Udemy QA boards

PROBLEM SOLVING PROCESS

**Identify the independent and dependent variables**

* Independent variable = feature dependent variable = label
* ‘Features’ are categories of data points that affect the value of a ‘label’

**Assemble a set of data related to the problem you’re trying to solve**

* Datasets almost always cleanup of formatting

**Decide on the type of output you are predicting**

* Regression used with continuous values, classification used with discrete values
* Classification – the value of our labels that belong to a discrete set (defined result)
* Regression – the value of our labels that belong to a continuous set (varied result)

**Based on the type of output, pick an algorithm that will determine a correlation between your features and labels**

* Many, many different algorithms exist, each with pros and cons

**Use model generated by algorithm to make a prediction**

* Models relate the value of features to the value of labels

Plinko

Goal: Given some data about where a ball is dropped from, can we predict what bucket it will end up in?

Array of objects approach:

[{dropPosition: 300, bounciness: 0.4, ballSize: 16, bucket: 4}]

Array of arrays approach (will be used in course):

[[300, 0.4, 16, 4], [350, 0.4, 25, 5], [416, 0.4, 16, 4], [722, 0.4, 16, 7]] = [dropPosition, bounciness, ballSize, bucket]

K-nearest neighbor (knn) “Birds of a feather flock together”

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970’s as a non-parametric technique.

K-nearest neighbor (one with one independent variable):

1. Drop a ball a bunch of times all around the board, record which bucket it goes into
2. For each observation, subtract drop point from 300px, get absolute value
3. Sort the results from least to greatest
4. Look at the ‘k’ top records. What was the most common bucket?
5. Whichever bucket came up most frequently is the one ours will probably go into

Lodash.com/docs

**Our prediction was bad!**

* Adjust the parameters of the analysis
* Add more features to explain the analysis
* Change the prediction point
* Accept that maybe there isn’t a good correlation

**Finding an ideal K**

* Record a bunch of data points
* Split that data into a ‘training’ set and a ‘test’ set
* For each ‘test’ record, run KNN using the ‘training’ data
* Does the result of KNN equal the ‘test’ record bucket?

Randomizing test data

Updating for KNN for multiple features

Feature Normalization

Original Data

Normalization

Standardization

Changes to drop position => predictable changes to output

Changes to ball position => changes our output, but not predictability

**Feature selection – deciding which features to include in analysis**

The end of the introduction

* Features vs labels
* Test vs training sets of data
* Feature normalization
* Common data structures (arrays of arrays)
* Feature selection

Lodash

Pros:

* Methods for just about everything we need
* Excellent API design (especially chain)
* Skills transferrable to other JS projects

Cons:

* Extremely slow (relatively)
* Not ‘numbers’ focused
* Some things are awkward (getting a column of values)

TensorFlow JS

Pros:

* Similar API to Lodash
* Extremely fast for numeric calculations
* Has a ‘low level’ linear algebra API + higher level API for ML
* Similar api to numpy – popular Python to numerical lib

Cons:

* Still in active development

The Plan:

* Learn some fundamental around Tensorflow JS
* Go through a couple of exercises with Tensorflow
* Rebuild KNN algorithm using Tensorflow
* Build other algorithms with Tensorflow

What is the fastest way to learn ML? It is to master fundamental operation around working with data

Js.tensorflow.org

Tensorflow’s #1 job (for us) is to make working with numbers in arrays of arrays really easy

Tensors – a JS object that wraps a collection of numbers (arrays)

1 dimensional

[200, 400, 600]

2 dimensional

[

[300, 0.4, 16, 4]

[350, 0.4, 25, 5]

]

3 dimensional

[

[

[5, 10, 17]

]

]

Count the number of opening square braces before you see a number

Shape – How many records in each dimension?

Imagine calling ‘.length’ once on each dimension from outside in

Tensor:

[

[5, 10, 17]

[18, 4, 2]

]

Shape:

[2,3]

Tensor:

[

[

[5, 10, 17]

]

]

Shape:

[1, 1, 3]

2d is the most important we will work with

[#rows, # columns]

**Broadcasting only works when:**

Take shape of both tensors

From right to left, the shapes are equal, or one is ‘1’

TensorA:

[

[1,2,3]

[4,5,6]

]

TensorB:

[

[1]

[1]

]

ShapeA: [2,3]

ShapeB: [2,1]

Tensorflow: KNN with regression

Steps:

* Apply a slightly different KNN algorithm in the browser with Tensorflow JS and fake data
* Move KNN algorithm to our code editor with real data and run in Node JS environment
* Do some optimization

Previously which bucket will a ball go into? – classification

What is the price of a house? – Regression

**KNN algorithm**

* Find distance between features and prediction point
* Sort from lowest point to greatest
* Take the top K records
* Average the label value of those top K records

Error =

Standardization =

Standardization =

Linear Regression

Pros:

Fast! Only train one time, then use for any prediction

Uses methods that will be very important in more complicated ML

Cons:

Lot harder to understand intuitively

Why not use Excel or Google spreadsheets? We can use many independent variables

Methods of Solving Linear Regression:

Ordinary Least Squares

Generalized Least Squares

…others

Gradient Descent –

Mean Squared Error = 2

Summation Symbol (Sigma) – take all guesses and all actual values and square the difference

Mean Squared Error is unlikely to ever be zero, but close to zero is target value

Price = m\* lot size + b

The better we guess m and b the smaller MSE is

‘m’ and ‘b’ will be as correct as they can be when MSE is a s low as possible

(House Price) = m \* (lot size) + b

Issues with this approach

* Don’t know the possible range of b
* Don’t know a step size for incrementing b
* Huge computational demands when adding in more features

Knowing the value of MSE at a given ‘b’ isn’t super useful, but knowing the slope, or rate of change of MSE is very valuable

Y = x^2 + 5

Dy/dx = 2x the derivative of an equation gives a new equation that tells us the slope at any location

**Gradient Descent**

* Pick a value for ‘b’
* Calculate the slope of MSE with b
* Is the slope very small? If yes, we are done!
* Multiply the slope by an arbitrary small value called a ‘learning rate’
* Subtract that from ’b’

Common Questions

* Why the learning rate?
* Why worry about derivatives? Just calculate MSE twice and compare the two values
* We want slope of 0, so why not set the derivative equal to zero and solve for b?

Slope of MSE with respect to B

**Gradient Descent**

Pick a value for ‘b’ and ‘m’

Calculate the slope of MSE with respect to ‘m’ and ‘b’

Are both slopes very small? If so, we are done!

Multiply both slopes by learning rate

Subtract results from ‘b’ and ‘m’

Miles per gallon = m \* (car horsepower) + b

Class LinearRegression

gradientDescent() – run one iteration of GD and update ‘m’ and ‘b’

train() – run GD until we get good values for ‘m’ and ‘b’

test() – use ‘test’ data set to evaluate the accuracy of our calculated ‘m’ and ‘b’

predict() – make a prediction using our calculated ‘m’ and ‘b’

We’re going to write a working but slow implementation of Gradient Descent

Use plain arrays of data and plain for loops so the calculation is easy to understand

Replace with a much faster version with way less code, but harder to understand

Use Tensorflow to dramatically simplify the code

Matrix Multiplication - Linear Algebra between two matrices (tensor)

* Are 2 matrices eligible to be multiplied together
* What is the output of matrix multiplication?
* How is matrix multiplication done?

Slope of MSE with respect to M and B

Labels – Tensor of our label data

Features – Tensor of our feature data

N – number of observations

Weights – M and B in a tensor

Transpose matrix to allow for matrix multiplication

Refactor

- Refactor constructor to make ‘features’ and ‘labels’ into tensors

- Append a column of one’s to the feature tensor

- Make a tensor for our weights as well

- Refactor ‘gradientDescent’ function to use the new equation

Accuracy

* Train the model with training data (already doing this)
* Use ‘test’ data to make predictions about observations with known labels
* Gauge accuracy by calculating ‘coefficient of determination’

Coefficient of Determination

SS tot = Total sum of squares

SS res = Sum of squares of residuals

Total Sum of squares – baseline accuracy value

Sum of squares of residuals – sum of differences between prediction and actual

Univariate Linear Regression, Multivariate Linear Regression

Learning Rate Optimization Methods – Adam, Adagrad, RMSProp, Momentum

**Custom Learning Rate Optimizer**

* With every iteration of GD, calculate the exact value of MSE and store it!
* After running an iteration of GD, look at the current MSE and the old MSE
* If the MSE went “up” then we did a bad update, so divide learning rate by 2
* If the MSE went “down” then we are going in the right direction! Increase LR by 5%

**Gradient Descent – Use entire feature set to update M and B**

**Batch Gradient Descent – Use a couple observations at a time to update M and B**

**Stochastic Gradient – Use one observation at a time to update M and B**

**Batch Gradient Descent**

* Guess a starting value of B and M (and M2, M3, etc.)
* Calculate the slope of MSE using a portion of observations in feature set and current M/B values
* Multiply the slope by the learning rate
* Update B and M

**Stochastic Gradient Descent (SGD)**

* Guess a starting value of B and M (and M2, M3, etc.)
* Calculate the slope of MSE using one observation in feature set and current M/B values
* Multiply the slope by the learning rate
* Update B and M

**Linear Regression – predicts continuous values**

**Logistic Regression – predicts discrete values (classification)**

**Binary Clasification (pass, not pass)**

Given a person’s age, do they prefer to read books or watch movies?

One feature- age

Two labels – read books, watch movies

Find a mathematical relationship (a formula) that relate’s a person’s age to whether they like to read books or watch movies

Preference = m \* age + b

Sigmoid =

E is Euler’s Constant ~2.718

Always produces values between 0 and 1

Probability of being the ‘1’ label =

**Logistic Regression Gradient Descent**

* Encode label values as either ‘0’ or ‘1’
* Guess a starting value of B and M (and M2, M3, etc.)
* Calculate slope of MSE using all observations in feature set and current M/B values
* Multiply the slope by the learning rate
* Update B and M

Given a vehicle’s weight, horsepower, and engine displacement, will it PASS or NOT PASS a smog emissions check?

**Multi-nominal Logistic Regression Multiple Classification Options**

Given the horsepower, weight, and displacement of a vehicle, will it have a high, medium, or low fuel efficiency?

Define bounds

**Marginal Probability Distribution – Considers one possible output case in isolation**

**Conditional Probability Distribution – Considers all possible output cases together**

Given the pixel intensity values in an image, identify whether the character is a hand-written 0,1,2,3,4,5,6 7,8,9

MNIST database

Training set: 60,000 images

Test set: 20,000 images