Saket Sathe and Charu C. Aggarwal
IBM T J Watson Research Center
Yorktown, NY 10598

Similarity Forests

ACM KDD Conference, 2017

Introduction

- Random forests are among the most popular classification methods used by real-world data scientists because of their extraordinary accuracy and effectiveness.

- A recent experimental comparison among more than 100 classifiers placed these methods as the top-performing techniques together with SVMs.

- Their use is primarily limited to multidimensional data because they sample features from the original data set.

- We propose a method for extending random forests to work with any arbitrary set of data objects.
  - The input is provided as similarities among the data objects rather than as multidimensional features.

Motivation

- In many applications, the input is provided in the form of similarities.

- SVMs can work with similarities, although they require all $O(n^2)$ similarity values to be specified
  - Similarity computations may be expensive in certain types of data domains

- SVMs and random forests provide better performance in different data sets.
  - It is therefore useful to make random forests work in cases where only similarities are available

- Domain-specific similarity matrices often have missing values.

Contributions

- Design a random forest to work with similarity values rather than multidimensional features.
- Requires the computation of only $O(n \cdot \log(n))$ similarity values.
- Can be used with missing values and noisy data sets.
- Can be used even in cases where the full dimensional representation is available
  - Provides competitive or slightly better performance than traditional random forests

Notations

- Set of n objects denoted by $O_1 \ldots O_n$.

- The similarity between O_i and O_j is denoted by S_{ij}.

  - An expensive domain-specific similarity function might compute S_{ij} (e.g., time-series) $\Rightarrow$ Similarities are computed on the fly as needed by the algorithm.

  - Algorithm design needs to minimize such computation

- It is assumed that the labels are drawn from a set of c different possibilities denoted by $L = \{1 \ldots c\}$.

- The ith object O_i in the training data is labeled as $l_i \in L$.

Definition

- Given a set of n objects $O_1 \ldots O_n$, which are labeled $l_1 \ldots l_n$, construct a random forest with the use of only computed pairwise similarities between these objects and no access to the multidimensional representations (if any) of these objects.

- For a given test object O with computable similarities with other training objects, a principled method must be proposed in order to perform the classification.
  - The testing process uses similar principles as the training phase.

SimForest: Broader Principles

- The basic idea is to assume that a multidimensional space exists in which the objects can be embedded (but do not explicitly compute it).

- Using kernel methods to extract a multidimensional representation of the objects defeats the purpose.
  - Such an approach can potentially use $O(n^2)$ space and $O(n^3)$ time.
  - Not practical for most real-world settings.

- All $O(n^2)$ pairs of similarities are not required to construct a similarity forest.

Splitting with Similarity Values

- The objects $O_1 \ldots O_n$ can be theoretically embedded in some multidimensional space as the points $\overline{X_1} \ldots \overline{X_n}$ (not explicitly computed).

- Just as a random forest samples features from a multidimensional data set, the similarity-based approach samples pairs of objects in order to define directions in the multidimensional space.

  – Sampling the object O_i and O_j results in the vector direction from $\overline{X_i}$ to $\overline{X_j}$.

- The data objects are then split into two groups using a hyperplane perpendicular to this direction (all done without explicit projection).

Recursive Splitting

- The splitting approach is applied recursively to construct each ensemble component of the similarity forest.

- During the testing phase, the traversal of the tree also requires such similarity computations.

- If the pairs of sampled objects $\overline{X_i}$ and $\overline{X_j}$ are chosen to belong to different classes, the randomized split directions will naturally tend to be more discriminating.

- The number of pairs of data objects that are required to be sampled at each node is analogous to the number of features that are sampled by the traditional random forest algorithm.
  - Our experimental results show that a small number of pairs such as 1 or 2 can achieve a high level of accuracy.

The SimForest Algorithm

- **Training:** Construct each tree in the forest using sampled pairs of points and splitting using similarity values.
  - Construct the tree down to the case where each leaf node has 100% accuracy.
- **Testing:** For each test instance, traverse the tree using only similarities between test instance and training instances.
  - Predict dominant label of leaf nodes
- **Key points:** Splitting (during training) and tree traversal (during testing) using only similarity values

Splitting

- Consider a pair of objects O_i and O_j that define the direction of the split.

- Consider an object O_k that needs to be projected along this direction.

- The unit direction of the embedding is given by $\frac{\overline{X_j}-\overline{X_i}}{||\overline{X_j}-\overline{X_i}||}$.

- The projection $P(\overline{X_k})$ of $\overline{X_k}$ on this direction is therefore given by the dot product of $\overline{X_k} - \overline{X_i}$ on this unit direction.

$$P(\overline{X_k}) = (\overline{X_k} - \overline{X_i}) \cdot \frac{\overline{X_j} - \overline{X_i}}{||\overline{X_j} - \overline{X_i}||}$$

- Need to express the above in terms of similarities!

Splitting

- One can express the numerator as dot products and then replace with similarities:

$$\begin{aligned} P(\overline{X_k}) &= \frac{\overline{X_k} \cdot \overline{X_j} - \overline{X_k} \cdot \overline{X_i} - \overline{X_i} \cdot \overline{X_j} + \overline{X_i} \cdot \overline{X_i}}{||\overline{X_j} - \overline{X_i}||} \\ &= \frac{S_{kj} - S_{ki} - S_{ij} + S_{ii}}{||\overline{X_j} - \overline{X_i}||} \\ &= \frac{S_{kj} - S_{ki} - S_{ij} + S_{ii}}{\sqrt{S_{ii} + S_{jj} - 2S_{ij}}} \end{aligned}$$

- We already have a projection purely in terms of similarities, and we can use it readily.

- But we can simplify even further!

Simplification

- The expression for projection can be simplified as follows:

$$\begin{aligned} P(\overline{X_k}) &= \frac{S_{kj} - S_{ki} - S_{ij} + S_{ii}}{\sqrt{S_{ii} + S_{jj} - 2S_{ij}}} \\ &= A \cdot (S_{kj} - S_{ki}) + B \end{aligned}$$

- Here, A and B are constants that do not depend on the object O_k being projected.

- We do not care about these constants–only the ordering of the objects on the projection line.

- So we can simply use $(S_{kj} - S_{ki})$ to order the points for the split!

Splitting Implementation

- Sort the data objects $\{O_k\}_{k=1}^{n}$ in order of $(S_{kj} - S_{ki})$ and use it to evaluate various splitting points.

- The splitting point is chosen such that it minimizes the weighted Gini index of the children nodes.

- The weighted Gini quality $GQ(N_1, N_2)$ of the children nodes is given by the following:
$$GQ(N_1, N_2) = \frac{n_1 G(N_1) + n_2 G(N_2)}{n_1 + n_2}$$

- Choose best over r pairs.

- Retain the split thresholds and the *ordered* split pair (O_i to O_j) for testing phase.

Testing Phase

- The testing phase uses an identical approach to project the test points on the line defined by the pairs of training points at each node.

- If (O_i, O_j) is the defining pair of objects for a given node, then the value of $S_{kj} - S_{ki}$ is computed for the test object O_k.

- The stored split point (say a) is used to determine which path in the decision tree to follow depending on whether or not $S_{kj} - S_{ki} \leq a$.

- This step is performed for each node on the path of the tree, until the object O_k is assigned to a leaf node.

- The label of the leaf node is reported as the prediction.

Distances in lieu of Similarities

- In some settings, it is easier to compute distances rather than similarities.
  - In the sequence domain, one might naturally use the edit distance or the dynamic time-warping distance.

- Natural solution: use cosine law with squared distance matrix Δ

$$S = -\frac{1}{2}(I - U/n)\Delta(I - U/n) \tag{1}$$

- **Disadvantage:** Requires all pairs of distances $\Rightarrow$ Loses some of the efficiency properties of similarity forests

Simplified Approach

- The cosine law is not really required if one can make some simplifying assumptions.

- **Assumption:** Each point is normalized to unit norm (might not be true in reality).

- Let $D(O_i, O_j)$ be the distance between data points O_i and O_j.

$$D(O_k, O_i)^2 - D(O_k, O_j)^2 = ||\overline{X_k} - \overline{X_i}||^2 - ||\overline{X_k} - \overline{X_j}||^2$$
$$= 2\overline{X_k} \cdot \overline{X_j} - 2\overline{X_k} \cdot \overline{X_i} + \underbrace{(||\overline{X_i}||^2 - ||\overline{X_j}||^2)}_{\text{Zero (Because of Normalization)}}$$
$$\propto S_{kj} - S_{ki}$$

- So we can simply use the difference in squared distances for splits instead of similarities!

Incompletely Specified Similarities

- Common problem in settings where similarity estimation is expensive.

- If similarities between pairs of objects (such as images) are obtained using crowd-sourcing, a cost may be incurred for each pairwise similarity.

- Not all objects can be partitioned between a pair of nodes at each split point.
  - The projection of data points along arbitrary directions may require unobserved similarities to be available.

Training with Incompletely Specified Similarities

- The selected pairs of objects (O_i, O_j) for splitting should always be such that similarity between them is observed. However, O_k may have unobserved similarities with either O_i or O_j.

- Since O_k cannot be assigned to one of the child nodes, we allow it to stay at the current node as its final destination.

- The class label of an internal node is defined by its majority class.

- Unlike the case of fully observed similarities, internal nodes also have a label.

Testing with Incompletely Specified Similarities

- When test instances are classified, the same procedure is applied.

- We compute $S_{kj} - S_{ki}$ (when similarity is available) and assigning the point O_k to a child node, depending on the relationship of $S_{kj} - S_{ki}$ to the split point.

- If these similarities are not available, the point O_k is assigned to an internal node.

- The label of the corresponding node is reported as the prediction.

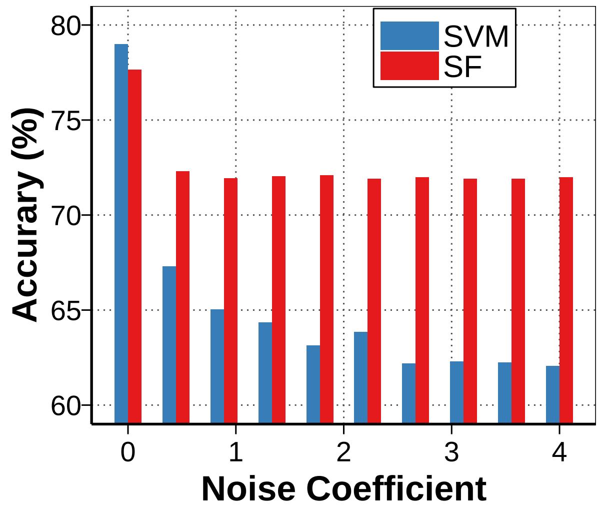
Experimental Results

- Extensive experimental results comparing *SimForest* with other competitive techniques.
- Extensively test *SimForest*'s resistance to noisy similarities.
- Evaluate how missing similarity values affect the accuracy of *SimForest*.
- Show results on multidimensional data.

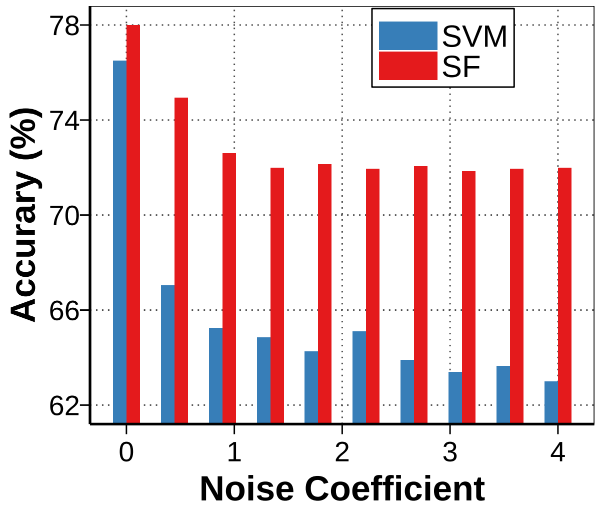
Similarity Matrix Generation

- Add noise to similarity matrix in controlled way with noise coefficient α.
  - Assume access to multidimensional data for building similarity matrices but not to algorithm or baselines.
  - Use cosine and RBF similarity on multidimensional data.
- Naturally used as kernels in support vector machines.
- SVMs are a natural baseline.

Effect of Noise



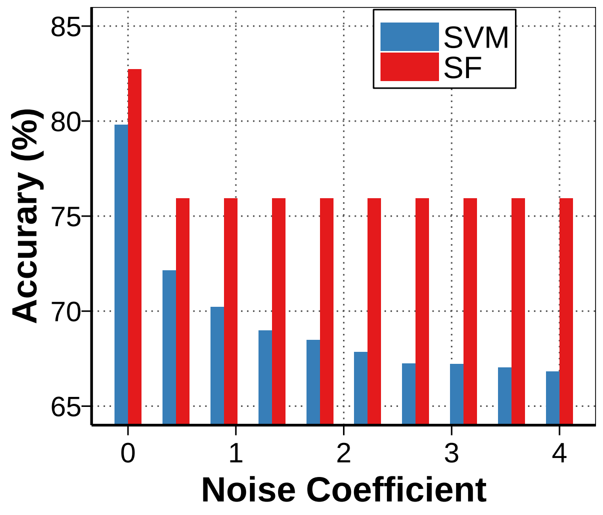
(a) RBF



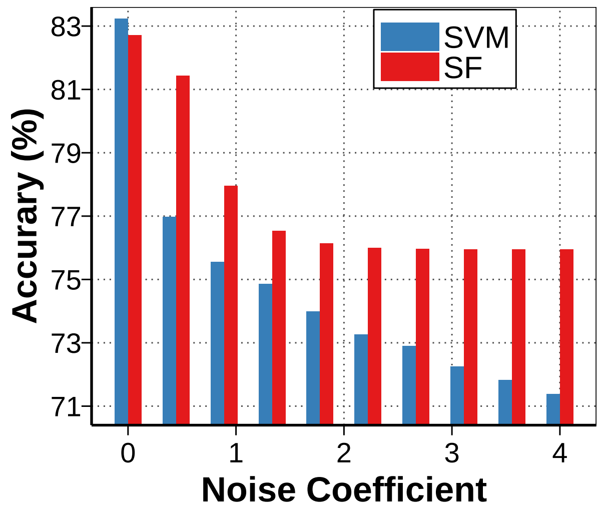
(b) Cosine

- German-number data set

Effect of Noise (Contd.)



(a) RBF



(b) Cosine

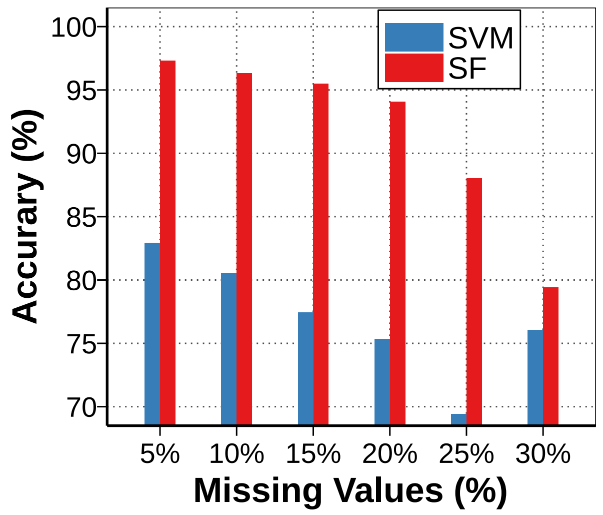
- A1A data set

Fixed Noise Level

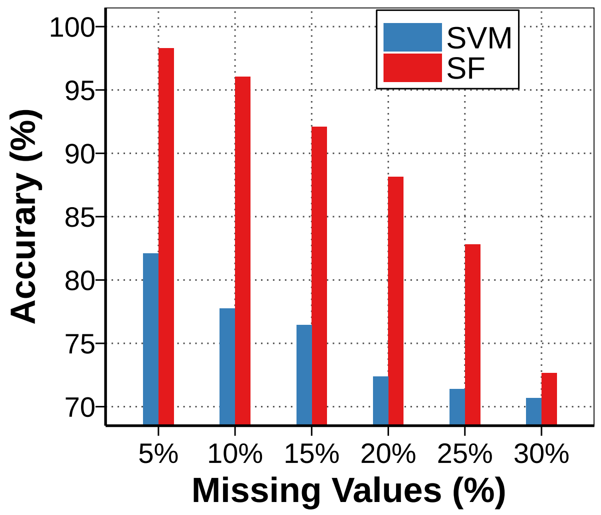
Data set	RBF Kernel		Cosine Similarity	
	SimForest	SVM	*SimForest*	SVM
Heart	**68.14**	67.22	**72.96**	68.70
Ionosphere	**71.69**	68.59	**74.22**	65.91
Australian	**61.88**	61.23	**85.79**	79.49
Diabetes	**68.44**	64.87	**67.33**	62.85
German-Numer	**72.00**	62.80	**71.85**	65.40
a1a	**75.94**	67.81	**75.97**	72.88

- Noise level fixed at $\alpha = 2.5$

Effect of Missing Data



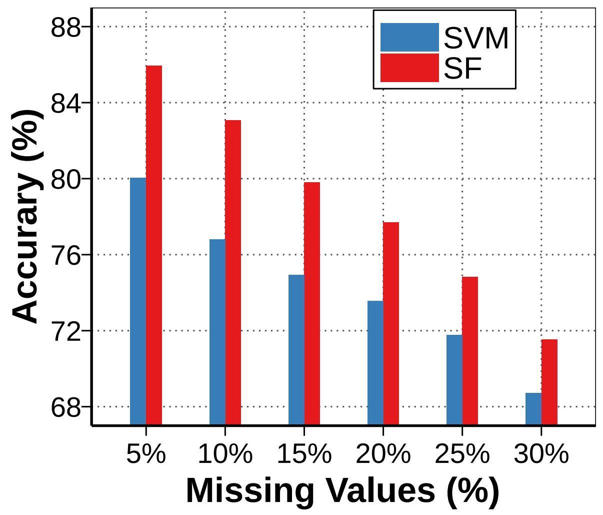
(a) RBF



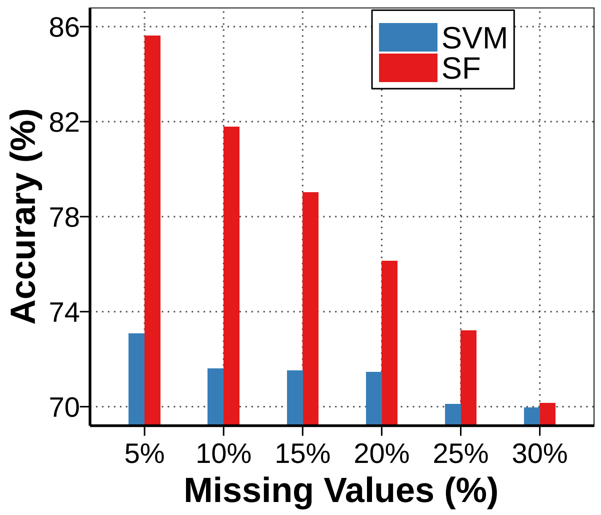
(b) Cosine

- Ionosphere data set

Effect of Missing Data



(a) RBF



(b) Cosine

- Splice data set

Fixed Missing Values

Data set	RBF Kernel		Cosine Similarity	
	SimForest	SVM	*SimForest*	SVM
German-Numer	**72.25**	67.60	**72.05**	64.25
Australian	**90.14**	83.47	**89.63**	86.23
Ionosphere	**95.49**	77.46	**92.11**	76.47
Breast-Cancer	**97.00**	90.94	**96.42**	92.07
Madelon	**56.11**	53.91	**55.65**	53.11
Splice	**79.81**	74.94	**79.02**	71.53

- Missing values fixed at 15%

Multidimensional data

Data set	Heart	ION	WBC	GN	SVMGuide3	a1a	Mushrooms
SimForest	**83.33**	**100.00**	**96.35**	**78.50**	**90.24**	82.81	**100.00**
Random Forest	79.62	94.36	**96.35**	77.00	87.80	82.63	**100.00**
SVM	81.48	87.32	**96.35**	76.00	58.53	**83.42**	**100.00**

- Previous experiments only expose the similarity matrices to the algorithms.
- *SimForest* can even be used for traditional multidimensional data
- Provides competitive performance in such cases

Conclusions and Summary

- Propose *SimForest*, which can be used with traditional similarity matrices
- Resistant to noise in the data
- Can work well with missing values
- Provides competitive performance for traditional multidimensional data