# Dataset2Vec: Learning Dataset Meta-Features

**Hadi S. Jomaa, Josif Grabocka, Lars Schmidt-Thieme**
Information Systems and Machine Learning Lab
University of Hildesheim
Samelsonplatz 1, 31141 Hildesheim, Germany
`{hsjomaa,josif,schmidt-thieme}@ismll.uni-hildesheim.de`

## Abstract

Machine learning tasks such as optimizing the hyper-parameters of a model for a new dataset or few-shot learning can be vastly accelerated if they are not done from scratch for every new dataset, but carry over findings from previous runs. Meta-learning makes use of features of a whole dataset such as its number of instances, its number of predictors, the means of the predictors etc., so called meta-features, dataset summary statistics or simply dataset characteristics, which so far have been hand-crafted, often specifically for the task at hand. More recently, unsupervised dataset encoding models based on variational auto-encoders have been successful in learning such characteristics for the special case when all datasets follow the same schema, but not beyond. In this paper we design a novel model, Dataset2Vec, that is able to characterize datasets with a latent feature vector based on batches and thus is able to generalize beyond datasets having the same schema to arbitrary (tabular) datasets. To do so, we employ auxiliary learning tasks on batches of datasets, esp. to distinguish batches from different datasets. We show empirically that the meta-features collected from batches of similar datasets are concentrated within a small area in the latent space, hence preserving similarity. We also show that using the dataset characteristics learned by Dataset2Vec in a state-of-the-art hyper-parameter optimization model outperforms the hand-crafted meta-features that have been used in the hyper-parameter optimization literature so far. As a result, we advance the current state-of-the-art results for hyper-parameter optimization.

## 1 Introduction

Meta-learning, or learning-to-learn is an important machine learning task that has gained an increasing amount of interest over the past several years, particularly for its success in fast adaptation of new domains in applications such as few-shot learning. Typical approaches for meta-learning focus on learning generic internal representations of models which is suitable for several tasks. Little attention has been given however for the meta-features learning problem, which aims at learning dataset characteristics that can vastly accelerate cross-dataset task learning, for example hyper-parameter optimization. One of the open challenges in meta-feature learning is that existing approaches focus on datasets with similar schema.

Task-dependent meta-features can be found in distribution regression approaches, which are typically achieved through kernel-based optimization techniques in the reproducing kernel Hilbert space [41, 19, 21, 2]. Recently, a permutation invariant set-based neural network approach [40] demonstrated the applicabilty multiple instance learning in regression tasks, for example in estimating population statistics, and mutli-label classification tasks. Task-independent meta-feature learning solutions also exist, however one of the major drawbacks however is that these summarization techniques are bound by the schema of the dataset, i.e. number of instances and number of features.

A dataset summarization technique that learns useful meta-features beyond a fixed schema should meet the following criteria:

Preprint. Under review.

- *Criterion 1*, Scalable: Estimates meta-features efficiently and quickly from unseen datasets.
- *Criterion 2*, Schema-agnostic: Provides descriptive embeddings regardless of the number of features and sample size of the available data.
- *Criterion 3*, Preserves Datasets Similarities: Meta-features of datasets generated from the same distribution should be closer to each other in the latent space than meta-features of datasets generated from different distributions.
- *Criterion 4*, Supports Meta-tasks: Learnt meta-features must be expressive enough to be employed in meta-tasks.

**Main Contributions**. Our contributions are as follows: (i) We investigate a model, Dataset2Vec that is able to characterize datasets with a latent feature vector based on batches and thus is able to generalize beyond datasets having the same schema to arbitrary (tabular) datasets. (ii) We employ auxiliary learning tasks on batches of datasets, esp. to distinguish batches from different datasets. (iii) In an experiment we show that using the dataset characteristics learned by Dataset2Vec in a state-of-the-art hyper-parameter optimization model outperforms both, expert-crafted meta-features as well as all hand-crafted meta-features that have been used in the hyper-parameter optimization literature so far. (iv) As a result, we advance the current state-of-the-art results for hyper-parameter optimization

## 2 Related Work

Meta-feature learning as a standalone task, to the best of our knowledge, is a new concept. In this section we summarize some of the related topics and fields which surround our approach.

**Dataset Summarization Techniques** refer to approaches that extract suitable dataset information, known as meta-features, which can be used for different machine learning tasks. Meta-features can be represented as general dataset characteristics [7, 26, 29] or even as measures extracted from models induced from training on the data [30, 11]. Meta-features can also be learned directly from the data. The neural statistician (NS) [9] proposes an extension to the variational auto-encoder [16] that models the latent distribution of the underlying generative process of each dataset, i.e. the item to be encoded is the dataset itself. The instances of the same dataset are conditioned on a shared latent variable, the context, which captures dataset-specific statistics. NS is improved upon by a hierarchical latent variable model [14], that uses subsampling to construct a lower bound on set instead of the complete data. *Available techniques for meta-feature learning require vast amount of data and are bound by a fixed schema of the dataset, unlike our approach that can accommodate any schema.*

**Meta-Learning** or learning-to-learn is the process of learning a scalable internal representation of a model that quickly adapts to new tasks [12] and has in the past several years attracted a lot of attention, specifically for its performance on few-shot learning tasks [18, 13, 39]. Existing approaches learn a generic initial model parameters through sampling tasks from a task-distribution with an associated train-validation datasets. Even within this line of research we notice that learning embeddings helps achieve state-of-the-art performance [28]. *Meta-learning approaches result in task-dependent meta-features, whereas we learn task-independent meta-features.*

**Multiple-Instance Learning** encompass learning algorithms that take as input a sample population for tasks such as regression [2, 25, 22] or classification [6]. More related to our work perhaps is the method proposed by [21], that learns feature representations from distributions using kernel-based optimization. Set-based end-to-end models for regression and classification [40] demonstrate good performance on such tasks for fixed sized inputs. *We propose a novel set-based architecture that accepts variable-sized inputs.*

**Embedding and Metric Learning Approaches** aim at learning semantic distance measures that position similar high-dimensional observations within close proximity to each other on a manifold. By transforming the data into embeddings, simple models can be trained to achieve significant performance [31, 4]. Learning these embeddings involves optimizing a distance metric [33] and making sure that local feature similarities are observed [42]. *For our method, we train a pairwise similarity objective that captures similarities and dissimilarities between datasets.*

**Hyper-parameter Optimization** plays an important role in the machine learning community and can be a main factor in deciding whether a trained model turns out to be the state-of-the-art or simply moderate. The use of meta-features for this task has led to a significant improvement especially when used for warm-start initialization of Bayesian optimization techniques based on Gaussian processes [34, 20] or on neural networks [24]. Surrogate transfer in sequential model-based optimization [15] is also improved with the use of meta-features as seen in the state-of-the-art [37] and similar approaches [36, 3, 10]. *We improve the state-of-the-art by replacing the hand-crafted dataset meta-features with the learned meta-features.*

In this work, we propose a novel meta-feature learning approach, *Dataset2Vec*, that learns expressive latent characteristics of a dataset, by designing a novel permutation invariant set-based architecture that extends beyond a fixed schema. We optimize the inter-dataset and intra-dataset similarities and demonstrate the benefit of using the learned meta-features on meta-tasks, namely hyper-parameter optimization.

## 3 The Meta-feature Learning Problem

A **meta-learning problem** is simply a supervised learning problem where the instances $x_n$ are themselves datasets, i.e., given a sample of $N$ pairs $(x_n, y_n)$ from an unknown distribution $p$ on a space $\mathcal{X} \times \mathcal{Y}$ and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ to learn a model $\hat{y} : \mathcal{X} \to \mathcal{Y}$ such that the expected loss on further samples $\mathbb{E}_{(x,y)\sim p}\ell(y, \hat{y}(x))$ is minimal. The loss of most meta-learning problems requires to solve a learning problem, hence the name meta-learning. To distinguish both problems, we will call the former meta-problem and denote its components $\mathcal{X}^{\text{meta}}$, $\mathcal{Y}^{\text{meta}}$, $\ell^{\text{meta}}$. and the latter target problem denote its components simply by $\mathcal{X}, \mathcal{Y}, \ell$.

For example for the special meta-learning problem **hyper-parameter optimization**, the meta-target $y_n$ is a hyper-parameter setting and the meta-loss $\ell$ the target validation loss of a target model trained on a train partition of the dataset with these hyper-parameters. For the meta-learning problem **few shot learning**, the meta-target $y_n$ is an initial parameter setting of the target model and the meta-loss $\ell$ the target validation loss of a target model trained on a few training samples (like 5 or 10) of the dataset being initialized by those parameter setting.

Different from multi-instance learning the schema of the datasets $x_n$ may vary: some datasets may have 10 predictors, others 10,000. To predict on a dataset $x_n$, they are represented by some dataset features (also called statistics or characteristics), e.g., the number of instances, the number of predictors, the mean of the means of the predictors etc. These dataset features are a vector of fixed size $K$ that does not depend on the schema of the dataset. But meta-features also can be learned!

We define the **meta-feature learning problem** as follows: given a meta-learning problem and a family of meta-feature-based meta-learning algorithms $a_K : (\mathbb{R}^K \times \mathcal{Y}^{\text{meta}})^* \to \text{maps}(\mathbb{R}^K \to \mathcal{Y}^{\text{meta}})$ find a dataset embedding dimension $K$ and a meta-feature extractor $\hat{\phi} : \mathcal{X}^{\text{meta}} \to \mathbb{R}^K$ such that the meta-model trained by $a_K$ on the dataset features extracted by $\hat{\phi}$ has minimal expected meta-loss.

## 4 The Meta-feature Extractor Dataset2Vec

**Base meta-feature extractor architecture.** For any specific dataset $D \in \mathcal{X}^{\text{meta}}$ we denote its size by $N^D$, its dimensionality by $M^D$, its number of targets by $T^D$, its predictors by $X^D \in \mathbb{R}^{N^D \times M^D}$ and its targets by $Y^D \in \mathbb{R}^{N^D \times T^D}$. The meta-feature extractor $\phi$ takes the two matrices $X^D$ and $Y^D$ as inputs. All meta-problems we are aware of, do not depend on the ordering of the instances, the ordering of the predictors or the ordering of the targets within a dataset. Not having to learn these independencies from data, we adopt an idea from DeepSet [40] to make the meta-feature extractor permutation invariant in these regards. As on the other side, most meta-problems depend on correlations between predictors and targets, we always kept a predictor and target cell each. The final architecture of the meta-feature extractor thus is as follows:

$$\phi^{\text{base}}(D) := h\left(\frac{1}{M^D T^D}\sum_{m=1}^{M^D}\sum_{t=1}^{T^D} g\left(\frac{1}{N^D}\sum_{n=1}^{N^D} f\left(X^D_{n,m}, Y^D_{n,t}\right)\right)\right)$$

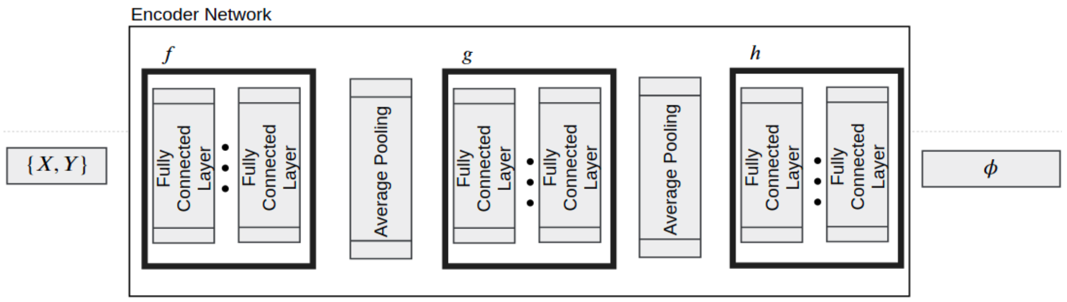


Figure 1: Overview of the encoder network

with $f : \mathbb{R}^2 \to \mathbb{R}^{K_f}$, $g : \mathbb{R}^{K_f} \to \mathbb{R}^{K_g}$ and $h : \mathbb{R}^{K_g} \to \mathbb{R}^K$ being three suitable functions, represented by neural networks with $K_f$, $K_g$ and $K$ output units, respectively (see also Figure 1).

**Batch Sampler.** While some dataset characteristics such as the number of predictors and the number of instances have to be computed on the whole dataset, most others such as (an estimate of) the mean of the means of the predictors could be computed more efficiently from samples. Especially for a learnt meta-feature extractor $\phi$ taking a dataset $D \in \mathcal{X}^{\text{meta}}$ as a whole as input, is impractical and does not scale well to larger datasets. We therefore propose to compute meta-features based on batches from a dataset.

For any specific dataset $D$ we define the batch for subsets $N' \subseteq \{1, \ldots, N^D\}, M' \subseteq \{1, \ldots, M^D\}, T' \subseteq \{1, \ldots, T^D\}$ of instance, predictor and target indices simply by the pair $(X', Y')$ of respective submatrices, i.e.,

$$X' := (X^D_{n,m})_{n \in N', m \in M'}, \quad Y' := (Y^D_{n,t})_{n \in N', t \in T'}$$

The batch sampler creates random index subsets $N'$, $M'$ and $T'$ of a size drawn uniformly at random from $\{2^q | q \in [4, 8]\}$, $[1, M]$ and $[1, T]$, drawing indices uniformly at random without replacement, Appendix B. Meta-features of a dataset $D$ then are computed by averaging the meta-features of $B$ many random batches:

$$\phi(D) := \frac{1}{B} \sum_{b=1}^{B} \phi^{\text{base}}(\text{randombatch}(D))$$

**End-to-end Learning the Meta-feature Extractor.** For any meta-task at hand such as hyper-parameter optimization, the meta-feature extractor can be learnt end-to-end by simply concatenating it with a meta-feature-based meta-model, for example also represented by a neural network:

$$\hat{y}^{\text{meta}} := \hat{y}^{\text{meta,mf}} \circ \phi : D \to \mathbb{R}^K \to \mathcal{Y}^{\text{meta}}, \quad \hat{y}^{\text{meta,mf}} : \mathbb{R}^K \to \mathcal{Y}^{\text{meta}}$$

But most meta-learning datasets are small, rarely containing more than a couple of thousands or 10,000s of meta-instances, as each such meta-instance itself requires to run a target learning process. Thus training a meta-feature extractor end-to-end directly, is not promising.

We propose to employ auxiliary meta-tasks with abundand data to extract meaningful meta-features, esp. the meta-task to identify if two batches of data stem from the same dataset or not, where a batch is a random sample of instances and predictors of a dataset.

**The Batch Identification Problem.** Given a sample of pairs of data set batches and same origin indicator $(x, x', i) \in \mathcal{X}^{\text{meta}} \times \mathcal{X}^{\text{meta}} \times \{0, 1\}$ from a joint distribution $p$, learn a batch identification model $\hat{i} : \mathcal{X}^{\text{meta}} \times \mathcal{X}^{\text{meta}} \to \{0, 1\}$ with minimal expected misclassification error

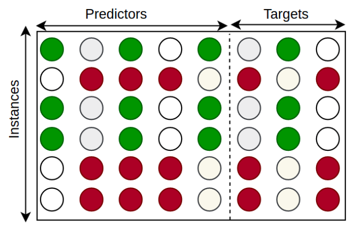
$$E_{(x,x',i) \sim p}(I(i \neq \hat{i}(x, x')))$$

where $I(\text{true}) := 1$ and $I(\text{false}) := 0$. See Figure 2 for an illustration of such batch sample pairs.

To force all information relevant for the batch identification task to be pooled in the meta-features, we use a very simple batch identification model:

$$\hat{i}(x, x') := e^{-\gamma ||\hat{\phi}(x) - \hat{\phi}(x')||}$$

Figure 2: The batch sampler for the dataset identification problem: two subsets (colored) are sampled randomly and assigned a similarity value of 1.



with $\gamma$ as a tuneable hyper-parameter. We train our model using the binomial negative-log likelihood objective function.

Dataset2Vec is trained on a large amount of batch samples from datasets in a meta-dataset, thus currently does not use any information of the subsequent meta-problem to solve and thus is a generic, in this sense unsupervised meta-feature extractor. Any other meta-task could be easily integrated into Dataset2Vec by just learning them jointly in a multi-task setting, esp. a batch reconstruction similiar to the dataset reconstruction of the NS [9] could be interesting, if one could figure out how to do this across different schemata. We leave this for future work.

## 5 Experiments

We claim that for a dataset summarization technique to learn useful meta-features, it must meet the following criteria: scalable (*Criterion 1*), schema-agnostic (*Criterion 2*), preserves dataset similarities (*Criterion 3*), and supports meta-tasks (*Criterion 4*). We evaluate if Dataset2Vec supports these claims by designing the following two experiments. Implementation can be found here[1].

### 5.1 Batch Identification

We train our model by optimizing the distance between meta-features of pairs of subsets. For each iteration we randomly sample equal number of positive and negative pairs. The reported results represent the average of a 5-fold cross validation experiment *Criterion 1*.

#### Baselines

As mentioned earlier, NS [9] learns meta-features as context information by encoding complete datasets with an extended variational autoencoder. However, since it is bound by a fixed dataset schema, we generate a 2-D labeled synthetic dataset to train the spatial model presented by the authors[2]. We use the same hyper-parameters used by the authors, considering that the we also train on 2-dimensional meta dataset. The network architecture for Datset2Vec, implemented in in Tensorflow [1], is described in Appendix A. NS and Dataset2Vec produce meta-features of size 64. We also set the number random batches for dataset summarization $B$ to 40.

#### Evaluation Metric

We evaluate the similarity between embeddings by means of pairwise classification accuracy with a cut-off threshold of $\frac{1}{2}$. We set the hyper-parameter $\gamma = \frac{1}{2}$, for NS after tuning it on a separate validation set, and keep $\gamma = 1$ for Dataset2Vec. We evaluate the pairwise classification accuracy over 16,000 pairs of batches containing equal number of postive and negative pairs.

#### UCI Meta Dataset

The UCI repository [8] contains a vast collection of datasets. We used 121 preprocessed classification datasets [3] of different schemas, *Criterion 2*, to train the meta-feature extractor by randomly sampling

[1]https://github.com/hadijomaa/dataset2vec.git
[2]https://github.com/conormdurkan/neural-statistician.git
[3]http://www.bioinf.jku.at/people/klambauer/data_py.zip

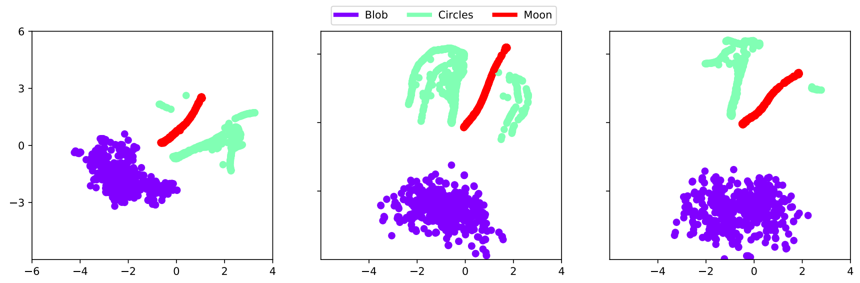


Figure 3: Projection of the latent representations estimated with Dataset2Vec on 3 distinct folds using multi-dimension scaling (MDS) [5]. Each point is a summary statistic for a single dataset whose summary is obtained as an average of 4 randomly sampled subsets.

pairs of subsets, Algorithm 1, along the number of instances, predictors, and targets. We achieve pairwise classification accuracy of 77.58% $\pm$ 3.13. In Table 1, we show several groups of datasets that have been collected by a 5-Nearest Neighbor based on the Dataset2Vec metafeatures. For the lack of groundtruth similarity that assesses the similarity between batches, we find within every group several datasets that are similarly named, and can be found in close proximity to each other.

Table 1: Groups of dataset based on the 5-Nearest Neighbors in the latent space

| **Group 1** | **Group 2** | **Group 3** |
|---|---|---|
| breast-cancer-wisc | pittsburg-bridges-TYPE | monks-2 |
| breast-tissue | pittsburg-bridges-MATERIAL | monks-1 |
| wall-following | pittsburg-bridges-REL-L | monks-3 |
| echocardiogram | pittsburg-bridges-T-OR-D | lenses |
| credit-approval | pittsburg-bridges-SPAN | balloons |

**Toy Meta Dataset**

We generate a collection of 10,000 2-D datasets each containing a varying number of samples. The datasets are created using the sklearn library [23] and belong to either circles, moons with 2 classes, or blobs with varying number of classes drawn uniformly at random within the bounds [2,8]. We also perturb the data by applying random noise, Appendix C.
We randomly sample a fixed-size subset (200 samples) from every dataset for both approaches, to ensure a fair comparison, and train until convergence. The perfomance of both models is summarized in Table 2. The similarity preserving aspect, *Criterion 3*, of Dataset2Vec can be seen in Figure 3 which depicts a 2-D plot of the learned meta-features.

## 5.2 Hyper-parameter Optimization

The task of hyper-parameter optimization is to identify an optimal hyper-parameter configuration $\lambda^* \in \Lambda$ that results in a model $M_{\lambda^*}$, such that the generalization error on the validation set is minimized:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathcal{L}(\mathcal{A}(D^{\text{train}}, \lambda), D^{\text{valid}}), \tag{1}$$

for a particular machine learning algorithm $\mathcal{A} : \mathcal{D} \times \Lambda \rightarrow \mathcal{Q}$ whose task is to estimate a model $Q_\lambda \in \mathcal{Q}$, from the space of all models $\mathcal{Q}$ with hyper-parameters $\lambda \in \Lambda$, that optimizes an objective function, for example a loss function $\mathcal{L}$, over a data set distribution $D \in \mathcal{D}$, where $\mathcal{D}$ is the set of all data sets and $D^{\text{train}} \bigcap D^{\text{valid}} = \emptyset$.

**Another Dataset Description**

Meta-features are commonly used to describe data sets, and are usually extracted as some statistics from a dataset [3], nevertheless selecting task-specific meta-features can be quite challenging [17].

Table 2: Dataset Classification Accuracy

| Method | Number of Parameters | Pairwise Classification Accuracy (%) |
|---|---|---|
| NS [9] | 2271402 | $58.75 \pm 0.84$ |
| Dataset2Vec | 50112 | $91.50 \pm 0.27$ |
| T-test Significance (p-value) | | 0.00014 |

For hyper-parameter optimization, a new set of meta-features have been proposed, which achieve state-of-the-art hyper-parameter tuning performance across several datasets [36]. We use 30 classification datasets obtained from the support vectore machines (SVM) dataset originally introduced in [35]. The dataset was generated by training an SVM on a grid of 288 hyper-parameter configurations with 6 hyper-parameters. For more information about the dataset, we refer the readers to [35].

## Baselines

The use of meta-features has led to significant performance improvement in hyper-parameter optimization. We select the state-of-the art [36] and evaluate the effectiveness of our learned meta-features compared to other characteristics, summarized in Table 3. We also compare with several well-known baselines that vary in terms of complexity and scalability.

1. Independent Gaussian Process [32] (I-GP): In this approach, the surrogate is modeled by a Gaussian process with a squared-exponential kernel and automatic relevance determination. Hyper-parameter tuning is done on each data set independently.
2. Spearmint [32]: Similar to I-GP, the surrogate is modeled by a Gaussian process with a Matérn 5/2 kernel.
3. Two-Stage Transfer Surrogate [36] (TST-R): As the name suggests, this approach models the surrogate model in two stages and leverages similarity across data sets. Particularly, in the first stage, a Gaussian process is trained as a surrogate model to estimate the hyper-parameter response of a new data set as well as previous data sets. In the second stage the surrogate models are averaged using normalized weights proportional to the similarity of the new unseen data set meta-features. The code for TST-R is available here[4].

Table 3: The list of all meta-features used for TST-R. NS is not appplicable as hyper-parameter optimization is done across datasets with different schemas.

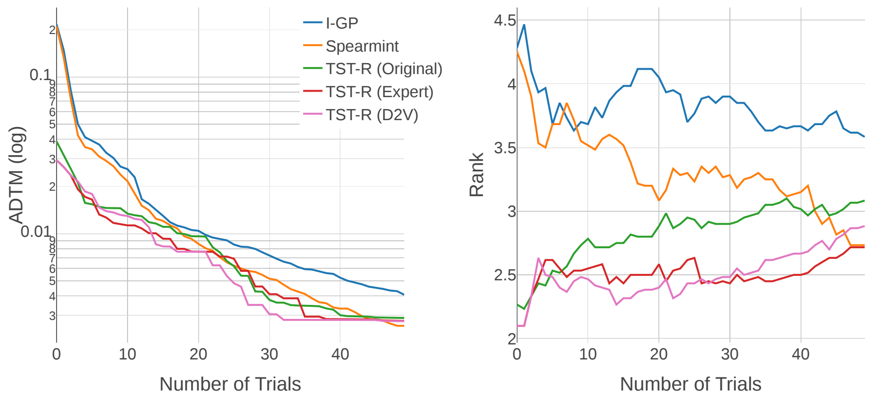
| Method Name | Count | Description |
|---|---|---|
| TST-R (Original) | 22 | Hand-crafted meta-features used for hyper-parameter optimization [35] |
| TST-R (Expert) | 90 | Expert-crafted meta-features [27] |
| TST-R (D2V) | 64 | Our learned meta-features |

I-GP and Spearmint cannot carry over information from previous experiments, i.e. on other data sets, whereas TST-R is a strong baseline that leverages meta-knowledge to reconstruct a scalable hyper-parameter response surface. It is worth-noting that the hand-crafted meta-features where specifically designed for the task of hyper-parameter optimization. The expert-crafted meta-features include a battery of summaries calculated as measures from information theory field [7], general dataset features and statistical properties [26] which require completely labeled data. For more information about meta-features, we refer the readers to [27].

## Evaluation Metrics

We follow the evaluation metrics of the state-of-the-art papers [36]. For the average rank (AR), we rank the performance of the best hyper-parameter configuration obtained by a tuning strategy with respect to other methods and then average over all data sets. This highlights the difference between

[4]https://github.com/wistuba/TST.git



(a) Average Distance to the Minimum (b) Average Rank

Figure 4: TST-R (D2V) demonstrates competitive performance against hand- and expert-crafted meta-features. This is indicative of the representational capacity of the learned meta-features in adequately representing the dataset. For both plots, lower is better.

different approaches, however it does not offer insight into how well the selected hyper-parameters are with respect to the global optimum. For that, we use the average distance to the minimum (ADTM) as the second evaluation metric. After $t$ trials, ADTM is defined as

$$\mathrm{ADTM}((\Lambda_t^D)_{D\in\mathcal{D}},\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{D\in\mathcal{D}} \min_{\lambda\in\Lambda_t^D} \frac{f(D,\lambda)-f(D)^{\min}}{f(D)^{\max}-f(D)^{\min}}$$

with $\Lambda_t^D$ as the set of hyper-parameters that have been selected by a hyper-parameter optimization method for data set $D$ in the first $t$ trials and $f(D)^{\min}$, $f(D)^{\max}$ the range of the loss function on the hyper-parameter grid $\Lambda$ under investigation.

### Results and Discussion

The results reported estimated using a leave-one-dataset-out cross-validation and are the average of ten repetitions. We notice that TST-R (Original) consistently outperforms I-GP and Spearmint, which is indicative of the importance of meta-features for hyper-parameter optimization, and has proven to outperform other hyper-parameter optimization approaches that make use of different meta-features [3, 38]. Replacing hand-crafted features with our learnt representations, TST-R (D2V), improves the overall performance as we achieve state-of-the-art ADTM across all baselines, *Criterion 4*. The learnt meta-features demonstrate competetive AR against expert-crafted meta-features as well, while we observe the lowest (best) rank initially for large budget of 25 trials. Another added advantage of TST-R (D2V) is that it does not require access to the whole dataset, unlike the rest of the meta-features.

## 6 Conclusion

We present a meta-feature learning approach, Dataset2Vec, that learns distinctive task-independent characteristics from datasets. Using meta-features learned for the batch identification problem provides meta-features for datasets that do not depend on the meta-task to solve, e.g., the same meta-features can be used for hyper-parameter optimization and few shot learning. It seems likely that meta-features that are learned jointly with the meta-task at hand will turn out to focus on the characteristics relevant for the meta-task and thus provide even better meta-losses, a direction of further research we are currently investigating.

## References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283. USENIX Association, 2016.

[2] François Bachoc, Fabrice Gamboa, Jean-Michel Loubes, and Nil Venet. A gaussian process regression model for distribution inputs. *IEEE Trans. Information Theory*, 64(10):6620–6637, 2018. doi: 10.1109/TIT.2017.2762322. URL `https://doi.org/10.1109/TIT.2017.2762322`.

[3] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. Collaborative hyperparameter tuning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 199–207, 2013. URL `http://jmlr.org/proceedings/papers/v28/bardenet13.html`.

[4] Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Class-balanced siamese neural networks. *Neurocomputing*, 273:47–56, 2018. doi: 10.1016/j.neucom.2017.07.060. URL `https://doi.org/10.1016/j.neucom.2017.07.060`.

[5] Ingwer Borg and Patrick Groenen. Modern multidimensional scaling: Theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.

[6] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018. doi: 10.1016/j.patcog.2017.10.009. URL `https://doi.org/10.1016/j.patcog.2017.10.009`.

[7] Ciro Castiello, Giovanna Castellano, and Anna Maria Fanelli. Meta-data: Characterization of input features for meta-learning. In *MDAI*, volume 3558 of *Lecture Notes in Computer Science*, pages 457–468. Springer, 2005.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[9] Harrison Edwards and Amos J. Storkey. Towards a neural statistician. In *ICLR*. OpenReview.net, 2017.

[10] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *CoRR*, abs/1802.02219, 2018. URL `http://arxiv.org/abs/1802.02219`.

[11] Andrey Filchenkov and Arseniy Pendryak. Datasets meta-feature description for recommending feature selection algorithm. In *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*, pages 11–18. IEEE, 2015.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1126–1135, 2017. URL `http://proceedings.mlr.press/v70/finn17a.html`.

[13] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, pages 9537–9548, 2018.

[14] Luke B. Hewitt, Maxwell I. Nye, Andreea Gane, Tommi S. Jaakkola, and Joshua B. Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. In *UAI*, pages 988–997. AUAI Press, 2018.

[15] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. Global Optimization*, 13(4):455–492, 1998.

[16] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[17] Rui Leite and Pavel Brazdil. Predicting relative performance of classifiers from samples. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 497–503. ACM, 2005.

[18] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, pages 3490–3497. AAAI Press, 2018.

[19] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representation. *CoRR*, abs/1807.08479, 2018. URL `http://arxiv.org/abs/1807.08479`.

[20] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *AAAI*, pages 1355–1362. AAAI Press, 2018.

[21] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 10–18, 2013. URL `http://jmlr.org/proceedings/papers/v28/muandet13.html`.

[22] Junier B. Oliva, Barnabás Póczos, and Jeff G. Schneider. Distribution to distribution regression. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1049–1057. JMLR.org, 2013.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Valerio Perrone, Rodolphe Jenatton, Matthias W. Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *NeurIPS*, pages 6846–6856, 2018.

[25] Barnabás Póczos, Aarti Singh, Alessandro Rinaldo, and Larry A. Wasserman. Distribution-free distribution regression. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pages 507–515, 2013. URL `http://jmlr.org/proceedings/papers/v31/poczos13a.html`.

[26] Matthias Reif, Faisal Shafait, Markus Goldstein, Thomas M. Breuel, and Andreas Dengel. Automatic classifier selection for non-experts. *Pattern Anal. Appl.*, 17(1):83–96, 2014.

[27] Adriano Rivolli, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C. P. L. F. de Carvalho. Towards reproducible empirical research in meta-learning. *CoRR*, abs/1808.10406, 2018.

[28] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *CoRR*, abs/1807.05960, 2018.

[29] Saddys Segrera, Joel Pinho Lucas, and María N. Moreno García. Information-theoretic measures for meta-learning. In *HAIS*, volume 5271 of *Lecture Notes in Computer Science*, pages 458–465. Springer, 2008.

[30] Kate Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1):6:1–6:25, 2008.

[31] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4080–4090, 2017. URL `http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning`.

[32] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

[33] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012. IEEE Computer Society, 2016.

[34] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Sequential model-free hyperparameter tuning. In *ICDM*, pages 1033–1038. IEEE Computer Society, 2015.

[35] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter search space pruning - A new component for sequential model-based hyperparameter optimization. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II*, pages 104–119, 2015. doi: 10.1007/978-3-319-23525-7_7. URL `https://doi.org/10.1007/978-3-319-23525-7_7`.

[36] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *ECML/PKDD*, volume 9851 of *Lecture Notes in Computer Science*, pages 199–214. Springer, 2016.

[37] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018. doi: 10.1007/s10994-017-5684-y. URL `https://doi.org/10.1007/s10994-017-5684-y`.

[38] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 1077–1085. JMLR.org, 2014.

[39] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NeurIPS*, pages 7343–7353, 2018.

[40] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. Deep sets. In *NIPS*, pages 3394–3404, 2017.

[41] Zhen Zhang, Mianzhi Wang, and Arye Nehorai. Optimal transport in reproducing kernel hilbert spaces: Theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[42] Zhedong Zheng, Liang Zheng, and Yi Yang. A discriminatively learned CNN embedding for person reidentification. *TOMCCAP*, 14(1):13:1–13:20, 2018.

# Appendices

## A Network Architecture

The proposed architecture for Dataset2Vec in constitutes of:

1. prepooling block representing function $f$ contains a dense layer, a residual block with three dense layers, followed by a final dense layer;
2. pooling block representing function $g$ consists of two dense layers
3. postpooling block representing function $h$ contains a dense layer, a residual block with three dense layers, followed by a final dense layer.

All layers have Rectified Linear Unit (ReLU) activations. We use 64 units for all layers.

## B Sample Batch Pairs

For any specific dataset $D \in \mathcal{X}^{\text{meta}}$ we denote its size by $N^D$, its dimensionality by $M^D$, its number of targets by $T^D$, its predictors by $X^D \in \mathbb{R}^{N^D \times M^D}$ and its targets by $Y^D \in \mathbb{R}^{N^D \times T^D}$.

Let $p$ be any distribution on pairs of data set batches and a binary indicator if both batches stem from the same data set (called same origin indicator), e.g., given a distribution of data sets $p_{\mathbb{D}}$:

**Algorithm 1: sample-batch-pairs**$(p_{\mathbb{D}})$

```
Input  : distribution over datasets $p_{\mathbb{D}}$
1  $D \sim p_{\mathbb{D}}$
2  if unif$([0,1]) < 0.5$ then
3      $D' \sim p_{\mathbb{D}}$
4      $i := 0$
5  else
6      $D' := D$
7      $i := 1$
8  end
9  $D_s :=$ sample-batch$(D)$
10 $D'_s :=$ sample-batch$(D')$
Return : $(D_s, D'_s, i)$
```

**Algorithm 2: sample-batch**$(D)$

```
Input  : Dataset $D$
1  $N' \sim \{2^q | \sim [4,8]\}$
2  $M' \sim [1, M^D]$
3  $T' \sim [1, T^D]$
4  $X' := (X^D_{n,m})_{n \in N', m \in M'}$
5  $Y' := (Y^D_{n,t})_{n \in N', t \in T'}$
6  $D' := (X', Y')$
Return : $D'$
```

## C Generating Synthetic Datasets

To generate the synthetic dataset, we use the sklearn-library [23] with the following commands: sklearn.datasets.make_moons, sklearn.datasets.make_circles, and sklearn.datasets.make_blobs. We fix the number of features $M$ to 2 and apply Algorithm 3

```
Algorithm 3: generate-set($M$)
Input :Number of Features $M$
1 $random\_state \sim [0, 100]$
2 $number\_of\_instances \sim \{2^q | q \in [11, 14]\}$
3 $type\_of\_dataset \sim$ {circles,blobs,moons}
4 $X, Y :=$ sklearn.datasets.make_$type\_of\_dataset(number\_of\_instances, random\_state, M)$
5 if $unif[0, 1] < 0.5$ then
6     $X := apply\_noise(X)$
7 end
Return : $X, Y$
```

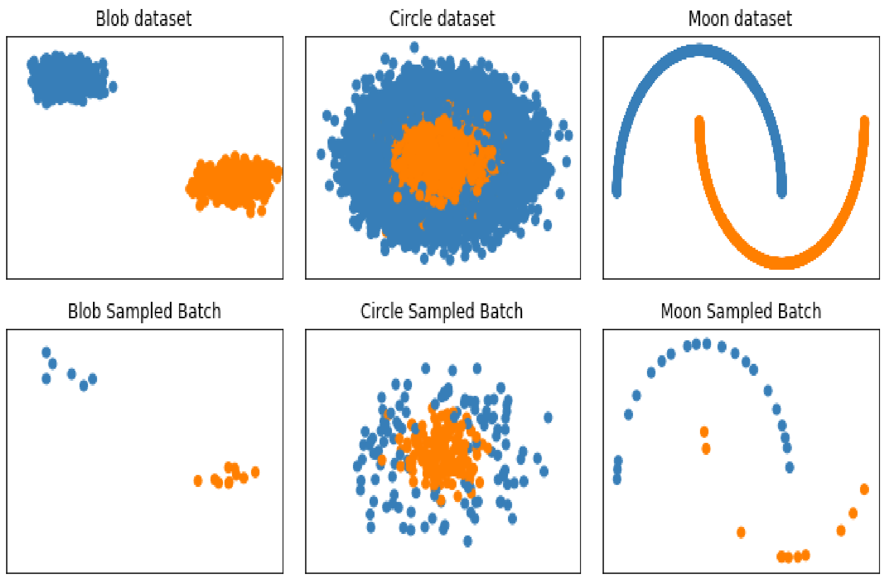


Figure 5: Examples of dataset