A comparison of meta-learning strategies

John Liddell

1200 N Dupont Hwy, Dover, DE 19901

11 01 2019

- Machine learning algorithm decision is not arbitrary
- No Free Lunch for Optimization
- Does NFL theory apply to meta-learners?

- Meta Learners
  - Brute Force - Cluter on all datasets in metabase
  - Active Meta Learning - Cluster on datasets in metabase with most information
  - Learning Curves - Choose item in metabase with learning curve most similar to new dataset
- Base Algorithms
  - Linear Regression - Fit Curve to data
  - Support Vector Machine - Sepearte data with hyperplane
  - K-Means Clustering - Draw borders around similar datapoints
  - Naive Bayes - Guess Likelihood of data class with Bayes theorem
  - Neural Networks - Decide class of data thru interconnected network of weights

- Brute Force Meta Learner
  - Most basic possible meta learning strategy
  - Gather run statistics for a group of datasets (metabase)
  - Take in new dataset, run clustering algorithm with metabase
  - Use algorithm of closest set in metabase
- Active meta-learning
  - Meta Learning strategy described in “Ranking of Classifiers based on Dataset Characteristics using Active Meta Learning”
  - Allows a dataset into metabase only if it has a higher uncertainty score than its peers
  - Relative uncertainty between two datasets:

$$\delta(V_x, d_i, V_x, d_j) = \frac{|V_x, d_i - V_x, d_j|}{Max_{k \neq i}(V_x, d_k) - Min_{k \neq i}(V_x, d_k)}$$

    - where:
    - V_x, d_k = value of metapamater V_x for dataset d_k
    - $Max_{k \neq i}(V_x, d_k)$ = Maximum V_x, d_k with dataset i removed
    - $Min_{k \neq i}(V_x, d_k)$ = Minimum V_x, d_k with dataset i removed

- Active meta-learning (cont)
  - Selection accomplished by summing uncertainties, ranking, then selecting highest ranked dataset
  - Process reduces training time and increases classification accuracy relative to Brute Force Learner
- Nearest Learning Curve Analysis
  - Gather algorithms classification accuracy for some dataset at various fractions of the training set
  - Plotting these accuracies reveals a learning curve
  - Categorization of the new datasets then accomplished in 3 steps:

    First Train model with each candidate algorithm at same fractions present in learning curves in metabase

    Second Get distance measure between this new curve and older curves

    Third Return algorithm that worked best on dataset represented by curve

Methodology

- Overall goal: Determine meta-learning dominance
- Requirements:
  - Set of meta learning strategies to compare
  - A pool of datasets from which to build metabases and on which to analyze performance
  - Analysis techniques to compare meta learners performances
- Program Flow:
  - Parse unprocessed datasets
  - Run base algorithms
  - Collect learning curves
  - Construct metabase sets
  - Populate meta learner guess tables
  - Compile results
  - Produce results charts

- Development Environment Description:
  - languages: python 3.7, bash
  - editors: emacs, pycharm
  - runtime environment: ipython in powershell
  - Personal pc metrics:
    - RAM: 16 GB
    - Processor: Intel i5-4460
    - OS: Windows 10
- Datasource Description:
  - Gathered with script from UCI Irvine Machine Learning Repository
  - Manual investigation of parability:

    First Write code to parse data

    Second Examine data to see if parsed

    Third If not parsed, discover why parse failed

    Fourth Repeat Sequence
  - Resulting parser flow:

    First Ensure file of allowed type then import if allowed

    Second Check each columns data type

    Third Transform unusable columns

- System required generally applicable “meta features”: features the meta learning algorithms could use to classify the datasets
- A set of meta features that met this criteria were:
  - weighted mean - The mean of a distribution normalized by its maximum value
  - coefficient of variation - A measure of the dispersion of a distribution.
  - skewness - A measure of the asymmetry of a probility distribution.
  - kurtosis - A measure of the “tailedness of a probibility distribution i.e. how many of much of the weight of a probabiliy distribution lies in its tails
  - shannon entropy - A measure of the minimum number of bits needed to encode a string of symbols. Is a measure of how much information is contained in a body of data.

- A vector is formed using these meta features via the following process:

First Apply meta feature to each column

Second Sum these values, divide by number of columns

$$F_{ad} = \frac{\sum_{c=i}^{N} f_{ai}}{N}$$

  - where:
  - F_{ad} Composite meta feature value
  - a = label of meta feature
  - d = label of dataset
  - c = iterator across columns in the dataset
  - f_{ai} = value of meta feature a in column i
  - N = overall number of columns in dataset

Third Repeat for other meta features

Fourth Craft vector with features

$$V_d = (F_{1d}, F_{2d}, ...F_{ad})$$

Database Description

- Database created and used via a python library called sqlalchemy
  - Object relation mapper - maps data structures to sql statements
- Database Tables:
  - algorithm - each row contains information on a basic algorithm
  - all_data - each row contains name,path, and vector representation of a dataset
  - runs_all - each row contains a combination of algorithm, dataset, training time, and test accuracy
  - learning_curves - each row contains test set accuracy at 10, 20, and 30 percent training set size
  - base_set_collection tables - each table contains a set of metabases
  - guesses tables - each table contains meta algorithm guesses
  - results - each row contains meta algorithm name, collection table name, metabase name, accuracy, training time

- Ran all algorithm/dataset combinations and stored in table
- Analysis session flow:

First Parse dataset for data matrix
Second Randomize order of the data matrix's rows
Third Use first 20 percent to train algorithm
Fourth For each algorithm:
  - Train a model with given algorithm
  - Analyze test set with trained model
  - save classification accuracy and training time to database

- Same procedure used to gather learning curves but training occured only at 10, 20, and 30 percent the size of the training set

- Metabase Collections:
  - 30 metabase collections
  - Each collection contains 10 metabase sets
  - Each metabase contains 10 datasets
  - Datasets choosen at random from pool of datasets
- Meta Learners Tested:

  First Metabase Collection, metabase, meta algorithm combination selected

  Second The meta learner uses its strategy to train a model

  Third For every dataset not in the current metabase, the model is to guess what algorithm would best classify that dataset

  Fourth Repeated with every combination of metabase collection, metabase, and meta algorithm
- Results Database table:
  - Each row contains meta algoriothm, collection table name, metabase name, training and accuracy combination

Findings

- A results matrix is crafted for each metabase collection:
  - Matrices contain “placement results”, how well meta algorithms did relative to one another
  - Row values always sum to 10
- Null Hypothesis - The meta learning algorithms are truly equal.
  - The null hypothesis being true would result in the average placement result being 3.3

	GuessesActive			GuessesEx			Gue	
	First	Second	Third	First	Second	Third	First	S
sample 1	1	4	5	6	2	2	3	
sample 2	1	4	5	5	2	3	4	
sample 3	1	3	6	7	3	0	2	
sample 4	1	5	4	6	3	1	3	
sample 5	0	6	4	8	2	0	2	
sample 6	3	3	4	5	4	1	2	
sample 7	4	3	3	4	4	2	2	
sample 8	2	3	5	7	2	1	1	
sample 9	1	3	6	3	5	2	6	
sample 10	0	4	6	7	3	0	3	
sample 11	0	6	4	7	3	0	3	
sample 12	1	5	4	7	2	1	2	
sample 13	3	3	4	5	4	1	2	

Average across samples for placement results

	GuessesActive	GuessesEx	GuessesSamp
First	1.70	3.8	4.50
Second	5.57	3.3	1.13
Third	2.73	2.9	4.37

Table: Average placement results across all samples

- Measures the likelihood of some data given some expectation
- Equation used to calculate it is:

$$t = \frac{\overline{x} - \mu}{\hat{\sigma}_{\overline{x}}} = \frac{\overline{x} - \mu}{\frac{s}{\sqrt{N}}}$$

- where:
  - s = sample standard deviation
  - N = number of samples
  - $\overline{x}$ = sample mean
  - μ = expected mean

Sample standard deviations

	GuessesActive	GuessesEx	GuessesSamp
First	1.42	1.30	1.48
Second	1.54	1.53	1.06
Third	1.36	1.33	1.58

Table: Placement results standard deviations

t SCORES OF PLACEMENT AVERAGES

	GuessesActive	GuessesEx	GuessesSamp
First	-6.29	1.98	4.32
Second	7.97	-0.11	-11.36
Third	-2.42	-1.77	3.61

Table: t scores of placement averages

Conclusion and recommendations

- Composite *t* score
  - Mean of the absolute value of the *t* scores
  - Obtain normalized *t* score of 4.42
  - Can thus reject the null hypothesis
- Desired Followup
  - Same experiment with 3000 datasets would remove possibility of data bias

Thank you for your time, have a nice day

References