

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

Machine Learning:

- Chapter 1: Introduction
- Chapter 2: Linear and Logistic Regression
- Chapter 3: Neural Networks
- Chapter 4: Optimization, Validation, and Regularization
- Chapter 5: Case Studies
- Chapter 6: Introduction to Tensorflow
- Chapter 7: 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

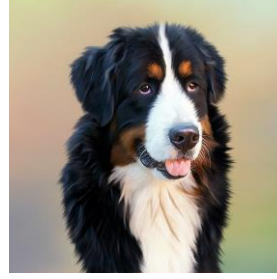
- Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.
- Machine learning algorithms build a mathematical model based on sample data, known 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	<p>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.</p> <p>Applications: regression, image classification, language translation, spam classification, auto-completion, etc</p>
Unsupervised Learning	<p>Process of learning previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision</p> <p>Applications: clustering, data mining</p>
Reinforcement Learning	<p>Process of learning what actions to take given the situation so as to maximize a reward</p> <p>Applications: Game playing (Atari, Chinese checkers, Chess, Go, etc), Industrial Control, etc</p>

Supervised Learning

Application	Input Information	Output Information/Label
Regression	<ul style="list-style-type: none">House features: lot area, floor area, # of bathrooms, # of floors, size of garage, etc	<ul style="list-style-type: none">Price
Image Classification	<ul style="list-style-type: none">Images of cats and dogsBone x-raysImages of animals	<ul style="list-style-type: none">cat/dognormal/brokenAnimal name
Language Translation	<ul style="list-style-type: none">English words and phrases	<ul style="list-style-type: none">French translations
Spam Filter	<ul style="list-style-type: none">Messages	<ul style="list-style-type: none">spam/not spam
Auto-Completion	<ul style="list-style-type: none">(2,3,4,5 ...) word phrases	<ul style="list-style-type: none">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:

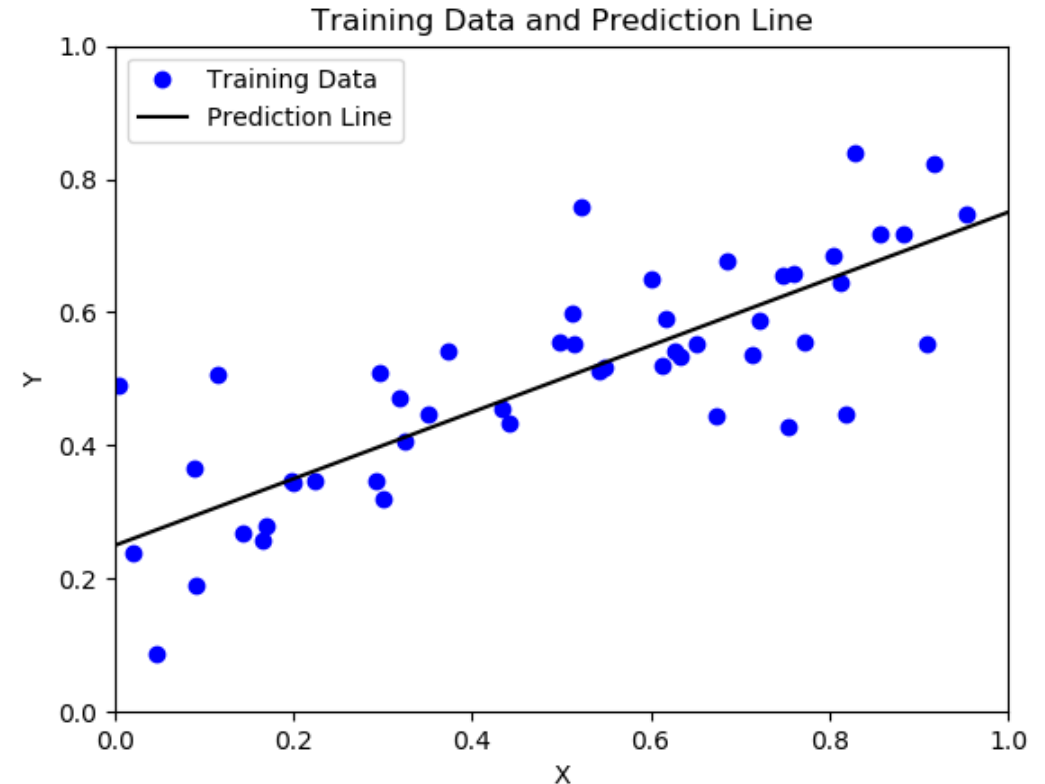
- Find straight line that best fits the training data

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 (x_0, x_1) plane
- Output Information: label 0 (red) or 1 (blue) for each point

Logistic Regression Goal:

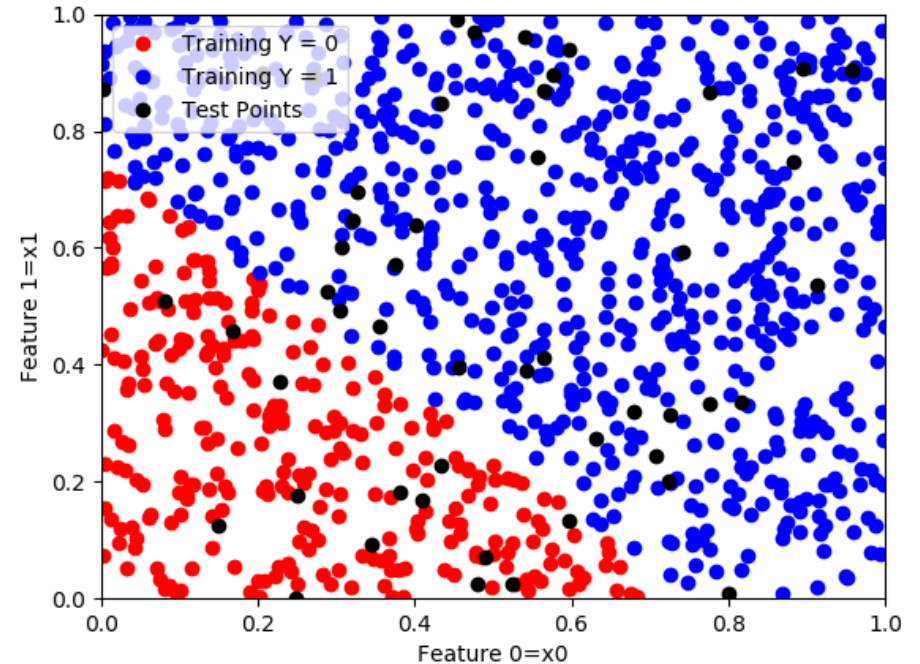
- Find function that best fits 0 and 1 labels in training data

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 (x_0, x_1) plane
- Output Information: label 0 (red) or 1 (blue)

Neural Networks Goal:

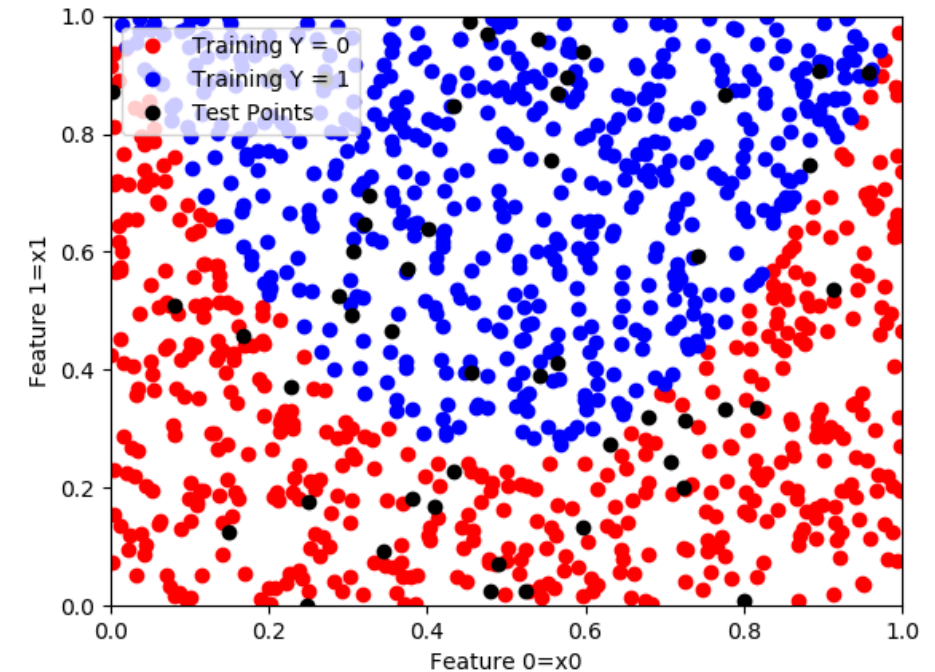
- Find function that best fits 0 and 1 labels in training data

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:

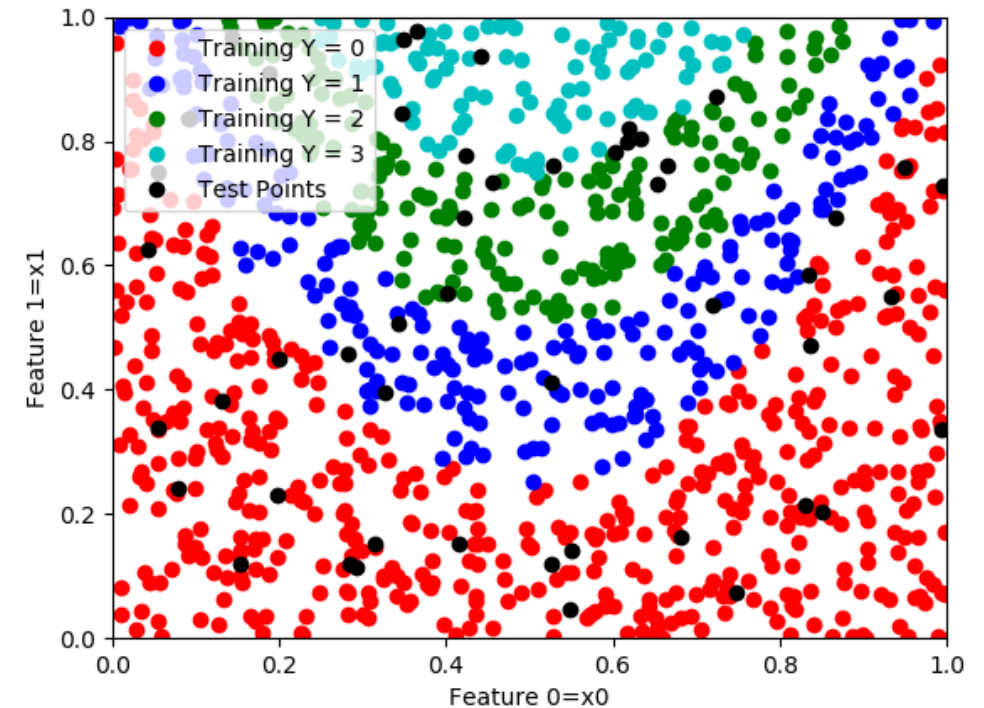
- Input Information: points in (x_0, x_1) plane
- Output Information: label 0 (red) or 1 (blue), 2 (green), 3 (cyan)

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

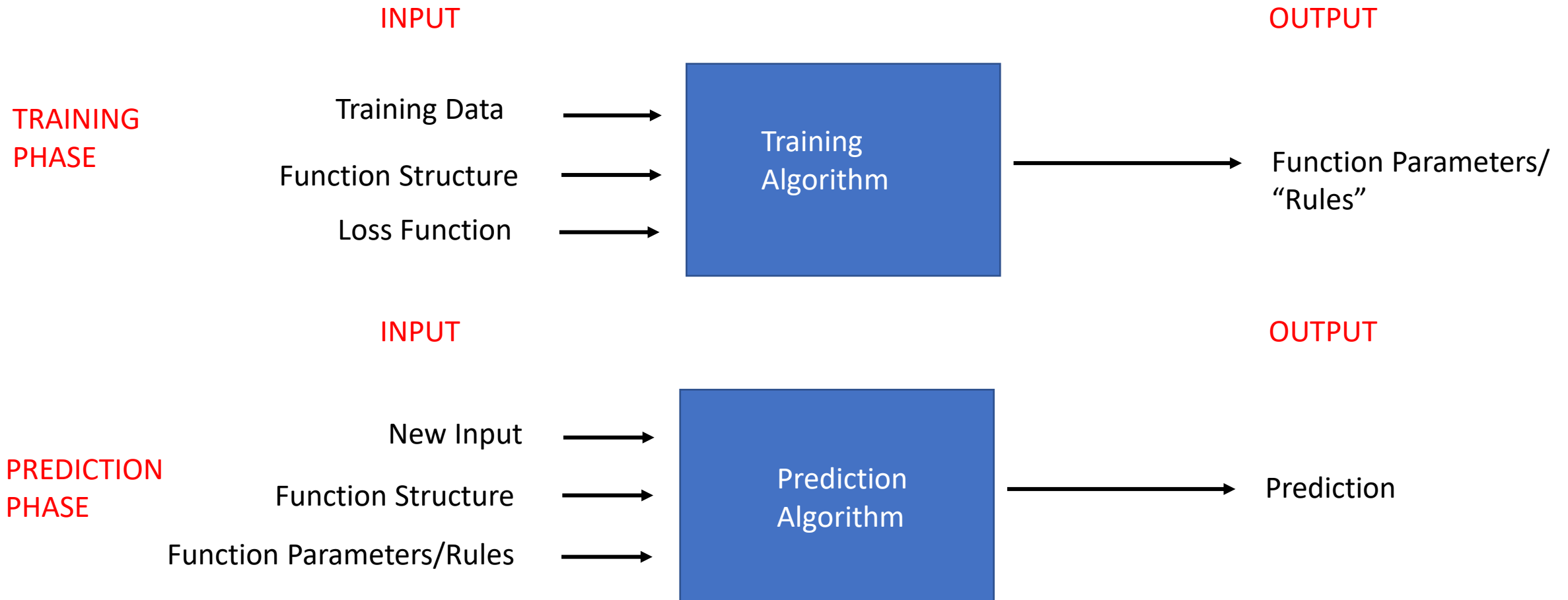
General approach for Supervised Learning:

- 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 $Y = W \cdot X + b$ (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.
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 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

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, 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

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
- All codes examples written in Python
- Course will run programs using both
 - Jupyter Notebook
 - Useful for short demos
 - Command Window
 - Will use to run main code base examples
- 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
 - Contains short python code snippets that complement examples in the presentations
- 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 Spam and MNIST case studies
Version 5.1	Add Tensorflow examples

Packages used in Course

Component	Version	Description/Comments
Python	3.7.1	<p>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.</p> <p>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.</p> <p>See: https://www.anaconda.com/ for download and installation of Anaconda. See: https://www.python.org for information about Python.</p>
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	<p>Package for machine learning. Will use its text processing functions for spam classification</p> <p>Scikit-learn: Machine Learning in Python, Pedregosa <i>et al.</i>, JMLR 12, pp. 2825-2830, 2011.</p>
tensorflow	1.13.1	Open source platform/framework for machine learning http://tensorflow.org
unittest, csv, time		These packages are part of the python release

Virtual Environment

- Useful to 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/Package Installation Demo

Task	Commands
Create virtual environment -In the Anaconda Prompt window (NAME is user specified name of environment)	<code>conda create -n NAME</code> <code>conda create -n NAME python=3.7.1</code>
List all virtual environments -In the Anaconda Prompt window	<code>conda env list</code>
Activate virtual environment -In the Anaconda Prompt window (NAME is user specified name of environment)	<code>conda activate NAME</code>
Deactivate virtual environment -In the Anaconda prompt window	<code>conda deactivate</code>
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”)	<code>pip install --user ipykernel</code> <code>python -m ipykernel install --user --name=NAME</code>
Installing packages - In the Anaconda Prompt window (after activating virtual environment):	<code>conda install numpy=1.18.1</code> <code>conda install matplotlib=3.0.3</code> <code>conda install pandas = 0.23.4</code> <code>conda install scikit-learn=0.20.1</code> <code>conda install tensorflow=1.13.1</code>
List packages: - In Anaconda Prompt window (after activating environment)	<code>conda list</code>

1.4 Introduction to Numpy Demo

Introduction to NumPy

NumPy is a Python package for scientific computing

- Underlying concept is numpy array (multi-dimensional object)
- 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

Introduction to NumPy Demo

File in Folder: IntroML\Examples\Chapter1	Examples
Chapter1.4_NumpyArrays.py	Examples of creation of numpy arrays
Chapter1.4_NumpyComponentwise.py	Examples of performing component-wise addition, multiplication, and scalar multiplication
Chapter1.4_NumpyFunctions.py	Examples of performing operations such as exponential, absolute value, and square
Chapter1.4_NumpyShapes.py	Examples of concatenating and reshaping arrays
Chapter1.4_NumpySum.py	Examples of using sum function

1.5 Review of Mathematical Concepts: Linear Algebra

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

Review of Mathematical Concepts

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

Linear Algebra – Vectors and Matrices

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

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

Denote the i 'th entry as W_i

- Example of a matrix with dimensions $d \times m$ (d rows and m columns)
(indices are $i=0, \dots, d-1$, and $j=0, \dots, m-1$)

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

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

Linear Algebra – Transpose

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

$$W = [W_0 \ W_1 \ \dots \ W_{d-1}] \quad W^T = \begin{bmatrix} W_0 \\ \vdots \\ W_{d-1} \end{bmatrix}$$

$\xrightarrow{\text{d entries}}$
 $\downarrow \text{d entries}$

- Let X be ($d \times m$) matrix. Transpose (dimensions $m \times d$) denoted by superscript T defined by: $X_{ij}^T = X_{ji}$

$$X = \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \vdots & \ddots & \vdots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix} \quad X^T = \begin{bmatrix} X_{00} & \dots & X_{d-1,0} \\ \vdots & \ddots & \vdots \\ X_{0,m-1} & \dots & X_{d-1,m-1} \end{bmatrix}$$

$\xrightarrow{\text{m entries}}$
 $\downarrow \text{d entries}$
 $\xrightarrow{\text{d entries}}$
 $\downarrow \text{m entries}$

Linear Algebra – Transpose Example

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

- Example 2:

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

Linear Algebra – Dot Product

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

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

- Dot product of W and X given by:

$$Z = W_0X_0 + W_1X_1 + W_2X_2 + \dots + W_{d-1}X_{d-1} = \sum_{i=0}^{d-1} W_iX_i$$

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

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

Linear Algebra – Dot Product Example

- Example

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

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

Linear Algebra – Vector/Matrix Multiplication

- Let W be row vector of length d , X be a matrix of dimension $(d \times m)$

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

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

$$Z_j = W_0 X_{0j} + W_1 X_{1j} + W_2 X_{2j} + \dots + W_{d-1} X_{d-1,j} = \sum_{i=0}^{d-1} W_i X_{i,j} \quad \text{for } j=0, \dots, m-1$$

- 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

$$Z = WX$$

Linear Algebra – Matrix/Matrix Multiplication

- Let W be a matrix of dimension $(n \times d)$, X be a matrix of dimension $(d \times m)$

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

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

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

- Z is an $n \times m$ 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

$$Z = WX$$

Linear Algebra – Multiplication Example 1

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

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

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

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

Linear Algebra – Multiplication Example 2

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

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

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

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

Linear Algebra – Broadcasting

- Let W be row vector of length d , X be a matrix of dimension $(d \times m)$, and b a scalar

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

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

$$Z_j = W_0 X_{0j} + W_1 X_{1j} + W_2 X_{2j} + \dots + W_{d-1} X_{d-1,j} + b = \sum_{i=0}^{d-1} W_i X_{ij} + b \text{ for } j=0, \dots, m-1$$

- Z is a row vector of length m

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

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)

Linear Algebra – Broadcasting Example 1

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

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

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

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

Linear Algebra – Broadcasting

- Let W be a matrix of dimension $(n \times d)$, X be a matrix of dimension $(d \times m)$, 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

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

- Z is an $n \times m$ 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

Linear Algebra – Broadcasting Example 2

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

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

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

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

Linear Algebra – Multiplication and Transpose

- Consider:

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

- Rewrite as:

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

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

- Consider:

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

- Rewrite as:

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

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

Linear Algebra – Jupyter Notebook Demo

- Open file IntroML/Examples/Chapter1/Chapter1_LinearAlgebra.ipynb
- Has examples of
 - Transpose
 - Matrix Multiplication
 - Matrix Multiplication with Broadcasting

1.6 Review of Mathematical Concepts: Multivariable Calculus

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

Review of Mathematical Concepts

- Multivariable Calculus
 - Partial Derivatives
 - Gradients
 - Chain Rule

Multivariable Calculus – Partial Derivatives

- Consider a function of m variables $L = L(Z_0, Z_1, Z_2, \dots, Z_{m-1})$
- 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:

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

Multivariable Calculus – Gradient

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

$$\nabla_Z L = \begin{bmatrix} \frac{\partial L}{\partial Z_0} & \frac{\partial L}{\partial Z_1} & \cdots & \frac{\partial L}{\partial Z_{m-1}} \end{bmatrix}$$

- Individual entries of gradient vector denoted by:

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

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

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

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

Multivariable Calculus – Chain Rule

- Consider a function of m variables $L = L(Z_0, Z_1, Z_2, \dots, Z_{m-1})$.
- Suppose: $Z_j = W_0 X_{0j} + W_1 X_{1j} + W_2 X_{2j} + \dots + W_{d-1} X_{d-1,j}$ for $j=0, \dots, m-1$

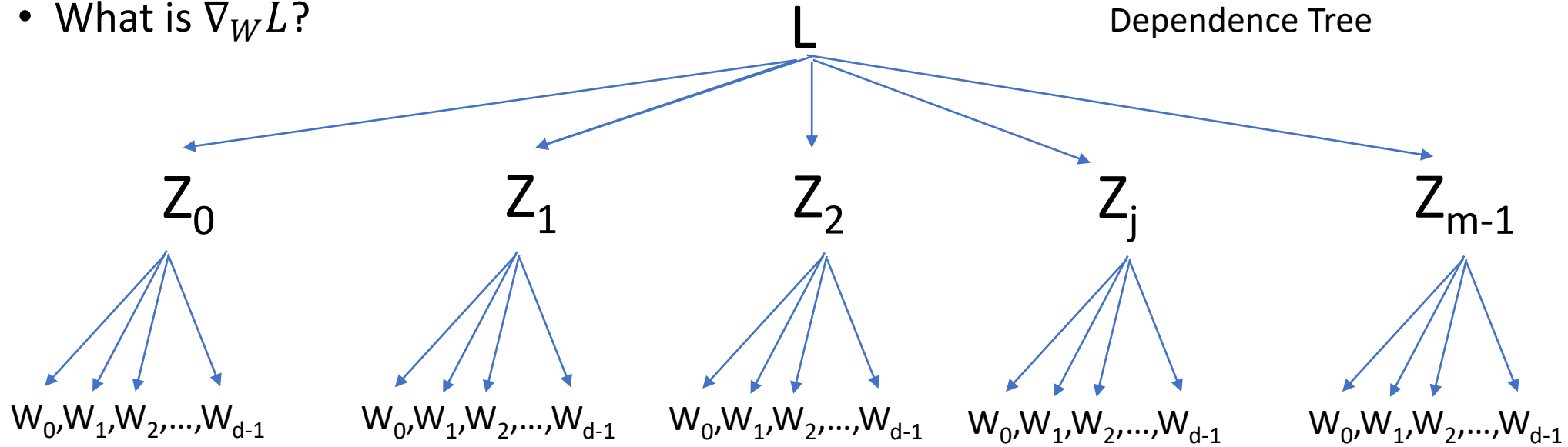
- In matrix terms:

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

- Question: what is $\nabla_W L$?

Multivariable Calculus – Chain Rule

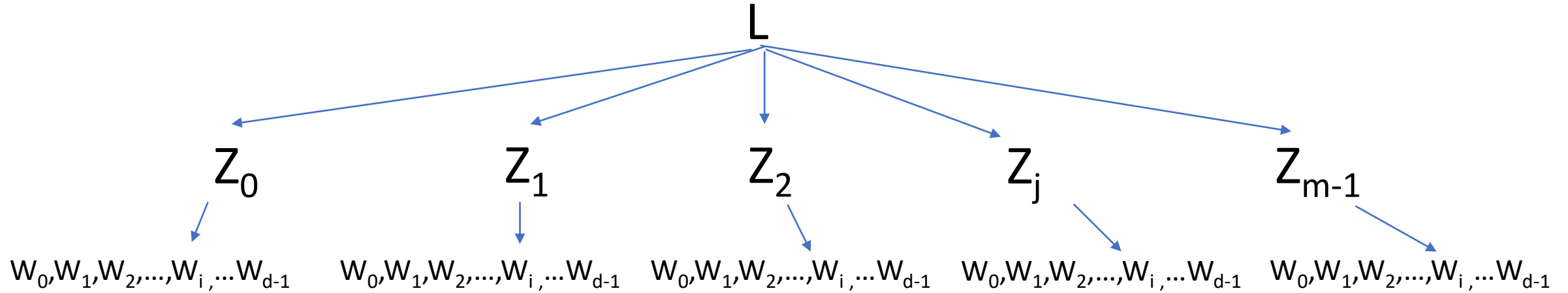
- What is $\nabla_W L$?



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

Multivariable Calculus – Chain Rule



1. For $\frac{\partial L}{\partial W_i}$ there are m paths: $L \rightarrow Z_j \rightarrow W_i$ $j=0, \dots, m-1$:
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 $i=0, \dots, d-1$)

- By definition: $\frac{\partial L}{\partial Z_j} = \nabla_Z L_j$

- From the last pages: $Z_j = W_0 X_{0j} + W_1 X_{1j} + W_2 X_{2j} + \dots + W_{d-1} X_{d-1,j}$ 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: $1 \times d = (1 \times m) \text{ times } (m \times d)$

Multivariable Calculus – Chain Rule

- Consider more general case: $L = L(Z)$
- Z is an $(n \times m)$ matrix, so L is a function of $n \times m$ variables
- Z defined by $Z = WX + b$: W is $n \times d$, X is $d \times m$ and b is $n \times 1$

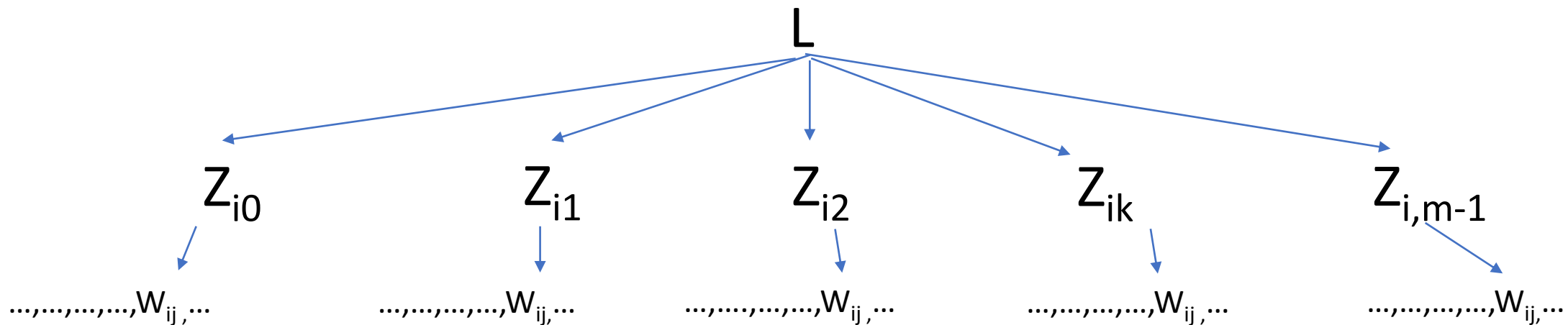
$$\begin{bmatrix} Z_{00} & \dots & Z_{0,m-1} \\ \dots & \dots & \dots \\ Z_{n-1,0} & \dots & Z_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} W_{00} & \dots & W_{0,d-1} \\ \dots & \dots & \dots \\ W_{n-1,0} & \dots & W_{n-1,d-1} \end{bmatrix} \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots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

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

Multivariable Calculus – Chain Rule

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

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

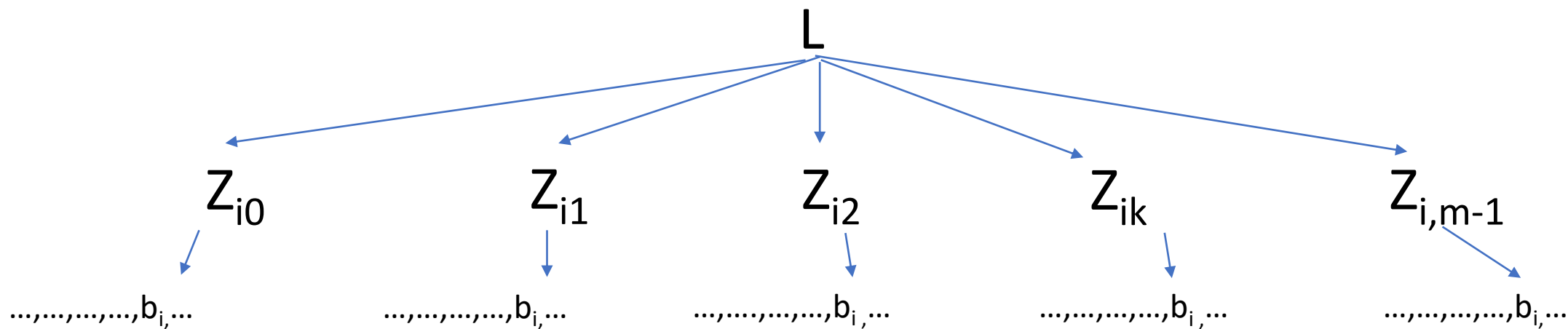


- Applying chain rule: $\nabla_W L_{ij} = \sum_{k=0}^{m-1} \frac{\partial L}{\partial Z_{ik}} \frac{\partial Z_{ik}}{\partial W_{ij}} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} X_{jk} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} X_{kj}^T$
- This is matrix/matrix multiplication: $\nabla_W L_{ij} = (\nabla_Z L X^T)_{ij}$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

Multivariable Calculus – Chain Rule

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

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

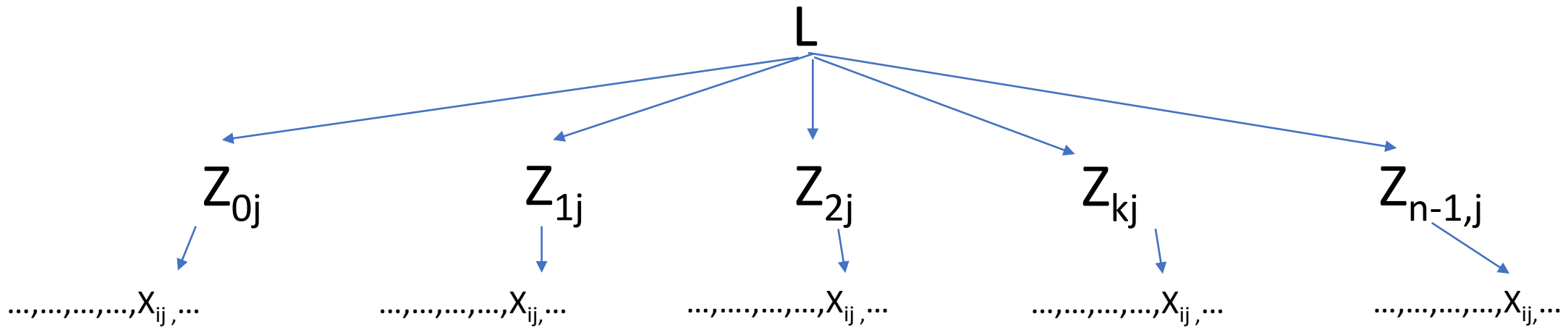


- Applying chain rule: $\nabla_b L_i = \sum_{k=0}^{m-1} \frac{\partial L}{\partial Z_{ik}} \frac{\partial Z_{ik}}{\partial b_i} = \sum_{k=0}^{m-1} \nabla_Z L_{ik} * 1 = \sum_{k=0}^{m-1} \nabla_Z L_{ik}$
- This is simply a sum along the i 'th row of the gradient $\nabla_Z L$

Multivariable Calculus – Chain Rule

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

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



- 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: $\nabla_X L = W^T \nabla_Z L$

Multivariable Calculus – Chain Rule Example

- Consider:

$$L = 2Z_0 + Z_1$$

- Gradient is

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

- Now assume:

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

so

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

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

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

Multivariable Calculus – Chain Rule Example

Start with:

$$L = 2W_0X_{00} + 2W_1X_{10} + 2b + W_0X_{01} + W_1X_{11} + b \quad \nabla_Z L = \begin{bmatrix} \frac{\partial L}{\partial z_0} & \frac{\partial L}{\partial z_1} \end{bmatrix} = [2 \quad 1]$$

$$\text{Direct Calculation: } \nabla_W L = \begin{bmatrix} \frac{\partial L}{\partial w_0} & \frac{\partial L}{\partial w_1} \end{bmatrix} = [2X_{00} + X_{01} \quad 2X_{10} + X_{11}]$$

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

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

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

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

$$\text{Chain Rule: } \nabla_b L = \sum_{j=0}^{m-1} \nabla_Z L_j = [3]$$

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) $\nabla_X L = W^T \nabla_Z L$

1.7 Review of Mathematical Concepts: Optimization

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

Review of Mathematical Concepts

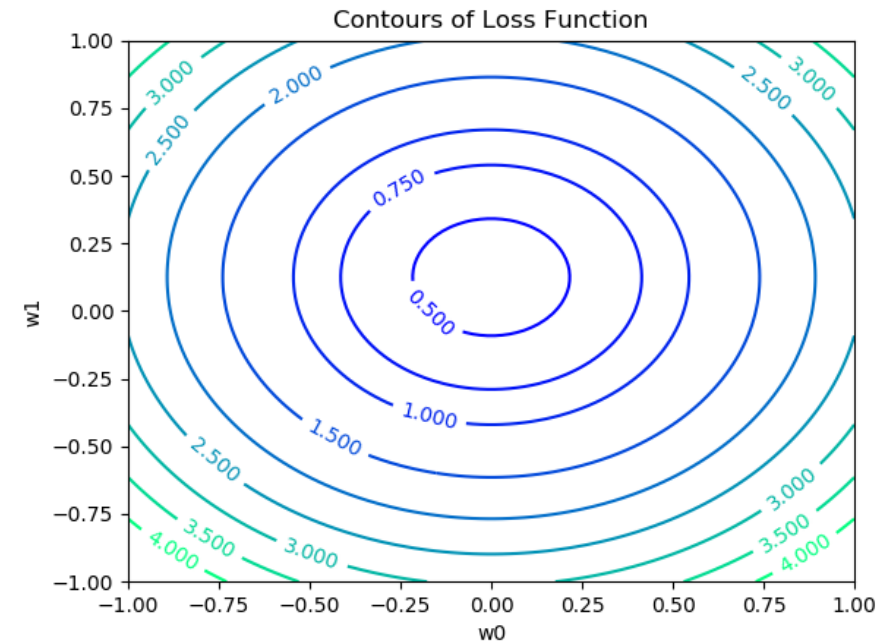
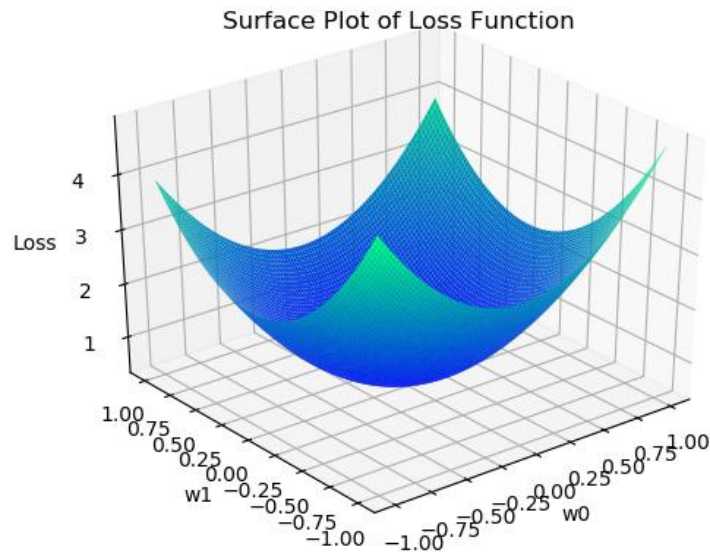
- Optimization
 - Gradient Descent for Minimizing a Function

Optimization

- Let $L(W_0, W_1, W_2, \dots, W_{d-1})$ denote a Loss function of d parameters
- Machine Learning Training Algorithm involves finding the parameters $W_0, W_1, W_2, \dots, W_{d-1}$ 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

Optimization - Surface and Contour Plots

- Use example of Loss function of 2 variables $\text{Loss} = L(W_0, W_1)$ 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)

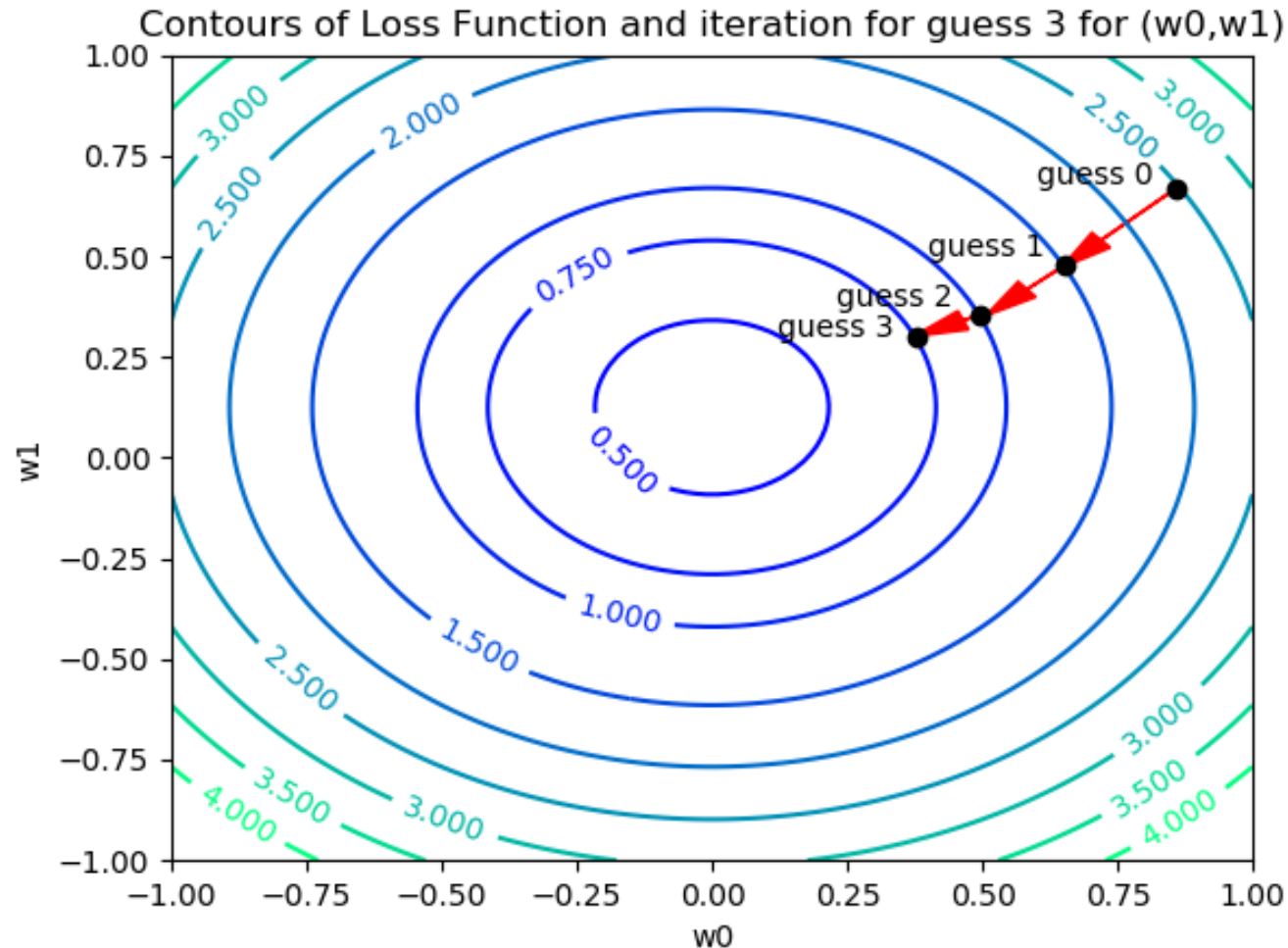


Optimization - Gradient Descent Algorithm

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 Loss
- Gradient orthogonal to contour lines
- To minimize the Loss, 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

Optimization - Gradient Descent Animation



Optimization: Gradient Descent Algorithm

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

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

1. Loop epoch $i = 1, 2, 3, \dots$

- Compute gradient vector $\nabla_W L_{\text{epoch}=i-1}$ at $W_{\text{epoch}=i-1}$
- Compute new guess using formula:
$$W_{\text{epoch}=i} = W_{\text{epoch}=i-1} - \alpha \nabla_W L_{\text{epoch}=i-1}$$
- Compute $L(W_{\text{epoch}=i})$

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

Note: $W_{\text{epoch}=i}$ may not converge or may converge to a local minimum

Optimization – Gradient Descent Example

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

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

- Choose $\alpha=0.1$ and initial guess

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

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

- Epoch 1:

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

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

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

- Epoch 2:

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

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

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

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

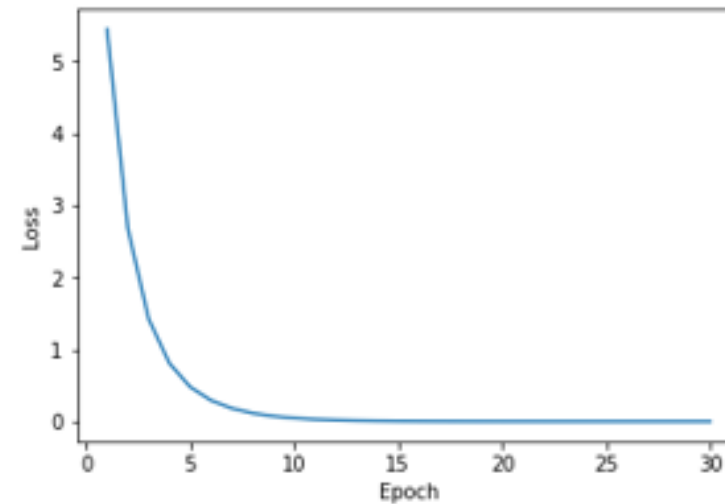
# iteration
loss_history = []
for epoch in range(nepoch):
    gradW = grad(W)
    W = W - alpha*gradW
    loss_history.append(loss(W))
print("After {} epochs".format(nepoch))
print("W: {}".format(W))
print("Loss: {}".format(loss_history[-1]))

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



General Formula for Optimization Algorithm

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

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

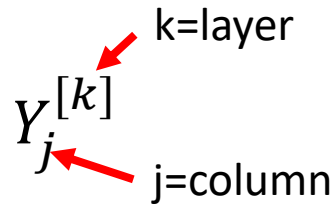
- Discuss additional optimization approaches in Chapter 4

Optimization – Jupyter Notebook Demo

- Open file IntroML/Examples/Chapter1/Chapter1_Optimization.ipynb
- Has examples of
 - Optimization using 2 epochs
 - Optimization using 30 epochs

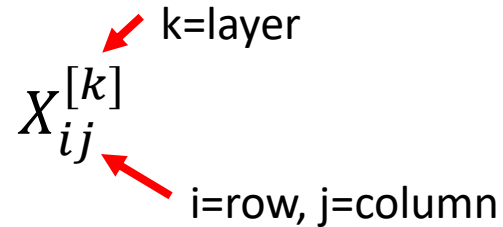
Notation

- 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:

$$Y_j^{[k]}$$


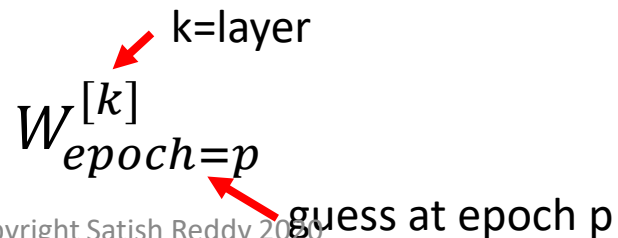
$k=\text{layer}$
 $j=\text{column}$

- Matrix:

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


$k=\text{layer}$
 $i=\text{row}, j=\text{column}$

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

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


$k=\text{layer}$
guess at epoch p