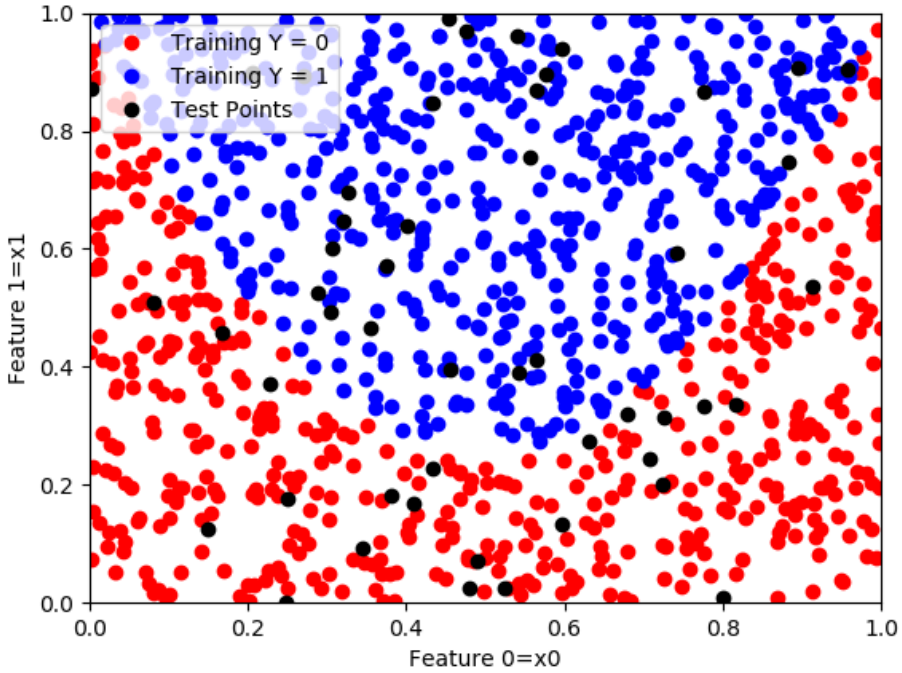
Neural Networks – Binary Classification

Training Data:
- Input Information: points in (x0,x1) plane
- Output Information: label 0 (red) or 1 (blue)

Neural Networks Goal:
- Find function that best fits 0 and 1 labels in training data

Prediction:
- Using function, determine labels for new input test points (black points in picture)

Neural Network
- Can be used in case of more complex boundary between 0 and 1 cases
- Can be used for classification with more than 2 classes



Neural Networks – Multi-class Classification

Training Data:

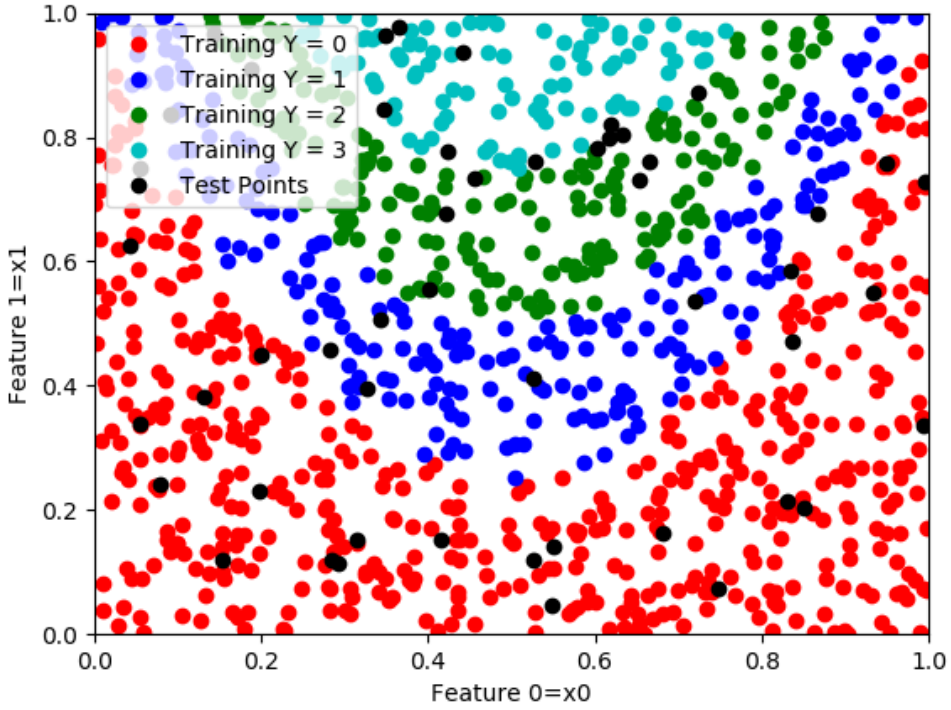
- Input Information: points in (x0,x1) plane
- Output Information: label 0 (red) or 1 (blue), 2 (green), 3 (cyan)

Neural Networks Goal:

- Find function that best fits 0,1,2,3 labels in training data

Prediction:

- Using function, determine labels for new input test points (black points in picture)



Approach for Supervised Learning

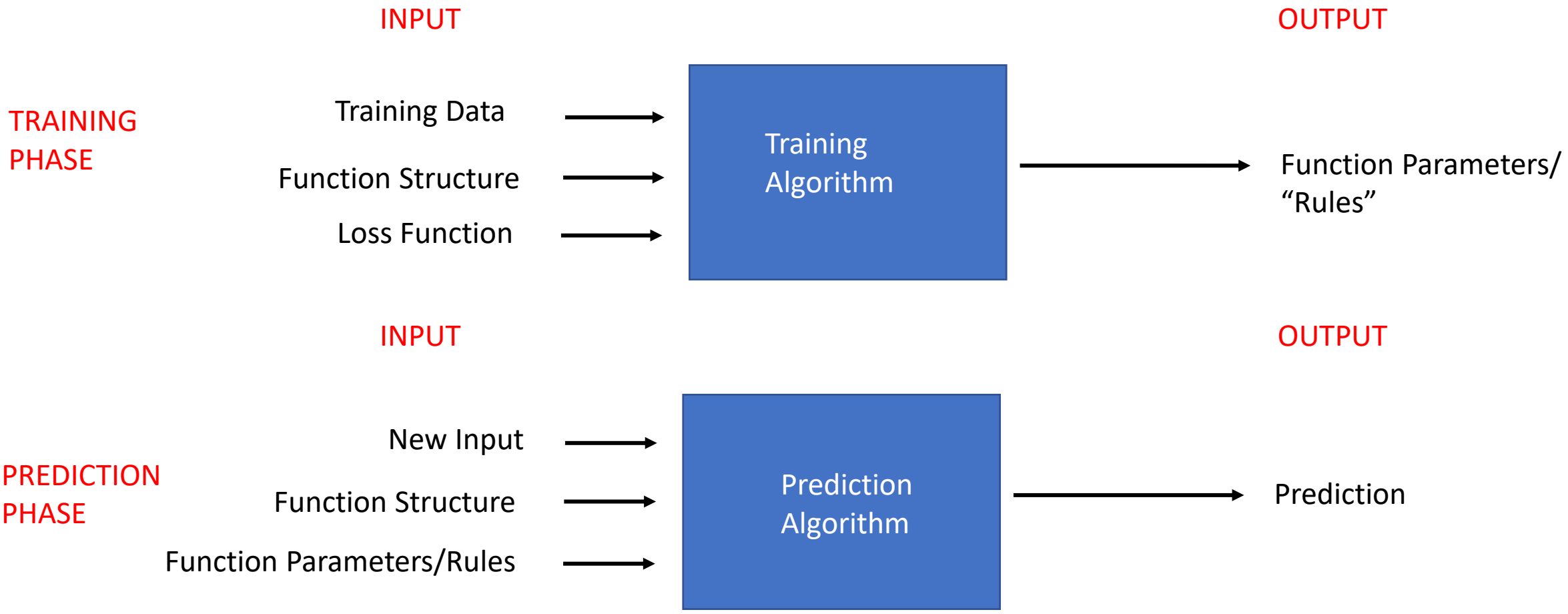
General approach for Supervised Learning:

- Training Data: Start with input/output information
- Function Structure: assume a functional form with unknown function parameters to map training input info to output info
- Define Loss Function: to measure how well function maps input info to output info
- Training Algorithm: find function parameters to minimize Loss function for training data
- Prediction Algorithm: use function structure and learned parameters to compute output info when new input info is provided

For Linear Regression (in 1 dimension):

- Training Data: points in X-Y plane
- Function Structure: assume Y = W*X + b (slope W, intercept b are “Function Parameters”/Rules)
- Loss function: squared error over training data
- Training Algorithm: find W and b to make Loss function as small as possible for training data
- Prediction Algorithm: use function structure and “learned” W, b to compute output Y when new input X values are provided

Supervised Learning: Training and Prediction



What this Course Covers

Course covers mathematics and programming for:

- Linear Regression, Logistic Regression and basic Neural Networks

Mathematics:

- Format of the training data
- Function structures
- Loss functions
- Training algorithms for determining function parameters
- Prediction algorithms

Programming:

- Course will walk students through development of class structures and codes (in Python language) for the function structures, training, and prediction algorithms
- Course will provide introduction to Tensorflow framework for employing Linear Regression, Logistic Regression, Neural Networks, and more advanced machine learning structures

1.2 About this Course

About this Course

This course provides an introduction to Supervised Machine Learning

Students completing this course will:

1. Gain an understanding of the mathematical foundations and algorithms for Linear Regression, Logistic Regression, and Neural Networks approaches for supervised machine learning
2. Gain an understanding of computer codes (written in Python) for Linear Regression, Logistic Regression, and Neural Networks. The course will walk through development of the codes from scratch.
3. Gain an understanding how to measure and improve performance of supervised machine learning systems
4. Gain a foundation for further study/practice of machine learning

Course Prerequisites

All courses have a starting point – an assumption about the base of students' knowledge. Courses cannot assume no knowledge and teach all the prerequisites. Students who don't have the prerequisite knowledge/skills listed below must acquire them before (or while) taking this course.

Prerequisites:

- Linear Algebra
  - Students should be familiar with vectors, matrices, transpose, matrix multiplication, least squares
- Multivariable Calculus
  - Students should be familiar with computing partial derivatives and employing the chain rule
- Python Programming
  - Students should be able to write and run Python 3 programs
  - It is preferable, but not required, that students be familiar with the NumPy package for scientific computing. Key features and functions will be explained in the course

Audience for this Course

This course is suitable for:

1. Students without any previous experience with machine learning
2. Students who have some knowledge of the subject and would like a refresher and/or gain a more detailed understanding of the mathematical foundations, algorithms, and implementation in Python

Course Approach

1. The underlying mathematics is explained in detail
   - The course will provide motivation and derivations for all the underlying mathematics
   - Examples provided in the presentations and Jupyter notebook demos
2. Algorithms are described in detail
   - The course will show how the underlying mathematics connects to the algorithms used in machine learning
3. Code development walkthrough is performed
   - This course walk through development full set of codes for Linear Regression, Logistic Regression, and Neural Networks
   - Walkthrough will emphasize how algorithms are translated into Python code and how a machine learning system can be created using object oriented approach

How to Get the Most from this Course

One cannot become proficient in a subject by simply watching someone else!

How to make the most from this course:

1. Take notes as you go through the material and work out the mathematical derivations by yourself
2. Ask questions on the forums
3. Do the programming

- Programming videos will start with objectives and high-level summary of the code structure, followed by video of code implementation
- You may want to just review the objectives and then design and implement the code by yourself.
- Alternatively, review objectives and high-level summaries, and then code for yourself. Can always check videos of code walkthroughs
- All codes are made available

Why Code from Scratch?

- There are many machine learning frameworks (Tensorflow, Keras, PyTorch, FastAI,scikit-learn, etc) that allow users to set up and solve machine learning problems in a few lines of code
- Ultimately, going forward for machine learning projects, research, and production development, you should use one of these frameworks, that has been tested and optimized. Some of these frameworks have been also developed for GPU (which are typically faster than CPU)
- My fundamental belief (and a reason for creating this course) is that to truly understand what is going on in these frameworks one must work through the mathematics and algorithms and write codes from scratch

About the Instructor

- Holds PhD in Applied Mathematics
- Worked for 10 years in university settings as post-doc and professor conducting research in applied math and numerical analysis and teaching undergraduate- and graduate-level courses
- Worked for 17+ years in financial risk management at a software start-up, financial information services company, and a large international bank

1.3 Software for the Course

Software Used in Course

- Instructor will use Windows 10 machine
  - Codes not specific to Windows – can use MacOS or Linux operating systems
- All codes examples written in Python
- Course will run programs using both
  - Jupyter Notebook
    - For demos
  - Command Window
    - For running main code base
- Should have text editor compatible with Python for writing and editing programs
  - Examples: Atom, Sublime, Notepad+, etc
  - Instructor will use Sublime, but you can use your favourite editor

Course Material

- Course codes located at:

https://github.com/satishchandrareddy/IntroML/

- Folder: Code
  - Contains working versions of the code
  - Series of folders(versions) mirror the lectures. Idea is to discuss underlying mathematics and algorithms and then do coding in manageable chunks as opposed to presenting all the theory and then do the coding.
  - Functionality will be added incrementally
- Folder: Examples
  - Jupyter notebook demos show code snippets that complement examples presented in the lectures
- Folder: Presentations
  - PDF files of presentations

Summary of Course Code Versions

Version	Description
Version 1.1	Initial functionality for Linear Regression and Derivative Testing
Version 1.2	Add Training and Prediction functionality for Linear Regression
Version 1.3	Add functionality for Logistic Regression
Version 2.1	Add functionality for Neural Networks for binary classification
Version 2.2	Add functionality for Neural Networks for multi-class classification
Version 3.1	Add functionality for Momentum, RmsProp, and Adam optimizers and mini-batch optimization
Version 3.2	Add functionality for validation and additional accuracy calculations
Version 3.3	Add functionality for hyperparameter searches and regularization
Version 4.1	Add functionality for house price, Spam, and MNIST case studies
Version 5.1	Add Tensorflow examples

Packages used in Course

Component	Version	Description/Comments
Python	3.7.1	It is assumed that students have a version of Python on their machine and have ability to run Jupyter notebooks. If you don't have Python on your machine, it is probably best to install Python using the Anaconda package, which aims to simplify package management. See: https://www.anaconda.com/ for download and installation of Anaconda. See: https://www.python.org for information about Python. I have found that Python 3.7.1 is consistent with the following package versions.
NumPy	1.18.1	Package for scientific computing. See https://numpy.org/ for details.
Matplotlib	3.0.3	Package for plotting. See https://matplotlib.org/ for details
pandas	0.23.4	Package containing data structures and data analysis tools – will use to load data from csv file. See https://pandas.pydata.org/
scikit-learn (sklearn)	0.20.1	Package for machine learning. Will use its text processing functions for spam classification. See https://scikit-learn.org/stable/index.html
tensorflow	1.13.1	Open source platform/framework for machine learning. See http://tensflow.org
copy, csv, time & unittest		These packages are part of the python release

Virtual Environment

- Should set up a “Virtual Environment” for this course
- Virtual environment will allow user to install specific versions of packages within environment without conflicting with different versions of packages used outside of that environment
  - Example: course uses Numpy version 1.18.1
  - You may have another version of Numpy on your machine for other purposes
  - Virtual environments will allow you to use both versions on your machine without conflicts
- There are many websites giving information on creating environments and installing packages. I have found the following to be useful:
  - https://janakiev.com/blog/jupyter-virtual-envs/ for details on setting up the virtual environment and connecting it to the Jupyter notebook
  - https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html for details on installing packages in conda environment

Virtual Environment and Package Installation

Task	Commands
Create virtual environment using python 3.7.1 -In the Anaconda Prompt window (NAME is user specified name of environment)	conda create -n NAME python=3.7.1
List all virtual environments -In the Anaconda Prompt window	conda env list
Activate virtual environment -In the Anaconda Prompt window (NAME is user specified name of environment)	conda activate NAME
Deactivate virtual environment -In the Anaconda prompt window	conda deactivate
Connect virtual environment to Jupyter Notebook -In the Anaconda Prompt window (NAME is user specified name of environment) (Use double dash - - before “user” and “name”)	pip install --user ipykernel python -m ipykernel install --user --name=NAME
Installing packages - In the Anaconda Prompt window (after activating virtual environment):	conda install numpy=1.18.1 conda install matplotlib=3.0.3 conda install pandas = 0.23.4 conda install scikit-learn=0.20.1 conda install tensorflow=1.13.1
List packages: - In Anaconda Prompt window (after activating environment)	conda list

Software for Course Demo

- Demo
  - Create a virtual environment
  - Install packages
  - Connect virtual environment to the Jupyter Notebook