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

Copyright Satish Reddy 2020

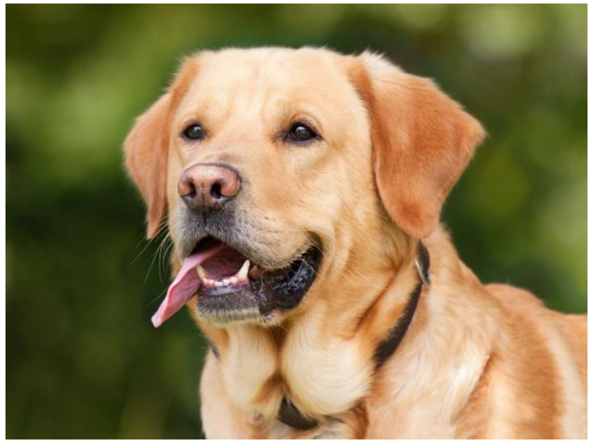
Machine Learning:

Copyright Satish Reddy 2020

1.1 What is Machine Learning?

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

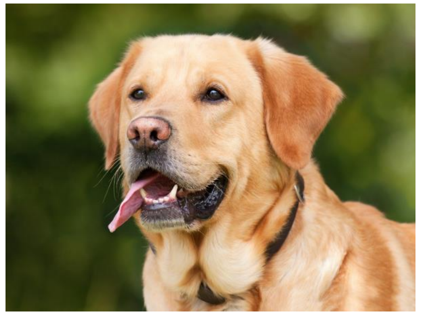
Machine Learning:

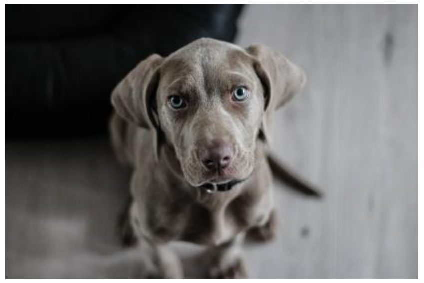
Three areas of application:

- Classification
- Clustering
- Game playing

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

Machine Learning: Clustering

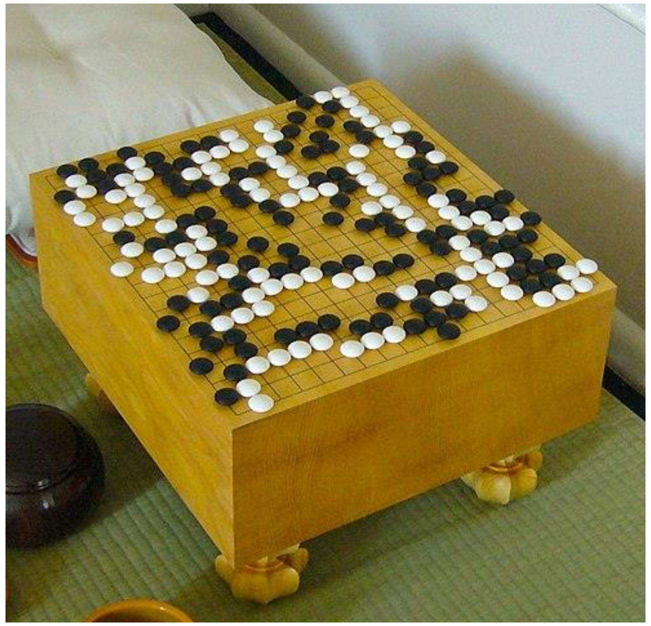
Pictures source: pixabay.com

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)

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

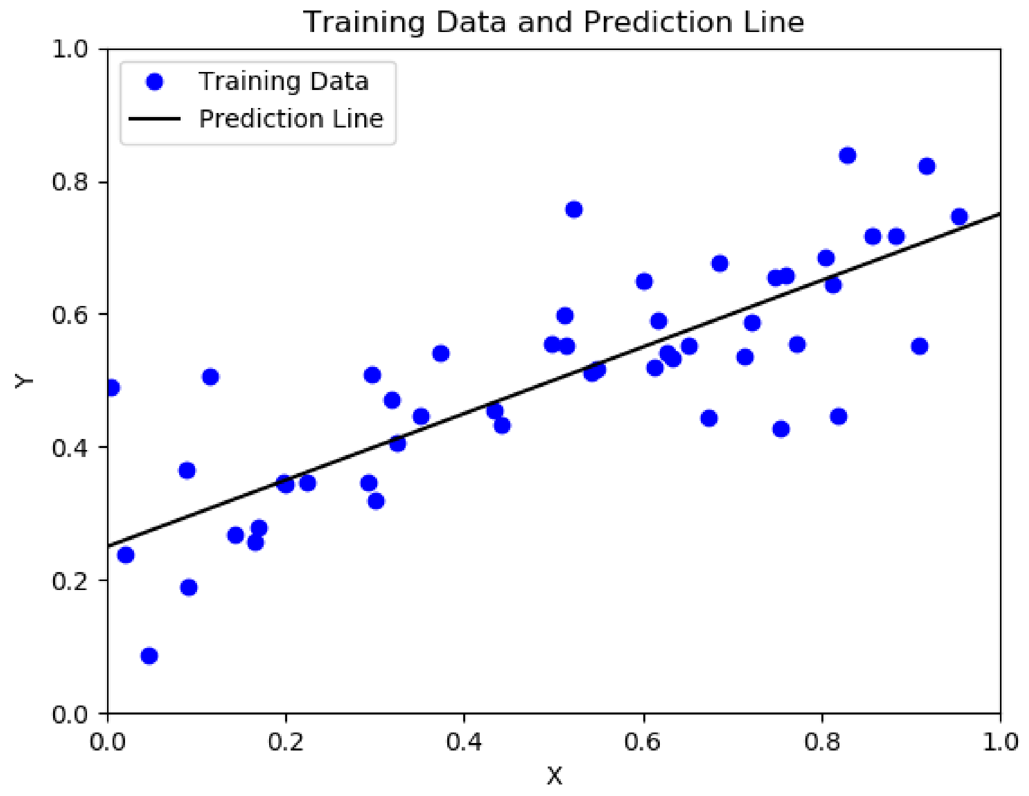
Linear Regression – Line Fitting

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



Copyright Satish Reddy 2020

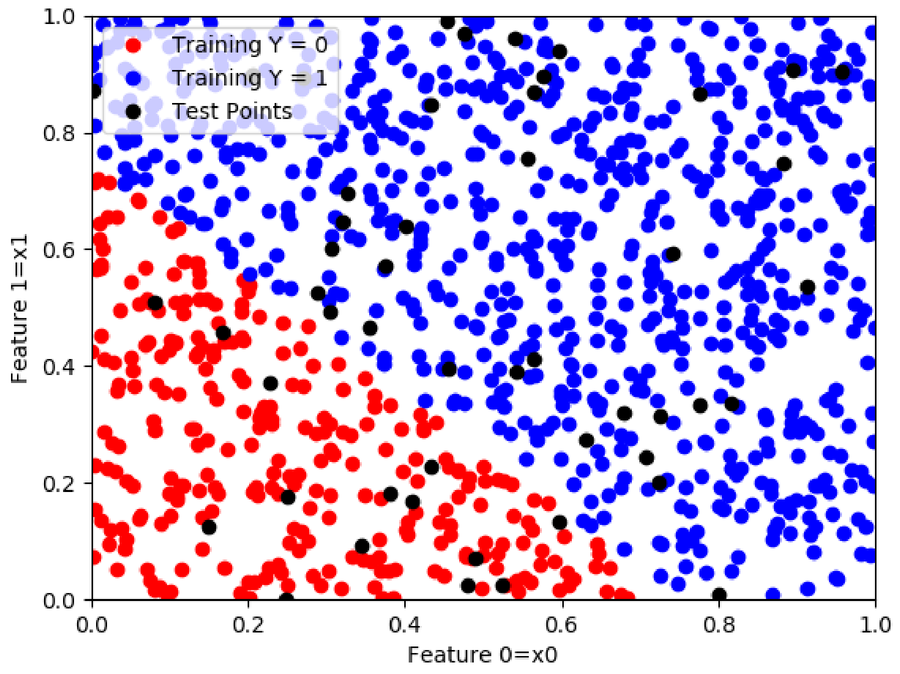
Logistic Regression – Binary Classification

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



Copyright Satish Reddy 2020

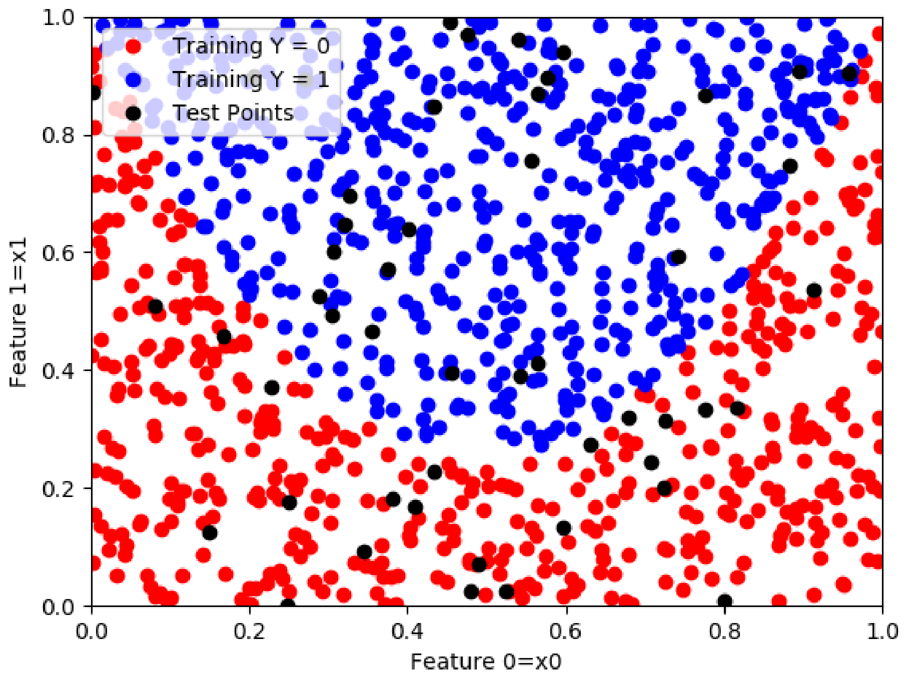
Neural Networks – Binary Classification

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



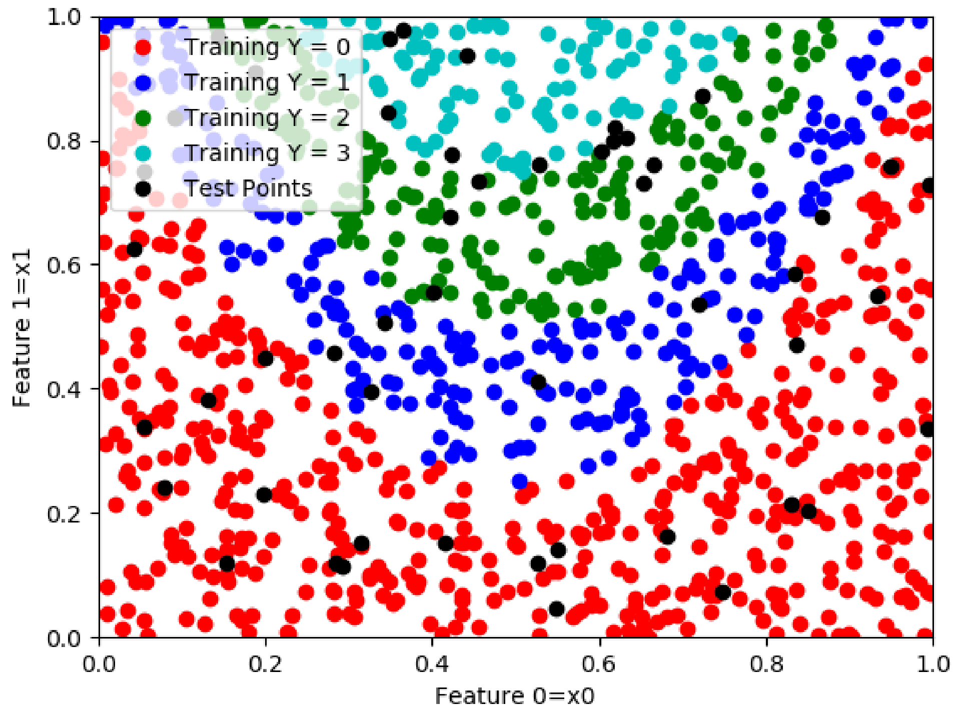
Copyright Satish Reddy 2020

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)



Copyright Satish Reddy 2020

Approach for Supervised Learning

General approach for Supervised Learning:

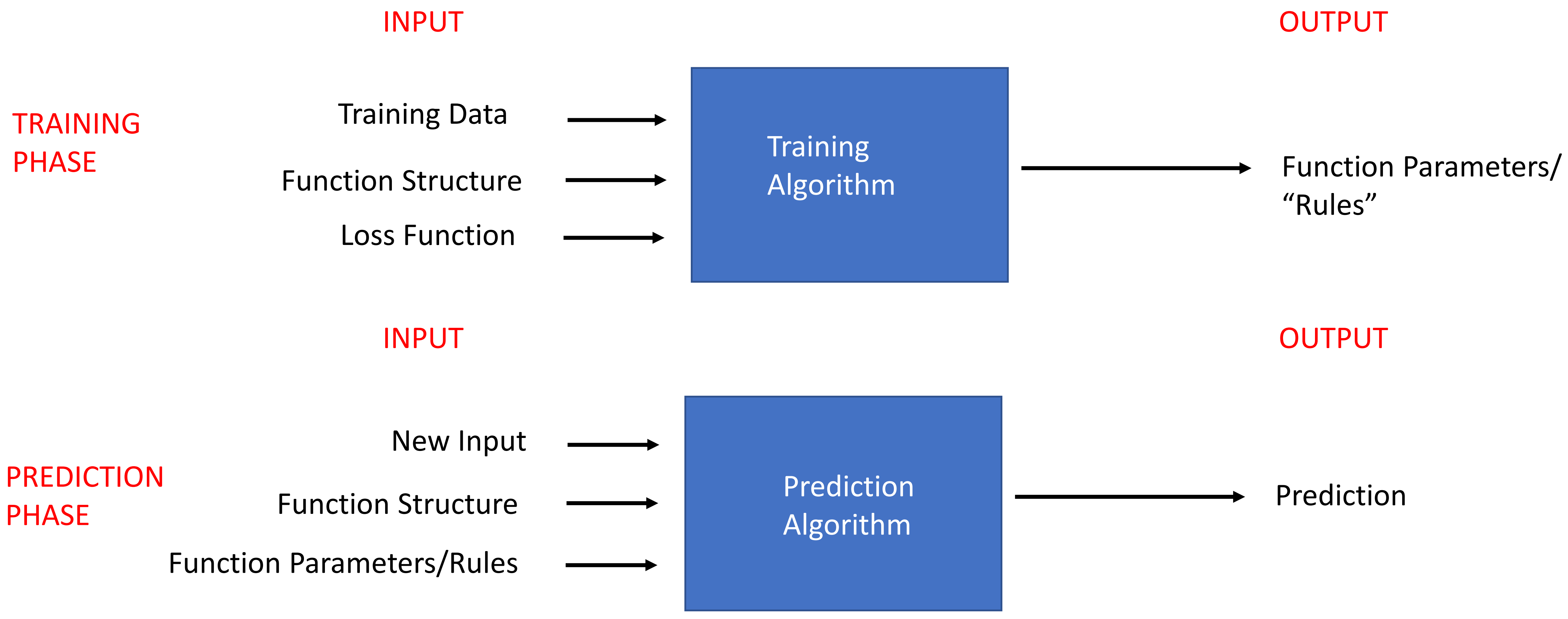
- Training Data: Start with input/output information
- Function Structure: assume a functional form with unknown function parameters to map training input info to output info
- Define Loss Function: to measure how well function maps input info to output info
- Training Algorithm: find function parameters 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

Copyright Satish Reddy 2020

Supervised Learning: Training and Prediction



Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

1.2 About this Course

Copyright Satish Reddy 2020

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.
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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

Course Approach

1. The underlying mathematics is explained in detail
   - The course will provide motivation and derivations for all the underlying mathematics
   - Examples will be provided in the presentations and in short code snippets
2. Algorithms are described in detail
   - The course will show how the underlying mathematics connects to the algorithms used in machine learning
3. Code walkthrough is performed
   - This course will provide walkthrough of a full set of codes for Linear Regression, Logistic Regression, and Neural Networks.
   - Walkthrough will emphasize how algorithms are translated into Python code

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

Why Code from Scratch?

- There are many machine learning frameworks (Tensorflow, Keras, PyTorch, FastAI, sklearn, etc) that allow users to set up and solve machine learning problems in a few lines of code
- Ultimately, going forward for your machine learning projects, research, and production development, you should use one of these frameworks, that have been tested, 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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

1.3 Software for the Course

Copyright Satish Reddy 2020

Software Used in Course

- Instructor will use Windows 10 machine
  - Codes not specific to Windows – can use MacOS or Linux
- All codes examples written in Python
- Course will run programs using both
  - Jupyter Notebook
    - Useful for showing short code snippets
  - Command Window
    - Useful for 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

Copyright Satish Reddy 2020

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
  - Contains short python code snippets that duplicate examples in presentations
  - These examples show how to convert “math” into “numpy” statements in python
- Folder: Presentations
  - PDF files of presentations

Copyright Satish Reddy 2020

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 Spam and MNIST case studies
Version 5.1	Add Tensorflow examples

Copyright Satish Reddy 2020

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. Need not specifically be 3.7.1, but best to have a recent version of Python 3. 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.
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. Note, you can use other plotting packages if you like, but this course will use Matplotlib
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 Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
tensorflow	1.13.1	Open source platform/framework for machine learning http://tensflow.org
unittest, csv, time		These packages are part of the python release

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

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

Copyright Satish Reddy 2020

Code for the Course – Demo

- Demo
  - Creating a virtual environment
  - Installing packages
  - Connecting environment to the Jupyter Notebook

Copyright Satish Reddy 2020

1.4 Introduction to Numpy Demo

Copyright Satish Reddy 2020

Introduction to NumPy

NumPy is a Python package for scientific computing

- Key object is multi-dimensional numpy array
- Numpy functions manipulate these arrays
  - Can perform standard matrix and vector operations
  - Can perform operations on entire array without explicit looping
- Course codes use numpy array as fundamental building block

Copyright Satish Reddy 2020

Key Numpy Commands and Functions

Operation	numpy functions
Array creation	numpy.array()
Array indexing	
Component-wise operations: addition, multiplication, scalar multiplication	+,*, * with scalar
Functions	numpy.exp(), numpy.absolute(), numpy.square()
Concatenation, shaping, removal, addition of axes	numpy.concatenate(), numpy.reshape(),numpy,squeeze(), numpy.expand_dims()
Sum entries of array	numpy.sum()
Array of zeros, array of ones	numpy.zeros(), numpy.ones()
Array of random numbers: setting seed, from uniform distribution, from normal distribution	numpy.random.seed(), numpy.random.rand(), numpy.random.randn()

Copyright Satish Reddy 2020

Introduction to Numpy Demo

- IntroML/Examples/Chapter1/Chapter1.4_NumpyExamples.ipynb
    - Jupyter Notebook showing key numpy functions to be used in this course

Copyright Satish Reddy 2020

1.5 Review of Mathematical Concepts: Linear Algebra

Copyright Satish Reddy 2020

Motivation

Supervised Learning: Process of learning a function that maps input information to labelled output information. The labelled input/output information is called the training data. The learned function is then used to predict outputs when new input information is provided.

Approach and underlying mathematics

- Assume “Function Structure” with unknown function parameters
- Function structures for Linear Regression, Logistic Regression, and Neural Networks are based on matrices and involve matrix multiplication and broadcasting
- Training Algorithm uses optimization to find function parameters that minimize loss function
- Optimization algorithms involve computation of gradients, which requires the chain rule and matrix multiplication with transposes

Copyright Satish Reddy 2020

Review of Mathematical Concepts

- Linear Algebra
  - Vectors, Matrices, and Transposes
  - Dot Product and Matrix Multiplication
  - Broadcasting
  - Multiplication and Transpose

Copyright Satish Reddy 2020

Linear Algebra – Vectors and Matrices

- Example of a row vector of length d (start index at 0)

$W = [W_0\, W_1\ \ldots W_{d-1}]$

Denote the i'th entry as W_i

- Example of a matrix with dimensions dxm (d rows and m columns) (indices are i=0,…,d-1, and j=0,…,m-1)

$$X = \begin{bmatrix} X_{00} & \cdots & X_{0,m-1} \\ \cdots & \cdots & \cdots \\ X_{d-1,0} & \cdots & X_{d-1,m-1} \end{bmatrix}$$

Denote the entry for row i and column j as X_{ij} (row index 1st, column index 2nd)

Copyright Satish Reddy 2020

Linear Algebra – Transpose

- Let W be a row vector of length d (dimensions 1xd)
- The transpose of W denoted W^T (dx1) is a column vector of length d

$$\begin{array}{c} W = [W_0\ W_1\ \ldots\ W_{d-1}] \\ \xrightarrow{\hspace{6em}} \\ \text{d entries} \end{array} \qquad W^T = \begin{bmatrix} W_0 \\ \ldots \\ W_{d-1} \end{bmatrix} \Big\downarrow \text{d entries}$$

- Let X be (dxm) matrix. Transpose (dimensions mxd) denoted by superscript T defined by: $X^T_{ij} = X_{ji}$

$$\begin{array}{c} X = \begin{bmatrix} X_{00} & \ldots & X_{0,m-1} \\ \ldots & \ldots & \ldots \\ X_{d-1,0} & \ldots & X_{d-1,m-1} \end{bmatrix} \Big\downarrow \text{d entries} \\ \xrightarrow{\hspace{10em}} \\ \text{m entries} \end{array} \qquad \begin{array}{c} X^T = \begin{bmatrix} X_{00} & \ldots & X_{d-1,0} \\ \ldots & \ldots & \ldots \\ X_{0,m-1} & \ldots & X_{d-1,m-1} \end{bmatrix} \Big\downarrow \text{m entries} \\ \xrightarrow{\hspace{10em}} \\ \text{d entries} \end{array}$$

Copyright Satish Reddy 2020

Linear Algebra – Transpose Example

$W = [1\ 2\ 3\ 4]$ $\quad W^T = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$

- Example 2:

$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$ $\quad X^T = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$

Copyright Satish Reddy 2020

Linear Algebra – Dot Product

- Let W and X both be row vectors of length d

$W = [W_0\ W_1\ \ldots W_{d-1}]$ and $X = [X_0\ X_1\ \ldots X_{d-1}]$

- Dot product of W and X given by:

$$Z = W_o X_o + W_1 X_1 + W_2 X_2 + \cdots + W_{d-1} X_{d-1} = \sum_{i=0}^{d-1} W_i X_i$$

- Can express this as a matrix multiplication (which is built upon dot product of row of first object and column of second)

$$Z = WX^T = [W_0\ W_1\ \ldots W_{d-1}] \begin{bmatrix} X_0 \\ X_1 \\ \ldots \\ X_{d-1} \end{bmatrix}$$

Copyright Satish Reddy 2020

Linear Algebra – Dot Product Example

- Example

$W = [1\ 2\ 3\ 4] \qquad X = [5\ 6\ 7\ 8]$

$Z = WX^T = [1\ 2\ 3\ 4]\begin{bmatrix}5\\6\\7\\8\end{bmatrix}$=(1)(5) + (2)(6) + (3)(7)+(4)(8)=70

Copyright Satish Reddy 2020

Linear Algebra – Vector/Matrix Multiplication

- Let W be row vector of length d, X be a matrix of dimension (dxm)

$$W = [W_0\ W_1\ \ \dots W_{d-1}] \qquad X = \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix}$$

- Define Z_j to be the dot product of W and the j’th column of X

$Z_j = W_o X_{oj} + W_1 X_{1j} + W_2 X_{2j} + \cdots + W_{d-1} X_{d-1,j} = \sum_{i=0}^{d-1} W_i X_{i,j}$ for j=0,…,m-1

- Z is a row vector of length m

$$[Z_0\ Z_1\ \ \dots Z_{m-1}] = [W_0\ W_1\ \ \dots W_{d-1}] \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix}$$

or

$Z = WX$

Copyright Satish Reddy 2020

Linear Algebra – Matrix/Matrix Multiplication

- Let W be a matrix of dimension (nxd), X be a matrix of dimension (dxm)

- W= $\begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix}$ $X = \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix}$

- Define Z_{ij} to be the dot product of row i of W and column j of X

$Z_{ij} = W_{i0}X_{0j} + W_{i1}X_{1j} + W_{i2}X_{2j} + \cdots + W_{i,d-1}X_{d-1,j} = \sum_{k=0}^{d-1} W_{ik}X_{kj}$ i=0,…,n-1, j=0,…,m-1

- Z is an nxm matrix

$$\begin{bmatrix} Z_{00} & \dots & Z_{0,m-1} \\ \dots & \dots & \dots \\ Z_{n-1,0} & \dots & Z_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix} \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix}$$

or

Z = WX

Copyright Satish Reddy 2020

Linear Algebra – Multiplication Example 1

$W = [1 \;\; 2 \;\; 3] \qquad X = \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}$

$$Z = WX = [1\;2\;3] \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}$$

$$Z = [(1)(4) + (2)(5) + (3)(6) \quad (1)(7) + (2)(8) + (3)(9)]$$

$$Z = [32 \quad 50]$$

Copyright Satish Reddy 2020

Linear Algebra – Multiplication Example 2

$$W = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}$$

$$Z = WX = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix}$$

$$Z = \begin{bmatrix} (1)(4) + (2)(5) + (3)(6) & (1)(7) + (2)(8) + (3)(9) \\ (2)(4) + (3)(5) + (4)(6) & (2)(7) + (3)(8) + (4)(9) \end{bmatrix}$$

$$Z = \begin{bmatrix} 32 & 50 \\ 47 & 74 \end{bmatrix}$$

Copyright Satish Reddy 2020

Linear Algebra – Broadcasting

- Let W be row vector of length d, X be a matrix of dimension (dxm), and b a scalar

$$W = [W_0\ W_1\ \ldots W_{d-1}] \qquad X = \begin{bmatrix} X_{00} & \cdots & X_{0,m-1} \\ \cdots & \cdots & \cdots \\ X_{d-1,0} & \cdots & X_{d-1,m-1} \end{bmatrix}$$

- Define Z_j to be dot product of W and column j of X plus b

$Z_j = W_o X_{oj} + W_1 X_{1j} + W_2 X_{2j} + \cdots + W_{d-1} X_{d-1,j} + b = \sum_{i=0}^{d-1} W_i X_{ij} + b$ for j=0,…,m-1

- Z is a row vector of length m

$$[Z_0\ Z_1\ \ldots Z_{m-1}] = [W_0\ W_1\ \ldots W_{d-1}] \begin{bmatrix} X_{00} & \cdots & X_{0,m-1} \\ \cdots & \cdots & \cdots \\ X_{d-1,0} & \cdots & X_{d-1,m-1} \end{bmatrix} + b$$

or $Z = WX + b$

- b is a scalar, so it does not have the same dimensions as Z. It is added for each element of Z. This is an example of broadcasting (this is term used in numpy documentation)

Copyright Satish Reddy 2020

Linear Algebra – Broadcasting Example 1

$W = [1 \;\; 2 \;\; 3] \qquad X = \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} \qquad b = 7$

$$Z = WX + b = [1\;2\;3]\begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} + 7$$

$$Z = [(1)(4) + (2)(5) + (3)(6) + 7 \quad (1)(7) + (2)(8) + (3)(9) + 7]$$

$$Z = [39 \quad 57]$$

Copyright Satish Reddy 2020

Linear Algebra – Broadcasting

- Let W be a matrix of dimension (nxd), X be a matrix of dimension (dxm), b be a column vector of length n

$$W= \begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix} \quad X = \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix} \quad b = \begin{bmatrix} b_0 \\ \dots \\ b_{n-1} \end{bmatrix}$$

- Define Z_{ij} to be the dot product of row i of W and column j of X plus i'th entry of b

$$Z_{ij} = W_{i0}X_{0j} + W_{i1}X_{1j} + W_{i2}X_{2j} + \cdots + W_{i,d-1}X_{d-1,j} + b_i = \sum_{k=0}^{d-1} W_{ik}X_{kj} + b_i \qquad \text{i=0,…,n-1, j=0,…,m-1}$$

- Z is an nxm matrix

$$\begin{bmatrix} Z_{00} & \dots & Z_{0,m-1} \\ \dots & \dots & \dots \\ Z_{n-1,0} & \dots & Z_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix} \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ \dots \\ b_{n-1} \end{bmatrix}$$

or

Z=WX+b

- Example of Broadcasting - entry i of b is added to each entry of row i of Z

Copyright Satish Reddy 2020

Linear Algebra – Broadcasting Example 2

$$W = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} \quad b = \begin{bmatrix} 11 \\ 12 \end{bmatrix}$$

$$Z = WX + b = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 4 & 7 \\ 5 & 8 \\ 6 & 9 \end{bmatrix} + \begin{bmatrix} 11 \\ 12 \end{bmatrix}$$

$$Z = \begin{bmatrix} (1)(4) + (2)(5) + (3)(6) + 11 & (1)(7) + (2)(8) + (3)(9) + 11 \\ (2)(4) + (3)(5) + (4)(6) + 12 & (2)(7) + (3)(8) + (4)(9) + 12 \end{bmatrix}$$

$$Z = \begin{bmatrix} 43 & 61 \\ 59 & 86 \end{bmatrix}$$

Copyright Satish Reddy 2020

Linear Algebra – Multiplication and Transpose

- Consider:

$Z_{ij} = \sum_{k=0}^{d-1} W_{ik} X_{jk}$ (dot product of row i of W and row j of X)

- Rewrite as:

$Z_{ij} = \sum_{k=0}^{d-1} W_{ik} X_{kj}^{T}$ (dot product of row i of W and column j of X^T)

- Can express this as a matrix multiplication: $Z = WX^T$
- Consider:

$Z_{ij} = \sum_{k=0}^{d-1} W_{ki} X_{kj}$ (dot product of column i of W and column j of X)

- Rewrite as:

$Z_{ij} = \sum_{k=0}^{d-1} W_{ik}^{T} X_{kj}$ (dot product of row i of W^T and column j of X)

- Can express this as matrix multiplication: $Z = W^T X$

Copyright Satish Reddy 2020

Linear Algebra – Jupyter Notebook Demo

- Open file IntroML/Examples/Chapter1/Chapter1.5_LinearAlgebraExamples.ipynb
- Has examples of
  - Transpose
  - Matrix Multiplication
  - Matrix Multiplication with Broadcasting

Copyright Satish Reddy 2020

1.6 Review of Mathematical Concepts: Multivariable Calculus

Copyright Satish Reddy 2020

Motivation

Supervised Learning: Process of learning a function that maps input information to labelled output information. The labelled input/output information is called the training data. The learned function is then used to predict outputs when new input information is provided.

Approach and underlying mathematics

- Assume “Function Structure” with unknown function parameters
- Function structures for Linear Regression, Logistic Regression, and Neural Networks are based on matrices and involve matrix multiplication and broadcasting
- Training Algorithm uses optimization to find function parameters that minimize loss function
- Optimization algorithms involve computation of gradients, which requires the chain rule and matrix multiplication with transposes

Copyright Satish Reddy 2020

Review of Mathematical Concepts

- Multivariable Calculus
  - Partial Derivatives
  - Gradients
  - Chain Rule

Copyright Satish Reddy 2020

Multivariable Calculus – Partial Derivatives

- Consider a function of m variables L = L(Z_0,Z_1,Z_2 ,…,Z_{m-1})
- Partial derivative of L with respect to Z_j measures rate of change of L if only Z_j changes. By definition this partial derivative is:

$$\frac{\partial L}{\partial Z_j} = \lim_{\varepsilon \to 0} \frac{L\left(Z_0, Z_1, \dots, Z_j + \varepsilon, \dots, Z_{m-1}\right) - L(Z_0, Z_1, \dots, Z_j, \dots, Z_{m-1})}{\varepsilon}$$

Copyright Satish Reddy 2020

Multivariable Calculus – Gradient

- Consider a function of m variables L = L(Z_0,Z_1,Z_2 ,…, Z_{m-1})
- The gradient is vector of length m defined by:

$$\nabla_Z L = \left[\frac{\partial L}{\partial Z_0} \quad \frac{\partial L}{\partial Z_1} \cdots \quad \frac{\partial L}{\partial Z_{m-1}}\right]$$

- Individual entries of gradient vector denoted by:

$$\nabla_Z L_j = \frac{\partial L}{\partial Z_j}, \quad \mathrm{j} = 0, \ldots, \mathrm{m} - 1$$

Copyright Satish Reddy 2020

Multivariable Calculus - Gradient Notation

Generalize the gradient notation to scalars and matrices

- If L is a function of a single variable: L = L(b). Gradient defined as:

$$\nabla_b L = \left[\frac{\partial L}{\partial b}\right]$$

- If L is function of nxm variables in matrix Z (Z_{ij} for i=0,…,n-1,j=0,…,m-1), $\nabla_Z L$ is a matrix with same dimensions as Z

$$\nabla_Z L = \begin{bmatrix} \frac{\partial L}{\partial Z_{00}} & \cdots & \frac{\partial L}{\partial Z_{0,m-1}} \\ \cdots & \cdots & \cdots \\ \frac{\partial L}{\partial Z_{n-1,0}} & \cdots & \frac{\partial L}{\partial Z_{n-1,m-1}} \end{bmatrix}$$

$$\nabla_Z L_{ij} = \frac{\partial L}{\partial Z_{ij}} \quad i = 0, \dots, n-1, j = 0, \dots, m-1$$

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- Consider a function of m variables L = L(Z_0,Z_1,Z_2 ,…, Z_{m-1}).

- Suppose: $Z_j = W_o X_{oj} + W_1 X_{1j} + W_2 X_{2j} + \cdots + W_{d-1} X_{d-1,j}$ for j=0,…,m-1
- In matrix terms:

$$[Z_0\ Z_1\ \ \ldots Z_{m-1}] = [W_0\ W_1\ \ \ldots W_{d-1}] \begin{bmatrix} X_{00} & \ldots & X_{0,m-1} \\ \ldots & \ldots & \ldots \\ X_{d-1,0} & \ldots & X_{d-1,m-1} \end{bmatrix}$$

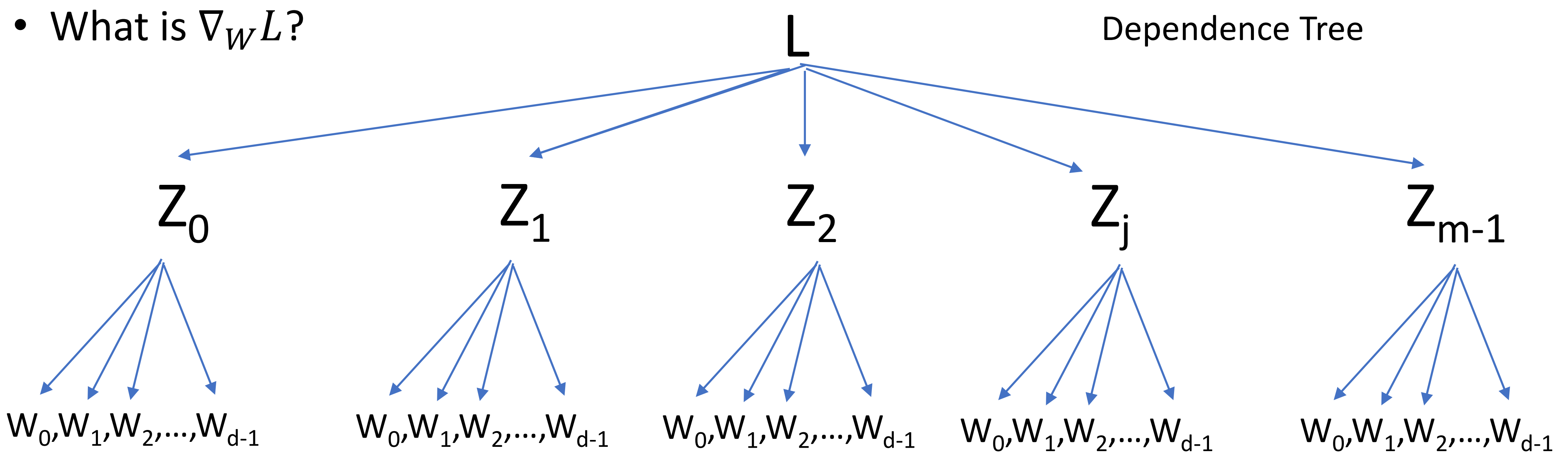
Z=WX

- Question: what is $\nabla_W L$?

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- What is $\nabla_W L$?


Dependence Tree
L
Z_0
Z_1
Z_2
Z_j
Z_{m-1}
$W_0, W_1, W_2, \ldots, W_{d-1}$
$W_0, W_1, W_2, \ldots, W_{d-1}$
$W_0, W_1, W_2, \ldots, W_{d-1}$
$W_0, W_1, W_2, \ldots, W_{d-1}$
$W_0, W_1, W_2, \ldots, W_{d-1}$

Applying the chain rule of multivariable calculus:

(1) To determine partial derivative of L with respect to W_i, find all paths from L to W_i

(2) Take partial derivatives along each path and multiply

(3) Sum over all the paths

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

$$\begin{tikzcd}
 & & L \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
Z_0 \arrow[d] & Z_1 \arrow[d] & Z_2 \arrow[d] & Z_j \arrow[d] & Z_{m-1} \arrow[d] \\
W_0,W_1,W_2,\dots,W_i,\dots W_{d-1} & W_0,W_1,W_2,\dots,W_i,\dots W_{d-1} & W_0,W_1,W_2,\dots,W_i,\dots W_{d-1} & W_0,W_1,W_2,\dots,W_i,\dots W_{d-1} & W_0,W_1,W_2,\dots,W_i,\dots W_{d-1}
\end{tikzcd}$$

1. For $\frac{\partial L}{\partial W_i}$ there are m paths: L –> Z_j –> W_i j=0,…,m-1:
2. Product of derivatives on each path: $\frac{\partial L}{\partial Z_j}\frac{\partial Z_j}{\partial W_i}$
3. Sum over all paths: $\frac{\partial L}{\partial W_i} = \sum_{j=0}^{m-1}\frac{\partial L}{\partial Z_j}\frac{\partial Z_j}{\partial W_i}$ (for i=0,…,d-1)

- By definition: $\frac{\partial L}{\partial Z_j} = \nabla_Z L_j$
- From the last pages: $Z_j = W_o X_{oj} + W_1 X_{1j} + W_2 X_{2j} + \cdots + W_{d-1} X_{d-1,j}$ so $\frac{\partial Z_j}{\partial W_i} = X_{ij}$

$\frac{\partial L}{\partial W_i} = \sum_{j=0}^{m-1} \nabla_Z L_j X_{ij} = \sum_{j=0}^{m-1} \nabla_Z L_j X_{ji}^T$

In matrix form: $\nabla_W L = \nabla_Z L X^T$ dimensions: 1xd = (1xm) times (mxd)

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- Consider more general case: L = L(Z)
- Z is an (nxm) matrix, so L is a function of nxm variables
- Z defined by Z = WX + b: W is nxd, X is dxm and b is nx1

$$\begin{bmatrix} Z_{00} & \dots & Z_{0,m-1} \\ \dots & \dots & \dots \\ Z_{n-1,0} & \dots & Z_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix} \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ \dots \\ b_{n-1} \end{bmatrix}$$

- What are: $\nabla_W L$, $\nabla_b L$, $\nabla_X L$?
- Note: these gradients come up in machine learning training algorithms

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- For $\nabla_W L_{ij} = \frac{\partial L}{\partial W_{ij}}$ (i=0,…,n-1, j=0,…,d-1)

Z=WX+b $\begin{bmatrix} \cdots & \cdots & \cdots \\ Z_{i0} & \cdots & Z_{i,m-1} \\ \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & W_{ij} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \cdots & \cdots & \cdots \\ X_{j0} & \cdots & X_{j,m-1} \\ \cdots & \cdots & \cdots \end{bmatrix} + \begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix}$

$$\begin{tikzcd}
 & & L \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
Z_{i0} \arrow[d] & Z_{i1} \arrow[d] & Z_{i2} \arrow[d] & Z_{ik} \arrow[d] & Z_{i,m-1} \arrow[d] \\
\ldots,\ldots,\ldots,\ldots,W_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,W_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,W_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,W_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,W_{ij},\ldots
\end{tikzcd}$$

- Applying chain rule: $\nabla_W L_{ij} = \sum_{k=0}^{m-1} \frac{\partial L}{\partial Z_{ik}} \frac{\partial Z_{ik}}{\partial W_{ij}} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} X_{jk} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} X_{kj}^T$
- This is matrix/matrix multiplication: $\nabla_W L_{ij} = (\nabla_Z L X^T)_{ij}$
- Hence: $\nabla_W L = \nabla_Z L X^T$

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- For $\nabla_b L_i = \frac{\partial L}{\partial b_i}$ (i=0,…,n-1)

Z=WX+b $\begin{bmatrix} \cdots & \cdots & \cdots \\ Z_{i0} & \cdots & Z_{i,m-1} \\ \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} + \begin{bmatrix} \cdots \\ b_i \\ \cdots \end{bmatrix}$

$$\begin{tikzcd}
 & & L \arrow[lld] \arrow[ld] \arrow[d] \arrow[rd] \arrow[rrd] & & \\
Z_{i0} \arrow[d] & Z_{i1} \arrow[d] & Z_{i2} \arrow[d] & Z_{ik} \arrow[d] & Z_{i,m-1} \arrow[d] \\
\ldots,\ldots,\ldots,\ldots,b_i,\ldots & \ldots,\ldots,\ldots,\ldots,b_i,\ldots & \ldots,\ldots,\ldots,\ldots,b_i,\ldots & \ldots,\ldots,\ldots,\ldots,b_i,\ldots & \ldots,\ldots,\ldots,\ldots,b_i,\ldots
\end{tikzcd}$$

- Applying chain rule: $\nabla_b L_i = \sum_{k=0}^{m-1} \frac{\partial L}{\partial Z_{ik}} \frac{\partial Z_{ik}}{\partial b_i} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} * 1 = \sum_{k=0}^{m-1} \nabla_Z L_{ik}$
- This is simply a sum along the i'th row of the gradient $\nabla_Z L$

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule

- For $\nabla_X L_{ij} = \frac{\partial L}{\partial X_{ij}}$ (i=0,…,d-1, j=0,…,m-1)

Z=WX+b $\begin{bmatrix} \ldots & Z_{0j} & \ldots \\ \ldots & \ldots & \ldots \\ \ldots & Z_{n-1,j} & \ldots \end{bmatrix} = \begin{bmatrix} \ldots & W_{0i} & \ldots \\ \ldots & \ldots & \ldots \\ \ldots & W_{n-1,i} & \ldots \end{bmatrix} \begin{bmatrix} \ldots & \ldots & \ldots \\ \ldots & X_{ij} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix} + \begin{bmatrix} \ldots \\ \ldots \\ \ldots \end{bmatrix}$

$$\begin{tikzcd}
 & & L \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
Z_{0j} \arrow[d] & Z_{1j} \arrow[d] & Z_{2j} \arrow[d] & Z_{kj} \arrow[d] & Z_{n-1,j} \arrow[d] \\
\ldots,\ldots,\ldots,\ldots,X_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,X_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,X_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,X_{ij},\ldots & \ldots,\ldots,\ldots,\ldots,X_{ij},\ldots
\end{tikzcd}$$

- Applying chain rule: $\nabla_X L_{ij} = \sum_{k=0}^{n-1} \frac{\partial L}{\partial Z_{kj}} \frac{\partial Z_{kj}}{\partial X_{ij}} = \sum_{k=0}^{n-1} \nabla_Z L_{kj} W_{ki} = \sum_{k=0}^{n-1} W_{ik}^T \nabla_Z L_{kj}$
- This is matrix/matrix multiplication: $\nabla_X L_{ij} = (W^T \nabla_Z L)_{ij}$
- Hence: $\nabla_X L = W^T \nabla_Z L$

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule Example

- Consider:

$L = 2Z_0 + Z_1$

- Gradient is

$\nabla_Z L = \begin{bmatrix} \frac{\partial L}{\partial Z_0} & \frac{\partial L}{\partial Z_1} \end{bmatrix} = [2 \;\; 1]$

- Now assume:

$Z = WX + b \quad \text{and } W = [W_0 \quad W_1], X = \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix}$

so

$Z = [Z_0 \;\; Z_1] = [W_0X_{00} + W_1X_{10} + b \quad W_0X_{01} + W_1X_{11} + b]$

- Substituting the expressions for Z_0 and Z_1 into L, we get:

$L = 2W_0X_{00} + 2W_1X_{10} + 2b + W_0X_{01} + W_1X_{11} + b$

Copyright Satish Reddy 2020

Multivariable Calculus – Chain Rule Example

Start with:

$L = 2W_0X_{00} + 2W_1X_{10} + 2b + W_0X_{01} + W_1X_{11} + b \quad \nabla_Z L = \begin{bmatrix}\frac{\partial L}{\partial Z_0} & \frac{\partial L}{\partial Z_1}\end{bmatrix} = \begin{bmatrix}2 & 1\end{bmatrix}$

Direct Calculation: $\nabla_W L = \begin{bmatrix}\frac{\partial L}{\partial W_0} & \frac{\partial L}{\partial W_1}\end{bmatrix} = \begin{bmatrix}2X_{00} + X_{01} & 2X_{10} + X_{11}\end{bmatrix}$

Chain Rule: $\nabla_W L = \nabla_Z L X^T = \begin{bmatrix}2 & 1\end{bmatrix}\begin{bmatrix}X_{00} & X_{10} \\ X_{01} & X_{11}\end{bmatrix} = \begin{bmatrix}2X_{00} + X_{01} & 2X_{10} + X_{11}\end{bmatrix}$

Direct Calculation: $\nabla_X L = \begin{bmatrix}\frac{\partial L}{\partial X_{00}} & \frac{\partial L}{\partial X_{01}} \\ \frac{\partial L}{\partial X_{10}} & \frac{\partial L}{\partial X_{11}}\end{bmatrix} = \begin{bmatrix}2W_0 & W_0 \\ 2W_1 & W_1\end{bmatrix}$

Chain Rule: $\nabla_X L = W^T \nabla_Z L = \begin{bmatrix}W_0 \\ W_1\end{bmatrix}\begin{bmatrix}2 & 1\end{bmatrix} = \begin{bmatrix}2W_0 & W_0 \\ 2W_1 & W_1\end{bmatrix}$

Direct Calculation: $\nabla_b L = \begin{bmatrix}\frac{\partial L}{\partial b}\end{bmatrix} = \begin{bmatrix}3\end{bmatrix}$

Chain Rule: $\nabla_b L = \sum_{j=0}^{m-1} \nabla_Z L_j = \begin{bmatrix}3\end{bmatrix}$

Copyright Satish Reddy 2020

Gradient Formulas Summary

Component		Description
Function	Z is nxm matrix W is nxd matrix X is dxm matrix b is nx1 matrix	$L = f(Z)$ $Z = WX + b$
Gradients		$\nabla_W L = \nabla_Z L X^T$ $\nabla_b L_i = \sum_{j=0}^{m-1} \nabla_Z L_{ij}$ (sum along i'th row of gradient) $\nabla_X L = W^T \nabla_Z L$

Copyright Satish Reddy 2020

1.7 Review of Mathematical Concepts: Optimization

Copyright Satish Reddy 2020

Motivation

Supervised Learning: Process of learning a function that maps input information to labelled output information. The labelled input/output information is called the training data. The learned function is then used to predict outputs when new input information is provided.

Approach and underlying mathematics

- Assume "Function Structure" with unknown function parameters
- Function structures for Linear Regression, Logistic Regression, and Neural Networks are based on matrices and involve matrix multiplication and broadcasting
- Training Algorithm uses optimization to find function parameters that minimize loss function
- Optimization algorithms involve computation of gradients, which requires the chain rule and matrix multiplication with transposes

Copyright Satish Reddy 2020

Review of Mathematical Concepts

- Optimization
  - Gradient Descent for Minimizing a Function

Copyright Satish Reddy 2020

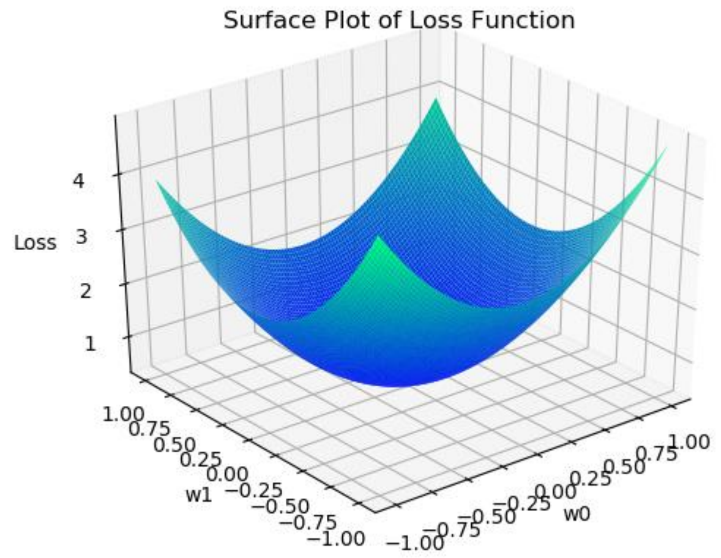
Optimization

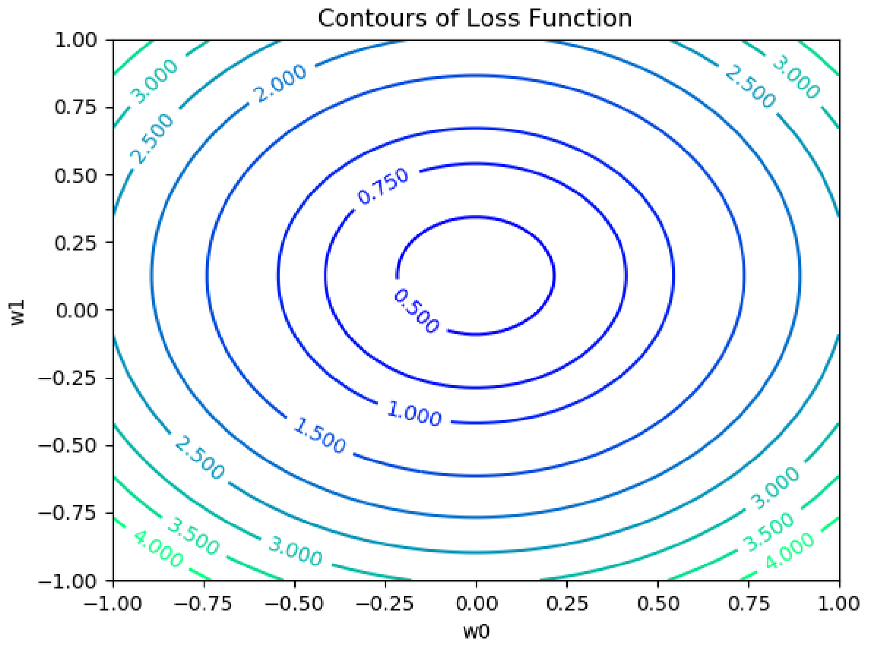
- Let $L(W_0, W_1, W_2, \ldots, W_{d-1})$ denote a Loss function of d parameters
- Machine Learning Training Algorithm involves finding the parameters $W_0, W_1, W_2, \ldots, W_{d-1}$ that minimize Loss function
- From multivariable calculus, gradient $\nabla_W L = 0$ at minimum
- $\nabla_W L$ has d components, so setting gradient to 0 involves solving d equations for d unknowns
- In general it is not feasible to solve these equations exactly
- Optimization is a process of finding the minimum (or getting close to minimum) using an iterative process making use of the gradient

Copyright Satish Reddy 2020

Optimization - Surface and Contour Plots

- Use example of Loss function of 2 variables Loss = L(W_0,W_1) to illustrate Gradient Descent optimization algorithm
- Surface plot: Loss function (3 dimensions)
- Contour plot: lines of constant Loss in (W_0,W_1) plane (2 dimensions)





Copyright Satish Reddy 2020

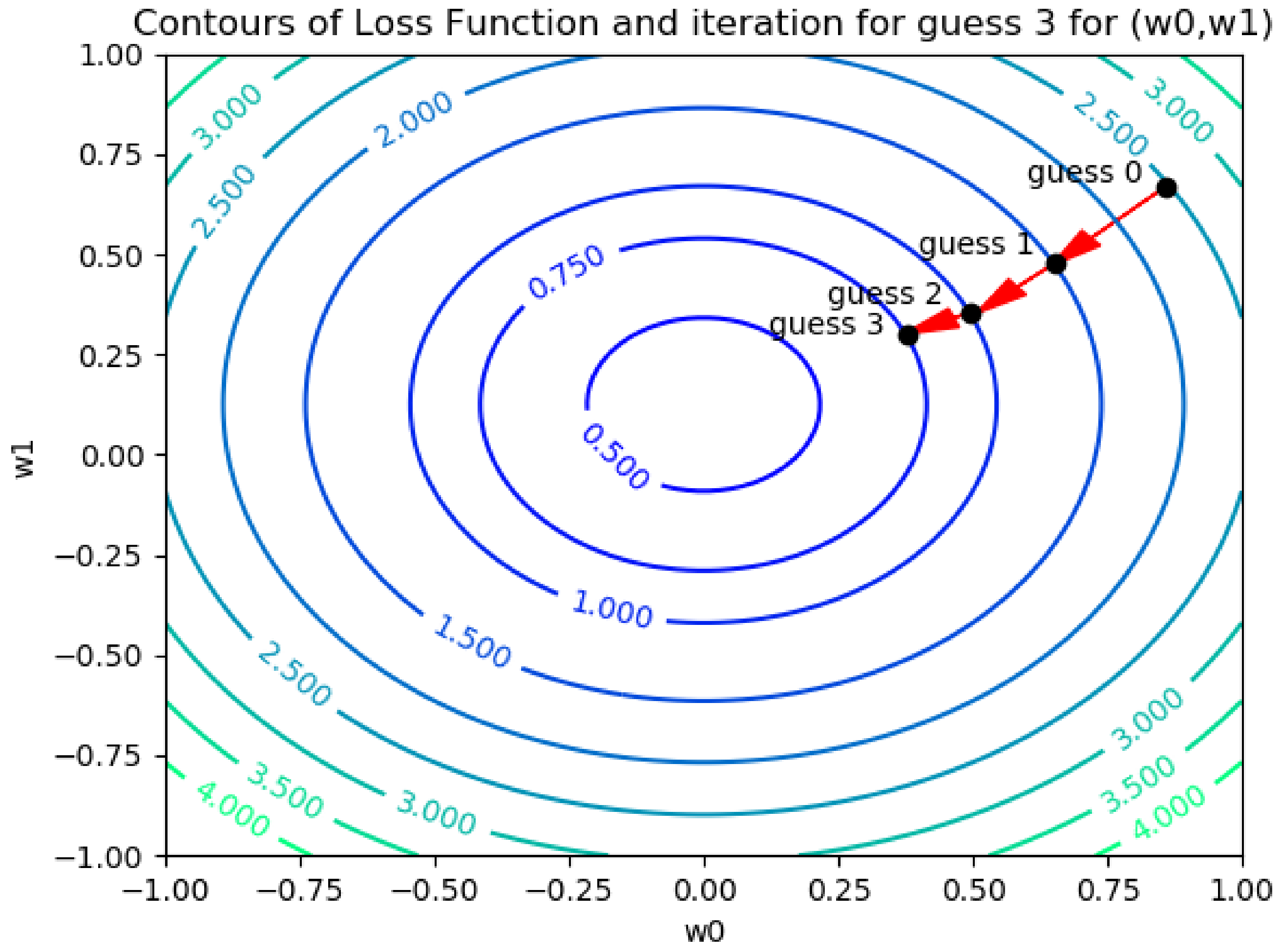
Optimization - Gradient Descent Algorithm

Key facts to be used in Gradient Descent Algorithm

- At each point in (W_0,W_1) plane one can compute gradient vector $\nabla_W L$,which points in direction of most rapid increase of L
- Gradient orthogonal to contour lines
- To minimize the L, take step in opposite direction of gradient
- Gradient Descent algorithm applies this process repeatedly:
    - Make guess for (W_0,W_1)
    - Loop: compute gradient at current point and take step in opposite direction
- Size of step depends on parameter $\alpha > 0$ called Learning Rate
    - If too large, then may overshoot minimum
    - If too small, then many steps may be required to get near minimum

Copyright Satish Reddy 2020

Optimization - Gradient Descent Animation



Copyright Satish Reddy 2020

Optimization: Gradient Descent Algorithm

Let W = [W_0 W_1 W_2 W_{d-1}] denote parameter vector of variables. Let L(W)= L(W_0,W_1,W_2 ,…, W_{d-1}) be a function of these parameters

Make initial guess $W_{epoch=0}$ and choose learning rate $\alpha > 0$

1. Loop epoch i = 1, 2, 3, …
   - Compute gradient vector $\nabla_W L_{epoch=i-1}$ at $W_{epoch=i-1}$
   - Compute new guess using formula: $\mathrm{W}_{epoch=i} = \mathrm{W}_{epoch=i-1} - \alpha \nabla_W L_{epoch=i-1}$
   - Compute $\mathrm{L}(\mathrm{W}_{epoch=i})$

Loop for fixed number of epoch (or if L(W) reduced sufficiently)

Note: this process may not lead to convergence to minimum or may converge to a local minimum

Copyright Satish Reddy 2020

Optimization – Gradient Descent Example

- Consider simple function (has minimum at [0 0])

$L(W) = L(W_0, W_1) = 2W_0^2 + W_1^2$ with gradient $\nabla_W L = [4W_0 \quad 2W_1]$

- Choose α=0.1 and initial guess

$W_{epoch=0} = [2\ 2]$

$L(W_{epoch=0}) = 2 * 2^2 + 2^2 = 12$

- Epoch 1:

$\nabla_W L(W_{epoch=0}) = [4W_0 \quad 2W_1] = [8\ \ 4]$

$W_{epoch=1} = W_{epoch=0} - \alpha\, \nabla_W L\left(W_{epoch=0}\right) = [2 \quad 2] - 0.1 * [8 \quad 4] = [1.2 \quad 1.6]$

$L(W_{epoch=1}) = 2 * 1.2^2 + 1.6^2 = 5.44$

- Epoch 2:

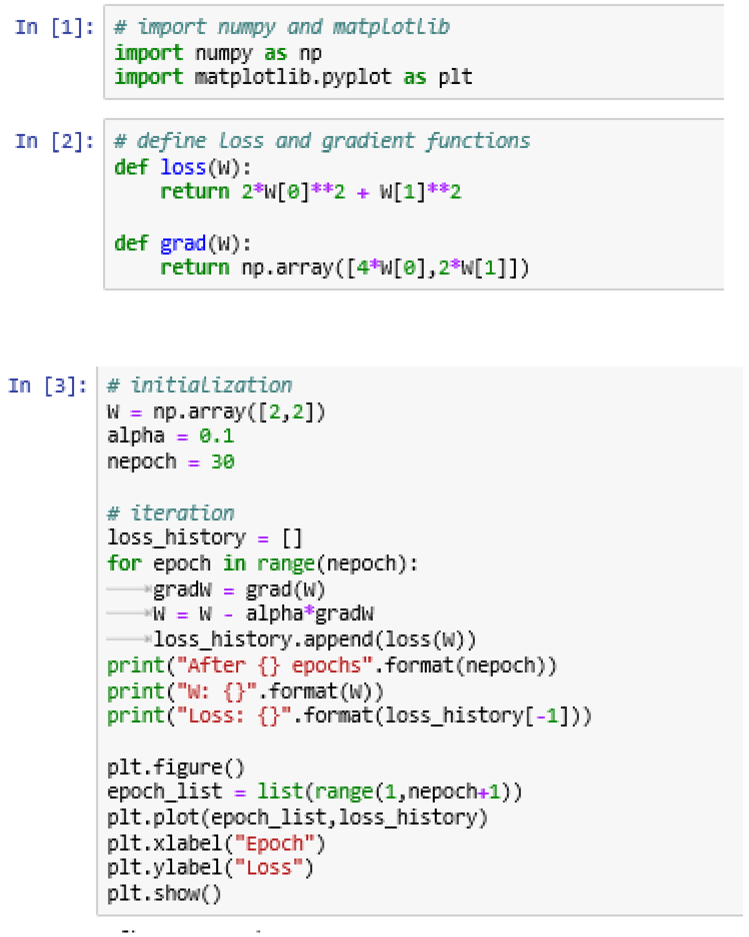
$\nabla_W L(W_{epoch=1}) = [4W_0 \quad 2W_1] = [4.8 \quad 3.2]$

$W_{epoch=2} = W_{epoch=1} - \alpha\, \nabla_W L\left(W_{epoch=1}\right) = [1.2 \quad 1.6] - 0.1 * [4.8 \quad 3.2] = [0.72 \quad 1.28]$

$L(W_{epoch=2}) = 2 * 0.72^2 + 1.28^2 = 2.6752$

Copyright Satish Reddy 2020

Optimization – Gradient Descent Example

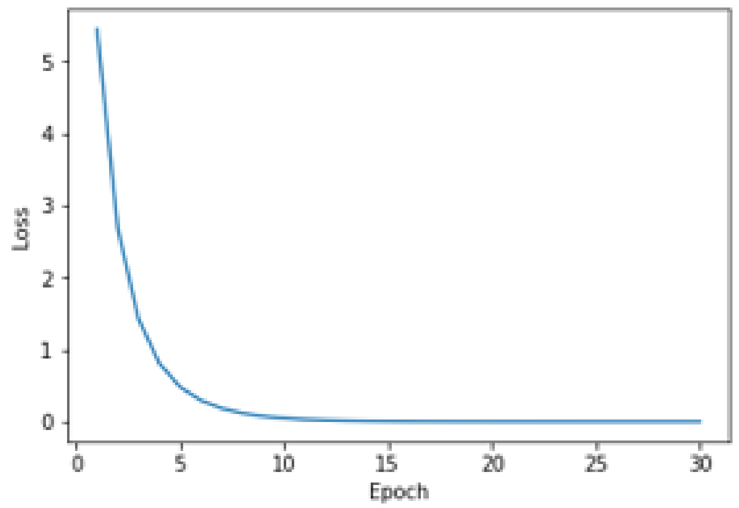


```
In [1]: # import numpy and matplotlib
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: # define loss and gradient functions
        def loss(W):
            return 2*W[0]**2 + W[1]**2

        def grad(W):
            return np.array([4*W[0],2*W[1]])
```

```
In [3]: # initialization
        W = np.array([2,2])
        alpha = 0.1
        nepoch = 30

        # iteration
        loss_history = []
        for epoch in range(nepoch):
            gradW = grad(W)
            W = W - alpha*gradW
            loss_history.append(loss(W))
        print("After {} epochs".format(nepoch))
        print("W: {}".format(W))
        print("Loss: {}".format(loss_history[-1]))

        plt.figure()
        epoch_list = list(range(1,nepoch+1))
        plt.plot(epoch_list,loss_history)
        plt.xlabel("Epoch")
        plt.ylabel("Loss")
        plt.show()
```

```
After 30 epochs
W: [4.42147839e-07 2.47588008e-03]
Loss: 6.129982554452983e-06
```



Copyright Satish Reddy 2020

General Formula for Optimization Algorithm

- Optimization algorithms use iteration to approximate W that minimizes loss
- Algorithms differ in update formula - in general formula is:

$$W_{epoch=i} = W_{epoch=i-1} + Update$$

- Discuss additional optimization approaches in Chapter 4

Copyright Satish Reddy 2020

Optimization – Jupyter Notebook Demo

- Open file IntroML/Examples/Chapter1/Chapter1.7_Optimization.ipynb
- Has examples of
  - Optimization using 2 epochs
  - Optimization using 30 epochs

Copyright Satish Reddy 2020

Notation

- Throughout course we will be dealing with vectors and matrices
- For Neural Networks we will be dealing with multiple layers
- Vectors will typically be row vectors:

k=layer

$$Y_j^{[k]}$$

j=column

- Matrix:

k=layer

$$X_{ij}^{[k]}$$

i=row, j=column

- Optimization (subscript epoch=p) corresponds to guess for p'th epoch. Here W could be a scalar, vector, or matrix

k=layer

$$W_{epoch=p}^{[k]}$$

guess at epoch p

Copyright Satish Reddy 2020