Do CDH Network Models Perform CBR Adaptation? An Audit

Caleb Kisby

February 20, 2021

Abstract

Crafting quality CBR adaptation rules can be challenging. Recent work proposes leveraging neural networks to learn adaptation rules, using the case difference heuristic. But this presents an issue: Are these network models performing CBR adaptation per se, or are they learning different strategies? In this paper, we formalize falsifiable and testable criteria that differentiate adaptation rules from the alternatives. We demonstrate the use of these criteria via a knowledge audit of various CDH network models.

Introduction

Case adaptation is a crucial component of case-based reasoning systems. But acquiring case adaptation rules comes with the classic frustrations of knowledge engineering. A major problem for CBR systems is how to automate the creation of these adaptation rules.

Many systems have been proposed to *learn* adaptation knowledge by observing successful adaptations. By far the most widespread approach is the case difference heuristic (CDH) (Hanney and Keane 1997). According to this approach, adaptation may be viewed as determining the difference in the retrieved and problem solutions, given the differences in their attributes. We can then learn adaptation knowledge by adjusting this solution difference in accordance to observed case differences.

Recently, (Liao, Liu, and Chao 2018) has naturally suggested to use neural network models to learn these case differences. (This was done prior to 2018, but not explicitly using CDH. See (Policastro, Carvalho, and Delbem 2003) and (Main, Dillon, and Witten 2007).) Leveraging the power of neural networks has been very successful, and this approach has since been applied to a variety of extensions on CDH (see (Ye et al. 2021b), (Ye et al. 2021a), and (Leake, Ye, and Crandall 2021).

But neural network models have a key drawback: Their internal behavior is not readily interpretable. In particular, it is not clear whether the knowledge a CDH network model is learning can be considered case adaptation. For example, the network may just be reasoning from scratch from a single case, discarding the case *difference* information. This is the problem I aim to address; in this paper, I will conduct a knowledge audit on CDH network models and assess whether they are learning case adaptation rules.

One solution that immediately comes to mind is *rule extraction*. For regular neural networks, it is possible to extract logical rules that are sound (the network provably believes them) and complete (they are all of the rules the network believes). See (Garcez, Broda, and Gabbay 2001) for a detailed account of this. One problem with using rule extraction for our audit is that current state-of-the-art rule extraction can deal with propositional and conditional reasoning, whereas real domains often use predicate reasoning. Additionally, we still need significant human effort in order to *interpret* the extracted rules. In other words, we would still need to solve the hard problem of determining whether the extracted rules are really case adaptation rules.

Instead, my approach is to pinpoint criteria that distinguish case adaptation from alternative forms of reasoning. I will then formulate these criteria as easily testable statistical hypotheses, which will allow us to say (with a certain degree of confidence) whether our CDH network model is indistinguishable from the alternatives. These tests require very little human interpretation, and can be applied to reasoning in any domain.

Criteria for Case Adaptation

We first need to identify criteria that characterize case adaptation. In particular, we need to be able to differentiate between rules exhibiting case adaptation and rules exhibiting alternative forms of reasoning.

Consider a CBR agent faced with problem case B. It retrieves A, the closest case in its case-base, with solution sol_A. The case adaptation component step of CBR is now tasked with adapting (A, sol_A) towards B in order to produce a potential solution sol_B to the problem case. In principle, there are many different ways to acheive this. For example, we could simply use the retrieved sol_A as the proposed solution to B. This is what is known as the *trivial adaptation*; we reason exclusively using the case (A, sol_A), discarding all information about B. Alternatively, we could scrap the retrieved case and simply produce a solution for B using external domain knowledge. I will call this *reasoning from scratch*. These two cases form the extremes that we need to guard against — neither can be considered "adaptation" in any meaningful way, since they do not leverage both the retrieved case and the problem case.

The above intuition can be formulated using the following criteria for case adaptation. This list aims to exclude trivial adaptation and reasoning from scratch, but is otherwise inclusive to a variety of adaptation rules. We say that an agent's reasoning is *case adaptation* if:

1. The agent takes both a problem case B and a retrieved case (A, sol_A) as input

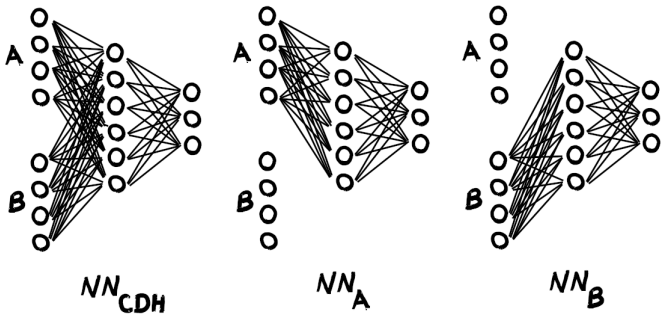

Figure 1: A CDH network model NN_{CDH}, ablated tby dropping the input B (NN_A), and again by dropping the input A (NN_B)

2. Typically, the agent's decisions would differ if it did not have both A and B available
3. Typically, the agent's internal state would differ if it did not have both A and B available

Another way to put this: The agent meaningfully combines both A and B (1) in its *input*, (2) in determining *output*, and (3) in its *internal state*.

It may superficially seem that criteria (2) and (3) are redundant, but it is necessary to separate them. For example, say we have an agent that considers both A and B, but then determines that it should defer to a from-scratch reasoner anyway. This behaviorally looks like a from-scratch reasoner, but internally is doing real case adaptation. We might also have (3) without (2), as in the case of an agent that reasons from scratch but changes its mind last-minute (due to some confidence estimate, say) and uses the trivial adaptation after all.

Note also the use of the word "typically" in (2) and (3). I include this word to emphasize that we do allow the agent to reason from scratch or use trivial adaptation on occasion — but for the majority of decisions we expect the agent to rely on significant case adaptation.

Audit Design

Of the case adaptation criteria above, (1) is easily ensured by just feeding both the retrieved and problem cases as input to our CDH network model. (We should also verify that the input weights are not zero, since we need the network to make full use of the inputs). But it's unclear how we should go about testing (2) and (3). Specifically, the requirement that the output and internal state *meaningfully combine* A and B (or *have both A and B available*) is vague. The main hurtle in our audit design to formulate (2) and (3) in a precise, falsifiable way.

Suppose we have a CBR agent that makes use of a trained neural network with a single hidden layer, NN_{CDH}, for its adaptation function (Figure 1). NN_{CDH} takes as input the problem case B, as well as the retrieved case A and its solution sol_A, and produces output sol_B. We can ablate this network by dropping the weights of some of its inputs, resulting in a network with more restricted reasoning. Let NN_A be the result of dropping weights for input B, and let NN_B be the result of dropping weights for input (A, sol_A). NN_A represents an agent that does trivial adaptation (it can only use information from (A, sol_A)), whereas NN_B represents an agent that reasons from scratch (it can only use information from B).

Rather than asking whether NN_{CDH} makes use of A and B in a meaningful way, we can ask a more directed question: Is NN_{CDH} distinguishable from NN_A and NN_B? At risk of sounding repetitive, this can be broken down into four points:

- Is NN_{CDH}'s output distinguishable from NN_A's output?
- Is NN_{CDH}'s output distinguishable from NN_B's output?
- Is NN_{CDH}'s internal state distinguishable from NN_A's internal state?
- Is NN_{CDH}'s internal state distinguishable from NN_B's internal state?

Each of these experimental questions are precise and falsifiable, and hence testable. For example, to test the first one for a given NN_{CDH}, we determine whether we can reject (with a certain degree of confidence) the null hypothesis

NN_{CDH}'s output is indistinguishable from NN_A's output; the mean difference between the two is 0.

For determining the mean difference between *internal states*, we check the number of neurons in the single hidden layer of the network.

Assessment

I will now try to demonstrate that this audit gives reasonable verdicts. In this assessment I will consider three generated scenarios, each with varying differences between NN_{CDH}, NN_A, and NN_{CDH}. The goal is to show that our audit correctly classifies the reasoning of NN_{CDH} in each scenario.

The Balance-Scale Domain

For testing, I used the Balance-Scale dataset (Shultz, Mareschal, and Schmidt 1994) from the UCI Machine Learn-

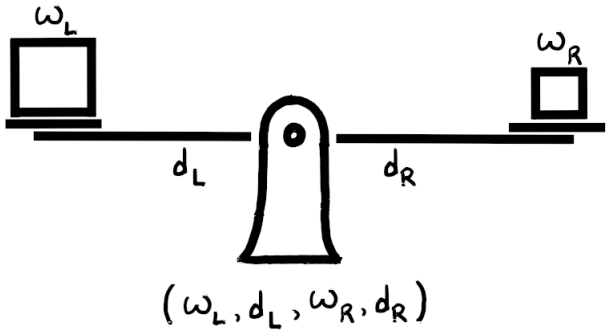

Figure 2: In the Balance-Scale task, our agent takes a left object's weight w_L and distance from center d_L and a right object's weight w_R and distance from center d_R. The task is to determine if the scale will tip 'Left', 'Right', or remain 'Balanced'.

ing Repository (Dua and Graff 2017). The task is to determine which direction a scale will tip ('Left', 'Right', or 'Balanced'), given a left object's weight w_L and distance from center d_L and a right object's weight w_R and distance from center d_R (see Figure 2). The possible weights and distances are positive integers between 1 and 5. So there are $5^4 = 625$ possible input vectors.

My primary reason for using this particular task is that it is amenable to all three of our alternative forms of reasoning. First, Balance-Scale obeys the CBR tenant that similar problems have similar solutions, which suggests that an agent using trivial adaptation could be suitable. Balance-Scale instances can also be reasoned from scratch, by checking which of the following is greater: $d_L \times w_L$, or $d_R \times w_R$. Finally, we can solve Balance-Scale using case adaptation — especially using case differences. For example, given a high enough Δ(left weight) between our retrieved and problem cases, we might decide to predict 'Left' or 'Balanced' rather than 'Balanced' or 'Right'. Because all three of these forms of reasoning are good for the task, a competent neural network will have a fair chance of learning any of these three strategies.

Another advantage of this dataset is that it is *complete* — we have access to every possible input! So our training set could potentially be a majority of all possible cases. This makes it easy to produce different scenarios by tuning the relative competence of the CBR's retrieval and adaptation. If, for instance, we want weak retrieval but strong adaptation, we can give the CBR very few cases while training NN_{CDH} on enough cases to become very competent.

System Architecture

As in our audit design, our testbed system is a CBR agent that uses the neural network NN_{CDH} to compute the result of its case adaptations. The CBR retrieves cases using k-nearest neighbors with uniform attribute weights. As for adaptation, NN_{CDH} is a simple network with a single hidden layer. It has 9 input nodes (4 for the retrieved case A, 1 for sol_A, and 4 for the problem case B). The hidden layer has dimension 6. The output has three nodes, one for each classification ('Left', 'Balanced', or 'Right').

Since NN_{CDH} is performing classification, its hidden and output neurons use softmax activation. We use categorical crossentropy for loss. Other parameters for the net (e.g. epochs, batchsize) can be found in the source code.

In each scenario, the weights of NN_{CDH} were trained on problem instances (B, sol_B) and their closest cases (A, sol_A). As described in the audit design, we then ablate NN_{CDH}'s input weights to create NN_A and NN_B. Note that we do not re-train these nets' weights, since we want to compare NN_{CDH}'s reasoning against *its own* reasoning without input A or B.

Results and Discussion

To demonstrate the effectiveness of the audit, I consider three randomly-generated scenarios (Table 1). Each of these scenarios reflect different degrees of similarity between NN_A, NN_B, and NN_{CDH}. The intention is that these scenarios exercise the unique considerations that go into deciding whether NN_{CDH} is performing case adaptation.

The first four columns of Figure 1 display the accuracy of the CBR system with varying adaptation strategies. Here, "No Adaptation" denotes that the CBR used no adaptation at all. This is different from using NN_A (trivial adaptation) for adaptation, since the remaining weights of NN_A still alter the final prediction. Notice that in all three cases, NN_A performs significantly worse than NN_{CDH}. So we can already conclude that NN_{CDH} is not using the trivial adaptation.

Differentiating between NN_{CDH} and NN_B is more challenging — especially given how close their accuracy is in each of these scenarios. Here we need to rely on our experimental questions. Specifically, in each case we ask whether we can distinguish between NN_{CDH}'s output internal state and those of NN_B. For each case, we compute the mean difference between NN_{CDH} and NN_B, in both their outputs and their internal (hidden) states. (We also show the mean difference between NN_{CDH} and NN_A for comparison.) 95%-confidence intervals for each mean are also shown.

Consider the first scenario. The mean difference between NN_{CDH}'s output and NN_B's output strictly excludes 0. This is also true of the mean difference between the activated nodes in their hidden states. So we can (with 95% confidence) reject the null hypothesis that there is no difference, and conclude that NN_{CDH} is not reasoning from scratch. Since it is also not using trivial adaptation, our verdict is that this network has successfully learned substantial case adaptation!

In the second scenario, both the mean difference in output and the mean difference in the hidden layers contain 0. Here we cannot reject either null hypothesis, and conclude that NN_{CDH} may be reasoning from scratch.

The third scenario is the most interesting of the three. If we examine the mean difference in the output, we see that this interval contains 0. So we cannot reject the null hypothesis that there is no difference in the *outputs*. But their hidden layers *do* differ — 12.9 ± 8.6 does not contain 0! So we can conclude that NN_{CDH} does not merely reason from scratch after all!

In fact, the agent in this third scenario is likely the agent mentioned above: The agent that uses both cases to *defer* to a from-scratch reasoner. This is a legitimate case adaptation strategy, since the entire decision makes full use of both retrieved and problem case. And this illustrates a key point about this sort of audit; namely, we cannot determine an agent's reasoning by treating it *purely* as a black-box system. We *must* be able to peer inside in order to make crucial distinctions such as the one in this last scenario.

Conclusion and Future Work

In this report, we have characterized case adaptation, and discussed the differences between adaptation, trivial adaptation, and reasoning from scratch. Furthermore, we have formalized criteria for case adaptation in a falsifiable and easily testable form. We then trained various neural networks on a domain that is amenable to all three alternative forms of reasoning, and demonstrated how we can use our criteria to determine which kind of reasoning each neural network

Table 1:

Accuracy				Mean Difference (Output)		Mean Difference (Hidden)	
No Adaptation	**NN$_A$**	**NN$_B$**	**NN$_{CDH}$**	**NN$_A$**	**NN$_B$**	**NN$_A$**	**NN$_B$**
74%	2%	92%	94%	98.4 ± 3.2	6.5 ± 6.3	188.7 ± 8.1	6.5 ± 6.3
82%	61%	92%	92%	37.1 ± 12.4	3.2 ± 4.5	93.6 ± 11.1	4.8 ± 5.5
82%	10%	94%	94%	93.6 ± 6.3	3.2 ± 4.5	177.4 ± 10.7	12.9 ± 8.6

is using. This knowledge audit was completely automated, and can in principle be adapted to any dataset and any CDH network model.

In the introduction, I mentioned that rule extraction is a natural approach when we want to peer into a neural network's reasoning. But even if we have rules characterizing the network, this still begs the question of how we can determine that *these* rules are performing case adaptation. I believe that this analysis would be similar to the one given here. That is, if we define rule systems R_A and R_B that capture trivial adaptation and reasoning from scratch, respectively, then we can compare their internal and external conclusions against the rules in question.

This comparison of rule-based systems would allow us to make more nuanced conclusions than we can about a neural network's activations. (In light of it, the tests in this report would seem relatively shallow.) But building the rule systems and defining similarity between their internal states is beyond the scope of this project.

One thing I can do next is run these ideas by Xiaomeng Ye. He's given this problem a good deal more thought than I have, and I'd like to see what other criteria he thinks are necessary to differentiate case adaptation from alternatives. And if he has already set up a system for this, it would be fun to combine our tests.

References

[Dua and Graff 2017] Dua, D., and Graff, C. 2017. UCI machine learning repository.

[Garcez, Broda, and Gabbay 2001] Garcez, A. d.; Broda, K.; and Gabbay, D. M. 2001. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence* 125(1-2):155–207.

[Hanney and Keane 1997] Hanney, K., and Keane, M. T. 1997. The adaptation knowledge bottleneck: How to ease it by learning from cases. In *International Conference on Case-Based Reasoning*, 359–370. Springer.

[Leake, Ye, and Crandall 2021] Leake, D.; Ye, X.; and Crandall, D. J. 2021. Supporting case-based reasoning with neural networks: An illustration for case adaptation. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

[Liao, Liu, and Chao 2018] Liao, C.-K.; Liu, A.; and Chao, Y.-S. 2018. A machine learning approach to case adaptation. In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 106–109. IEEE.

[Main, Dillon, and Witten 2007] Main, J.; Dillon, T. S.; and Witten, M. 2007. Adaptation knowledge from the case base. In *Australasian Joint Conference on Artificial Intelligence*, 579–588. Springer.

[Policastro, Carvalho, and Delbem 2003] Policastro, C. A.; Carvalho, A. C.; and Delbem, A. C. 2003. Hybrid approaches for case retrieval and adaptation. In *Annual Conference on Artificial Intelligence*, 297–311. Springer.

[Shultz, Mareschal, and Schmidt 1994] Shultz, T. R.; Mareschal, D.; and Schmidt, W. C. 1994. Modeling cognitive development on balance scale phenomena. *Machine learning* 16(1):57–86.

[Ye et al. 2021a] Ye, X.; Leake, D.; Jalali, V.; and Crandall, D. 2021a. Learning adaptations for case-based classification: A neural network approach.

[Ye et al. 2021b] Ye, X.; Zhao, Z.; Leake, D.; Wang, X.; and Crandall, D. 2021b. Applying the case difference heuristic to learn adaptations from deep network features. *arXiv preprint arXiv:2107.07095*.