CPDIST: Deep Siamese Networks for Learning Distances Between Structured Preferences

Andrea Loreggia  
University of Padova  
andrea.loreggia@gmail.com

Nicholas Mattei  
IBM Research  
n.mattei@ibm.com

Francesca Rossi  
IBM Research  
University of Padova  
francesca.rossi2@ibm.com

K. Brent Venable  
Tulane University  
kvenabl@tulane.edu

Abstract

Preferences are central to decision making by both machines and humans. Representing, learning, and reasoning with preferences is an important area of study both within computer science and across the sciences. When working with preferences it is necessary to understand and compute the distance between sets of objects, e.g., the preferences of a user and the descriptions of objects to be recommended. We present CPDIST, a novel neural network to address the problem of learning to measure the distance between structured preference representations. We use the popular CP-net formalism to represent preferences and then leverage deep neural networks to learn a recently proposed metric function that is computationally hard to compute directly. CPDIST is a novel metric learning approach based on the use of deep siamese networks which learn the Kendall Tau distance between partial orders that are induced by compact preference representations. We find that CPDIST is able to learn the distance function with high accuracy and outperform existing approximation algorithms on both the regression and classification task using less computation time. Performance remains good even when CPDIST is trained with only a small number of samples compared to the dimension of the solution space, indicating the network generalizes well.

Introduction

Preferences are central for both individual and group decision making by both computer systems and humans. Due to this central role in decision making the study of representing preferences is an important area of study both within computer science and beyond (Goldsmith et al. 2008). The size of the description of the statements, as is often assumed in work with CP-nets (Boutilier et al. 2004) but also mention soft constraints (Bistarelli, Montanari, and Rossi 1997; Rossi, Venable, and Walsh 2011) and GAI-nets (Gonzales and Perny 2004) as they have been used widely across computer science and other disciplines and possess complementary properties in terms of expressivity, reasoning efficiency, and ability to be used for explanation.

CP-nets are a compact graphical model used to capture qualitative conditional preferences over features (variables) (Boutilier et al. 2004). Consider a car that is described by values for all its possible features: make, model, color, stereo options. A CP-net consists of a dependency graph and a set of statements of the form, “all else being equal, I prefer x to y.” For example, in a CP-net one could say “Given that the car is a Honda Civic, I prefer red to yellow.”, where condition sets the context for the preference statement over possible alternatives. These preferences are qualitative as there is no quantity expressing how much I prefer one action over another one. Qualitative preferences are an important formalism as there is experimental evidence that qualitative preferences may more accurately reflect humans’ preferences in uncertain information settings (Roth and Kagel 1995; Allen et al. 2015).

A CP-net induces an over all possible combinations of features or outcomes. This is a partial order if the dependency graph of the CP-net is acyclic, i.e., the conditionality of the statements, as is often assumed in work with CP-nets (Goldsmith et al. 2008). The size of the description of the CP-net may be exponentially smaller than the partial order
it describes. Hence, CP-nets are called a compact representation and reasoning and learning on the compact structure, instead of the full order, is an important topic of research. Recent work by Loreggia et al. (2018a) proposes the first formal metric to describe the distance between CP-nets in a rigorous way. What is important is not the differences in the surface features of the CP-nets, e.g., a single statement or dependency, but rather the distance between their induced partial orders. Even a small difference in a CP-net could generate a very different partial order, depending on which feature is involved in the modification. While the metrics proposed by Loreggia et al. (2018a) are well grounded, they are hard to compute, in general, and approximations must be used.

Following work in metric learning over structured representations (Bellet, Habrard, and Sebban 2015; 2013), we wish to learn the distance between partial orders represented compactly as CP-nets. We do not want to work with the partial orders directly as they may be exponentially larger than the CP-net representation. Informally, given two CP-nets, we wish to estimate the distance between their induced partial orders. Notice that this is a fundamentally different task to metric learning over graphs as, although we estimate the distance between graphs (partial orders), we start from a compact representation and not the induced graphs as input.

In addition to being an interesting fundamental problem there are practical applications as well. As the number of features to over there are practical applications as well. As the number of compact representation and not the induced graphs as input. the distance between graphs (partial orders), we start from a partial orders directly as they may be exponentially larger compounded as CP-nets. We do not want to work with the partial orders because they may be exponentially larger than the CP-net representation. Informally, given two CP-nets, we wish to estimate the distance between their induced partial orders. Notice that this is a fundamentally different task to metric learning over graphs as, although we estimate the distance between graphs (partial orders), we start from a compact representation and not the induced graphs as input.

In addition to being an interesting fundamental problem there are practical applications as well. As the number of CP-nets grows extremely fast, from 481,776 for 4 binary features to over 5.24 × 10^{40} with 7 binary features (Allen et al. 2017) and the computation time of the metrics proposed by Loreggia et al. (2018a) scale linearly with the number of features, new methods must be explored as working with the CP-nets becomes unwieldy quickly. Leveraging the inferential properties of neural networks may help us make CP-nets more useful as a preference reasoning formalism.

The aim of this work is not introducing a new graph learning method, an important topic in machine learning (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016), but rather to merge work in the decision theory area with machine learning techniques. Like earlier work in preference learning (Fürnkranz and Hüllermeier 2010), we wish to apply the power of machine learning, specifically neural networks, to the problems presented by representing and reasoning with preferences. To our knowledge this is the first attempt to use neural nets to learn distances between structured, graphical preference representations.

Contributions We formalize the problem of metric learning on CP-nets, a compact preference representation, that combines elements of graph embeddings, metric learning, and preference reasoning into one problem. We then present CPDIST, which is a siamese network (Bromley et al. 1993) trained using pair of CP-nets represented through their normalized Laplacian matrices and list of cp-statements. We observe that we can decompose the problem into two steps (1) learning a vector representation of the CP-nets and (2) learning the distance metric itself, thus exploring various types of transfer learning is important. We explore both of these options and discuss the use of autoencoders (Hinton and Salakhutdinov 2006) and their impact on performance. We evaluate our approach both quantitatively, by judging the accuracy and mean absolute error (MAE) of CPDIST; and qualitatively, by judging if given two CP-nets we can determine which is closer to a reference point. CPDIST is able to learn a good approximation of the distance function and out perform in terms of both accuracy and speed the current best approximation algorithms. CPDIST gives good performance even when the network is trained with a small number of samples.

CP-nets

CP-nets, short for Conditional Preference networks, were first proposed by Boutilier et al. (2004) are a graphical model for compactly representing conditional and qualitative preference relations. CP-nets are comprised of sets of ceteris paribus preference statements (cp-statements). For instance, the cp-statement, “I prefer red wine to white wine if meat is served,” asserts that, given two meals that differ only in the kind of wine served and both containing meat, the meal with red wine is preferable to the meal with white wine. CP-nets have been extensively used in the preference reasoning (Brafman and Dimopoulos 2004; Cornielio et al. 2013; Rossi, Venable, and Walsh 2011), preference learning (Chevaleyre et al. 2011) and social choice (Brandt et al. 2016; Lang and Xia 2009; Mattei et al. 2013) literature as a formalism for working with qualitative preferences (Domschlag et al. 2011). CP-nets have even been used to compose web services (Wang et al. 2009) and other decision aid systems (Pu et al. 2011).

Formally, a CP-net has a set of features (or variables) \( F = \{ X_1, \ldots, X_n \} \) with finite domains \( D(X_1), \ldots, D(X_n) \). For each feature \( X_i \), we are given a set of parent features \( Pa(X_i) \) that can affect the preferences over the values of \( X_i \). This defines a dependency graph in which each node \( X_i \) has \( Pa(X_i) \) as its immediate predecessors. An acyclic CP-net is one in which the dependency graph is acyclic. Given this structural information, one needs to specify the preference over the values of each variable \( X_i \) for each complete assignment to the the parent variables, \( Pa(X_i) \). This preference is assumed to take the form of a total or partial order over \( D(X_i) \). A cp-statement for feature \( X_i \) that has parents \( Pa(X_i) = \{ x_2, \ldots, x_n \} \) and domain \( D(X_i) = \{ a_1, \ldots, a_m \} \) is a total ordering over \( D(X_i) \) and has general form: \( x_2 = v_1, x_3 = v_2, \ldots, x_n = v_n : a_1 > \ldots > a_m \), where for each \( X_j \in Pa(X_i) \) : \( x_i = v_i \) is an assignment to a parent of \( X_i \) with \( v_i \in D(X_i) \). The set of cp-statements regarding a certain variable \( X_i \) is called the cp-table for \( X_i \).

![Figure 1: A CP-net with n = 4 features.](image)
For a concrete example, consider the CP-net depicted graphically in Figure 1 with features are \( A, B, C, \) and \( D \). Each variable has binary domain containing \( f \) and \( \overline{f} \) if \( F \) is the name of the feature. All cp-statements in the CP-net are:

\[
\begin{align*}
\alpha &\rightarrow \pi, \quad \beta \rightarrow \overline{\beta}, \quad (a \land b) : c \rightarrow \pi, (a \lor b) : \overline{c} \rightarrow \pi, \\
(a \land \overline{b}) : c \rightarrow \pi, \quad (a \lor \overline{b}) : \overline{c} \rightarrow c, \\
(\pi \land b) : \overline{c} \rightarrow c, \quad c : d \rightarrow \overline{d}, \quad \overline{c} : d \rightarrow d.
\end{align*}
\]

Here, statement \( \alpha \rightarrow \pi \) represents the unconditional preference for \( A = a \) over \( A = \overline{a} \), while statement \( c : d \rightarrow \overline{d} \) states that \( D = d \) is preferred to \( D = \overline{d} \), given that \( C = c \).

The semantics of CP-nets depends on the notion of a worsening flip: a change in the value of a variable to a less preferred value according to the cp-statement for that variable. For example, in the CP-net above, passing from \( ab\overline{c}d \) to \( ab\overline{c}d \) is a worsening flip since \( c \) is better than \( \overline{c} \) given \( a \) and \( b \). One outcome \( \alpha \) is preferred to \( \beta \) or dominates another outcome \( \beta \) (written \( \alpha \triangleright \beta \)) if and only if there is a chain of worsening flips from \( \alpha \) to \( \beta \). This definition induces a preorder over the outcomes, which is a partial order if the CP-net is acyclic (Boutilier et al. 2004). The complexity of dominance and consistency testing in CP-nets is an area of active study in preference reasoning (Goldsmith et al. 2008; Rossi, Venable, and Walsh 2011).

Finding the optimal outcome of a CP-net is NP-hard (Boutilier et al. 2004), in general. However, in acyclic CP-nets, there is only one optimal outcome and this can be found in linear time by sweeping through the CP-net, assigning the most preferred values in the cp-tables. For instance, in the CP-net above, we would choose \( A = a \) and \( B = b \), then \( C = c \), and then \( D = d \). In the general case, the optimal outcomes coincide with the solutions of a set of constraints obtained replacing each cp-statement with a constraint (Brafman and Dimopoulos 2004).

### Metric Learning on CP-nets

Metric learning algorithms aim to learn a metric (or distance function) over a set of training points or samples. The importance of metrics has grown in recent years with the use of these functions in many different domains: from clustering to information retrieval and from recommender systems to active constraint (Brafman and Dimopoulos 2004). In many different domains: from clustering to information retrieval and from recommender systems to active constraint (Brafman and Dimopoulos 2004).

Formally, a metric space is a pair \((M, d)\) where \( M \) is a set of elements and \( d \) is a function, formally a distance or metric, \( d : M \times M \rightarrow \mathbb{R} \) where \( d \) satisfies four criteria. Given any three elements \( A, B, C \in M \), \( d \) must:

1. \( d(A, B) \geq 0 \), there must be a value for all pairs;
2. \( d(A, B) = d(B, A) \), \( d \) must be symmetric;
3. \( d(A, B) \leq d(A, C) + d(C, B) \);
4. \( d(A, B) = 0 \) if and only if \( A = B \);
5. \( d(a, b) \) can be zero if and only if the two elements are the same.

Xing et al. (2002) first formalized the problem of metric learning, i.e., learn the metric directly from samples rather than formally specifying the function \( d \). This approach requires training data which means that for each pair of samples we have some oracle that is able to give the value of the metric. The success of deep learning in many different domains (Chopra, Hadsell, and LeCun 2005; Krizhevsky, Sutskever, and Hinton 2012) has lead many researchers to apply these approaches to the field of metric learning, resulting in a number of important results (Bellet, Habrard, and Sebban 2015; 2013).

In this work we focus on metric spaces \((M, d)\) where \( M \) is a set of CP-nets, we want to learn the distance \( d \) which best approximates the Kendall tau distance (KTD) (Kendall 1938) between the induced partial orders. This distance metric, formally given in Definition 1, was formally defined and proved to be a metric on the space of CP-nets by Loreggia et al. (2018a). To extend the classic KTD to CP-nets a penalty parameter \( p \) defined for partial rankings (Fagin et al. 2006) was extended to the case of partial orders. Loreggia et al. (2018a) assume that all CP-nets are acyclic and in minimal (non-degenerate) form, i.e., all arcs in the dependency graph have a real dependency expressed in the cp-statements, a standard assumption in the CP-net literature (see e.g., (Allen et al. 2016; 2017; Boutilier et al. 2004)).

**Definition 1.** Given two CP-nets \( A \) and \( B \) inducing partial orders \( P \) and \( Q \) over the same unordered set of outcomes \( U \):

\[
KTD(A, B) = KT(P, Q) = \sum_{i \neq j} K_{ij}^P(P, Q) \tag{1}
\]

where \( i \) and \( j \) are two outcomes with \( i \neq j \) (i.e., iterate over all unique pairs), we have:

1. \( K_{ij}^P(P, Q) = 0 \) if \( i, j \) are ordered in the same way or are incomparable in \( P \) and \( Q \);
2. \( K_{ij}^P(P, Q) = 1 \) if \( i, j \) are ordered inversely in \( P \) and \( Q \);
3. \( K_{ij}^P(P, Q) = p, 0.5 \leq p < 1 \) if \( i, j \) are ordered in \( P \) and incomparable in \( Q \) (resp. \( Q, P \)).

To make this distance scale invariant, i.e., a value in \([0, 1]\), it is divided by \(|U|\).

CP-nets present two important and interesting challenges when used for metric learning. The first is the fact that we are attempting to learn a metric via a compact representation of a partial order. We are not learning over the partial orders induced by the CP-nets directly, as they could be exponentially larger than the CP-nets themselves. The second challenge is the encoding of the graphical structure itself. Graph learning with neural networks has only recently started receiving serious attention in the literature (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016) including the popular Graph Convolutional Neural Network (GraphGCN) (Kipf and Welling 2016) and methods to speed up the learning of graphs (Chen, Ma, and Xiao 2018). Goyal and Ferrara (2017) give a complete survey of recent work as well as a python library of implementations for many of these techniques. However, most of these works rely on finding good embeddings for the nodes of the network and then using collections of these nodes to represent the graphs. The focus of these techniques is for learning properties of the structure of the graph or properties of the nodes themselves. None of these techniques have been applied to embedding graphs for metric learning.
Structure of CPDIST

The architecture of CPDIST is depicted in Figure 2. In this section we will discuss the encoding used for the CP-nets and the design of our autoencoders, depicted in Figure 3 that are used for transfer learning in this domain. We would like to leverage transfer learning in this domain as training examples becomes prohibitively expensive to compute at higher values of $n$ since computing KTD requires exponential time in the size of the CP-net. Hence, if we can learn a good encoding for CP-nets it may be possible to train a network for small $n$ and use it for problems with larger CP-nets.

![CPDIST Network](image)

**Figure 2: Structure of CPDIST:** CP-nets are provided to the encoder as a normalized Laplacian matrix and a list of cp-statements. The encoders output a compact representation of the CP-nets which is then concatenated and passed to the fully connected layers that connect to an $M$ class classifier over $[0,1]$ to predict KTD.

In our task the metric space $(M, d)$ where $M$ is a set of compact, graphical preferences that induce a partial order and our goal is to learn the metric $d$ only from the compact, graphical representation. The key challenge is to find a vector representation of not only the graph but the cp-statement. We represent a CP-net $I$ over $m$ using two matrices. First is the adjacency matrix $adj_I$ which represents the dependency graph of the CP-net and is a $m \times m$ matrix of 0s and 1s. The second matrix represents the list of cp-statements $cpt_I$, which is a $m \times 2^{m-1}$ matrix, where each row represents a variable $X_i \in F$ and each column represents a complete assignment for each of the variables in $F \setminus X_i$. The list is built following a topological ordering of variables in the CP-net. Each cell $cpt_I(i,j)$ stores the preference value for the $i$th variable given the $j$th assignment to variables in $F \setminus X_i$.

In graph learning the central research topic is how to redefine operators, such as convolution and pooling, as to generalize convolutional neural network (CNN) to graphs (Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016). The most promising research uses a spectral formulation of the problem (Shuman et al. 2013; Bruna et al. 2013) and integrating this into our models is an important direction for future work. We follow in the spirit of the work by Kipf and Welling (2016) for GCN and use a simple convolutional network structure removing pooling layers from CPDIST, as we do not define any pooling operator over the graph structure. In graph spectral analysis, the Laplacian matrix is preferred as it has better properties for encoding, e.g., density, compared to just the adjacency matrix. The Laplacian matrix $L = D - A$, where $D$ is the degree matrix, a diagonal matrix whose $i$th diagonal element $d_i$ is equal to the sum of the weights of all the edges incident to vertex $i$, and $A$ is the adjacency matrix representing the graph. The normalized Laplacian $L = I - D^{-\frac{1}{2}} \times A \times D^{-\frac{1}{2}}$ (Shuman et al. 2013). The set of training examples $X = \{x_1, \ldots, x_n\}$ is made up of pairs of CP-nets represented through their normalized Laplacians and the cp-statements. The set of corresponding labels $Y = \{y_1, \ldots, y_n\}^T$, where each $y_i \in Y$, $y_i \in [0,1]$ is the normalized value of KTD between the CP-nets in $x_i$. Each $x_i \in X$ is then a tuple $(L_A, cpt_A, L_B, cpt_B)$ representing a pair of CP-net $(A, B)$.

![Siamese Autoencoder](image)

**Figure 3: Structure of Siamese Autoencoder:** this version of the autoencoder uses a combined representation for the adjacency matrix and the cp-statements.

The purpose of the two input components of CPDIST, labeled Encoder in Figure 2, is to output a compact representation of CP-nets. To improve performance with networks of this structure, a well-established practice is to train an autoencoder (Hinton and Salakhutdinov 2006; Lecun and Bengio 1995) separately, and then transfer the weights of the portion of the network to the main network. We will evaluate two different approaches to transfer learning in our setting as well as not using any transfer learning. First, we use two different autoencoders: one for the normalized Laplacian matrix and the other for the cp-statements. The two autoencoders are trained separately and then weights are transferred to the main network. We denote this approach as Autoencoder in subsequent experiments. In the second approach, shown in Figure 3 and denoted as Siam. Autoencoder, we use a unique autoencoder designed to combine the two components of CP-nets. Informally, the output of two encoders are concatenated and then split into their components to be decoded. We conjecture that this combination should allow more information about the CP-net to be used for learning the compact representation.

**Experiments**

We train CPDIST to learn the KTD metric, varying the number of features of the CP-nets $n \in \{3, \ldots, 7\}$ and using two
different autoencoder designs. We evaluate our networks on both the regression and classification tasks and measure their performance against the current best approximation algorithm for computing the KTD between two CP-nets called I-CPD Loreggia et al. (2018a). In the regression task the network computes the distance value exactly while in the classification task we divide the output in \( m = 10 \) intervals and the network must select the correct interval. CP-nets are represented by their normalized Laplacian and the list of the cp-statements in the CP-tables following a topological order of the variables. The encoder components of the network is composed by two CNNs (Lecun and Bengio 1995); one receives as input the normalized Laplacian matrixes of the CP-net and the other the lists of cp-statements. The aim of the encoder to output a compact representation of each CP-net. We will discuss data generation, training and mention challenges due to the unique properties of CP-nets.

**Data Generation and Training**

For each number of features \( n \in \{3, \ldots, 7\} \) we generate 1000 CP-nets at uniformly at random using the generators from Allen et al. (Allen et al. 2016; 2017). This set of CP-nets is split into a training-generative-set (900 CP-nets) and test-generative-set (100 CP-nets). From these two datasets we compute the training and test dataset comprised of all possible pairs of CP-nets from the training-generative-set and test-generative-set, respectively, along with the value of KTD for that pair. These two datasets are unbalanced by design since CP-nets distance values are not distributed uniformly. Figure 4 shows the distribution of CP-net pairs over 20 intervals for all CP-nets generated for \( n \in \{3, \ldots, 7\} \). While our classification experiments are for \( M = 10 \) classes, dividing the interval into 20 classes provides a better visualization of the challenge of obtaining training samples at the edges of the distribution. Much of this comes from the fact that since KTD is a formal metric, there is only one CP-net that induces distance 1 and one CP-net to induce distance 0: a copy of the CP-net under test and the CP-net where all cp-statements are exactly reversed.

All training was done on a machine with 2 x Intel(R) Xeon(R) CPU E5-2670 @ 2.60GHz and one NVidia K20 128GB GPU. We train CPDIST for 70 epochs using the Adam optimizer (Kingma and Ba 2014). For each number of features of the CP-net \( n \) we use all \( \binom{900}{2} \) pairs in the training-set. There are only 488 binary CP-nets with 3 features (Allen et al. 2017), hence, for \( n = 3 \) the training-set is 17K samples while for \( n > 3 \) the number of samples in the training-set is 800K. Both the Autoencoder and Siamese Autoencoder models are trained for 100 epochs using the Adam optimizer (Kingma and Ba 2014) using the same training-set. Model weights from the best performing epoch are saved and subsequently transferred to the deep neural network used to learn the distance function.

The training and validation loss for the autoencoder is shown in Figure 5. Observe that the loss for the CPT representation approaches zero after only 3 epochs for both the training and validation phases. The same trend is true for the adjacency matrix, though the loss converges to \( \approx 0.15 \).

### Quantitative Performance: Classification and Regression

The first task for CPDIST is classifying the distance between two CP-nets, \( A \) and \( B \), into the same one of \( M = 10 \) intervals of \([0, 1]\) where the value of KTD lies. Table 2 gives the F-score, Cohen’s Kappa (Cohen 1995) and mean absolute error (MAE) for the task with no auto encoder and each of the two autoencoder variants. Cohen’s \( \kappa \) is a measure of inter-rater agreement where are two raters are the particular instance of CPDIST and the actual value of KTD. We measure mean absolute error as a value over the number of intervals between the value returned by CPDIST and KTD. For example, a MAE of 1.0 means that CPDIST is off by one interval, on average. In this setting, using a random classifier to guess the interval with \( M = 10 \) possible intervals and a normal distribution like the one seen in Figure 4 would give an F-score \( \approx 0.19 \).

Looking at Table 2 we see that CPDIST state of the art performance across the test instances when compared to the I-CPD approximation algorithm. The overall accuracy, measured as F-score, is above 70% across all CP-net sizes and we see that on average it is off by less than 0.5 intervals as measured by the MAE. The values for Cohen’s \( \kappa \) indicate good agreement between the two methods and this is borne out by high accuracy numbers. The most interesting overall effect in Table 2 is that the performance does not decay too much as we increase the number of features. Indeed, the F-score remains very stable across the range. We interpret this to mean that CPDIST is learning a good generalization of the distance function even when the solution space is exponentially larger than the number of training examples.

Table 3 we see the results of the much harder regression task. Again we see that CPDIST is able to out perform the state of the art I-CPD approximation across the board. While for \( n = 3 \) the values are similar, for \( n \in \{4, \ldots, 7\} \) CPDIST is giving a \( \approx 30\% \) decrease in error, \( \approx 0.015 \) absolute decrease. Looking at results from Table 1 we can see that CPDIST is doing this significantly faster than I-CPD as well. In is interesting to note that in Table 3 all versions of our network are outperforming I-CPD, whether or not we first train the autoencoder.

Turning to the question of transfer learning for this task we see that the use of the autoencoders strictly increases the performance of the network on the classification and regression task. In both cases the best performing networks use one of the two autoencoder variants we tested. The Siamese Autoencoder slightly out performs the plain Autoencoder when looking at MAE for the classification task, though the results are more mixed for F-score and Cohen-\( \kappa \). In the regression task the Siamese Autoencoder is better at the end points and the two networks are statistically indistinguishable for \( n \in \{4, 5, 6\} \). These results indicate that the use of an autoencoder can significantly help in this task, though the exact design of that autoencoder remains an important question for future work. Important future work is using an autoencoder trained for a smaller number of features to bootstrap learning for larger numbers of features.
Figure 4: Histogram of the number of CP-net pairs per interval across all experimental datasets. CP-nets pairs are not distributed uniformly in the class intervals.

Table 1: Comparison of the mean runtime for a single triple over 1000 trials on the qualitative comparison task of the neural network and I-CPD (Loreggia et al. 2018a).

| N   | I-CPD       | Autoencoder Neural Network |
|-----|-------------|----------------------------|
| 3   | 0.75 (0.43) msec | 0.087 (0.004) msec         |
| 4   | 1.04 (0.21) msec | 0.098 (0.004) msec         |
| 5   | 1.78 (0.41) msec | 0.100 (0.005) msec         |
| 6   | 3.45 (0.50) msec | 0.114 (0.003) msec         |
| 7   | 6.79 (1.33) msec | 0.138 (0.001) msec         |

Table 2: Performance of CPDIST on the classification task with and without the autoencoders. Numbers in parenthesis are standard deviations. Mean absolute error is computed as the number of intervals between the true and predicted values for the classification task.

| N   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   |
|-----|---------|---------|-------|---------|---------|-------|---------|---------|-------|---------|---------|-------|
|     | No Autoencoder | Autoencoder |     | Siam. Autoencoder |       | I-CPD |
| 3   | 0.6643 (0.0275) | 0.6113 | 0.3449 | 0.7051 (0.0306) | 0.6578 | 0.2986 | 0.7295 (0.0501) | 0.6860 | **0.2734** | 0.4235 |
| 4   | 0.7424 (0.0096) | 0.6762 | 0.2582 | 0.7483 (0.0085) | 0.6824 | **0.2525** | 0.7459 (0.0088) | 0.6796 | 0.2548 | 0.4515 |
| 5   | 0.7074 (0.0111) | 0.6146 | 0.3015 | 0.7271 (0.0084) | 0.6385 | 0.2833 | 0.7278 (0.0077) | 0.6393 | **0.2831** | 0.3875 |
| 6   | 0.6945 (0.0130) | 0.5799 | 0.3194 | 0.7157 (0.0198) | 0.6073 | 0.2971 | 0.7161 (0.0141) | 0.6081 | **0.2969** | 0.3645 |
| 7   | 0.6887 (0.0227) | 0.5571 | 0.3256 | 0.6497 (0.0892) | 0.4957 | 0.3830 | 0.6884 (0.0274) | 0.5549 | **0.3266** | 0.3340 |

Table 3: MAE of CPDIST on the regression task with and without the autoencoders.

| N   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   | F-score | Cohen-κ | MAE   |
|-----|---------|---------|-------|---------|---------|-------|---------|---------|-------|---------|---------|-------|
|     | No Autoencoder | Autoencoder |     | Siam. Autoencoder |       | I-CPD |
| 3   | 0.0470 | 0.0426 | **0.0421** | 0.0460 |
| 4   | 0.0247 | **0.0242** | 0.0243 | 0.0434 |
| 5   | 0.0269 | **0.0261** | 0.0262 | 0.0380 |
| 6   | 0.0256 | **0.0255** | 0.0256 | 0.0352 |
| 7   | 0.0256 | 0.0257 | **0.0252** | 0.0335 |

Qualitative Comparison Task Performance
For many applications we are not concerned with the true value of the distance but rather deciding which of two preferences is closer to a reference point. For example, in product recommendation we may want to display the closer of two objects and not care about computing the values (Pu et al. 2011). Formally, the qualitative comparison takes takes a set of CP-nets triples \((A, B, C)\), where \(A\) is a reference CP-net and the task is to decide which other CP-net \(B\) or \(C\) is closer to \(A\). We generate uniformly at random 1000 triples of CP-nets for each \(n \in \{3, \ldots, 7\}\). For each triple \((A, B, C)\) we compute both \(\text{KTD}(A, B)\) and \(\text{KTD}(A, C)\) to
Table 4: Performance of the various network architectures on the qualitative comparison task as well as performance of I-CPD (Loreggia et al. 2018a). While our networks do not achieve the best performance on this task they are within the margin of error ($\approx 1.0\%$).

|   | No Autoencoder | Autoencoder | Siam. Autoencoder | I-CPD |
|---|----------------|-------------|-------------------|-------|
| N | Accuracy on Triples | Accuracy on Triples | Accuracy on Triples | Accuracy on Triples |
| 3 | 85.01% (2.01%) | 85.76% (2.29%) | 85.47% (2.32%) | 91.80% |
| 4 | 91.17% (0.92%) | 91.38% (1.10%) | 91.78% (1.13%) | 92.90% |
| 5 | 88.40% (0.91%) | 89.36% (1.08%) | 89.18% (1.08%) | 90.80% |
| 6 | 87.33% (0.80%) | 87.17% (1.33%) | 86.79% (1.84%) | 90.10% |
| 7 | 84.79% (1.16%) | 84.57% (1.14%) | 85.12% (0.86%) | 89.90% |

Table 4: Performance of the various network architectures on the qualitative comparison task as well as performance of I-CPD (Loreggia et al. 2018a). While our networks do not achieve the best performance on this task they are within the margin of error ($\approx 1.0\%$).
Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. arXiv abs/1312.6203.

Chen, J.; Ma, T.; and Xiao, C. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In Proceedings of the 6th International Conference on Learning Representations (ICLR).

Chevalley, Y.; Koriche, F.; Lang, J.; Mengin, J.; and Zanuttini, B. 2011. Learning ordinal preferences on multi-attribute domains: The case of CP-nets. In Preference Learning. Springer. 273–296.

Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a similarity metric discriminatively, with application to face verification. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, 539–546.

Cohen, P. R. 1995. Empirical Methods for Artificial Intelligence. MIT Press.

Cornelio, C.; Goldsmith, J.; Mattei, N.; Rossi, F.; and Venable, K. 2013. Updates and uncertainty in CP-nets. In Proceedings of the 26th Australasian Joint Conference on Artificial Intelligence (AUSAI).

Cover, T., and Hart, P. 2006. Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. 13(1):21–27.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In Proceedings of the 30th International Conference on Neural and Information Processing (NIPS), 3837–3845.

Domshlak, C.; Hübnermeier, E.; Kaci, S.; and Prade, H. 2011. Preferences in AI: An overview. Artificial Intelligence 175(7):1037–1052.

Fagin, R.; Kumar, R.; Mahdian, M.; Sivakumar, D.; and Vee, E. 2006. Comparing partial rankings. SIAM J. Discret. Math. 20(3):628–648.

Fürnkranz, J., and Hüllermeier, E. 2010. Preference Learning. Springer.

Goldsmith, J., and Junker, U. 2009. Preference handling for artificial intelligence. AI Magazine 29(4).

Goldsmith, J.; Lang, J.; Truszczynski, M.; and Wilson, N. 2008. The computational complexity of dominance and consistency in CP-nets. Journal of Artificial Intelligence Research 33(1):403–432.

Gonzalez, C., and Perny, P. 2004. GAI networks for utility elicitation. In Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR).

Goyal, P., and Ferrara, E. 2017. Graph embedding techniques, applications, and performance: A survey. CoRR abs/1705.02801.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. arXiv abs/1506.05163.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. Science 313(5786):504–507.

Kendall, M. G. 1938. A new measure of rank correlation. Biometrika 30(1/2):81–93.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. arXiv/1412.6980.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. arXiv abs/1609.02907.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, 1097–1105.

Lang, J., and Xia, L. 2009. Sequential composition of voting rules in multi-issue domains. Mathematical Social Sciences 57(3):304–324.

Lecun, Y., and Bengio, Y. 1995. Convolutional Networks for Images, Speech and Time Series. The MIT Press. 255–258.

Li, M., and Kazimipour, B. 2018. An efficient algorithm to compute distance between lexicographic preference trees. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), 1898–1904.

Lloyd, S. 2006. Least squares quantization in pcm. IEEE Trans. Inf. Theor. 28(2):129–137.

Loreggia, A.; Mattei, N.; Rossi, F.; and Venable, K. B. 2018a. On the distance between CP-nets. In Proceedings of the 17th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).

Loreggia, A.; Mattei, N.; Rossi, F.; and Venable, K. B. 2018b. Preferences and ethical principles in decision making. In Proceedings of the 1st AAAI/ACM Conference on AI, Ethics, and Society (AIES).

Loreggia, A.; Mattei, N.; Rossi, F.; and Venable, K. B. 2018c. Value alignment via tractable preference distance. In Yampolskiy, R. V., ed., Artificial Intelligence Safety and Security, CRC Press, chapter 16.

Mattei, N., and Walsh, T. 2013. PrefLib: A library for preferences, http://www.PrefLib.org. In Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT).

Mattei, N.; Pini, M. S.; Rossi, F.; and Venable, K. B. 2013. Bribery in voting with CP-nets. Annals of Mathematics and Artificial Intelligence 68(1–3):135–160.

Pigozzi, G.; Tsoukias, A.; and Viappiani, P. 2015. Preferences in artificial intelligence. Annals of Mathematics and Artificial Intelligence 77:361–401.

Pu, P.; Faltings, B.; Chen, L.; Zhang, J.; and Viappiani, P. 2011. Usability guidelines for product recommenders based on example critiquing research. In Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds., Recommender Systems Handbook. Springer. 511–545.

Rossi, F.; Venable, K.; and Walsh, T. 2011. A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice. Morgan and Claypool.

Roth, A., and Kagel, J. 1995. The Handbook of Experimental Economics, volume 1. Princeton University Press Princeton.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. 30(3):83–98.

Wang, H.; Shao, S.; Zhou, X.; Wan, C.; and Bouguettaya, A. 2009. Web service selection with incomplete or inconsistent user preferences. In Proc. 7th International Conference on Service-Oriented Computing. Springer. 83–98.

Wang, H.; Zhang, J.; Wan, C.; Shao, S.; Cohen, R.; Xu, J.; and Li, P. 2010. Web service selection for multiple agents with incomplete preferences. In 2010 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 565–572.

Xing, E. P.; Ng, A. Y.; Jordan, M. I.; and Russell, S. J. 2002. Distance metric learning with application to clustering with side-information. In Proceedings of the 15th International Conference on Neural and Information Processing (NIPS), 505–512.