Automated Feature Engineering for Predictive Modeling

Udayan Khurana, Horst Samulowitz, Fatemeh Nargesian (University of Toronto), Tejaswini Pedapati, Elias Khalil (Georgia Tech), Gregory Bramble, Deepak Turaga, Peter Kirchner

Saket Sathe @ ICDM
“Kernel-Based Feature Extraction For Collaborative Filtering”
[also has a very nice paper at KDD-17: “Similarity Forests”]
Data Science Workflow

Ingestion
- Retrieval
- Storage
- Formatting
- ...

Selection
- Data Source Selection
- Data Composition
- Data Linkage
- Concept Extraction
- Filtering
- ...

Preparation
- Missing Values
- Smoothing
- Normalization
- ...

Generation
- Aggregation
- Construction
- Labelling
- Data Augmentation
- ...

Transform
- Feature selection
- Feature space transformation
- ...

Model
- Regression
- Classification
- ...

Operations
- (Re)-Deployment, Re-Training, Monitor
- Explanations
- Written Report
- Best-Worst case scenarios

Oil Rig Monitoring

Noisy Sensor Streams

Cleaned sensor streams

Model
Data Science Workflow – and its Automation

Ingestion:
- Retrieval
- Storage
- Formatting
- ...

Selection:
- Data Source Selection
- Data Composition
- Data Linkage
- Concept Extraction
- Filtering
- ...

Preparation:
- Missing Values
- Smoothing
- Normalization
- ...

Generation:
- Aggregation
- Construction
- Labelling
- Data Augmentation
- ...

Transform:
- Feature selection
- Feature space transformation
- ...

Model:
- Regression
- Classification
- ...

Report:
- Explanations
- Written Report
- Best-Worst case scenarios
- ...

Automated Feedback
Feature Representation

- Why does feature representation matter?
  - Consider building a classifier using a straight line

picture source: Deep Learning by Goodfellow et al.
Feature Engineering Example

- Problem: Kaggle DC bikeshare rental prediction
- Regression with Random Forest Regressor
- Original features *(relative abs. error = 0.61)*:

<table>
<thead>
<tr>
<th>datetime</th>
<th>season</th>
<th>holiday</th>
<th>workingday</th>
<th>weather</th>
<th>temp</th>
<th>atemp</th>
<th>humidity</th>
<th>windspeed</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-01-01 00:00:00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9.84</td>
<td>14.395</td>
<td>81</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>2011-01-01 01:00:00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9.02</td>
<td>13.635</td>
<td>80</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>2011-01-01 02:00:00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9.02</td>
<td>13.635</td>
<td>80</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>

- Added features *(relative abs. error = 0.20)*:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>0</td>
<td>4.394449</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1.711228</td>
</tr>
<tr>
<td>2011</td>
<td>1</td>
<td>4.382027</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1.711030</td>
</tr>
<tr>
<td>2011</td>
<td>2</td>
<td>4.382027</td>
<td>0</td>
<td>0.5</td>
<td>1.442695</td>
<td>-1.710832</td>
</tr>
</tbody>
</table>
Overview: Feature Engineering

- Feature Engineering describes the transformation of a given dataset’s feature space:
  - In order to improve learning accuracy.
  - Through generating new features, removing unnecessary ones.
  - Performed by a data scientist.
  - Occupies a bulk of the modeling time.

Original Data

<table>
<thead>
<tr>
<th>0</th>
<th>2,4,6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,4.2,0</td>
</tr>
<tr>
<td>0</td>
<td>1,5,4.2</td>
</tr>
<tr>
<td>1</td>
<td>6.9,4,3</td>
</tr>
</tbody>
</table>

Transformed Data

<table>
<thead>
<tr>
<th>0</th>
<th>2,4,6,1,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,4.2,0,9</td>
</tr>
<tr>
<td>0</td>
<td>1,5,4,2,1</td>
</tr>
<tr>
<td>1</td>
<td>6.9,4,3,47.6</td>
</tr>
</tbody>
</table>
Ways of Feature Engineering

(1) Data scientist's **expertise** gives a hunch on which transformations to try.

log(x), x+y, x², frequency(x)

(2) Performed iteratively through **trial and error**.
Ways of Feature Engineering

1. **Data scientist’s expertise** gives a hunch on which transformations to try.

2. Performed iteratively through trial and error.

**Original Data**

<table>
<thead>
<tr>
<th>log(x), x+y, x², frequency(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

**Transformed Data**

**Model Building and Validation**
How to get a good hunch?

- Make a theoretical model
  - Consult an expert in the domain.
  - Build hypothesis and verify them with data
  - Come up with data enhancement options

- Limitations
  - Dependent on human effort
  - Expensive and time consuming
  - Data and theory are different
  - Dataset may not be descriptive

Entity Relationship diagram for Amazon Resource Kaggle challenge dataset
Feature Engineering and its Automation

• Introduction
• Problem Definition and Complexity
• Performance driven methods
  • Hierarchical function based approach
  • Reinforcement Learning based policy-learning
    [“Feature Engineering for Classification using Reinforcement Learning”, AAAI’2018]
• Learning Feature Engineering
  [“Learning Feature Engineering for Classification, IJCAI’2017”]
• Combined Demo: “Dataset Evolver”
  https://www.youtube.com/watch?v=4T8KaeOn-2Y
• Automated Model Selection
• Automated Neural Network Composition
Problem Definition

• Given a predictive modeling task:
  • Set of $m$ features: $F = \{f_1, f_2 \ldots f_m\}$
  • target vector: $y$
  • a modeling algorithm, $M$
    • $A_M(F, y)$ reflects model performance
  • $k$ transform functions: $t_1, t_2, \ldots t_k$

• A sequence of transformations $S = t^{(1)}(t^{(2)} \ldots (f_i))$

• **Problem**: Find a set of sequences of transformations $S = \{s_1, .. s_r\}$
  • $F_{\text{new}} = F' + S$, where $F'$ is a subset of $F$
  • $\text{argmax}(S) P_M(F_{\text{new}}, y)$
Complexity and Brute force

- For $k$ features and $r_1$ = unary transforms, $d$ = depth
  - $s_1 = (k \times r_1)^{d+1}$
  - For $k=10, r_1=10, d=5; s_1 = 10^{12}$
- For $r_2$ binary transforms
  - $s_2 = (C_{(k,2)} \times r_2)^{d+1}$
  - For $k=10, r_2=10, d=5; s_2 = 10^{15}$
- For each case, verification involves training and evaluating a model
- It is clearly computationally infeasible to verify all possibilities
Existing approaches...

- **Expand-Reduce**
- DSM [Kanter et al. DSAA 2015], OneBM [arXiv 2017]
  - Applies all transforms at once \( \{f_1, f_2 \ldots f_m\} \times \{t_1, t_2, \ldots t_k\} \Rightarrow (m \times k) \) features
  - Followed by a feature selection (FS) step and parameter tuning
  - Positive: One modeling step (excluding FS)
  - Limitation: Doesn’t consider compositions of functions
  - Limitation: FS is a performance bottleneck due to large \( (m \times k) \) features
Existing approaches...

- **Evolution-centric**
  - ExploreNet [Katz et al. ICDM 2016]
  - Adds one feature at a time and performs model building and verification.
  - Runs in a greedy manner until time runs out
  - Positive: More scalable than expand-reduce method
  - Limitation: Extremely Slow
Hierarchical organization of transformations

- Transformation $T$ applied to feature set applies function to all valid input features and appends new columns to the existing ones.
- Exploration using search strategy guided by performance accuracy under a constrained budget.

Example of a Transformation Graph, which is a directed acyclic graph. The start node $D_0$ corresponds to the given dataset; that and the hierarchical nodes are circular. The sum nodes are rectangular. In this example, we can see three transformations, log, sum, and square, as well as a feature selection operator $FS_1$.

Khurana et al.: Cognito: Automated Feature Engineering for Supervised Learning [ICDM ’16]
Selected factors influential in policy decisions

- **Node n’s Accuracy**: Higher accuracy of a node incentivizes further exploration from that node, compared to others.

- **Transformation, t’s average performance until Gi**: Using t’s mean performance in the transformation graph so far, we compare potential gains from it, compared to others.

- **Frequency of a transform in the path from root node to n**: can be used to discount the potential gains from application of t if it has already been applied to a descendant of n.

- **Accuracy gain for n’s parents**: While n’s accuracy itself is a factor, so is the gain from its parent (and the same for its parent), indicating the focus on more promising regions of the graph.

- **Node Depth**: A higher value is used to penalize the relative complexity of the transformation sequence (overfitting).

- **Remaining budget fraction**: The budget is measured by the number of maximum nodes allowed for the graph. At each step, the fraction of remaining budget is a factor in determining the trade-off in exploration versus exploitation.

- **Ratio of feature counts in n to the original dataset**: This indicates the bloated factor of the dataset.

- **Is transformation a feature selector or not (augmenter)?**
Hierarchical organization of transformations

• Emulates a human trial and error process

• Performance Driven Traversal Strategies:
  • Depth Oriented
  • Breadth Oriented
  • Mixed (budgeted)
  • Reinforcement learning-based (next)

• Advantages:
  • Allows composition of transforms
  • Batching improves performance
  • Data-level transformations are logical blocks for measuring performance

• Demo:
  • Cognito: https://www.youtube.com/watch?v=hJlG0mvynDo
Examples of different policies

Breadth

Depth

Mixed (RL)
Policy Learning with Reinforcement Learning

- Strategy learned with experience over various datasets.
- Consider it is a *Markov Decision Process (MDP)*.
- A state (snapshot) of TG is a state of MDP.
  - A state is described by an array of TG factors and remaining budget.
  - We learn transitions from one state to another
- Objective in learning:
  - Short term goal: Balance exploration and exploitation.
  - Final Goal: maximize the final delta in accuracy
RL Modeling

- Immediate reward
  \[ r_i = \max_{n' \in \Theta(G_{i+1})} A(n_{i+1}) - \max_{n \in \Theta(G_i)} A(n_i) \]

- Cumulative reward
  \[ R(s_i) = \sum_{j=0}^{B_{\text{max}}} \gamma^j r_{i+j} \]

- Q-function
  \[ Q(s, c) = r(s, c) + \gamma R^{\Pi}(\delta(s, c)) \]

- Optimal Policy:
  \[ \Pi^*(s) = \arg \max_c [Q(s, c)] \]

- Approximation of Q-function:
  \[ Q(s, c) = w^c . f(s) \]

- Update rule for \( w_c \):
  \[ w_{cj}^{c_j} \leftarrow w_{cj}^{c_j} + \alpha (r_j + \gamma \max_{n', t'} Q(g'_{n'}, c') - Q(g, c)).f(g, b) \]
System Overview

Preprocessing
- Preparation: Type inference, format conversion, missing values completion
- EDA: Explores correlations, feature ranks, outliers, redundant columns, infers entities
- Tagging: Tags column based upon inferences, types and correlations

Feature Engineering
Explores feature space in Transformation Graph combination of construction and selection options through reinforcement learning.

- Utilizes off the shelf model selection to work with a few good models.
- Employs sampling and estimation to scale by further pruning the graph

Post Feature Engineering Exploration
Trains models on a diverse set nodes based on exploration of transformation graphs. Ensembles are constructed based on combinations of such models and the best overall is suggested for deployment.

Nodes
{ O O O O O }

Learning Algos
{ RF LR DT NB }

Models
M_1 M_2 M_3 ...... M_K

Ensembles
E_1 E_r
### Results

Comparing accuracy between base dataset (no FE), Our FE, DSM inspired FE, Random FE, and Cognito using 24 datasets. Performance here is unweighted average FScore for classification and (1 rel. absolute error) for regression.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source</th>
<th>C/R</th>
<th>Rows</th>
<th>Features</th>
<th>Base</th>
<th>Cognito RL</th>
<th>DSM FE</th>
<th>Random FE</th>
<th>Cognito Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higgs Boson</td>
<td>UCIrvine</td>
<td>C</td>
<td>50000</td>
<td>28</td>
<td>0.718</td>
<td>0.729</td>
<td>0.682</td>
<td>0.699</td>
<td>0.702</td>
</tr>
<tr>
<td>Amazon Employee</td>
<td>Kaggle</td>
<td>C</td>
<td>32769</td>
<td>9</td>
<td>0.712</td>
<td>0.806</td>
<td>0.744</td>
<td>0.740</td>
<td>0.714</td>
</tr>
<tr>
<td>PimaIndian</td>
<td>UCIrvine</td>
<td>C</td>
<td>768</td>
<td>8</td>
<td>0.721</td>
<td>0.756</td>
<td>0.712</td>
<td>0.709</td>
<td>0.732</td>
</tr>
<tr>
<td>SpectF</td>
<td>UCIrvine</td>
<td>C</td>
<td>267</td>
<td>44</td>
<td>0.686</td>
<td>0.788</td>
<td>0.790</td>
<td>0.748</td>
<td>0.780</td>
</tr>
<tr>
<td>SVMGuide3</td>
<td>LibSVM</td>
<td>C</td>
<td>1243</td>
<td>21</td>
<td>0.740</td>
<td>0.776</td>
<td>0.711</td>
<td>0.753</td>
<td><strong>0.766</strong></td>
</tr>
<tr>
<td>German Credit</td>
<td>UCIrvine</td>
<td>C</td>
<td>1001</td>
<td>24</td>
<td>0.661</td>
<td>0.724</td>
<td>0.680</td>
<td>0.655</td>
<td>0.662</td>
</tr>
<tr>
<td>Bikeshare DC</td>
<td>Kaggle</td>
<td>R</td>
<td>10886</td>
<td>11</td>
<td>0.393</td>
<td>0.798</td>
<td>0.693</td>
<td>0.381</td>
<td>0.790</td>
</tr>
<tr>
<td>Housing Boston</td>
<td>UCIrvine</td>
<td>R</td>
<td>506</td>
<td>13</td>
<td>0.626</td>
<td><strong>0.680</strong></td>
<td>0.621</td>
<td>0.637</td>
<td>0.652</td>
</tr>
<tr>
<td>Airfoil</td>
<td>UCIrvine</td>
<td>R</td>
<td>1503</td>
<td>5</td>
<td>0.752</td>
<td><strong>0.801</strong></td>
<td>0.759</td>
<td>0.752</td>
<td>0.794</td>
</tr>
<tr>
<td>AP-omentum-ovary</td>
<td>OpenML</td>
<td>C</td>
<td>275</td>
<td>10936</td>
<td>0.615</td>
<td><strong>0.820</strong></td>
<td>0.725</td>
<td>0.710</td>
<td>0.758</td>
</tr>
<tr>
<td>Lymphography</td>
<td>UCIrvine</td>
<td>C</td>
<td>148</td>
<td>18</td>
<td>0.832</td>
<td><strong>0.895</strong></td>
<td>0.727</td>
<td>0.680</td>
<td>0.849</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>UCIrvine</td>
<td>C</td>
<td>351</td>
<td>34</td>
<td>0.927</td>
<td><strong>0.941</strong></td>
<td>0.939</td>
<td>0.934</td>
<td><strong>0.941</strong></td>
</tr>
<tr>
<td>Openml_618</td>
<td>OpenML</td>
<td>R</td>
<td>1000</td>
<td>50</td>
<td>0.428</td>
<td><strong>0.587</strong></td>
<td>0.411</td>
<td>0.428</td>
<td>0.532</td>
</tr>
<tr>
<td>Openml_589</td>
<td>OpenML</td>
<td>R</td>
<td>1000</td>
<td>25</td>
<td>0.542</td>
<td><strong>0.689</strong></td>
<td>0.650</td>
<td>0.571</td>
<td>0.644</td>
</tr>
<tr>
<td>Openml_616</td>
<td>OpenML</td>
<td>R</td>
<td>500</td>
<td>50</td>
<td>0.343</td>
<td><strong>0.559</strong></td>
<td>0.450</td>
<td>0.343</td>
<td>0.450</td>
</tr>
<tr>
<td>Openml_607</td>
<td>OpenML</td>
<td>R</td>
<td>1000</td>
<td>50</td>
<td>0.380</td>
<td><strong>0.647</strong></td>
<td>0.590</td>
<td>0.411</td>
<td>0.629</td>
</tr>
<tr>
<td>Openml_620</td>
<td>OpenML</td>
<td>R</td>
<td>1000</td>
<td>25</td>
<td>0.524</td>
<td><strong>0.683</strong></td>
<td>0.533</td>
<td>0.524</td>
<td>0.583</td>
</tr>
<tr>
<td>Openml_637</td>
<td>OpenML</td>
<td>R</td>
<td>500</td>
<td>50</td>
<td>0.313</td>
<td><strong>0.585</strong></td>
<td>0.581</td>
<td>0.313</td>
<td>0.582</td>
</tr>
<tr>
<td>Openml_586</td>
<td>OpenML</td>
<td>R</td>
<td>1000</td>
<td>25</td>
<td>0.547</td>
<td><strong>0.704</strong></td>
<td>0.598</td>
<td>0.549</td>
<td>0.647</td>
</tr>
<tr>
<td>Credit Default</td>
<td>UCIrvine</td>
<td>C</td>
<td>30000</td>
<td>25</td>
<td>0.797</td>
<td><strong>0.831</strong></td>
<td>0.802</td>
<td>0.766</td>
<td>0.799</td>
</tr>
<tr>
<td>Messidor_features</td>
<td>UCIrvine</td>
<td>C</td>
<td>1150</td>
<td>19</td>
<td>0.691</td>
<td><strong>0.752</strong></td>
<td>0.703</td>
<td>0.655</td>
<td>0.762</td>
</tr>
<tr>
<td>Wine Quality Red</td>
<td>UCIrvine</td>
<td>C</td>
<td>999</td>
<td>12</td>
<td>0.380</td>
<td><strong>0.387</strong></td>
<td>0.344</td>
<td>0.380</td>
<td>0.386</td>
</tr>
<tr>
<td>Wine Quality White</td>
<td>UCIrvine</td>
<td>C</td>
<td>4900</td>
<td>12</td>
<td>0.677</td>
<td><strong>0.722</strong></td>
<td>0.654</td>
<td>0.678</td>
<td>0.704</td>
</tr>
<tr>
<td>SpamBase</td>
<td>UCIrvine</td>
<td>C</td>
<td>4601</td>
<td>57</td>
<td>0.955</td>
<td><strong>0.961</strong></td>
<td>0.951</td>
<td>0.937</td>
<td>0.959</td>
</tr>
</tbody>
</table>
Results

Various Search Policies

![Bar chart showing the performance of different search policies with 'Handcrafted' and 'Learned' labels.]
Predicting Transformations

• Do we really need to trial and error?
  • Can’t we just see and tell which transforms are useful?

• Patterns not visible to the human eye
  • Use machine (learning) to crunch the patterns?

• Challenges in learning across multiple datasets:
  • Datasets have varying shapes
  • Datasets represent different problems

• We present LFE (Learning Feature Engineering):
  • Novel representation of data using data sketches
  • Predict the most useful transform for each feature
  • Learn across thousands of open source classification datasets
Learning-based Feature Engineering

- Learn correlations between feature distributions, target distributions and transformations
  - Build meta-models to predict good transformations through past observation
  - Generalize over 1000s of datasets across a variety of domains
- **Main Challenge:** Features vectors are of different sizes
- **Solution:** Quantile Sketch Array to capture the essential character of a feature.

An example of feature representation using **quantile sketch array (QSA)**. The feature $f1$’s values are binned into 10 equiwidth bins, separately for classes $-1$ and $+1$. The two resulting vectors are then concatenated and fed into the trans- formation $t_i$’s classifier, which in turn recommends for or against applying $t_i$ on $f1$.

Nargesian e al. Feature Engineering for Classification [IJCAI 17]
Experiments

The percentage of datasets, from a sample of 50, for which a feature engineering approach results in performance improvement (measured by F1 score of 10 fold cross validation for Random Forest and Logistic Regression)
Experiments

Statistics of Datasets and F1 Score of LFE and Other Feature Engineering Approaches with 10-fold Cross Validation of Random Forest.
Live Demo [Cognito + LFE]
Combined: Explorer+Predictor

- Coming soon:

**Automating Feature Engineering**
Udayan Khurana, Fatemeh Nargesian, Horst Samulowitz, Elias Khalil, Deepak Turaga
*NIPS workshop on Artificial Intelligence for Data Science, 2016*
How to deal with model and data dependency?
Data Allocation using Upper Bounds (DAUB)

• **Question**: How can one robustly project the accuracy at $n$ data points to the expected accuracy at $N$ data points?

• **Proposal**: Apply principle of “Optimism under Uncertainty”

  • **DAUB**’s upper bound is based on first order Taylor expansion of unknown reward function $f(N)$

  • Using discrete derivative $f'(n,s) = (f(n) - f(n-s)) / s$ where $s$ a natural number

  • Allocate more training data to approach with best expected performance
Bandit-based Algorithm – An Example

- Logistic Regression
- Random Forest
- SVM

# Additional Data points

Built Model

Prediction Accuracy versus #data points

Training Data

Upper bound estimate on performance
Theoretical Support for DAUB

• **Using following assumptions:**
  1. Learning curves are monotone
  2. Diminishing returns
  3. Access to *true* accuracy/cost (instead of *observed* quantities)
    • 1+2 can be enforced in practice, but 3. would be too expensive in practice – however as number of allocated data points increases observed accuracy converges to true accuracy

• **Analysis of idealized DAUB = DAUB***
  • Training data allocation sequence chosen by DAUB* is essentially optimal
Bounded Regret of DAUB*

• Learner \( f \) is called **D-optimal** iff \( f(N) \geq f^*(N) - D \)

• **Cost**\((f)\) = computational cost of training \( f \) on \( N \) data points

• **D-Regret**\((f)\) = cost spent on **D-suboptimal** learner \( f \)

**Theorem**#: If DAUB* selects \( f' \) and \( f \) is any D-suboptimal learner, then

1. \( f' \) is D-optimal

2. D-regret\((f)\) is sub-linear in cost\((f')\)

3. The bound on D-regret\((f)\) is tight up to a constant factor

[#for more details on assumptions etc. please refer to the paper “]
Things to consider in practice

• Learning curves are not always monotone, but....
  • Simple to employ techniques that ensure monotonicity (e.g. monotone regression)

• Ways to improve accuracy estimate
  • Projected accuracy can be very inaccurate (especially in the beginning)
    • In particular it can be often above 100% (e.g., 294%), one could cap at 100%, but loses ranking information
  • Use Training Performance to improve estimate
    • Assuming that training and test data come from the same distribution the training performance (accuracy at sample size) can serve as upper bound!
    • **Assumption:** Training Error < Validation Error
    • Upper Bound Estimate = choose minimum of projected accuracy and training accuracy
The graph illustrates the prediction accuracy (%) over the number of training instances for various machine learning models. The models include SimpleCart, LADTree, J48, RandomForest, DTNB, and REPTree.

The table below summarizes the full training and DAUB (Data Augmentation for Unsupervised Bias) results for different datasets in terms of allocation, time (s), iterations, allocation, time (s), speedup, and loss:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Application Area</th>
<th>Full Training</th>
<th>DAUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buzz</td>
<td>social media</td>
<td>1,578k, 56,519</td>
<td>57, 302k, 5,872</td>
</tr>
<tr>
<td>Cover Type</td>
<td>forestry</td>
<td>1,578k, 43,578</td>
<td>13, 160k, 3,848</td>
</tr>
<tr>
<td>HIGGS</td>
<td>signal processing</td>
<td>1,578k, 49,905</td>
<td>56, 372k, 2,001</td>
</tr>
<tr>
<td>Million Songs</td>
<td>music</td>
<td>1,578k, 115,911</td>
<td>53, 333k, 17,208</td>
</tr>
<tr>
<td>SUSY</td>
<td>high-energy physics</td>
<td>1,578k, 26,438</td>
<td>31, 214k, 837</td>
</tr>
<tr>
<td>Vehicle Sensing</td>
<td>vehicle management</td>
<td>1,578k, 68,139</td>
<td>50, 296k, 5,603</td>
</tr>
</tbody>
</table>
Feature Engineering with Neural Networks
FE with Deep Learning

• Deep nets perform feature engineering
• Incredibly successful in several domains

• However,
  • Need extensive amount of data
  • Lack interpretability
  • Not known to work generally in all domains
  • Need architectural configuration for a new problem
Automated Composition of Neural Networks
Automated Composition of Neural Networks

- Problem in Deep Learning is not so much the large variety of analytics, but enormous configuration space of a single Deep Neural Network (DNN)
- Wide range of settings and combinations to choose from:
  - Learning Rate
  - Number of layers
  - Number of nodes per layer
  - Activation function per layer
  - Pre-Train yes or no
  - Drop-Out rate
  - ...

- Ideas:
  - Combine learning curve estimation procedure with hyper-parameter optimization
  - Apply mathematical optimization

Bandit-Based approach applied to NN search

- Use the number of epochs vs performance to estimate performance of a NN configuration
  - Use slope of learning curve to estimate performance on full training

- Support of arbitrary framework (e.g., THEANO, TORCH, TENSORFLOW) through wrapper interface
  1. Start with a default network, parameters and ranges are specified in JSON
  2. Perform hyper-parameter optimization (how many nodes per layer, activation function, learning rate, drop out, etc.) but only allow limited number of epochs
  3. Perform DAUB estimation on new configurations and allocate more epochs based on estimated performance
- Deploy on GPU cluster
Automated Composition of Neural Networks

Idea — Optimize NN Model architectures and parametrization to solve a specific ‘cognitive’ task

Optimizer
generate new params

parameters
architecture
learning rate
kernel size
# filters

generate Neural Net

train model on all data

evaluate model

0.78 acc