MITx: 6.86x Machine Learning with Python - From Linear Models to Deep Learning

Overview

December 2019

Comment: This overview has been written in R Markdown that provides an authoring framework for data science. Learn more at https://bookdown.org/yihui/rmarkdown/

MITx's MicroMasters Program in Statistics and Data Science

From probability and statistics to data analysis and machine learning, master the skills needed to solve complex challenges with data.

In september 2019. a great MIT's movement, to open their doors to the whole world, has come to the field of Data Science. Anyone from the EdX platform for online learning, showing interests for Data Science has gotten an invitation to join. In the next year or so, the future students were asked to pass 4 MIT exams and one Capstone exam, so they could get the MIT's credentials in Data Science area. If they achieve the goal, they could also apply for the second semester on MIT's campus and finish **Masters of Statistics and Data Science** in a hybrid generation of online and on campus students.

Mastering the foundations of data science, statistics, and machine learning using programming languages like **Python** and **R** students are prepared for jobs like: Data Scientist, Data Analyst, Business Intelligence Analyst, Systems Analyst, Data Engineer. More about the program could be found **here**.

MITx: 6.86x Machine Learning with Python

An in-depth introduction to the field of machine learning, from linear models to deep learning and reinforcement learning, through hands-on Python projects. – Course 4 of 4 in the MITx MicroMasters program in Statistics and Data Science.

MITx: 6.86x is advanced 14 week long instructor paced online course given by MIT's Department of Electrical Engineering and Computer Science. Prerequisites for this course are:

- MIT: 6.00.1x Introduction to Computer Science and Programming Using Python or proficiency in Python programming
- MITx: 6.431x Probability - The Science of Uncertainty and Data or equivalent probability theory course
- College-level single and multi-variable calculus
- Vectors and matrices

Course is presented in 6 Units, with 19 Lectures, 7 Homeworks and 5 Python Projects, through topics like:

- Representation, over-fitting, regularization, generalization, VC dimension
- Clustering, classification, recommender problems, probabilistic modeling, reinforcement learning
- On-line algorithms, support vector machines, and neural networks/deep learning

Stidents are gaining knowledge about principles and algorithms for turning training data into effective automated predictions. In it's second edition, the course already has 50 thousands students enrolled.

Unit 0: Brief Review of Vectors, Planes, and Optimization

(total video duration - 21:09)

In this Unit basic overview of the next topics was made:

- **vectors** in n-dimensional space (addition and subtraction of two vectors, norm of an vector, dot product of two vectors)
- **plane definition** in n-dimensional spaces (used later in classification for decision boundary) as: a point x of n-dimensional space belongs to some plane if it satisfies the equation $\theta \cdot x + \theta_0 = 0$, where θ is a vector perpendicular to the given plane and θ_0 an offset of the plane from it's origin.
- **loss function** ($L(x, y; \theta) = \frac{1}{n} \sum_{i=1}^{n} |\hat{y} - y_i|$), **gradient descent** ($\hat{\theta} = \theta - \gamma \nabla_\theta L(x, y; \theta)$) and **chain rule**

Homework 0

(total number of questions - 32)

The homework for this Unit referred to the next topics:

1. Points and vectors
2. Planes
3. Matrices
4. Probability density functions
5. Univariate Gaussians
6. Optimization and gradients

Project 0

(number of coding questions: 5)

Initial Python setup and packages installation, introduction to Numpy, Neural network exercise, introduction to ML packages and one debugging exercise.

Unit 1: Linear Classifiers and Generalizations (2 weeks)

Lecture 1. Introduction to Machine Learning

(total video - 33:00, total number of questions - 16)

Objectives

- Understand the goal of machine learning from a movie recommender example
- Understand elements of **supervised learning**, and the difference between the **training set** and the **test set**
- Understand the difference of **classification** and **regression -** two representative kinds of supervised learning

Machine Learning definition was introduced: Machine learning is a discipline aims to design, understand, and apply computer programs that learn from experience (i.e. data) for the purpose of modelling, prediction, and control. Main focus of this lecture is on the prediction, explained through examples on the market value of a stock measured as a function of time or in a self-driving vehicle, one can try to predict whether a collision is about to happen, in a medical context, one might predict the risk of acquiring or getting a recurrence for a disease. There are other types of prediction such as someone liking a movie or not, predicting the properties of an image or in machine translation, translating one sentence from English to Spanish language for example.

Supervised Learning was introduced: machine learning tasks are really provided in terms of examples of correct behavior, putting large number of images into an annotated categories and then automating the process of finding the solution to a task. Defining a set of possible mappings depending of an image and a parameter outputting the category of an image is an optinal solution. A different value for these parameters will specify a different mapping. The aim is to find out of all the possible parameters the one that agrees the best with the given set of examples.
A Concrete Example of a Supervised Learning Task: Movie Recommender Problem. Each movie has a **feature vector** , $x^{(i)}$, with binary values answering specific questions about a given movie and a as output, $y^{(i)}$, +1 if one liked the movie and -1 if not. This is how one gets a training set of feature vectors (examples) with the associated **labels**. The task is to learn a mapping from those vectors to the labels. This mapping is then tested on the test set hoping that it will predict well the labels. Then naturally **training error** and **test errors** are introduced as a measure how good this mapping, a **classifier** $h\colon \chi \to \{-1, 1\}$, performs. The problem can occur if the training error is too small, that could mean possibility of overfiting the classifier to the given data. In that case the test error would be to high. Types of machine learning:

- **Supervised learning** - explicitly specify the correct behavior
- **Unsupervised learning** - strip the labels, only given the examples, model/find the irregularities in how those examples vary.
- **Semi-supervised learning** - mix of supervised and unsupervised
- **Active learning** - the algorithm can itself ask for useful, additional examples
- **Transfer learning** - training the method based on one scenario and applying the method in a different scenario
- **Reinforcement learning** - learning to control, learning to take actions in the world so as to optimize some criteria

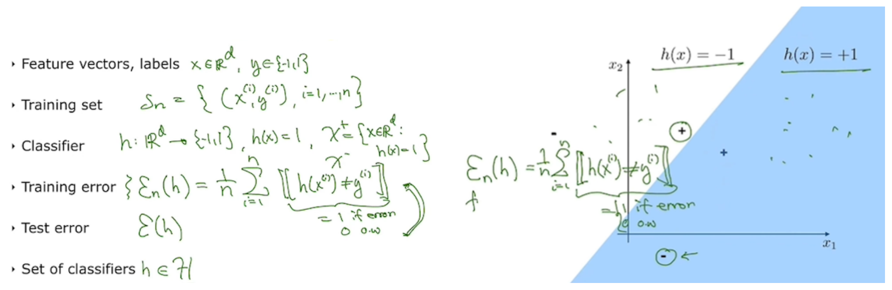
Lecture 2. Linear Classifier and Perceptron

(total video: 45:28, total number of questions: 19)

Objectives

- Understand the concepts of **feature vectors** and **labels**, **training set** and **test set**, **classifier**, **training error**, **test error**, and the **set of classifiers**
- Understand the **mathematical** presentation of linear classifiers
- Understand the **intuitive** and **formal** definition of linear separation
- Understand the **perceptron** algorithm with and without offset

All the basic concepts are introduced:


Feature vectors, labels $x \in \mathbb{R}^d$, $y \in \{-1,1\}$
Training set $S_n = \{(x^{(i)}, y^{(i)}), i=1,\dots,n\}$
Classifier $h: \mathbb{R}^d \to \{-1,1\}$, $h(x)=1$, $\mathcal{X}^+ = \{x \in \mathbb{R}^d : h(x)=1\}$
Training error $\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} [[h(x^{(i)}) \neq y^{(i)}]]$
$= 1$ if error, 0 o.w.
Test error $\mathcal{E}(h)$
Set of classifiers $h \in \mathcal{H}$
$h(x) = -1$
$h(x) = +1$
x_2
x_1
$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} [[h(x^{(i)}) \neq y^{(i)}]]$
$=1$ if error, 0 o.w.

Decision boundary for a specific classifier is defined as $\{x : \theta_1 x_1 + \theta_2 x_2 = 0\}$ or $\{x : \theta \cdot x = 0\}$ for $\theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}$ and $x = \begin{bmatrix}x_1 \\ x_2\end{bmatrix}$. It separates the plane into two subplanes, $\{x : \theta \cdot x > 0\}$ and $\{x : \theta \cdot x < 0\}$, where direction of θ is oriented to the positive one.

Lecture 3. Hinge loss, Margin boundaries and Regularization

(total video: 24:17, total number of questions: 10)

Objectives

- Understand the need for **maximizing** the margin
- Know how to **pose** linear classification **as an optimization problem**
- Understand **hinge loss**, **margin boundaries** and **regularization**

Homework 1

(total number of questions: 36)

1. Perceptron mistakes
2. Perceptron performance
3. Decision boundaries
4. Feature vectors

Lecture 4. Linear Classification and Generalization

(total video: 27:58, total number of questions: 11)

Objectives

- Understanding **optimization** view of learning
- Optimization algorithms - **gradient descent**, **stochastic** gradient descent, **quadratic** program

Homework 2

(total number of questions: 12)

1. Linear support vector machines
2. Passive-aggressive algorithm
3. Perceptron updates

Project 1: Automatic Review Analyzer

(number of coding questions: 9, number of additional questions 30)

The goal of this project is to design a classifier to use for sentiment analysis of product reviews. The training set consists of reviews written by Amazon customers for various food products. The reviews, originally given on a 5 point scale, have been adjusted to a +1 or -1 scale, representing a positive or negative review, respectively.

Python solutions on the given topics were required:

1. Hinge loss
2. Perceptron algorithm
3. Algorithm discussion
4. Classification and accuracy
5. Parameter tuning
6. Feature engineering

Unit 2: Nonlinear Classification, Linear regression, Collaborative Filtering

(2 weeks)

Lecture 5. Linear Regression

(total video: 50:46, total number of questions: 10)

Objectives

- Write the **training error** as least squares criterion for linear regression
- Use **stochastic gradient descent** for fitting linear regression models
- Solve **closed-form** linear regression solution
- Identify **regularization** term and how it changes the solution, **generalization**

Lecture 6. Nonlinear Classification: Kernels

(total video: 49:13, total number of questions: 13)

Objectives

- Understand **non-linear classifiers** from feature maps
- Understand moving from coordinate parameterization to weighting examples
- Learn how to find **kernel functions** induced from feature maps
- Understand **kernel perceptron** and **kernel linear regression**
- Understand **properties** of kernel functions

Lecture 7. Recommender Systems

(total video: 36:41, total number of questions: 9)

Objectives

- Understand the problem definition and assumptions of **recommender systems**
- Understand the impact of similarity measures in the **K-Nearest Neighbor method**
- Understand the need to impose the **low rank** assumption in collaborative filtering
- Iteratively find values of U and V (given $X = UV^T$) in collaborative filtering

Homework 3

(total number of questions: 26)

1. Collaborative filtering, kernels, linear regression
2. Feature vectors transformation
3. Kernels
4. Linear regression and regularization

Project 2: Digit recognition (Part 1)

(number of coding questions: 11, number of additional questions 20)

The MNIST database contains binary images of handwritten digits commonly used to train image processing systems. The digits were collected from among Census Bureau employees and high school students. The database contains 60,000 training digits and 10,000 testing digits, all of which have been size-normalized and centered in a fixed-size image of 28 $\times$ 28 pixels. Many methods have been tested with this dataset and in this project, one will get a chance to experiment with the task of classifying these images into the correct digit using some of the methods you have learned so far.

Python solutions on the given topics were required:

1. Support vector machine
2. Multinomial (Softmax) regression and Gradient descent
3. Classification using manually crafted features
4. Dimensionality reduction using PCA
5. Cubic features
6. Kernel methods

Unit 3 Neural networks

(2.5 weeks)

Lecture 8. Introduction to Feedforward Neural Networks

(total video - 36:08, total number of questions - 17)

Objectives

- Recognize the number of **layers** of a **feedforward neural network** and the number of **units** in each layer
- Write down common **activation functions** such as the hyperbolic tangent function tanh , and the **rectified linear function (ReLU)**
- Compute the output of a simple neural network possibly with **hidden layers** given the **weights** and **activation functions**
- Determine whether data after transformation by some layers is linearly separable, draw decision boundaries given by the weight vectors and use them to help understand the behavior of the network

Lecture 9. Feedforward Neural Networks, Back Propagation, and Stochastic Gradient Descent (SGD)

(total video - 22:13, total number of questions - 5)

Objectives

- Use **back-propagation** and write down **recursive relations** to compute the gradient of the loss function with respect to the weight parameters
- Use the **stochastic descent algorithm** to train a feedforward neural network
- Recognize when a network has **overcapacity**

Lecture 10. Recurrent Neural Networks 1

(total video - 26:18, total number of questions - 17)

Objectives

- Know the difference between feed-forward and recurrent neural networks
- Understand the role of **gating** and **memory cells**
- Understand the process of **encoding** and **decoding**

Lecture 11. Recurrent Neural Networks 2

(total video - 35:58, total number of questions - 18)

Objectives

- Understand **Markov models**
- Understand **Feature based** Markov models and **Temporal/Sequence** problems
- Understand RNNs for **sequences**
- Understand RNN **decoding**

Homework 4

(total number of questions: 26)

1. Neural networks
2. LSTM
3. Backpropagation
4. Word embeddings

Lecture 12. Convolutional Neural Networks

(total video - 23:07, total number of questions - 11)

Objectives

- Understand **Convolution neural networks**
- . . .

Project 3: Digit recognition (Part 2)

(number of coding questions - 7, number of additional questions - 8)

In this section, we are going to use deep neural networks to perform the same classification task as in previous sections. We will use PyTorch, a python deep learning framework.

Midterm Exam (1 week)

(total number of questions - 38)

Unit 4 Unsupervised Learning

(2 weeks)

Lecture 13. Clustering 1

(total video - 48:58, total number of questions - 21)

Objectives

- Understand the definition of **clustering**
- Understand **clustering cost** from different similarity measures
- Understand the **k-means algorithm**

Lecture 14. Clustering 2

(total video - 28:22, total number of questions - 10)

Objectives

- Understand the **limitations** of the K-Means Algorithm
- Understand how the **K-Medoids algorithm** is different from the K-Means Algorithm
- Understand the **computational complexity** of the K-Means and the K-Medoids algorithms
- Understand the importance of well-determining the **number of clusters**
- Understand elements that can be supervised in unsupervised learning

Lecture 15. Generative Models

(total video - 53:05, total number of questions - 22)

Objectives

- Understand what **generative models** are and how they work
- Understand **estimation** and **prediction** phases of generative models
- Derive a **relation** connecting generative and discriminative models
- Derive **Maximum Likelihood estimates** for multinomial, Gaussian generative models

Lecture 16. Mixture Models; EM algorithm

(total video - 39:43, total number of questions - 19)

Objectives

- Review **Maximum Likelihood Estimation** (MLE) of mean and variance in Gaussian statistical model
- Define **Mixture Models**
- Understand and derive ML estimates of mean and variance of Gaussians in an **observed** Gaussian mixture model
- Understand EM algorithm to estimate mean and variance of Gaussians in an **unobserved** Gaussian mixture model

Homework 5

(total number of questions: 32)

1. K-means and K-medoids
2. Maximum likelihood estimation
3. EM algorithm

Project 4: Collaborative Filtering via Gaussian Mixtures

(number of coding questions - 8, number of additional questions - 14)

Your task is to build a mixture model for collaborative filtering. You are given a data matrix containing movie ratings made by users where the matrix is extracted from a much larger Netflix database. Any particular user has rated only a small fraction of the movies so the data matrix is only partially filled. The goal is to predict all the remaining entries of the matrix.

Unit 5 Reinforcement Learning

(2 weeks)

Lecture 17. Reinforcement Learning 1

(total video - 57:35, total number of questions - 15)

Objectives

- Introduction **Reinforcement Learning** (RL)
- Understand RL terminology
- Understand **utility** function, **policy**, **value** functions
- Understand **Bellman equations**
- Understand value iteration and **Q-value** iteration