Summary of:
Case-Based Learning Algorithms

By David W. Aha
What is CBL (Case-Based learning)

- A subset of CBR (Case based reasoning)
  - Focus on learning issues
  - Limited to feature-value case representations
- CBL systems are well suited for processing a set of training cases and use them to predict values for subsequently present cases
- CBL has been used in: Clinical audiology, word pronunciation, predicting power load levels, appraising oil prospecting sites, pole balancing, speech recognition, robot control tasks...
- Example of CBL algorithms are Protos\(^1\) and MBRtalk\(^2\)

\(^1\) Bareiss, 1989a; 1989b 
\(^2\) Stanfill, 1987; Stanfill & Waltz, 1988
How is CBL restricted?

- A paper from 1984 examining a simple CBL algorithm revealed several deficiencies
  - Computationally expensive because they save and compute similarities to all training cases
  - Intolerant of noise
  - Intolerant of irrelevant features
  - Sensitive to the choice of the algorithm’s similarity function
  - No simple way to process symbolic-valued features
  - Give little usable information regarding the structure of the data
- The paper did not investigate potential solutions to these problems
Framework for Case-Based Learning Algorithms

1. Pre-processor: Prepares the input for processing (e.g. normalizing range, raw input -> cases)
2. Similarity: Function assess similarities of a given case with the previously stored cases
3. Prediction: Inputs similarity assessments and generates prediction for the value of the given case’s goal feature
4. Memory Updating: Modifying or abstracting previously stored cases, forgetting cases presumed to be noisy, or updating a feature’s relevance weight setting.
CBL1

- Pre-processor linearly normalizes all numeric feature values
- Easy similarity function

\[
\text{Similarity}(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{i \in P} \text{Feature_dissimilarity}(C_{1i}, C_{2i})}}
\]

\[
\text{Feature_dissimilarity}(C_{1i}, C_{2i}) = \begin{cases} 
(C_{1i} - C_{2i})^2 & \text{if feature } i \text{'s values are numeric} \\
0 & \text{if } C_{1i} = C_{2i} \\
1 & \text{otherwise}
\end{cases}
\]
CBL1

- Prediction function is the k-nearest neighbour function
- Memory updating function simply stores all training cases in its concept description
- Training is trivial; just stores normalized cases
CBL2

- An extended version of the CBL1 algorithm
- Purpose: Reduce storage requirements
- Achieved by only storing cases which are *incorrectly* classified
- Storage requirement is around 30-40% of CBL1 for most domains
- Classification accuracy is slightly lower (down ~5%) than CBL1 for all domains
CBL3

• An extension of the CBL2 algorithm
• Purpose: Tolerate noisy training cases
• Such cases can be recognised: they misclassify future test cases
• CBL3 seeks to prevent noisy cases to participate in the classification of future test cases
CBL3

• This is achieved in this way: the frequency of which stored cases (when used to classify the current case) correctly classify the current case

• A statistical significance test is performed to ensure that stored cases with significantly high frequencies will participate in the classification
CBL3

- CBL3 outperforms CBL2 in every test domain (both classification accuracy and storage requirements), and its accuracy is comparable to CBL1
- However, CBL3 loses to C4 (decision tree learning) in domains containing many irrelevant features
Case based learning

1 Unresolved Issues

2 learn (more) domain specific information
possible solution: automated solutions for constructing new features

3 more elaborate case representation
example: legal reasoning

4 Where on the compute-encode scale?
How to determine what is the correct balance between encoding and computing for a case?

5 Summary

6 Most previous case studies
...were for only one application

7 This paper
...shows that it is possible to define simple and general applicable CBL that
• reduced storage requirements
• noise tolerance
• irrelevant features tolerance