Recognizing Variables from their Data via Deep Embeddings of Distributions

Jonas Mueller, Alex Smola

jonasmue@amazon.com

Amazon Web Services

Automating machine learning and analytics

Given a new dataset to model:

Current AutoML:

- Try applying many models and report which one works best
- Expensive, brute-force, suboptimal search for best model
- Has no idea where the data comes from

Human Analyst:

- Understand what variables generated the data
- Recall previously-analyzed datasets generated from similar variables
- Use previous experience to propose promising models for the new data

Ingredients for successful AutoML

1. Organized repository of different datasets annotated with informative metadata regarding the performance of various models [1]

2. Ability to recognize which repository datasets (and best models associated with them) are relevant when presented with new data

[1] Examples: OpenML, kaggle.com

Overview

Objective: Given new data from unknown variable, identify which previously-seen datasets stem from the same variable

Applications: AutoML, semantic labeling (eg. identifying PII data), automated transforms, schema-matching, dataset search

Approach: Use neural network to embed each dataset as vector, such that similar variables' data have nearby embeddings

Setup

- Repository $\mathcal{R}$ containing N datasets $\mathcal{D}_1, \ldots, \mathcal{D}_N$
- Each dataset $\mathcal{D}_i$ stems from a single variable v_i
  and is comprised of IID observations $x_1, \ldots, x_{n_i} \sim P_i$
- Some datasets in $\mathcal{R}$ annotated as matched variables $v_i = v_j$
- Identify which of $\mathcal{D}_1, \ldots, \mathcal{D}_N$ stem from same variable as new $\mathcal{D}_*$

Statistical similarity

Many measures of statistical difference can be expressed:

$$d(P_1, P_2) = \left|\left|\mathbb{E}_{P_1}[h(x)] - \mathbb{E}_{P_2}[h(x)]\right|\right|$$

with some feature map h (eg. summary statistics, histograms, RKHS)

Let $h : x \rightarrow \mathbb{R}^k =$ neural network used to embed datasets[2]

$$d_h(\mathcal{D}_1, \mathcal{D}_2) = ||h(\mathcal{D}_1) - h(\mathcal{D}_2)||_2^2 \text{ with } h(\mathcal{D}_i) = \frac{1}{|\mathcal{D}_i|} \sum_{x \in \mathcal{D}_i} h(x)$$

[2] Zaheer et al. (2017). *Deep Sets*.

Key issues

- Provided labels (ie. column names) for datasets are often uninformative, or not standardized across groups

  ⇒ we use raw data values to gauge variable similarity

- Standard statistical similarity measures fail to:

  1. Ignore natural variation between datasets containing measurements of the same type of variable (eg. *temperature* in Celsius vs Fahrenheit)
  2. Distinguish different variables whose data distributions happen to be identical (eg. binary-valued variables: *true*/*false* or *yes*/*no*)
  3. Facilitate efficient identification of datasets with matched variables (our vector embeddings enable approximate nearest neighbor search)

Modeling variable matches

- $p_{ij} = \exp(-D_{ij}) :=$ probability $\mathcal{D}_i$ and $\mathcal{D}_j$ stem from same variable
- $D_{ij} = d_h(\mathcal{D}_i, \mathcal{D}_j) + g(\mathcal{D}_i) + g(\mathcal{D}_j)$
- $g : x \to \mathbb{R}^+ =$ another deep sets neural network to adjust probability
- Networks h, g trained jointly based on cross-entropy between p_{ij} and the match/no-match labels in the repository $\mathcal{R}$
- g learns to output large values for datasets with common distributions shared by many different variables (eg. binary-valued Bernoulli)

Techniques to improve performance

- Triplet training with anchor samples
- Subsample datasets when calculating stochastic gradients
- Augment training set of variable-matches by splitting single dataset into two matched datasets
- For numeric data: $h =$ feedforward network that operates on 32-bit binary representation of values instead of floats
- For text data: $h =$ feedforward network that operates on pretrained embedding of individual text fields (eg. fastText, Bert)
- For arbitrary string data: $h =$ LSTM that produces vector embedding

Experiments on OpenML

- Repository of thousands of datasets created by splitting columns of hundreds of data tables taken from OpenML[3]
- "True" variable matches identified based on column labels (ignoring common/generic column names)
- For query dataset with column-name = age (from survival data table): the top 3 matches are all columns named age, from tables annotated as audiology, diabetes, and breast tumor data
- Dataset with highest probability adjustment value ($\text{argmax}_j\, g(\mathcal{D}_j)$) is column where 58% of values = `true` and the rest = `false`. There are many similar datasets in OpenML with diverse column names.

[3] `http://www.openml.org/`

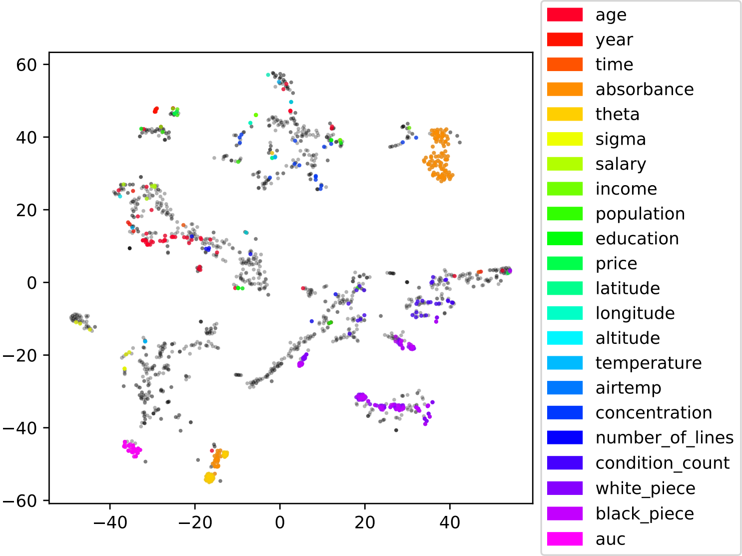

Figure: t-SNE of embeddings for 1K held-out numeric OpenML datasets. Datasets colored based on column name (if amongst topmost frequently-occurring names)

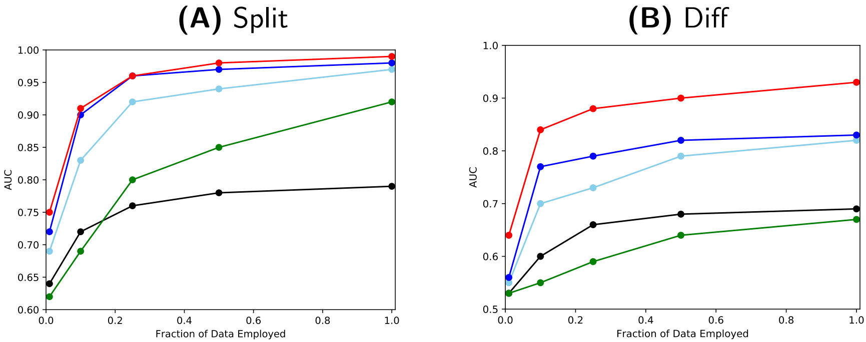

Figure: Match/no-match classification performance of different methods on various-sized subsamples of held-out numeric OpenML datasets:

1. *Mean+StdDev difference* (black)
2. *Kolmogorov-Smirnov* p-value (green)
3. *Maximum Mean Discrepancy* (light blue)
4. *SCF improved-MMD estimator* (blue)
5. *Ours* (red)

Method	$k = 1$	$k = 5$	$k = 10$	Recall
MeanSD	0.4	0.51	0.58	-
KS	0.46	0.59	0.67	-
MMD	0.48	0.62	0.69	-
SCF	**0.49**	0.65	0.72	-
Ours	0.48	**0.66**	**0.74**	-
MeanSD	0.35	0.44	0.53	0.52
KS	0.36	0.46	0.59	0.6
MMD	0.33	0.45	0.52	0.6
SCF	0.33	0.4	0.55	0.62
Ours	**0.42**	**0.61**	**0.67**	**0.71**

Table: Number of correct matches @k for retrieving datasets from $\mathcal{R}$ in *Split* (unshaded) and *Diff* (shaded) settings (averaged over 100 query datasets)

Recognizing Variables from their Data via Deep Embeddings of Distributions

Objective: Given new data from unknown variable, identify which previously-seen datasets stem from the same variable

Applications: AutoML, semantic labeling (eg. identifying PII data), automated transforms, schema-matching, dataset search

Approach: Use neural network to embed each dataset as vector, such that similar variables' data have nearby embeddings

Contact: jonasmue@amazon.com