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

Chapter 7.1 Summary and Thank You

Summary and Thank You

Goal of this Section:

- Present a summary of the course material
- Provide suggestions for further study
- Thank You

Supervised Learning

Recall definition of Supervised Machine 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.

This course has focused on Linear Regression, Logistic Regression and Neural Network approaches for Supervised Machine Learning

- Each of these approaches assumes a different function structure for the learned function

Components of Supervised Learning

(1) Training Data

- Input information
- Labelled Output information

(2) Function Structure

- Defines general form of the function to be learned
- Will have unknown parameters
- Process of applying function structure is called Forward Propagation

(3) Loss Function

- Used to measure effectiveness of function structure and choice of parameters

(4) Training Phase

- Uses optimization technique to determine function parameters that minimize loss function for training data
- Process of computing derivatives used in optimization is called Back Propagation

(5) Prediction Phase

- Applies forward propagation using parameters determined in Training Phase to predict outputs when new input data is provided

Training Data - Summary

Case	Details
Input/Output	Assume m data points Data point j: Input information: feature vector: $\begin{bmatrix} X_{0,j} \\ X_{1,j} \\ \dots \\ X_{d-1,j} \end{bmatrix}$ assume d features Labelled output information: Y_j Define the feature matrix (dxm) and output vector (1xm): $X = \begin{bmatrix} X_{00} & \dots & X_{0,m-1} \\ \dots & \dots & \dots \\ X_{d-1,0} & \dots & X_{d-1,m-1} \end{bmatrix}$ $Y = [Y_0 \quad \dots \quad Y_{m-1}]$
Linear Regression	Y_j is a real value
Binary Classification (Logistic Regression/ Neural Network)	Y_j is 0 or 1
Multiclass Classification	Y is one of 0,1,…,c-1 (assume c classes)

Function Structure – Summary

Component	Details
Linear Regression	Input: X is feature matrix (dxm) Parameters: W is row vector of length d, b is scalar $Z = WX + b$ (row vector of length m) Activation function: $f(z) = z$ $A = f(Z)$ (row vector of length m)
Logistic Regression	Input: X is feature matrix (dxm) Parameters: W is row vector of length d, b is scalar $Z = WX + b$ (row vector of length m) Activation function: $f(Z) = \frac{1}{1+e^{-Z}}$ (sigmoid activation) $A = f(Z)$ (row vector of length m)
Neural Network	Input: X is feature matrix (dxm) Assume N layers Parameters: $W^{[k]}, b^{[k]}$ for layers k=1,…,N For each layer k =1, …, N Linear: $Z^{[k]} = W^{[k]}A^{[k-1]} + b^{[k]}$ $A^{[0]} = X$ Activation: $A^{[k]} = f^{[k]}(Z^{[k]})$ (use sigmoid/softmax activation in layer N for binary/multiclass classification)

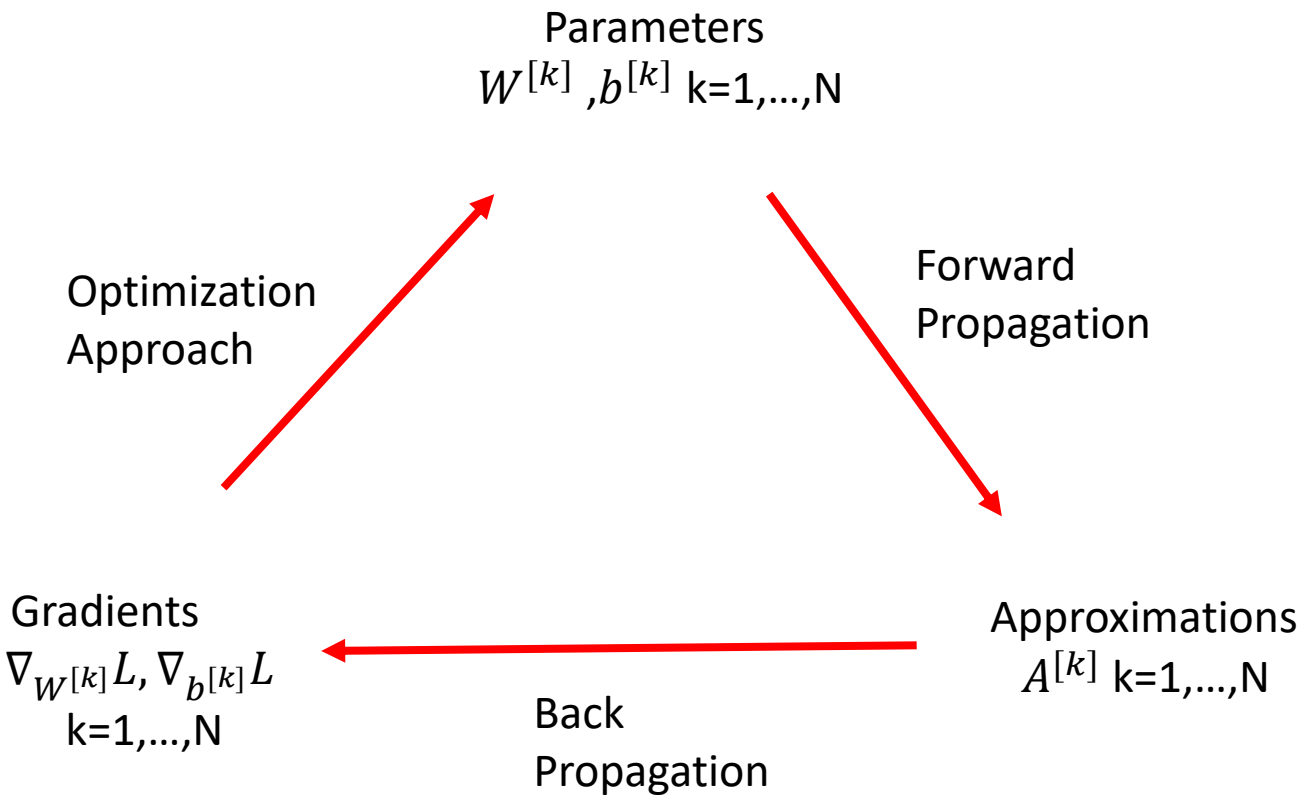
Loss Function – Summary

Component	Details
General	$Y = [Y_0 \quad \dots \quad Y_{m-1}]$ is training output information (m data points) $Y^h = [Y_0^h \quad \dots \quad Y_{m-1}^h]$ is the one-hot matrix (m one-hot vectors for mult-iclass classification) $A = [A_0 \quad \dots \quad A_{m-1}]$ is output of forward propagation for Linear Regression/Logistic Regression $A^{[N]} = [A_0^{[N]} \quad \dots \quad A_{m-1}^{[N]}]$ is output of forward propagation for Neural Network
Linear Regression Mean Squared Error	$L = \frac{1}{m}\sum_{j=0}^{m-1}(A_j - Y_j)^2$
Logistic Regression/ Neural Network: Binary Cross Entropy	$L = -\frac{1}{m}\sum_{j=0}^{m-1} Y_j \ln A_j^{[N]} + (1 - Y_j)\ln(1 - A_j^{[N]})$
Neural Network: Cross Entropy (for multi-class classification)	$L = -\frac{1}{m}\sum_{j=0}^{m-1} Y_{ij}^h \ln A_j^{[N]}$
L2 Regularization Penalty Term	$P = \sum_{k=1}^{N} \lambda^{[k]} \sum_{i=0}^{n^{[k]}-1} \sum_{j=0}^{n^{[k-1]}-1} \left(W_{ij}^{[k]}\right)^2$ added to loss function

Back Propagation – Summary

Component	Details
Input	Feature matrix: X Value vector: Y or Y^h (one-hot vector in case of multiclass classification
Compute gradients of loss with respect to $W^{[k]}$ and $b^{[k]}$	Perform Forward Propagation to compute $A^{[k]}$ for k=1,…,N Compute $\nabla_{A^{[N]}} L$ For k = N,…,1 • Compute $\nabla_{Z^{[k]}} L$ • $\nabla_{W^{[k]}} L = \nabla_{Z^{[k]}} L A^{[k-1]^T} + \lambda^{[k]} W^{[k]}$ (includes penalty term for L2 regularization) • $\nabla_{b^{[k]}} L_i = \sum_{j=0}^{m-1} \nabla_{Z^{[k]}} L_{ij}\,, \quad i = 0, \dots, n^{[k]} - 1$ • If k>1:$\nabla_{A^{[k-1]}} L = W^{[k]T} \nabla_{Z^{[k]}} L$

Training Algorithm



- Training Algorithm is used to find parameters $W^{[k]}$,$b^{[k]}$ k=1,…,N that minimize Loss function
- With initial $W^{[k]}$,$b^{[k]}$ k=1,…,N, use Forward Propagation to compute $A^{[k]}$ k=1,…,N
- Use Back Propagation to compute gradients
- Use Optimization Approach to update parameters:
  - $W^{[k]} \leftarrow W^{[k]} + Update(W^{[k]})$
  - $b^{[k]} \leftarrow b^{[k]} + Update(b^{[k]})$
  - Update depends on optimization approach
- Process is repeated

Prediction Algorithm – Summary

Component	Details
Input	New input feature matrix $\tilde{X}$ (d features x p samples)
Linear Regression	Perform Forward Propagation to compute $\tilde{A}$ (1xp), the prediction for values
Binary Classification Logistic Regression/ NeuralNetwork	Perform Forward Propagation to compute $\tilde{A}^{[N]}$ (1xp) Round values to 0 if less than 0.5 and to 1 if greater than or equal to 0.5 to get predicted labels
Multiclass Classification	Perform Forward Propagation to compute $\tilde{A}^{[N]}$(c classes x p samples) Predicted class label for each sample is row index of $\tilde{A}^{[N]}$ with largest value

Overview of Machine Learning Approach

Component	Details
Set up Neural Network	Pick number of layers Units per layer Activation functions
Data	Split data into training, validation, and test sets
Train	Perform training monitoring accuracy for the training and validation data sets
Underfitting	In case of underfitting: increase complexity of neural network (add more layers/units)
Overfitting	In case of overfitting: add regularization or additional training data or reduce complexity of neural network
Hyperparameter Search	Perform training and monitor accuracy for range of hyperparameters (regularization parameters, number of units and layers, optimization parameters) to determine best parameter set
Final test	Train using “optimal” parameters from hyperparameter search and compute accuracy using test set
Prediction	Use results of training with optimal parameters for prediction

Supervised Learning – Topics for Further Study

- Data
  - Feature Engineering
  - Natural Language Processing - how to convert words to feature matrix
  - Image Classification
- Function Structures
  - Recurrent Neural Networks for Natural Language Processing
  - Convolutional Neural Networks for Image Classification
  - Multiple objects
- Training Algorithms
  - Stopping criterion
  - Regularization
  - Optimization techniques
- Machine Learning Frameworks and Packages
  - Tensorflow, Pytorch, FastAI, sklearn, etc
- Higher Level Tools
  - Azure, AWS, etc
- Hyperparameter Search
  - Use multiple processors to speed up search

Machine Learning Resources

Some additional resources

- Online Courses
  - Coursera
  - Udemy
- Online Resources
  - Medium
  - Machine Learning Mastery
- Sources of Data for Practice
  - Kaggle website (Data Science Competitions)
    - www.kaggle.com
    - Many datasets
    - Collaborative community – many tutorials and publically available Jupyter notebooks available for learning purposes
  - UCI (University of California, Irvine) Machine Learning Repository
    - https://archive.ics.uci.edu/ml/index.php
    - Many datasets

Thank You

- Thank you for taking this course