Case Adaptation with Neural Networks: Capabilities and Limitations

Xiaomeng Ye[0000−0002−2289−1022], David Leake[0000−0002−8666−3416] and David Crandall

Luddy School of Informatics, Computing, and Engineering
Indiana University, Bloomington IN 47408, USA
{xiaye, leake, djcran}@indiana.edu

Abstract. Neural network architectures for case adaptation in case-based reasoning (CBR) have received considerable attention. However, architectural gaps and general questions remain. First, existing architectures focus on adaptation of numeric attributes alone. Second, some proposed neural network adaptation architectures operate directly on pairs of cases, so could be performing direct prediction instead of adaptation. Third, it is unclear how the effectiveness of CBR systems with neural network components compares to that of networks alone. This paper addresses these questions. It extends a neural network-based case difference heuristic (NN-CDH) approach to handle both numeric and nominal attributes, in an architecture that applies to both regression and classification domains. The network predicts solution difference based on problem difference, ensuring that it learns adaptations. The paper presents experiments for both classification and regression tasks that compare performance of a neural network to a baseline CBR system and CBR variants with different retrieval schemes and adaptation schemes, on both real data and controlled artificial data sets. In these tests, CBR with the extended NN-CDH generally performs comparably to the baseline neural network, and NN-CDH consistently improves the results from naive retrieval but may worsen the results of network-based retrieval.

Keywords: Case Adaptation, Case Difference Heuristic, Hybrid Systems, Neural Network-Based Adaptation, Nominal differences

1 Introduction

Neural network architectures for case adaptation have been the subject of considerable study within the case-based reasoning community. However, gaps remain both in architectural capabilities and in fundamental questions of system behavior. First, existing architectures focus on adaptation of numeric attributes; adapting nominal attributes is an open challenge, made harder because of the lack of standard methods for expressing such adaptations in network architectures. Second, some proposed neural network adaptation architectures take as input a query and retrieved case and generate a solution, which raises a question about whether they truly learn adaptation——in principle, they could learn

to ignore the prior case and simply generate a solution from scratch. Third, the performance benefit of using CBR systems with neural network components over other models is unclear. This paper presents research on these questions.

This paper introduces a method for neural network adaptation of nominal attributes based on expressing the difference between two one-hot encoded nominal values as a vector, and extends the NN-CDH adaptation approach [12] to predict a solution difference given a problem difference, where the differences can involve nominal attributes. In contrast to the previous version of NN-CDH for classification, this guarantees learning adaptation knowledge. Extensive experiments on standard regression and classification data sets, and on artificial data sets, compare the extended NN-CDH with other models. Some trends of results paralleled those obtained in previous tests [12, 20], providing additional support for those trends as general characteristics of neural network adaptation.

The experimental results suggest that NN-CDH is most useful when (1) retrieval is relatively good, in the sense of providing a starting point harmonized to learned adaptation knowledge (cf. [9, 18]) and (2) queries are relatively novel, so adaptation is needed, and yet not too novel, which could require adaptation capabilities beyond the learned adaptation knowledge.

The results illustrate three lessons. First, good case retrieval alone may surpass other ML methods such as neural networks. Second, learned adaptation may actually worsen the retrieval result if the two processes are not harmonized. This point is beyond the scope of this paper but examined in our previous work [11]. Third, directly integrating neural networks into a CBR system may lead to results comparable to the counterpart neural network, but is not a "magic bullet" for superior performance. Even with comparable performance, CBR offers benefits such as interpretability. However, the results suggest that to reliably surpass statistical AI methods, symbolic knowledge will be necessary.

2 Background

Neural Network Adaptation Learning by the Case Difference Heuristic: CBR researchers have explored many machine learning techniques for acquiring case adaptation knowledge. Because of the difficulty of codifying adaptation knowledge, especially in poorly understood domains, there has been particular interest in data-driven methods [2–4, 13, 17]. Hanney and Keane's case difference heuristic (CDH) approach [5] is a method for learning adaptation knowledge from the case base, by comparing pairs of cases and attributing their solution difference to their problem difference. Originally, the CDH approach was applied to learning adaptation rules, but other variants have been explored, including network-based approaches. Liao, Liu, and Chao [13] train a neural network on case pairs to predict solution difference based on problem difference. Leake, Ye, and Crandall [12] take a similar approach but train the neural network with adaptation context in addition to problem/solution differences. They call this method the neural network-based case difference heuristic (NN-CDH) approach.

Case Adaptation Involving Nominal Attributes: The work mentioned in Section 2 all focuses on case adaptation for regression tasks, based on numeric attributes. As noted by Craw et al. [3], adaptation involving nominal attributes is difficult compared to numerical attributes because there is no natural calculation for symbolic similarity (for retrieval) or dissimilarity (for adaptation). Some previous works have studied non-neural methods for learning case adaptation involving nominal attributes. For example, Jarmulak, Craw and Rowe [8] present a case-based method that learns adaptation cases for both numeric and nominal features. Jalali, Leake, and Forouzandehmehr [6] combine a statistical method with an ensemble approach. Craw et al. [3] propose two methods to adapt nominal attributes: first, a coarse-grained method indicating whether a nominal attribute should remain the same or change, and second, a multi-class adaptation which proposes the target solution directly.

Initial Version of NN-CDH for Classification: The initial version of NN-CDH for classification followed the second route proposed by Craw et al. [3]; Ye et al. [20] modify NN-CDH into an approach called C-NN-CDH which predicts a target solution (class label) based on a source case and a target problem. It was tested in comparison to other classification algorithms including SVMs and neural networks. Those experiments supported that C-NN-CDH can achieve state-of-art case adaptation and classification results.

However, C-NN-CDH operates differently from other CDH methods (including NN-CDH) that work directly with problem/solution differences. C-NN-CDH takes a source case and a target problem as inputs. It is not guaranteed to make use of the source case to truly perform adaptation; in principle it could learn to find a target solution based on the target problem only. In response, this paper proposes a single adaptation model that learns from differences of nominal features/labels and therefore guarantees the learning of adaptation knowledge.

NN-CDH can handle nominal feature differences but not nominal label differences. To address this, we use the following encoding scheme. If a nominal feature is one-hot encoded into a vector of numbers (with a single "1" and multiple "0"s), it can be treated as multiple numerical features and processed by the neural network. Two such vectors can be subtracted from one another to form a nominal difference, which can be then passed to the input layer of NN-CDH. However, NN-CDH's last layer uses a single neuron with the linear activation function to predict a numerical output. This numerical output is the solution difference in a regression task. This output neuron cannot represent a nominal difference in the solution.

Both the studies on NN-CDH and C-NN-CDH methods (we will refer to both as NN-CDHs for short) compared CBR systems using NN-CDHs with their counterparts, neural network systems that predict target solution directly from target problem [12, 20]. Ye et al. [20] show that NN-CDH performs comparably to the counterpart neural network system on certain data sets. Leake, Ye and Crandall [12] hypothesize that NN-CDHs can outperform end-to-end network methods, for example, when the query is novel in a high dimensional space.

However more evidence is needed to assess this hypothesis. This paper reports experiments to elucidate NN-CDH behavior.

Siamese Networks for Similarity Assessment: Siamese networks [1] are a neural network architecture that can predict the distance between two input samples. A siamese network is composite of two identical feature extraction networks and a subtraction layer. Given two input vectors (in our context, two feature vectors, one describing a query and the other the problem addressed by a prior case), the feature extraction networks extract their features and the subtraction layer compares their features and outputs a value indicating the distance (often using a sigmoid function) between the two cases. Siamese networks have been used as the similarity measure in case retrieval [14, 15], with good performance. This study refers to a KNN retrieval process that uses a siamese network as similarity measure as *SN retrieval*.

The Relationship Between Retrieval and Adaptation: Smyth and Keane observed that for efficient adaptation, CBR retrieval and adaptation knowledge must be tightly connected [18], and Leake, Kinley and Wilson [9] illustrated the efficiency benefit of coordinating similarity and adaptation learning. Leake and Ye [11] showed that even when strong network models are trained for both retrieval and adaptation, adapted solutions might be less accurate than the retrieved solution if the retrieval and adaptation knowledge are not coordinated—*i.e.*, if the retrieved cases are close to the real solution by the similarity metric but not easily adaptable, or if the adaptation model is not trained to adapt such retrieval results. The experiments in this study tested a CBR system adapting results from either 1-NN retrieval or SN retrieval. The experiments further support that when both the retrieval model and the adaptation model are well trained individually but not in harmony with each other, adaptation may worsen the result of retrieval.

3 NN-CDH for both Classification and Regression

The research reported in this paper extends NN-CDH to perform adaptation for both regression and classification task domains. If the domain is regression, the model works identically to the original NN-CDH [12]. The following first explains the case adaptation process applicable to both classification and regression domains, then introduces a network method for handling nominal difference, and last discusses the neural network structure of NN-CDH and how it works with both numerical and nominal differences.

3.1 General Model of Case Adaptation

Given a query describing a target problem, the standard CBR adaptation process first retrieves a case whose problem is similar to the query. The system calculates the difference between the retrieved problem and the query problem,

and modifies the solution of the retrieved case based on the problem difference. NN-CDH augments the difference information by using the retrieved problem as context for the adaptation. It uses both that context and the problem difference to predict a solution difference.

If the system task is a regression task, NN-CDH calculates the difference of two values over a numerical attribute by subtracting the two values. However, in classification tasks, each case contains a solution description that is a nominal label (Multi-class labeling is not within the scope of this study). The neural network of NN-CDH needs to handle outputs of differences of nominal attributes.

3.2 1-Hot/1-Cold Nominal Difference

NN-CDH handles nominal differences as follows. Given a nominal-valued attribute n that can take any of a set d different possible values for that attribute, one-hot encoding converts n into a group of d binary bits $\{n_1, n_2, ...n_d\}$, where n_i is '1' and all others bits are '0's. Given two nominal values $n = \{n_1, n_2, ...n_d\}$ and $m = \{m_1, m_2, ...m_d\}$ of the same attribute, their 1-hot/1-cold nominal difference (or "nominal difference" for short) is defined as

$$n - m = \{n_1 - m_1, n_2 - m_2, ...n_d - m_d\} \tag{1}$$

Given two cases in a classification task, their problem difference is calculated as a vector of individual feature differences concatenated, which may or may not involve nominal differences; Their solution difference is the nominal difference of their class labels. If the two cases are of the same class, then the solution difference is all '0's, indicating no difference between their solutions; Otherwise, the solution difference contains exactly one '1' and one '-1' (hence the name "1-hot/1-cold"), indicating changing from one class into another. Nominal differences are vectors that can be either input or output of NN-CDH, allowing NN-CDH to learn the projection from problem difference to solution difference in a classification task.

As a side benefit, the problem difference may contain both nominal and numerical differences, allowing interaction between the two categories of feature differences. This benefit is similar to that of a classification neural network which learns one-hot encoded nominal values and numerical values together.

3.3 Neural Network Structure of NN-CDH

Given a neural network structure for a given domain and task, the network structure of the NN-CDH for it is similar. The main difference is in its first and last layers. The first layer needs to have more neurons to accommodate both adaptation context and problem difference. The last layer needs to express predicted solution difference. The last layer is a dense layer using the *tanh* activation function, as opposed to the linear activation function in the original NN-CDH.

For regression, the last layer is a single neuron, producing a solution difference between -1 and 1 to increase or decrease a retrieved solution. For classification,

Algorithm 1 Pseudocode for the Training and Usage of NN-CDH

```
1: procedure Training(cases)
2:     pairs ← assembled pairs of cases
3:     NN-CDH ← new neural network
4:     CDH_data ← {}
5:     for each source and target in pairs do
6:         CDH_data.append(
7:         [prob(source),prob(source) − prob(target):sol(source) − sol(target)])
8:     NN-CDH.fit(CDH_data)
9:     return NN-CDH
10: procedure Adapt(query,retrieval)
11:     retrieved ← retrieval(CB, query)
12:     sol_diff ← NN-CDH.predict(prob(retrieved), prob(retrieved) − prob(query))
13:     if Task is regression then
14:         r ← sol(retrieved) − sol_diff
15:     else if Task is classification then
16:         r_vector ← sol(retrieved) − sol_diff  #element-wise subtraction
17:         r ← argmax(r_vector)
18:     return r
```

the last layer is a dense layer with d neurons, where d is the dimension of a one-hot encoding of the classification label. Each neuron can have a value between -1 (indicating "change to this class") and 1 (indicating "change from this class").

3.4 Training and Adaptation Procedure

The training and adaptation procedures follow the original NN-CDH, but with some modification. They are illustrated in Algorithm 1.

During training, pairs of training cases are assembled as training data for NN-CDH. For each pair, their problem difference and solution difference are calculated. One problem description of the two cases is used as the adaptation context. The problem difference and the adaptation context are concatenated to form the input of the NN-CDH and the solution difference is the expected output of the NN-CDH. Last, the NN-CDH is trained using backpropagation and the mean squared error as the loss function.

During adaptation, the NN-CDH predicts a solution difference based on a problem difference and an adaptation context. Depending on the task domain, the retrieved solution is modified according to the solution difference in one of two ways. If it is a regression task, the retrieved solution subtracting the predicted solution difference forms the final solution (as in the original NN-CDH). If it is a classification task, the retrieved solution is a one-hot encoding of the class of the retrieved case. The retrieved solution subtracts the predicted solution difference element by element. The maximum bit of the resulting vector is used to determine the final solution (Ties are broken arbitrarily).

4 Evaluation

Our evaluation addresses two questions: Does NN-CDH consistently improve the result of retrieval, and how does a CBR system using NN-CDH perform when compared to a neural network of equivalent computational power?

4.1 Systems Being Compared

This study tests six different methods and compares their performance in terms of prediction accuracy. They are: 1) a baseline neural network, 2) k-nearest neighbor (k-NN), 3) a CBR system whose retrieval is either 1-NN retrieval or SN retrieval and whose adaptation is either a rule-based CDH or an NN-CDH. The rule-based CDH is a baseline adaptation method (referred as CB_A in the work of Craw et al. [3]) where the case pairs are stored in an adaptation case base. The rule-based CDH uses a non-optimised 1-NN retrieval to select the case pairs based on the adaptation context and problem difference and applies the solution difference as the adaptation. Because there are two variations of retrieval and two of adaptation, four variations of CBR systems are tested. The six different methods are referred as "all models" in the following text.

The neural networks may have more or fewer layers and neurons per layer to accommodate the varying complexity of different data sets. In our implementation, the neural networks have 2-4 hidden layers, each of which has 8-128 neurons. For every task domain, the baseline neural network is almost identical to the NN-CDH and they share the same number of layers, neurons and activation functions, except for the first layer because NN-CDH takes in adaptation context and problem difference. This is to ensure fair comparison of the two models because they share similar complexity. The structure of the NN-CDH is as discussed in Section 3.3. The last layer of the baseline neural network uses a sigmoid activation function for regression or softmax activation function for classification. The baseline neural network model is trained with the loss function of mean squared error in regression or with the loss function of categorical cross entropy for classification. Such designs of neural networks for regression and classification are widely used.

Both k-NN and 1-NN are default implementations from the scikit-learn package [16]. They use Euclidean distance over problem feature values to calculate distance between cases. All cases are weighted equally. Our k-NN used k=3. When the CBR system uses a siamese network over the Euclidean distance for similarity measure, a separate siamese network is trained to predict the similarity of two given cases.

The siamese network for case retrieval measures the similarity between two cases. The feature extraction network is composite of three dense layers (of dimension 32, 32 and 16) with dropout layers (dropout rate = 0.1) in between. For harder problems, the number of neurons in each layer is multiplied by 4.

The two features extracted from two cases are then passed to the subtraction layer which outputs the element-wise feature distance. Last, this is passed into a final dense layer to output a single similarity score. The final layer uses the

ReLU activation function for regression or the sigmoid activation function for classification. The siamese network is trained with the mean average error loss for regression or the contrastive loss for classification.

4.2 Assembling Case Pairs for Training

While the baseline neural network and the k-NN are trained from cases directly, both the siamese network and the NN-CDH need case pairs for training. Using all possible pairs may be infeasible, as a case base of n cases would have n^2 pairs. Multiple strategies exist for selective case pair assembly [7]. In all experiments of this study, we use n neighboring pairs (each case is paired with its nearest neighbor) and $10n$ random pairs (each case is randomly paired with another case). From these pairs, 90% are used for training and the rest for validation. We observed that this design choice can heavily influence the models, for example, a model may not accommodate two highly different input cases if the model is not trained with enough random pairs. However, this design choice is not the focus of this study.

To train the siamese network, we assemble pairs of cases' problem descriptions as input and determine their distance as the expected output of the siamese network. For regression, the distance is the absolute value of the distance between the two cases' solutions. For classification, the distance is 0 if the two solutions are the same and 1 otherwise. To train the NN-CDH, we use problem differences and solution differences of the case pairs.

4.3 Data Sets

This experiments test on five standard regression data sets (Airfoil, Car, Student Performance, Yacht, Energy Efficiency) and five classification data sets (Credit, Balance, Car, Yeast, Seeds), as well as on artificial data described in Section 4.4. As preprocessing, all numeric attributes are scaled to the range of [0,1] and all nominal attributes are one-hot encoded. This way the expected output values match the output range of the NN-CDH. Each data set is tested with 10-fold cross validation, and on three different settings:

- The normal setting: The standard setting where 90% of the cases are used as the training set (out of which 90% of the cases are used for training and the rest for validation) and 10% as the test set. The case pairs are assembled from the training set. The models are first trained and then tested on the whole test set.
- The novel setting (X): Similar to the normal setting with the difference that: For every test case, we remove its top R% neighbors (based on Euclidean distance on their *problem* descriptions) in the train set to form a trimmed train set. All models are trained on the *trimmed train set* and then tested on *that single test case*. This follows the design of a previous novel setting experiment [12].

– The novel setting (Y): Similar to the novel setting (X), we still remove top $R\%$ neighbors of a test case but the Euclidean distance is based on the solution description rather than the problem description. All models are trained on the trimmed train set and then tested on that single test case. This is a modification of another previous novel setting experiment [21].

The novel setting (X) simulates when the models are not trained with cases whose problem descriptions are similar to the query problem. The novel setting (Y) simulates when the models are not trained with cases sharing similar solutions as the query solution. The novel setting (Y) is arguably harder than the novel setting (X) because a CBR system in the later setting may still retrieve a case with good enough solution. The results for 1-NN retrieval under the novel setting (Y) are not reported because the trend is already clear in the novel setting (X). Moreover, the novel setting (Y) does not apply to classification data sets because two cases can be very different but still share the same class label. Removing cases based on class label does not necessarily make the query novel.

The data sets and various settings have been used in previous studies [12, 20, 21]. This study tests both classification and regression using the same models. We chose $R\%$ as 40%. For the normal setting of simpler data sets, experimental results are averaged over one ten-fold cross validation run. However, the novel settings have very high computational costs. Even one run on the whole test set is extremely expensive, because each test case requires a re-training of all the models. We resort to randomly choosing only 50 cases from a test set for testing.

In the novel settings, the performance of any model on test cases can vary tremendously as the novelty and difficulty of the test cases vary. Consequently almost all the differences in novel settings are not statistically significant.

Experimental Results on Real Data Sets We observed the error rate of models on regression in Table 1 (lower is better) and accuracy of models on classification in Table 2 (higher is better). It is important to note that in novel settings the variance is very high (not shown in the tables) and performance differences of the models are not statistically significant. However, the general trend still reveals some interesting comparisons between the models:

1. SN retrieval is almost always better than 1-NN retrieval.
2. The relation between retrieval methods and NN-CDH is complicated:
   (a) NN-CDH consistently improves the result of 1-NN retrieval in regression, but less so in classification. We believe this is largely because (1) it is easier to retrieve a case with the same label in classification than to retrieve a case with the exact same solution in regression, and (2) nominal attributes hide subtle differences between cases (ex. a major problem difference may lead to no class change or a minor problem difference may lead to a class change) and therefore case pairs are harder for NN-CDH to learn.
   (b) NN-CDH often fails to improve the result of SN retrieval. When the retrieval process is very good but not in synchronization with the adap-

	Setting	Retrieval	Neural Network	k-NN	Retrieval	Rule CDH	NN-CDH
Airfoil	Normal	1-NN	7.42%	6.87%	6.94%	5.80%	5.71%
	Normal	Siamese	7.60%	6.97%	6.49%	17.06%	8.82%
	Novel(X)	1-NN	9.34%	16.88%	17.99%	23.85%	12.02%
	Novel(X)	Siamese	9.51%	16.20%	8.45%	18.16%	13.84%
	Novel(Y)	Siamese	8.90%	16.49%	17.18%	22.29%	13.44%
Car	Normal	1-NN	1.52%	1.95%	1.63%	1.81%	1.38%
	Normal	Siamese	1.47%	1.92%	2.19%	6.89%	1.72%
	Novel(X)	1-NN	5.00%	7.42%	8.48%	8.87%	4.31%
	Novel(X)	Siamese	5.41%	7.36%	2.76%	8.45%	3.61%
	Novel(Y)	Siamese	5.06%	7.05%	6.20%	9.32%	4.77%
Student Performance	Normal	1-NN	21.61%	25.47%	31.59%	31.03%	29.63%
	Normal	Siamese	21.38%	25.64%	26.39%	31.66%	30.62%
	Novel(X)	1-NN	16.42%	18.73%	23.30%	27.00%	24.33%
	Novel(X)	Siamese	16.32%	19.30%	21.00%	21.90%	25.82%
	Novel(Y)	Siamese	23.73%	26.17%	32.00%	28.70%	33.38%
Yacht	Normal	1-NN	7.53%	13.77%	11.48%	6.85%	8.05%
	Normal	Siamese	5.94%	13.87%	2.18%	17.11%	7.71%
	Novel(X)	1-NN	10.50%	13.15%	16.72%	26.96%	10.50%
	Novel(X)	Siamese	10.14%	12.96%	3.52%	24.25%	6.43%
	Novel(Y)	Siamese	15.64%	19.56%	13.68%	19.77%	23.22%
Energy Efficiency	Normal	1-NN	7.36%	7.53%	14.62%	15.06%	13.04%
	Normal	Siamese	7.20%	7.55%	2.17%	12.51%	4.89%
	Novel(X)	1-NN	17.96%	23.46%	25.83%	22.29%	14.88%
	Novel(X)	Siamese	17.82%	23.44%	16.40%	22.41%	13.20%
	Novel(Y)	Siamese	17.07%	24.19%	23.72%	20.74%	12.95%

Table 1. Error Rates of Models on Regression Data Sets

tation, in this case NN-CDH, the adaptation model does not necessarily improve the retrieval result. See discussion in Section 2.

3. Neural network, SN retrieval and NN-CDH all may achieve best performance in various settings of different data sets. It is unclear which model will be most suitable in a given situation but some general trends are observed:
   (a) Under normal settings, the neural network is often best performing. Under novel settings, performance of all models degrades.
   (b) Under the novel setting (X), 1-NN is much worse than in the normal setting, but siamese retrieval performs relatively well and is often best.
   (c) Under the novel setting (Y), the siamese retrieval also suffers. We observe that NN-CDH may improve (but sometimes worsen) the retrieval results.
   (d) The neural network and the NN-CDH have comparable performance. This is because the two models share the same structure and computational power, although trained with different data for different purposes. Often NN-CDH is slightly worse. This may be because NN-CDH is working on the retrieval result. If the retrieval result is bad or if the retrieval and the adaptation are not in synchronization, it is harder to adapt.
   (e) As the neural network and the NN-CDH are similar in terms of problem solving power, if the neural network is underperforming (for example, if it performs worse than the retrieval method), then NN-CDH is likely underperforming as well and likely to worsen the retrieval result.

4.4 Artificial Data Sets

Considering the nature of different models, we hypothesize that the locality of the task domain (whether local regions follow a specific pattern that is sufficiently

	Setting	Retrieval	Neural Network	k-NN	Retrieval	Rule CDH	NN-CDH
Credit	Normal	1-NN	86.06%	85.34%	85.34%	80.37%	81.10%
	Normal	Siamese	85.91%	85.06%	80.78%	75.73%	76.34%
	Novel(X)	1-NN	77.80%	58.60%	58.60%	58.80%	70.20%
	Novel(X)	Siamese	68.00%	48.00%	70.00%	52.00%	76.00%
Balance	Normal	1-NN	97.21%	79.00%	79.00%	72.21%	97.15%
	Normal	Siamese	97.12%	79.44%	98.40%	73.66%	97.28%
	Novel(X)	1-NN	84.00%	46.00%	46.00%	56.00%	70.00%
	Novel(X)	Siamese	88.00%	46.00%	90.00%	42.00%	70.00%
Car	Normal	1-NN	99.87%	86.36%	86.36%	79.21%	99.27%
	Normal	Siamese	99.71%	84.84%	97.63%	71.12%	97.05%
	Novel(X)	1-NN	82.00%	52.00%	52.00%	52.00%	84.00%
	Novel(X)	Siamese	78.00%	60.00%	82.00%	64.00%	62.00%
Yeast	Normal	1-NN	58.83%	54.65%	54.65%	50.68%	52.90%
	Normal	Siamese	58.84%	54.42%	48.18%	45.19%	49.03%
	Novel(X)	1-NN	42.00%	40.00%	40.00%	38.00%	34.00%
	Novel(X)	Siamese	40.00%	26.00%	26.00%	26.00%	30.00%
Seeds	Normal	1-NN	93.71%	92.67%	92.67%	89.33%	94.29%
	Normal	Siamese	94.76%	93.33%	94.29%	90.48%	95.71%
	Novel(X)	1-NN	46.00%	26.00%	28.00%	28.00%	48.00%
	Novel(X)	Siamese	42.00%	14.00%	44.00%	32.00%	40.00%

Table 2. Accuracy Rates of Models on Classification Data Sets

different from the global landscape) is one factor causing disparity between the performances of models. Obviously, there are many other factors of a data set that might influence a model's performance, for example, the sparsity of feature values, the dimension, and the time-spatial relationship between features.

To further study the trends and find scenarios where one model may outperform the others, we created a way to generate artificial data sets with variable locality. The artificial cases take the form of

$$\{x_1, x_2, ...x_k : y\}, 0 \leq x_i \leq 1$$

where $\{x_1, x_2, ...x_k\}$ is a problem description with k features and y is the solution. k weights $\{w_1, w_2, ...w_k\}$ and k biases $\{b_1, b_2, ...b_k\}$ are randomly sampled from 0 to 1, each corresponding to one feature. For each case, x_i are randomly sampled from 0 to 1 and y is determined by two steps: 1) Find the first integer i such that $x_1 \leq i/k$; 2) Calculate the value of y as $y = w_i * x_i + b_i$.

This data set can be converted to include nominal features and even nominal solutions. For example, the first feature x_1 can be converted to an nominal feature of k possible values. If converted this way, the data set shows a perfect example where the nominal and numerical features are independent and yet interact, as the first feature (nominal) determines which numerical feature to use in calculating the solution.

On a high level, the first feature x_1 is globally used while the other features are only locally used depending on the value of x_1. Therefore this data set demonstrates a good example of task domains involving both global and local landscapes. By tuning the number of features k, we can modify the locality of the data set. When $k = 1$, the data set follows a single global rule; When k is large, the data set follows many local rules.

We generate data sets of 1000 cases respectively using $k = 3$, 5, and 7. For each data set, we test it with all models and three settings: normal, novel (X),

and novel (Y). For each novel setting, we also vary $R\%$ to be 10%, 20%, and 30% in order to render gradual influences of the novel settings. To ensure fair comparison between models, we use the same seed to generate random cases for each choice of k and to select test queries.

Experimental Results on Artificial Data Sets We observed the performance of models in Table 3 where the CBR system uses 1-NN retrieval and in Table 4 where the CBR system uses SN retrieval. We recorded both the error rate and standard deviation of each model. Each model is tested with 50 samples under each novel setting. The standard deviation of all models is very high in novel settings. This renders the comparisons in novel setting statistically insignificant, but we still believe the average error rates provide interesting data.

	Setting	Neural Network	k-NN	1-NN Retrieval	Rule CDH	NN-CDH
3 features	normal	17.47%(5.37)	9.677%(1.55)	9.818%(2.81)	12.31%(2.42)	9.906%(3.35)
remove on X	novel 0.1	17.23%(16.4)	14.86%(15.3)	18.57%(19.6)	24.90%(27.3)	9.148%(12.0)
	novel 0.2	19.90%(15.8)	19.09%(19.9)	23.94%(21.4)	27.62%(25.1)	11.41%(13.3)
	novel 0.3	22.42%(16.0)	21.00%(17.5)	24.40%(22.3)	30.25%(26.3)	15.41%(15.7)
5 features	normal	12.41%(1.66)	16.87%(2.13)	20.68%(1.98)	25.92%(2.12)	11.82%(2.82)
remove on X	novel 0.1	11.39%(9.71)	17.76%(14.7)	19.28%(22.6)	26.61%(23.4)	9.961%(9.01)
	novel 0.2	12.68%(11.6)	22.81%(15.7)	24.98%(23.7)	31.35%(23.9)	12.61%(11.8)
	novel 0.3	13.08%(11.9)	21.13%(18.7)	24.36%(22.4)	35.70%(27.8)	12.89%(11.7)
7 features	normal	24.42%(2.69)	25.70%(1.87)	32.08%(2.09)	39.37%(3.18)	25.49%(2.58)
remove on X	novel 0.1	20.02%(13.6)	25.53%(15.8)	34.56%(22.7)	37.55%(29.3)	23.15%(14.5)
	novel 0.2	21.38%(12.2)	25.70%(19.1)	33.92%(23.3)	40.67%(26.6)	23.84%(20.0)
	novel 0.3	22.98%(12.2)	28.35%(16.1)	31.02%(20.7)	36.22%(22.6)	21.82%(15.4)
3 features	normal	17.47%(5.37)	9.677%(1.55)	9.818%(2.81)	12.31%(2.42)	9.906%(3.35)
remove on Y	novel 0.1	15.59%(17.6)	13.77%(19.8)	15.27%(22.6)	14.24%(23.7)	10.63%(17.3)
	novel 0.2	20.65%(18.0)	17.97%(22.5)	21.64%(23.8)	21.19%(25.4)	16.74%(21.8)
	novel 0.3	22.33%(19.2)	24.68%(21.8)	26.23%(23.2)	25.05%(24.9)	19.62%(22.0)
5 features	normal	12.41%(1.66)	16.87%(2.13)	20.68%(1.98)	25.92%(2.12)	11.82%(2.82)
remove on Y	novel 0.1	12.11%(12.3)	16.81%(14.9)	18.15%(19.6)	21.64%(23.1)	11.83%(11.8)
	novel 0.2	18.73%(17.3)	24.74%(18.0)	28.06%(21.5)	26.95%(20.2)	20.88%(18.5)
	novel 0.3	23.25%(20.7)	31.04%(19.4)	33.68%(19.6)	34.40%(19.2)	21.99%(19.2)
7 features	normal	24.42%(2.69)	25.70%(1.87)	32.08%(2.09)	39.37%(3.18)	25.49%(2.58)
remove on Y	novel 0.1	29.76%(14.4)	29.85%(19.4)	33.85%(22.2)	37.84%(21.4)	31.37%(20.5)
	novel 0.2	32.92%(17.9)	35.60%(19.3)	38.07%(20.4)	38.89%(22.9)	39.12%(20.7)
	novel 0.3	35.73%(15.9)	39.03%(17.8)	40.11%(18.6)	41.49%(21.0)	39.42%(19.8)

Table 3. Error rates (and standard deviation) of Models on the Artificial Regression Data Sets. CBR uses 1-NN retrieval and adaptation is based on the retrieval result.

Many of the observations mesh with those in Section 4.3. We also observed additional phenomena in the experiments with the artificial data sets:

1. To our surprise, the baseline network performs relatively well (and sometimes best) in data sets with high locality. This contradicts our hypothesis.
2. Rule-based CDH always downgrades the retrieval result. Rule-based CDH finds the best case pairs to apply using the naive 1-NN, which works poorly in this special domain where only two features matter.
3. All models perform better in low-dimension data sets than high-dimension ones. 1-NN performs well in low-dimension spaces. The NN-CDH further improves the result of 1-NN retrieval, and often achieves the best performance.

	Setting	Neural Network	k-NN	SN Retrieval	Rule CDH	NN-CDH
3 features	normal	16.93%(4.94)	9.677%(1.55)	4.338%(1.09)	15.28%(4.23)	9.087%(3.05)
remove on X	novel 0.1	16.83%(16.5)	14.86%(15.3)	8.870%(15.4)	21.35%(29.8)	8.562%(10.9)
	novel 0.2	17.36%(14.3)	19.09%(19.9)	11.36%(15.8)	32.46%(27.5)	13.35%(12.3)
	novel 0.3	22.95%(14.9)	21.00%(17.5)	16.28%(18.0)	36.49%(25.8)	19.17%(18.0)
5 features	normal	13.01%(2.44)	16.87%(2.13)	7.322%(1.72)	25.59%(2.94)	12.51%(2.68)
remove on X	novel 0.1	11.03%(10.6)	17.76%(14.7)	6.017%(10.1)	28.25%(22.7)	9.115%(8.58)
	novel 0.2	11.38%(11.3)	22.81%(15.7)	6.954%(9.07)	25.10%(21.2)	12.42%(10.4)
	novel 0.3	12.72%(10.9)	21.13%(18.7)	9.769%(13.1)	22.93%(21.4)	12.48%(11.0)
7 features	normal	21.16%(2.38)	25.70%(1.87)	17.92%(4.98)	36.13%(3.47)	23.44%(2.90)
remove on X	novel 0.1	21.60%(13.3)	25.53%(15.8)	19.76%(19.8)	32.57%(25.6)	24.73%(19.0)
	novel 0.2	22.37%(13.0)	25.70%(19.1)	22.13%(20.3)	40.46%(33.8)	25.75%(17.9)
	novel 0.3	25.39%(10.8)	28.35%(16.1)	23.00%(20.6)	38.59%(27.1)	27.47%(18.3)
3 features	normal	16.93%(4.94)	9.677%(1.55)	4.338%(1.09)	15.28%(4.23)	9.087%(3.05)
remove on Y	novel 0.1	15.02%(18.3)	13.77%(19.8)	10.46%(13.9)	13.31%(16.6)	9.129%(11.6)
	novel 0.2	22.83%(19.4)	17.97%(22.5)	18.77%(21.5)	25.64%(24.3)	16.33%(24.6)
	novel 0.3	21.29%(20.2)	24.68%(21.8)	22.06%(18.3)	28.12%(25.8)	21.04%(23.8)
5 features	normal	13.01%(2.44)	16.87%(2.13)	7.322%(1.72)	25.59%(2.94)	12.51%(2.68)
remove on Y	novel 0.1	10.88%(10.4)	16.81%(14.9)	6.869%(8.68)	21.01%(20.2)	9.752%(9.32)
	novel 0.2	19.78%(17.1)	24.74%(18.0)	17.66%(19.0)	33.89%(26.6)	20.01%(16.9)
	novel 0.3	22.53%(20.2)	31.04%(19.4)	23.35%(20.4)	34.45%(24.1)	24.40%(22.3)
7 features	normal	21.16%(2.38)	25.70%(1.87)	17.92%(4.98)	36.13%(3.47)	23.44%(2.90)
remove on Y	novel 0.1	29.28%(15.8)	29.85%(19.4)	27.28%(22.3)	35.69%(22.4)	33.14%(21.8)
	novel 0.2	34.26%(17.3)	35.60%(19.3)	36.82%(21.0)	46.45%(27.8)	40.74%(22.9)
	novel 0.3	36.16%(15.5)	39.03%(17.8)	40.23%(20.4)	41.37%(22.8)	41.06%(20.7)

Table 4. Error rates (and standard deviation) of Models on the Artificial Regression Data Sets. CBR uses SN retrieval and adaptation is based on the retrieval result.

However, as the dimensionality increases, only two features are relevant and 1-NN retrieves worse results, making adaptation harder for NN-CDH. We observe when $k = 7$, the neural network outperforms 1-NN retrieval and NN-CDH consistently. This trend is not as obvious for SN retrieval.

4. SN retrieval can consistently outperform other models, except under novel settings (Y). Under a novel setting (Y), even the best possible neighbor's solution will be different from the target solution, as all cases with close solutions are removed. In such situations, we see the neural network and NN-CDH can outperform SN retrieval.
5. The artificial data set can be perfectly solved if a symbolic system somehow learns the rule to 1) find the first integer i such that $x_1 \leq i/k$; and 2) calculate the value of y as $y = w_i * x_i + b_i$. However, none of the systems tested achieve similar proficiency.

5 Conclusion

This paper proposed a way to handle nominal differences in network-based adaptation and extended NN-CDH to case adaptation for both classification and regression domains. Experiments with both real and artificial data sets align with results from previous studies of NN-CDH. In general, NN-CDH achieves comparable performance to its counterpart, the traditional neural network. Moreover, SN retrieval may outperform both the neural network and NN-CDH, and NN-CDH can worsen the result from SN retrieval. This illustrates the need to harmonize retrieval and adaptation methods [11].

In extensive experimental results, integrating neural network methods into CBR did not give CBR power beyond the neural network, even on data designed to test a hypothesized advantage for CBR. There are more sample pairs to train NN-CDH than samples to train a baseline neural network, which might provide an advantage [19] but we do not observe that here. On the other hand, CBR generally achieves performance comparable to the network, making it a competitive option, especially in tasks for which the intrinsic interpretability of CBR would make it preferable to a network approach.

The artificial data set, which can be perfectly solved using simple rules, illustrates a situation in which neither the neural network or knowledge-light CBR learns the underlying abstraction well. Symbolic CBR provides the opportunity to integrate knowledge and task-optimized representations. We envision that a balanced blend of domain knowledge, brought to bear by symbolic AI methods, with network-learned methods, has the potential to achieve a performance edge. We consider the study of such integration to be a key step in advancing the performance of case adaptation [10].

6 Acknowledgments

This work was funded by the the Department of the Navy, Office of Naval Research (Award N00014-19-1-2655). We thank the members of the Indiana University Deep CBR group for valuable discussions.

References

1. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: Proceedings of the 6th International Conference on Neural Information Processing Systems. pp. 737–744. NIPS'93, Morgan Kaufmann, San Francisco (1993)
2. Corchado, J., Lees, B.: Adaptation of cases for case based forecasting with neural network support. In: Soft Computing in Case Based Reasoning, pp. 293–319. Springer, Berlin (2001)
3. Craw, S., Wiratunga, N., Rowe, R.: Learning adaptation knowledge to improve case-based reasoning. Artificial Intelligence 170, 1175–1192 (2006)
4. F. Zhang, M. Ha, X. Wang, X. Li: Case adaptation using estimators of neural network. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826). vol. 4, pp. 2162–2166 vol.4 (2004)
5. Hanney, K., Keane, M.: Learning adaptation rules from a case-base. In: Proceedings of the Third European Workshop on Case-Based Reasoning. pp. 179–192. Springer, Berlin (1996)
6. Jalali, V., Leake, D., Forouzandehmehr, N.: Learning and applying case adaptation rules for classification: An ensemble approach. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 4874–4878 (2017)
7. Jalali, V., Leake, D.: Extending case adaptation with automatically-generated ensembles of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2013. pp. 188–202. Springer, Berlin (2013)

8. Jarmulak, J., Craw, S., Rowe, R.: Using case-base data to learn adaptation knowledge for design. In: Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2. pp. 1011–1016. IJCAI-01, Morgan Kaufmann, San Francisco (2001)
9. Leake, D., Kinley, A., Wilson, D.: Learning to integrate multiple knowledge sources for case-based reasoning. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. pp. 246–251. Morgan Kaufmann (1997)
10. Leake, D., Crandall, D.: On bringing case-based reasoning methodology to deep learning. In: Case-Based Reasoning Research and Development, ICCBR-20. pp. 343–348. Springer, Cham (2020)
11. Leake, D., Ye, X.: Harmonizing case retrieval and adaptation with alternating optimization. In: Case-Based Reasoning Research and Development, ICCBR 2019. pp. 125–139. Springer (2021)
12. Leake, D., Ye, X., Crandall, D.: Supporting case-based reasoning with neural networks: An illustration for case adaptation. In: Proceedings of AAAI Spring Symposium AAAI-MAKE 2021: Combining Machine Learning and Knowledge Engineering (2021), https://www.aaai-make.info/program
13. Liao, C., Liu, A., Chao, Y.: A machine learning approach to case adaptation. In: 2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). pp. 106–109 (2018)
14. Martin, K., Wiratunga, N., Sani, S., Massie, S., Clos, J.: A convolutional siamese network for developing similarity knowledge in the SelfBACK dataset. In: Proceedings of the ICCBR 2017 Workshops, Doctoral Consortium, and Competitions. pp. 85–94. CEUR Workshop Proceedings (2017), `http://hdl.handle.net/10059/2490`
15. Mathisen, B.M., Aamodt, A., Bach, K., Langseth, H.: Learning similarity measures from data. Progress in Artificial Intelligence pp. 129–143 (10 2019)
16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
17. Policastro, C.A., Carvalho, A.C., Delbem, A.C.: Automatic knowledge learning and case adaptation with a hybrid committee approach. Journal of Applied Logic 4(1), 26–38 (2006)
18. Smyth, B., Keane, M.: Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. Artificial Intelligence 102(2), 249–293 (1998)
19. Ye, X., Leake, D., Huibregtse, W., Dalkilic, M.: Applying class-to-class siamese networks to explain classifications with supportive and contrastive cases. In: Case-based reasoning research and development, ICCBR-20. pp. 245–260. Springer (2020)
20. Ye, X., Leake, D., Jalali, V., Crandall, D.J.: Learning adaptations for case-based classification: A neural network approach. In: Case-based reasoning research and development, ICCBR-21. pp. 279–293. Springer (2021)
21. Ye, X., Zhao, Z., Leake, D., Wang, X., Crandall, D.J.: Applying the case difference heuristic to learn adaptations from deep network features. CoRR abs/2107.07095 (2021), `https://arxiv.org/abs/2107.07095`