Appendix A

SPECIFYING EXAMPLES AND RULES

The rules-to-network translator (Section 2.2) requires two sets of information from a user to describe each problem. The first set of information is a specification of the features that are used to describe the examples. The second set of information is a set of rules that encode the knowledge of the problem which is supplied to the system. The following two sections define the syntax of each of these information types, and how they are handled by KBANN's rules-to-network translator.

A.1 Information about Features

The features used to describe an example take the general form:

`(feature_name feature_type feature_values)`

where:

`feature_name` is simply the name for a feature (e.g., size, color, etc),

`feature_type` is one of the following types in common usage in machine learning:

nominal	from among a named list (e.g., red, green, blue)
boolean	values must be either T or F
hierarchical	values exist in an ISA hierarchy
linear	values are numeric and continuous
ordered	values are non-numeric but have a total ordering

`feature_values` is a specification of the allowable values of a feature. Each type of feature has a different format in which values are expected.

The following subsections supply four pieces of information about each feature type:

1. a brief definition,
2. an example of how the feature type can be used,
3. an example of how the feature type is written for presentation to KBANN,

4. a description of how the feature is encoded by KBANN.

Missing values are handled in a consistent manner across the feature types. Namely, when the value of a feature is not known, all of the units used to encode it are given an equal activation. In addition, the sum of the activations of the units is set so that the total activation of the units is equal to the total activation of the units had the value been known. In this way, the network cannot learn to distinguish missing values merely on the basis of activation. Empirical tests [Shavlik91] have shown that this method of encoding missing values results in better generalization than other methods (such as setting all activations to zero.)

A.1.1 Nominal

This is the simplest type of feature. All the possible values of the feature are named. For the feature `color` of a coffee cup might be given by:

```
    (color nominal (red blue green))
```

KBANN would translate this feature into three units. The sole criteria for the activation of nominal features is that the summed activation of the units corresponding to a feature is equal to one. Hence, when there is information about a nominal feature, one of the units will have an activation of one and the others will have an activation of zero. When there is no information, each unit in normally given an activation equal to $\frac{1}{N}$ where N is the number of values of that feature. Alternately, the activations of each unit could be set to the prior probability of the feature having the value.

A.1.2 Binary

Binary-valued features are a special subclass of nominal features that have values of only two values (e.g., T, F). To reduce the number of input units, binary-valued features are given only a single input unit. The following is a binary feature:

```
    (temperature-is-hot binary (T F))
```

When there is no information available about a binary feature, its activation is 0.5. Note that this is a departure from the consistent total activation approach to handling missing variables.

A.1.3 Hierarchical

Hierarchical features are features defined in the context of an *ISA* hierarchy. For example, the following creates a feature which uses seven units to encode the information appearing in Figure A.1:

```
    (material hierarchy (material (insulating (styrofoam)
                                              (open-cell-foam))
                                  (non-insul  (paper)
                                              (ceramic))))
```

Hierarchical features act much like nominal features in that exactly one of the base-level units can be active. When no information is available concerning a hierarchical feature, then the activation of units at each level in the hierarchy is the reciprocal of the number of units in that level. Hence, in the above example, `styrofoam`, `open-cell-foam`, `paper` and `ceramic`

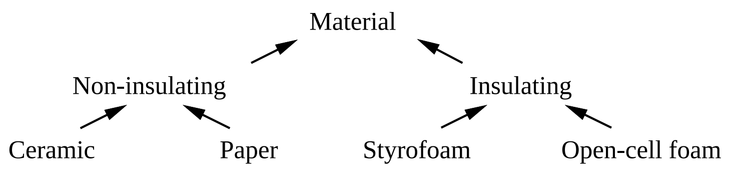

Figure A.1: A hierarchy of cup materials.

Table A.1: Equations used to encode a linear feature.

$$z_i = \frac{1}{1 + exp(\frac{abs(v-[upB-lowB]/2)}{upB-lowB})} \tag{A.1}$$

$$a_i = \frac{(z_i)^2}{\sum_j (z_j)^2} \tag{A.2}$$

where	upB	is the top of the subrange
	$lowB$	is the bopttom of the subrange
	v	is the exact value of the feature
	z_i	is an intermediate stage in the calculation of the activation
	a_i	is the activation of the unit
	i,j	are indices ranging over the units used to encode a feature

would all have activations of $\frac{1}{4}$; `insulating` and `non-insul` would have activations of $\frac{1}{2}$, and `material` would have an activation of 1.

A.1.4 Linear

Linear features have numeric values. Normally these values are continuous, but any feature with numeric values may be described using an linear feature. For example, the following feature describes the price of a cup in terms of three subranges.

```
    (price linear ((low-price 0 3)
                   (med-price 3 7)
                   (high-price 7 10)))
```

KBANN uses one unit to encode subrange. The activation of units is determined by Equations A.1 and A.2. Figure A.2 plots the activation of each feature over the whole range for `price`. So, if the value of the `price` feature is 6, then `low-price` would have and activation of 0.10, `med-price` would have an activation of 0.61 and `high-price` would have an activation of 0.29. When there is no information about a linear feature, each unit is activated as if the value was just outside of the range covered by the feature.

This encoding scheme allow users to write rules that refer to ranges rather than specific prices. Hence, this encoding creates a convenient shorthand for writing rules. However, the

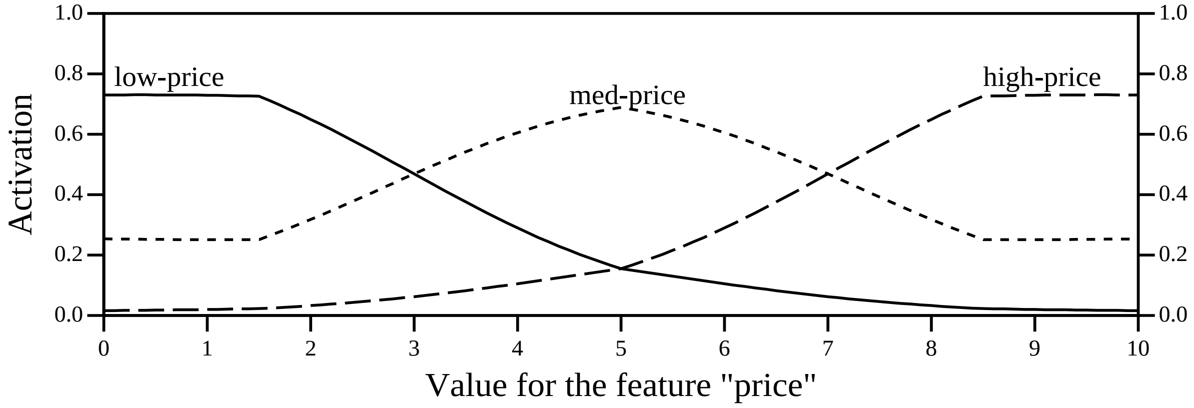

Figure A.2: The activation of units encoding a linear-valued feature.

user-defined ranges are unlikely to perfectly fit the data. As a result, schemes that just assign activations of zero or one to units for a linear features may not work. For instance, if the value of a feature indicates that the price is high, it may still be correct to consider the price as medium. However, it is rather unlikely that the price should really be considered low.

Thus, the activation scheme described above activates the inputs units according to how closely the value of the feature agrees with the range that each unit encodes. The idea behind this encoding scheme is that the ranges are not precise and exclusive. That is, because something has a high price does not mean that the price could not also be considered medium. However, when some thing has a price in the high range, it is rather unlikely that the price should really be considered low. Note that this approach to handling numeric features is very similar to "coarse coding", a common approach to handling real-values inputs in neural networks [Hinton86].

As with nominal features, when no information is available about a linear feature, each unit take a value of $\frac{1}{N}$, where N is the number of units used to encode the feature.

A.1.5 Ordered

Ordered features are a special type of nominal features for which the values are totally ordered. For example, `size` could be represented using the totally ordered set { `tiny small medium large huge` }. The specification for this feature would be:

```
(size ordered (tiny small medium large huge))
```

Like nominal features, ordered features are handled by creating one input unit for each possible value. However, ordered features cannot be treated like simple nominal features because the boundaries between subsequent values in the ordering are typically indistinct. Therefore, when an object is described as having `medium` size, it might be equally correctly be described as being either `large` or `small`. To account for the fuzziness [Zadeh83] inherent in ordered features, all values of an ordered feature are activated according to their distance from the given value using the formula given in Equation A.3.

$$activation = \frac{1}{2}^{distance\_from\_given\_value} \tag{A.3}$$

Using this equation, when an object is described as `small`, the unit corresponding to small would have an activation of 1.0, while the units corresponding to `tiny` and `medium` would have activations of 0.5. The activations of `large` and `huge` would be 0.25 and 0.125, respectively. When information about an ordered feature is missing, activations are set exactly as for a nominal feature.

A.2 Information about Rules

This section describes the syntax of rules that KBANN is currently able to translate into a neural network and specifies how KBANN translates them. The general form of a rule is:

```
    (indicator consequent antecedent1 antecedent2 ...)
```

Consequents and antecedents may take either of the two following forms.

```
    (feature value)        or        name
```

In the antecedents of a rule, the first form is normally used for reference to input units while the second form is used for reference to an intermediate conclusion.

Except as described in Sections A.2.2 and A.2.3 a consequent is true when all of the antecedents are true. That is, individual rules are purely conjunctive. Disjuncts are must be encoded as multiple rules with the same consequent. Thus, the rule which says that a consequent `A` is true if `F` is true and either `G`, or `H` or `J` is true is written for use in KBANN as the following three rules:

(<- A F G)
(<- A F H)
(<- A F J)

The following two sections describe options in the "rule indicator" and antecedents. A final part of this section describes the use of variables in the writing of rules. Variables are used only as a shorthand notation, they are not used in the network.

A.2.1 Rule types

There are two kinds of rule types in KBANN:

<= A "definitional" rule.
Definitional rules are rules that are not to be changed during the training phase. (For instance, a definitional rule might be used for "blots" in backgammon.)As a result, the consequents of fixed rules are connected only to those antecedents mentioned in the rule and the weights on the connections to the antecedents are not allowed to change during training.
Note, because definitional rules never change, they act like inputs to the network. Thus, units corresponding to the consequents of definitional rules are counted as if they were input units for the purposed of adding links to the network. (See Section 2.2.1 for details of link addition.)

<- A "variable" rule.
Variable rules are standard rules; they are subject to change during training. Therefore, consequents may be connected to antecedents other than those mentioned in the rule, and

weights on the links entering into the consequent may be changed during backpropagation training.

A.2.2 Antecedents

In addition to simply listing positive antecedents to form a simple conjunctive rule, KBANN allows the following special predicates in the antecedents of its rules.

Gnot	Negate an antecedent.
Greater-than	Only for "linear" and "ordered" features.
Less-than	Only for "linear" and "ordered" features.
N-true	Allows compact specification of "n-of-m" concepts.

The specification of antecedents is recursive. Hence, the user may nest any of these types of antecedents within each other.

The format for including special predicates in rules, and the situations in which special predicates can be used are described below. In addition, each of these descriptions includes a picture showing the translation of a rule containing one of the special antecedents into a network.

Gnot

Gnot[1] allows rules to state that an antecedent must not be true for the consequent to be true. When multiple antecedents must not be true, each one should be in its own *Gnot* clause. Thus, the rule:

```
A :- B ∧ ¬ C ∧ ¬ D
```

would be written for KBANN as:

```
(<- A B (GNOT C) (GNOT D))
```

Figure A.3a shows how KBANN translates this rule into a neural network. Briefly, this figure shows that the negated antecedents are attached to the consequent with a weight that is the negative of the weight on the unnegated antecedents. Section 2.2.3 contains a proof that this encoding of negated antecedents is correct.

Greater-than and less-than

Greater-than and *less-than* both are defined only for "linear" and "ordered" features. (See the previous section (A.1) for descriptions of these feature types.) For instance, given a linear feature defined by:

```
(size linear ((breadbox 0 10) (car 10 50)
              (bus 50 100) (house 100 200)))
```

then a rule that states "an object must be smaller than a bus" would be written:

```
(<- goodsize (less-than size bus))
```

Figure A.3b shows how KBANN translates this rule into a neural network. The figure shows that the network has negative weight links from `bus` and `house` and positive weight links from `breadbox` and `car`. Hence, is the example is either car or breadbox sized, the net activation

[1]The term "Gnot" is used rather than simply "Not" simply to differentiate it from a function in the language used to implement KBANN.

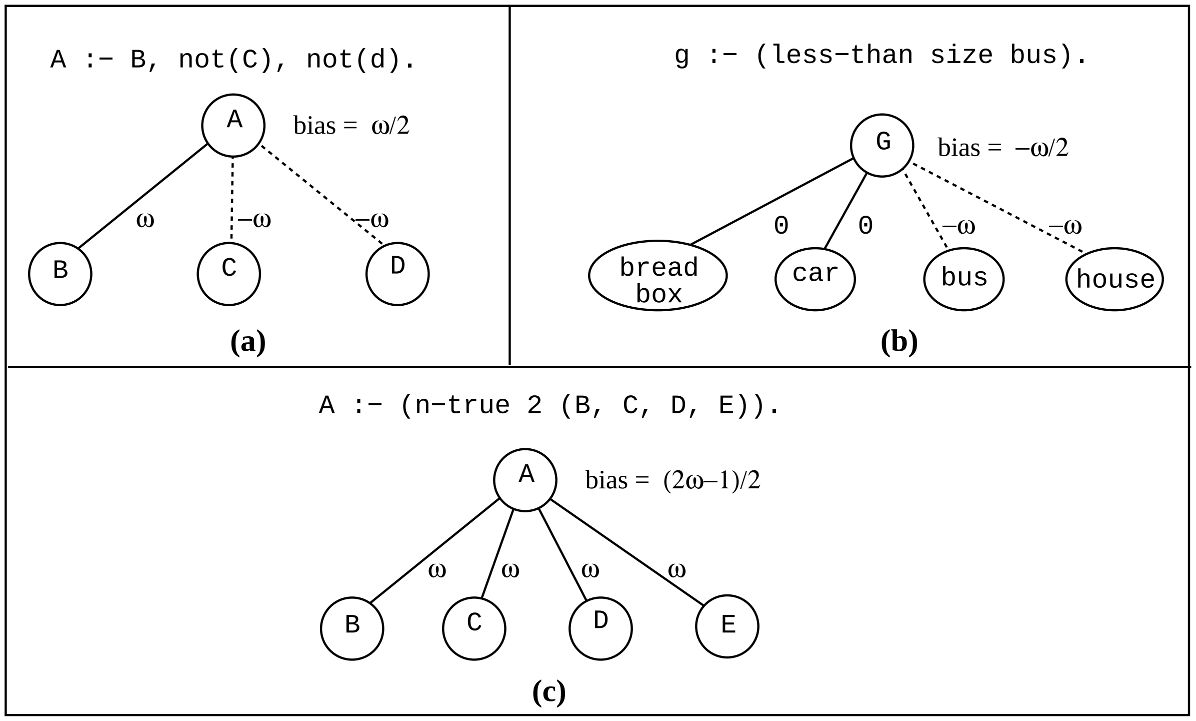

Figure A.3: Encoding rules in a neural network.

to the unit encoding the consequent will be greater than the bias. As a result, the unit will be active. Otherwise, the net activation will be less than the bias so the unit will be inactive.

N-true

N-true is a way of compactly specifying concepts characterized by requiring that n of m antecedents ($m > n$) be true. The antecedents are written in the form:

(`N-TRUE` n `(antecedent1 ...antecedentM)`)

Figure A.3b shows how KBANN translates an n-of-m rule into a neural network. This figure shows that the network to encode an n-of-m rule is exactly equivalent to a network to encode a conjunct except for the bias which is reduced to reflect the fact that not every antecedent must be true for the consequent to be satisfied.

A.2.3 Variables

Recall that KBANN only allows the user to specify propositional knowledge. Hence, predicate-calculus variables are not allowed in rules. However, KBANN is able to use variables in a very limited way to reduce the number of rules that the user needs to write. The essential idea is to replace the *value* in a *(feature value)* antecedent with a variable. (Variables are denoted by a '?' prefix. Thus, *?name* is treated by KBANN as a variable.) KBANN expands all variables for which there is a known set of possible values by creating multiple rules, each with one value from the known set. When multiple values of a single rule are replaced by the same variable name, then KBANN expands the variable with values that appear in both sets.

For example, consider the problem of comparing two objects to determine whether or not they are similar. One of the requirements for similarity might be that the objects are the same color. If the possible colors for the objects are {`red`, `green`, `blue`} then the user could encode the comparison of the two objects using the following three rules:

```
(<- samecolor (obj1 red) (obj2 red))
(<- samecolor (obj1 green) (obj2 green))
(<- samecolor (obj1 blue) (obj2 blue))
```

Alternately, these three rules could be written using the following single rule by merely using variables in the antecedents appropriately.

```
(<- samecolor (obj1 ?y) (obj2 ?y))
```

Appendix B

PSEUDOCODE

This appendix contains pseudocode for the major parts of KBANN. Following this pseudocode, readers should be able to re-implement the code used in this thesis. In combination with the datasets available from the collection at UC-Irvine, this makes possible the reproduction of every experiment described in this thesis.

Note: in all of the following pseudocode, lines beginning with semicolons are comments.

B.1 Pseudocode for the Rules-to-Network Translator

The pseudocode in this section is for a basic implementation of the rules-to-networks translator. It assumes that all features are nominal and all antecedents are positive. This translator is sufficient for the promoter recognition problem and needs only a small expansion (to handle n-of-m style predicates) for the splice-junction determination problem.

Rules-to-Network()

```
Let: $\mathcal{U}$ be a data structure for units with the following fields
            linksFrom - an initially empty list
            ruleCount - initially 0
            bias - initially 0
            level - initially 0
            name - no initial value
     $\epsilon$ be a very small number
     $\omega$ be the initial weight for links specified by knowledge
Given: $\mathcal{R}$ - a list of rules with no internal disjunctions and
            no external disjunctions with more than one antecedent
            ;; See Appendix A.2 and Chapter 2
       $\mathcal{F}$ - a list of nominal features where each item in the list is in
            the form (featureName value1 value2 ... ):
            ;; See Appendix A.1
;; First create all the input units for the KBANN-net by going through each nominal
;; feature and making a unit with the name $fearureName.valueName$
;; Note that $\mathcal{U} : name$ refers to the name field of unit $\mathcal{U}$
$\forall(f \in \mathcal{F})$
  $\forall(v \in < values\ of\ f >)$
    create a unit $\mathcal{U}$ with
      $\mathcal{U} : name = f.v$
      $\mathcal{U} : level = 1$
;; Next create hidden and output units by creating a unit that corresponds
;; to every unique consequent and antecedent in the set of rules. Also,
;; link each consequent to its antecedents and increment the bias of each
;; consequent on the assumption that the rule is conjunctive.
;; This code above assumes that all disjunctive rules with more than one
;; antecedent originally in $\mathcal{R}$ have been rewritten according to Section 2.2.
$\forall(r \in \mathcal{R})$
  Let $\mathcal{U}_r$ be the unit with $\mathcal{U}_r.name =< consequent\ of\ r >$
  if $\mathcal{U}_r$ does not exist then create it
  Increment $\mathcal{U}_r.ruleCount$
  $\forall(a \in < antecedentsofr >)$
    Let $\mathcal{U}_a$ be the unit with $\mathcal{U}_a.name = a$
    if $\mathcal{U}_a$ does not exist then create it
    Create a link from $\mathcal{U}_a$ to $\mathcal{U}_r$ with weight $\omega$ and put it in $\mathcal{U}_r.linksFrom$
    Add $\omega$ to $\mathcal{U}_r.bias$
;; Now adjust the bias of each unit so that it is either $\frac{2P-1}{2}\omega$
;; for conjunctive units or $\frac{\omega}{2}$ for disjunctive units
;; If a unit encodes a disjunct, then its rulecount will be greater then one.
$\forall u \in < listofunits >)$
  if $(u.rulecount > 1)$ then $u.bias = \frac{\omega}{2}$
  else Decrement $u.bias$ by $\frac{\omega}{2}$
Establish levels of units according to one of the schemes described in Section 2.2.
;; Add low weight links to the network by fully connecting all units
;; separated by one level
$\forall u \in < listofunits >)$
  $\forall uu \in < listofunits >)$
    if $((u.level = uu : level + 1)$ and (there is no existing link between $u$ and $uu$))
      THEN create a link from $uu$ to $u$ with weight 0
;; Finally, slightly perturb the network by adding to each weight a number close to zero.
;; This prevents problems resulting from symmetries in the network.
$\forall(u \in < listofunits >)$
  $\forall(l \in u : linksFrom)$
    add a random number in the range $[-\epsilon \ldots + \epsilon]$ to the weight of $l$
  add a random number in the range $[-\epsilon \ldots + \epsilon]$ to $u : bias$
```

B.2 Pseudocode for the Network-to-Rules Translator

B.2.1 The SUBSET algorithm

```
SUBSET():
GIVEN: $U$, a set initially containing each output unit
       $Z$, a positive number (typically 1.0)
       $\beta_n$, a beam width for subsets of negatively weighted links
       $\beta_p$, a beam width for subsets of positively weighted links
;; Form rules for only the units which participate in other rules or lead to a final conclusion
WHILE $U \neq \{\}$
  ;; begin by finding combinations of the positively weighted links that exceed the bias
  LET $v = pop(U)$
      $\phi = < bias\ on\ v > + Z$
      ExceedingSubsets = BBSEARCH(positively weighted links to $v$, $\phi$,$\beta_p$)
;; then find combinations for the negatively weighted links that drive
;; each positive combination below the bias
  $\forall(es \in ExceedingSubsets)$
    ADD to $U$ the units in $es$ that have not previously been seen
    LET $\Phi = < sum\ of\ link\ weights\ in\ es > + Z - < bias\ on\ v >$
        NegatingSubsets = BBSEARCH(negatively weighted links to $v$, $\Phi$, $\beta_n$)
    $\forall(ns \in NegatingSubsets)$
      ADD to $U$ units in $ns$ that have not previously been seen
      FORM a rule: if $es$ and (not $ns$) then $v$.
```

```
BBSEARCH(links, threshold, beamWidth)
;; this algorithm simply implements a beam search
;; with a branch-and-bound component
LET branches be every link in links
$\forall(b \in branches)$
  LET branchSum be weight of link
LOOP
  Eliminate all branches for which threshold is greater than
    <sum of the weights of all links of lesser weight than any link in the branch>
      + branchSum
  KEEP only beamWidth of branches by eliminating branches with smallest branchSum
  IF every member of branches has $branchSum > threshold$ then Terminate LOOP
  IF there are no ways to expand any member of branches then Terminate LOOP
  WITH every branch
    Form new branches using every link of lesser weight than any link in the branch
    Increment branchSum of new branches by the weight of then added link
Return all members of branches with $branchSum > threshold$
```

B.2.2 The NofM algorithm

N-of-M()

```
GIVEN $U$, a set initially containing each output unit
        $exs$, the training examples
;; $U$ is a set containing all consequents that have not yet been expanded into rules
WHILE $U \neq \{\}$
  LET $v = pop(U)$
      $i = 0$, $sum = 0$
      $\phi$ = bias on $v$
      $\gamma$ be a set of groups found using GROUPER(links to $v$)
      $\mathcal{R}v$ be a rule corresponding to the unit $v$;
          initially $\mathcal{R}v$ has no antecedents and a threshold $= \phi$
  $\gamma = DELETE\_GROUPS(\gamma, \phi, exs)$
  ;; With only important antecedents remaining, rewrite the result as a single rule
  $\forall(g \in \gamma)$
    $\forall(ga \in g)$
      Add to $\mathcal{R}v$ an antecedent $ga$ with weight $\gamma.weight$
      Add to $U$ $ga$ unless $ga$ is an input units or has already been added to $U$

Regularize the rules by making the threshold on every rule the same
  and adjust antecedent weights accordingly
Re-express all of the rules as a neural network using the rules-to-network translator.
Freeze weights on all links
Optimize thresholds using training examples and backpropagation
Simplify rules by eliminating weights on antecedents and thresholds
```

DELETE_GROUPS(*groups*, *threshold*, *examples*)

```
;; This function heuristically eliminates clusters of links that are not useful
;; for the classification of training examples
Associate with each group a counter $need\_group$ which has an initial value of 0.
$\forall(e \in examples)$
   Compute the activation of every unit
   If the current example is correctly classified
      ;; eliminate the activation traveling along each group to see if that group
      ;; is important for the correct classification of the current example
      $\forall(g_i \in groups)$
         Set the weight on all links in $g_i$ to 0.
         Compute the activation of every unit
         if the state of any consequent changes
            then increment $need\_group_i$
         Reset the weight of the links in $g_i$ -
   $\forall(g_i \in groups)$
      if ($need\_group_i = 0$ then remove every link in $g_i$ from the network
      Return $groups$
```

```
GROUPER(links)
;; The group finding algorithm for NofM that runs in $O(n * lg(n))$ time
Sort links in descending order by weight magnitude
Delete all links whose magnitude is below a threshold $\tau$   ; typically $\tau = 0.1$
LET $\gamma = \{\}$
      $g = \{\}$
      $g\_avg = 0$
      $\mathcal{T}$ = a user defined threshold from [0...1] (Typically 0.75)
$\forall(L \in links)$
  LET $lw = < magnitude\ of\ weight\ on\ L >$
  IF $(g = \{\})$ THEN
  ;; the current group is empty so start a new one
    $g\_max = lw$
    PUSH $L$ onto $g$
  ELSE
    IF $(lw > \mathcal{T} * g\_avg)$ THEN
      ;; add link to the existing group as it is close enough to the group's average
      PUSH $L$ onto $g$
      $g\_avg = average($<magnitude of link weights in g>$)$
    ELSE
      ;; The link is too small to be added so close the current group and start a new one
      PUSH $g$ onto $\gamma$
      $g\_avg = lw$      $g = \{ L \}$      $g\_max = lw$
    IF $(g\_max > g\_avg * \frac{1}{\mathcal{T}})$ THEN
      ;; the addition of a link drags the average down so that the highest weight link is
      ;; too large. So eliminate the current link from the current group, close the current
      ;; group and start a new group
      REMOVE $L$ from $g$
      PUSH $g$ onto $\gamma$
      $g\_avg = lw$      $g = \{ L \}$      $g\_max = lw$
RETURN $\gamma$
```

B.3 Pseudocode for the DAID Algorithm

In the pseudocode of this section, the notation $\rho[true]\rho[incorrect]\phi[present]$ is used to identify a single counter from among the eight set up at each input unit from each low level consequent. Replacing a name inside '[]' with a ? indicates that either value is acceptable. Hence, $\rho[?]\rho[incorrect]\phi[present]$ is equivalent to $\rho[true]\rho[incorrect]\phi[present]+\rho[false]\rho[incorrect]\phi[present]$.

```
DAID()
$\forall(\phi \in InputFeatures)$          ;; First initialize counters
    $\forall(\rho \in Rules)$
        Set up the counters for each of the following 8 situations:
            $\rho[True|False]\rho[Correct|Incorrect]\phi[Present|Absent]$
FOREACH $(\rho \in Rules)$
    Establish a state-variable $\rho\_consequentCorrect$
$\forall(\epsilon \in Examples)$      ;; scan each example incrementing the appropriate counters
    Compute truth value of each rule
    $\forall(fc \in FinalConclusions)$
        BackUpAnswer($fc$, correctAnswer, false)
    $\forall(\phi \in InputFeatures)$
        $\forall(\rho \in Rules)$
            Increment the appropriate counter from among
                $\rho[True|False]\rho[Correct|Incorrect]\phi[Present|Absent]$
;; with counters complete, now compute a number that captures the contribution
;; of each feature-value pair to correcting errors of each of the
;; lowest level antecedents $\forall(\phi \in InputFeatures)$
    $\forall(\rho \in Rules)$
        $on = \frac{\rho[true]\rho[correct]\phi[?]}{\rho[true]\rho[?]\phi[?]} * (\frac{\rho[true]\rho[incorrect]\phi[present]}{\rho[true]\rho[incorrect]\phi[?]} - \frac{\rho[false]\rho[?]\phi[present]}{\rho[false]\rho[?]\phi[?]})$
        $off = \frac{\rho[false]\rho[correct]\phi[?]}{\rho[false]\rho[?]\phi[?]} * (\frac{\rho[true]\rho[?]\phi[present]}{\rho[true]\rho[?]\phi[?]} - \frac{\rho[false]\rho[incorrect]\phi[present]}{\rho[false]\rho[incorrect]\phi[?]})$
        $on = max(0, on)$
        $off = min(0, off)$
        Replace the 8 counters of $\phi$ with: $\rho\_useweight$ = if ($on >$ abs($off$)) $on$ else $off$
```

```
BackUpAnswer(consequent, correctAnswer, currentlyCorrect)
;; a recursive procedure for stepping backward through a rule set, following
;; trails of correct and incorrect reasoning by the rules.
$\rho\_consequentCorrect$ = (currentlyCorrect or ValueOfConsequent = correctAnswer)
;; first determine if the truth-value of the consequent is correct.
;; It is correct if either its parent is correct or the truth-value is the same as $correctAnswer$.
FOREACH $(d \in DependenciesOfConsequent)$
    IF <at a negative dependency>
        dd = not(correctAnswer)
        ELSE dd = correctAnswer
    BackUpAnswer(d, dd, $\rho\_consequentCorrect$)
```

Appendix C

KBANN TRANSLATION PROOFS

This appendix contains detailed proofs that the scheme used by the rules-to-network translator accurately translates both conjunctive and disjunctive rules. These proofs complement the proofs appearing in Section 2.2.3. Specifically, Theorems 3 and 5 restate Theorems 1 and 2 but use a significantly different approach in the proof.

The following terms are used in the proof of Theorem 3:

C_a	the minimum activation for a unit to be considered *active*
A	a number such that $C_a \leq A \leq 1.0$
C_i	the maximum activation for a unit to be considered *inactive*
I	a number such that $0.0 \leq I \leq C_i$
$\mathcal{F}(x)$	$1/(1+e^{-x})$, the standard logistic activation function
w_p	the weight on links corresponding to positive dependencies
w_n	the weight on links corresponding to negated dependencies
θ	the bias
P	the number of positive antecedents to the rule
N	the number of negated antecedents to the rule
K	the total number of antecedents to a rule ($K = P + N$)
p	a number such that $1 \leq p \leq P$
n	a number such that $1 \leq n \leq N$.

Theorem 3: *(Repeats Theorem 1) Setting:*

1. $w_p = -w_n = \omega > 0$
2. $\theta = -\frac{2P-1}{2}\omega$

correctly encodes conjunctive rules into networks that reproduce the behavior of the rules under the conditions specified below.

The proof of this theorem uses Lemmas 3A and 3B.

Lemma 3A: *Part 1 of Theorem 3, that $w_p = -w_n = \omega$, results in an accurate encoding of conjunctive rules.*

Proof of Lemma 3A:

The method for mapping conjunctive rules into a KBANN-net is correct when Inequalities C.1 – C.4 are satisfied. Briefly, Inequality C.1 describes the only condition under which the consequent of a conjunctive rule is true; namely that all of its positive dependencies are true and none of its negative dependencies are true. Inequality C.1 simply translates this statement into a network, saying that for a unit to be active all of its positive antecedents must be active and none of its negated antecedents can be active. Inequality C.2[1] states the condition that a unit is inactive when fewer than all of its positive antecedents are active. Similarly, Inequality C.3 states the condition that a unit is inactive when at least one of its negated antecedents in active. Inequality C.4 gives the fourth possible condition on a unit; a unit is inactive when fewer than all of its positive antecedents are active and more that one of its negated antecedents are active. Hence, these inequalities describe all of the conditions which a conjunctive unit created by KBANN may experience.

$$C_a \leq \mathcal{F}[Pw_pA + Nw_nI + \theta] \tag{C.1}$$

$$C_i \geq \mathcal{F}[(P-p)w_pA + pw_pI + Nw_nI + \theta] \tag{C.2}$$

$$C_i \geq \mathcal{F}[Pw_pA + (N-n)w_nI + nw_nA + \theta] \tag{C.3}$$

$$C_i \geq \mathcal{F}[(P-p)w_pA + pw_pI + (N-n)w_nI + nw_nA + \theta] \tag{C.4}$$

The interesting conditions on these four inequalities are when they are at their boundaries. For instance, the interesting case for Inequality C.1 is when the net incoming activation to a unit is minimized. This occurs when each of the positive antecedents has an activation of exactly C_a and each of its negative antecedents has an activation of exactly C_i. If the resulting activation at this boundary condition only equals C_a then Inequality C.1 will always be satisfied because the every change to the network for which the rule remains true must increase the net incoming activation. Hence, any change must increase the activation of the unit. Equation C.5 expresses this boundary condition on Inequality C.1.

For units that should be inactive – because the consequent of the underlying rule is not satisfied – the converse of the preceding discussion applies. Thus, interesting boundary condition on Inequality C.2 occurs when the net incoming activation to a unit is maximized. This occurs exactly one of positive antecedents inactive, the activation of that antecedent is C_i, the activation of the remaining $(P-1)$ positive antecedents is 1.0, and the activation of every negative antecedent is 0.0. If, at this maximum, the unit has an activation of C_i, then Inequality C.2 will always be satisfied. Equation C.6 expresses this boundary condition on Inequality C.2.

[1]Inequality C.2 should be written

$$C_a \leq \mathcal{F}[\sum_{j=1}^{P} w_pA_j + \sum_{i=1}^{N} w_nI_i + \theta]$$

to be precisely correct. The complete statement is not made simply for brevity.

The final interesting boundary condition (on Inequality C.3) is very similar to the boundary condition on Inequality C.2. It occurs when exactly one of the negative antecedents is active. Specifically, the net incoming activation is maximized when the activation of each of the positive antecedents is 1.0, the activation of the active negated antecedent is C_a, and the activation of every negative antecedent is 0.0. Again, at this maximum, the unit must have an activation no greater than C_i. Equation C.7 expresses this boundary condition on Inequality C.3.

Equations C.6 and C.7 capture both of the interesting boundary conditions that might result from Inequality C.4. Hence, Equations C.5 – C.7 capture all of the interesting boundary conditions on the activation of a unit that encodes a conjunct. Equations C.8–C.10 simply restate these equations in more algebraically tractable forms.

$$C_a = \mathcal{F}[Pw_pC_a + Nw_nC_i + \theta] \tag{C.5}$$

$$C_i = \mathcal{F}[(P-1)w_p1.0 + w_pC_i + Nw_n0.0 + \theta] \tag{C.6}$$

$$C_i = \mathcal{F}[Pw_p1.0 + (N-1)w_n0.0 + w_nC_a + \theta] \tag{C.7}$$

$$-lg(\frac{1}{C_a} - 1) = Pw_pC_a + Nw_nC_i + \theta \tag{C.8}$$

$$-lg(\frac{1}{C_i} - 1) = (P-1)w_p1.0 + w_pC_i + \theta] \tag{C.9}$$

$$-lg(\frac{1}{C_i} - 1) = Pw_p1.0 + w_nC_a + \theta] \tag{C.10}$$

Finally, let $C_i = 1 - C_a$, $(C_a > C_i)$; this is reasonable as it yields a nice symmetric solution. Simply subtracting Equation C.10 from Equation C.9 results in $w_p = -w_n$. Hence, Lemma 1A is correct.[2]
□

Lemma 3B: *The maximum number of antecedents for which a network can consistently encode a conjunctive rule is given by* $K < \frac{C_a}{1-C_a}$.
Consider the following equations:

$$Pw_pC_a + Nw_nC_i + \theta > (P-1)w_p1.0 + w_pC_i + \theta \tag{C.11}$$

$$P\omega C_a - (K-P)\omega(1-C_a) + \theta > \omega(P-1) - \omega(1-C_a) + \theta \tag{C.12}$$

$$K < \frac{C_a}{1-C_a} \tag{C.13}$$

Inequality C.11 states the relationship between the right sides of equations Equations C.8 and C.9. This inequality simply states that the net incoming activation to a unit that is

[2]Note that the strict equality of w_p and $-w_n$ results for the assumption that Equations C.6 and C.7 both equal C_i. Relaxing this assumption to results in defining a range for w_n given a selection for w_p. Specifically: $-w_p - C_i \leq w_n \leq -w_p + C_i$. As $w_p = -w_n$ is at the center of this range, the results in the rest of this section will use these values.

active must be greater than the net incoming activation to a unit that is inactive. Substituting $1 - C_a$ for C_i (as in Lemma 1A), ω for w_p and $-\omega$ for w_n (the result of Lemma 1A) and $K - P$ for N yields Inequality C.12. This inequality reduces to Inequality C.13. Thus, Lemma 1B, which bounds the total number of antecedents admissible in a rule, is correct.

□

Proof of Theorem 3:

Lemma 3A proves that first part of Theorem 3.

One last step before completing Theorem 3; in addition to the work they have already done, the above equations can be used to determine values for the link weights. Specifically, substituting Lemma 3A that $w_p = -w_n$ into Equations C.8 and C.10, subtracting Equation C.10 from C.8 and solving for w_p yields Equation C.14. (The denominator of this equation re-expresses the bound defined by Lemma 3B.) This equation shows that the weight on links to encode a conjunctive rule is a function of both the amount a unit is allowed to differ from one (i.e., C_a) or zero and the number of antecedents in the rule.

$$w_p = \frac{lg(\frac{1}{C_i} - 1) - lg(\frac{1}{C_a} - 1)}{C_a + K(C_a - 1)} \tag{C.14}$$

At long last, the second part of Theorem 1 can be approached. This part of the theorem, which states that $\theta = -\frac{2P-1}{2}\omega$, is not algebraically correct. However, it is very close to correct. The following establishes bounds on the error for this method of setting for the bias.

Equation C.15 is Equation C.8 into which has been substituted Lemma 1A that $w_p = -w_n$. Solving Equation C.15 for θ (by substituting the setting for w in Equation C.14) yields Equation C.18. (Equations C.16 and C.17 show two steps of the simplification).

So, the formula for setting the bias given by part 2 of Theorem 1 is exactly correct when $1 = C_a + KC_i$. Unfortunately, this situation only occurs when $K = 1$. As rules rarely have only a single antecedent, the setting proposed in part 2 of this theorem is rarely exactly correct. However, substituting Lemma 1B that $K < \frac{C_a}{1-C_a}$ into $C_a + KC_i$ yields the result that $C_a + KC_i$ must always be in the range [1...2]. Therefore, $\theta = -\frac{2P-1}{2}\omega$ is never far from correct.

$$\theta = -lg(\frac{1}{C_a} - 1) - PwC_a + NwC_i \tag{C.15}$$

$$\theta = -lg(\frac{1}{C_a} - 1) - [K(C_a - 1) + P]\frac{-2lg(\frac{1}{C_a} - 1)}{C_a + K(C_a - 1)} \tag{C.16}$$

$$\theta = \frac{[-lg(\frac{1}{C_a} - 1)][C_a + K(C_a - 1)] - [K(C_a - 1) + P] - 2lg(\frac{1}{C_a} - 1)}{C_a + K(C_a - 1)} \tag{C.17}$$

$$\theta = \frac{-2P + C_a + KC_i}{2}w \tag{C.18}$$

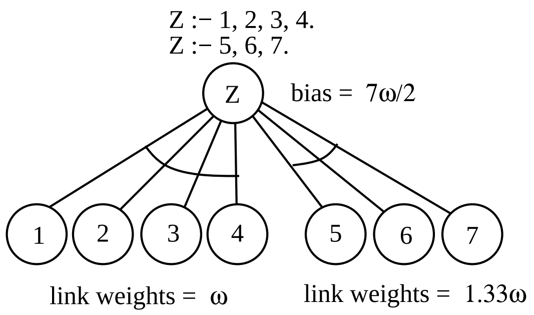

Figure C.1: The encoding of two disjunctive rules with multiple antecedents.

Moreover, while there will always be differences when using the simple function $\theta = -\frac{2P-1}{2}\omega$ to set the bias, these differences are too small to have an important effect on the resulting encoding of a conjunctive rule. Hence, $\theta = -\frac{2P-1}{2}\omega$ results in an approximately correct setting for the bias.

Therefore, given that $K < \frac{C_a}{1-C_a}$, setting $w_p = -w_n = \omega$ and $\theta = -\frac{2P-1}{2}\omega$ yields an encoding of conjunctive rules that is very close to correct.
□

Recall that the first step of the rules-to-network algorithm (Table 2.2) rewrites disjuncts so that all rules with the same consequent have exactly one antecedent. Figure C.1 illustrates the need for this rewriting. The figure shows two rules, one with four antecedents (named 1, 2, 3, and 4) and one with three antecedents (named 5, 6, and 7) all feeding unit Z. The bias on unit Z, and the links on to all of the antecedents are set so that Z will be active if either rule is true. However, Z will also be active when neither rule is true. For instance, if the units corresponding to antecedents 2, 3, 4, and 5 are all active, then Z will be active.

This example illustrates the need for rewriting disjunctive collections of rules to eliminate rules with more than one antecedent. The following theorem proves the necessity of this rewriting.

Theorem 4: *There is no way to correctly encode a set of disjunctive rules each of which has more than one antecedent into a neural network that uses a single unit to encode the consequent.*

Proof:

This proof will proceed by assuming that the theorem is false and showing a contradiction. The following terms are used:

v	a consequent of a set of disjunctive rules.
v_i	a rule for v
N_i	the number of antecedents in rule v_i
θ	the bias on the unit corresponding to v
ω_i^j	the weight on the link corresponding to the jth antecedent of rule v_i

The proof makes three nonrestrictive assumptions: (1) that there are no negative antecedents to rules (this assumption can easily be enforced through the addition of a unit that encodes *not*), (2) that there exists two rules v_1 and v_2 such that no combination of the antecedents of these two rules forms some other rule v_i and (3) that all units have an activation of either zero or one (this assumption is made only for simplicity).

The network encoding this set of rules must satisfy, for each rule i:

$$-\theta < \sum_{j=1}^{N_i} \omega_i^j \tag{C.19}$$

Assume the opposite of the theorem, namely that the only combination of link weights which exceed θ contain at least one of the rules v as a subset. Under these assumptions, the following inequalities are true:

$$-2\theta < \sum_{i=1}^{2}\sum_{j=1}^{N_i} \omega_i^j \tag{C.20}$$

$$-\theta > \omega_1^1 + \omega_2^1 \tag{C.21}$$

Equation C.20 simply states that the sum of two rules must be greater than twice $-\theta$ since each rule by itself must activate the unit. Equation C.21 says that the sum of the first antecedents of rules v_1 and v_2 is less than $-\theta$. This follows from the assumption that no unwanted combinations exceed θ. Subtracting $\omega_1^1+\omega_2^1$ from each side of Equation C.20 yields Equation C.22. Note that the term to the left of the inequality in Equation C.22 subtracts from -2θ an number that is less than $-\theta$. Hence, Equation C.23 follows directly from Equation C.22.

$$-2\theta - \omega_1^1 - \omega_2^1 < \sum_{i=1}^{2}\sum_{j=2}^{N_i} \omega_i^j \tag{C.22}$$

$$-\theta < -2\theta - \omega_1^1 - \omega_2^1 < \sum_{i=1}^{2}\sum_{j=2}^{N_i} \omega_i^j \tag{C.23}$$

But, Equation C.23 indicates that there is a combination of link weights of v_1 and v_2 that exceeds $-\theta$. But, by assumption (2) no rules are composed of the antecedents of v_1 and v_2. This contradiction proves that the theorem is true.
□

The following terms are used in the next theorem:

C_a	the minimum activation for a unit to be considered *active*
C_i	the maximum activation for a unit to be considered *inactive*
$\mathcal{F}(x)$	$1/(1+e^{-x})$, the standard logistic activation function
w_p	the weight on links corresponding to positive dependencies
θ	the bias
R	the number of rules encoding the disjunct
r	a number such that $1 \leq r \leq R$

Theorem 5: *Setting:*

1. $w_p = \omega$
2. $\theta = -\frac{\omega}{2}$

results in a unit that accurately encodes a disjunct under the conditions specified below.

Proof: The method for mapping disjunctive sets of rules into a KBANN-net is correct when Inequalities C.24 and C.25 are satisfied. These two equations simply state the condition that a unit should be active when one or more of its inputs are active. It only should be inactive when all of its inputs are inactive.

$$C_a \leq \mathcal{F}[rw_pA + (R-r)w_pI + \theta] \tag{C.24}$$

$$C_i \geq \mathcal{F}[Rw_pI + \theta] \tag{C.25}$$

As in Theorem 3, this proof will proceed by analysis of the extreme cases. By ensuring that Inequalities C.24 and C.25 are satisfied in the extreme cases, the non-extremes are guaranteed.

The extreme for Inequality C.24 occurs when one of the antecedents is true and carries the smallest signal indicative of truth (i.e., C_a) while all of the other antecedents are false and carry the smallest signal indicative of falsehood (i.e., 0). This extreme condition is given by Equation C.26. The parallel extreme condition for falsehood occurs when every antecedent is false and carries a signal of C_i. This extreme condition is given by Equation C.27.

Equations C.28 and C.29 simply restate Equations C.26 and C.27 in a more algebraically tractable form.

$$C_a = \mathcal{F}[w_pC_a + (R-1)w_p0.0 + \theta] \tag{C.26}$$

$$C_i = \mathcal{F}[Rw_pC_i + \theta] \tag{C.27}$$

$$-lg(\frac{1}{C_a} - 1) = w_p C_a + \theta \tag{C.28}$$

$$-lg(\frac{1}{C_i} - 1) = R w_p C_i + \theta \tag{C.29}$$

$$-2lg(\frac{1}{C_a} - 1) = w_p C_a - R w_p C_i \tag{C.30}$$

$$w_p = \frac{-2lg(\frac{1}{C_a} - 1)}{C_a - R(1 - C_a)} \tag{C.31}$$

Following Theorem 1, let $C_i = (1 - C_a)$. So, subtracting Equation C.29 from Equation C.28 yields Equation C.30 (note that $lg(\frac{1}{C_i} - 1) = -lg(\frac{1}{C_a} - 1)$). Equation C.31 results from solving Equation C.30 for w_p. There are two notes about Equation C.31. First, Theorem 3 has the same limitation as that imposed on Theorem 1 by Lemma 1B, as the Equation C.31 is only correct when $R < \frac{C_a}{1-C_a}$ (since $w_p > 0$). Second, this equation suggests setting $w_p \approx 4$ given the fairly-common conditions that $C_a = 0.9$ and $R = 4$. Hence, the empirically derived value of $\omega = 4$ is supported analytically.

Finally, substituting the value for w_p in Equation C.29 gives Equation C.32. Solving for θ yields Equation C.33 (following much the same process as was used to solve for θ in Theorem 1). Hence, when $C_a + RC_i = 1$ the setting proposed by Theorem 3 of $\theta = -0.5\omega$ is exactly correct. As in Theorem 1, the bound on R of $R < \frac{C_a}{C_i}$ restricts $C_a + RC_i$ to the range [1...2]. Thus, the setting for the bias proposed in theorem 3 can never be far from correct.

$$-lg(\frac{1}{C_i} - 1) = RC_i \frac{-2lg(\frac{1}{C_a} - 1)}{C_a - R(1 - C_a)} + \theta \tag{C.32}$$

$$\frac{-C_a - RC_i}{2} w = \theta \tag{C.33}$$

Therefore, the scheme for mapping disjuncts into KBANN-nets results in networks that accurately encode disjunctive rules under a wide range of conditions.
□

In conclusion, the proofs of Theorems 3 and 5 show that the simple methods for translating rules into neural networks are close to correct. The proofs also exactly specify correct methods for translating rules into networks. In practice, it is unnecessary to translate with complete fidelity because training of the network is easily able to overcome errors resulting from inexact translation.

Appendix D

ADDITIONAL EXPERIMENTAL RESULTS

This appendix contains the results of tests not reported elsewhere in the thesis. In particular, these results complement the comparisons of the algorithms of Section 3.2. Testing methodology is not described here, it follows Section 3.2.

Tables D.1 and D.2 report statistical comparisons of the results of ten-fold cross-validation testing on each dataset. The splice-junction table repeats data appearing in Figure 3.8. However, a similar table could not be built for the promoter dataset because Figure 3.7 uses a method of summarizing the data that reduces eleven tests to a single result. Hence, the results for promoters in Table D.1 are for ten-fold cross-validation testing follows the procedures used in Section 3.2 for testing splice-junctions. Briefly, each datapoint represents the average over eleven cross-validation tests to average out the effects of example presentation order and testing versus training set splits. In addition, the results for standard ANNs and KBANN-nets are average over ten trials using different initial weight sets to smooth the differences in learning that result from small differences in the initial weights of the neural networks.

Table D.1, which reports tests for promoters, shows KBANN to be significantly superior to the six empirical learning systems with which it is compared. The algorithms are compared using the "relative information score" (RIS); an information theoretic measure of learning efficiency suggested by Kononenko and Bratko [Kononenko91]. Figure D.1 presents the leaving-one-out results that appear in Figure 3.7 alongside the results of ten-fold cross-validation.

Large discrepancies between the two measures point to systems which benefited significantly from the error-counting method used for the leaving-one-out study. For instance, the relative information score for Cobweb differs by more than 30 points between ten-fold cross-validation and leaving-one-out. Recall that examples are counted as incorrect in the leaving-one-out study only when the example was misclassified on a majority of the eleven runs. Hence, this method of error counting enhances the results of systems such as Cobweb which frequently make between one and five classification errors but rarely make more than six errors. The effect of the differences in error counting is quite apparent in Tables D.3 - D.5 which show the results for leaving-one-out and ten-fold cross-validation for all of the 106 examples in the promoter training set.

Tables D.3 - D.5 present the results of both ten-fold cross-validation and leaving-one-out for every promoter example. Hence, these tables allow a visual comparison of the results

Table D.1: Statistical analysis of learning for promoter recognition.

System Name	RIS Avg.	RIS Std. Dev.	t statistic Cbwb	ID3	Near Neigh.	Ptron	PEBLS	BP
KBANN	90.4	2.95	18.4	17.2	20.2	15.8	5.7	8.8
BackProp	82.3	3.62	16.0	13.8	10.4	5.2	1.1	—
PEBLS	80.8	3.84	12.8	13.3	7.9	3.6	—	—
Perceptron	75.4	2.38	12.9	10.7	6.3	—	—	—
Near.Neigh	65.4	4.64	6.3	4.5	—	—	—	—
ID3	53.7	6.04	1.3	—	—	—	—	—
Cobweb	51.1	6.25	—	—	—	—	—	—

The t statistic is for one-tailed paired-sample comparisons of the systems. When $t > 1.8$ then the system whose name is given by the row is superior with 95% confidence to the system whose name is given by the column.

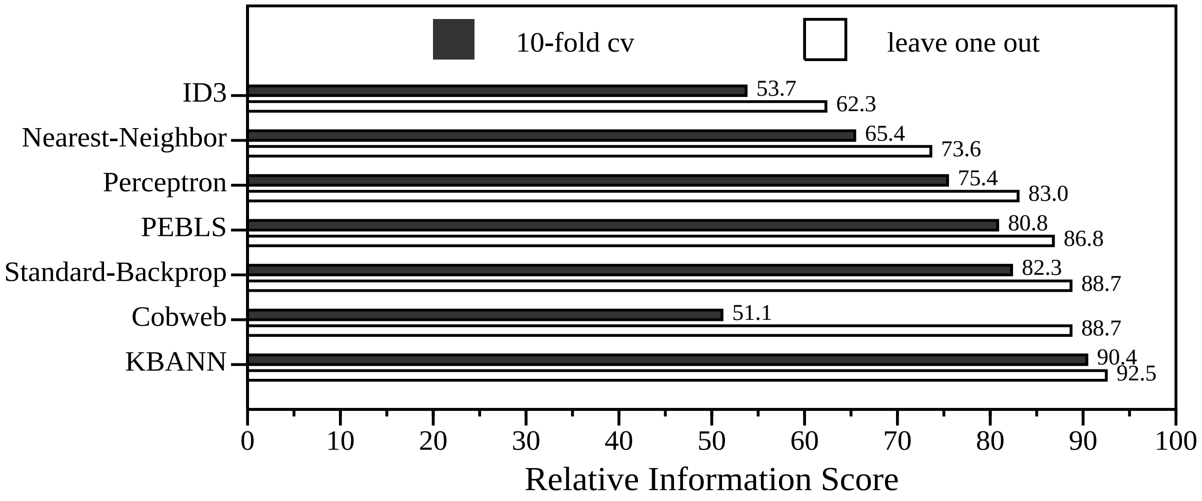

Figure D.1: Promoter recognition scores for both leaving-one-out and ten-fold cross-validation.

under two different testing methods. Numbers in bold in the ten-fold cross-validation columns indicate that the example was classified wrong in the majority of cases. The ten-fold cross-validation column for Perceptron reports decimals because Perceptron often could not make a clear decision. Frequently, it identified an example as being both a promoter and a non-promoter or, it identified en example as being neither. In both of these cases, the error count was incremented by the probability of randomly choosing the correct answer (i.e., 0.5).

An interesting comparison can be made between the results in Tables D.3 - D.5 and Figure D.2 which presents a cluster analysis of the 106 examples. The cluster analysis shows a tight central core of promoters that have names 'RR*'. The seven learning systems detailed in Table D.2 make virtually no errors on these examples. While the non-examples of promoters have no similar central cluster, the examples that are farthest away from the central promoter

Table D.2: Significance of differences in the splice-junction domain.

System Name	RIS Avg.	RIS Std. Dev.	*t* statistic: Near Neig.	Cbwb	ID3	Ptron	PEBLS	BP
KBANN	90.2	0.31	74.6	14.8	27.2	22.2	7.2	2.2
BackProp	89.5	1.07	58.6	13.1	13.3	12.5	1.9	—
PEBLS	88.7	0.63	56.1	10.7	18.6	12.3	—	—
KBANN	88.0	0.76	60.5	9.0	13.2	9.5	—	—
Perceptron	83.9	0.84	52.2	3.4	0.2	—	—	—
ID3	83.9	0.78	43.0	2.8	—	—	—	—
Cobweb	81.9	1.86	20.5	—	—	—	—	—
Near.Neigh	69.3	0.82	—	—	—	—	—	—

The t statistic is for one-tailed paired-sample comparisons of the systems. When $t > 1.8$ then the system whose name is given by the row is superior with 95% confidence to the system whose name is given by the column.

cluster are consistently correctly classified. To be specific, the 20 examples of non-promoters that cluster farthest from any promoter have about half the average error rate for the full set of examples. The 20 promoters closest to the central promoter cluster have about $\frac{1}{5}$ of the average error rate. On the other hand, the 20 promoters and non-promoters nearest to boundary between promoters and non-promoters have about double the average error rate.

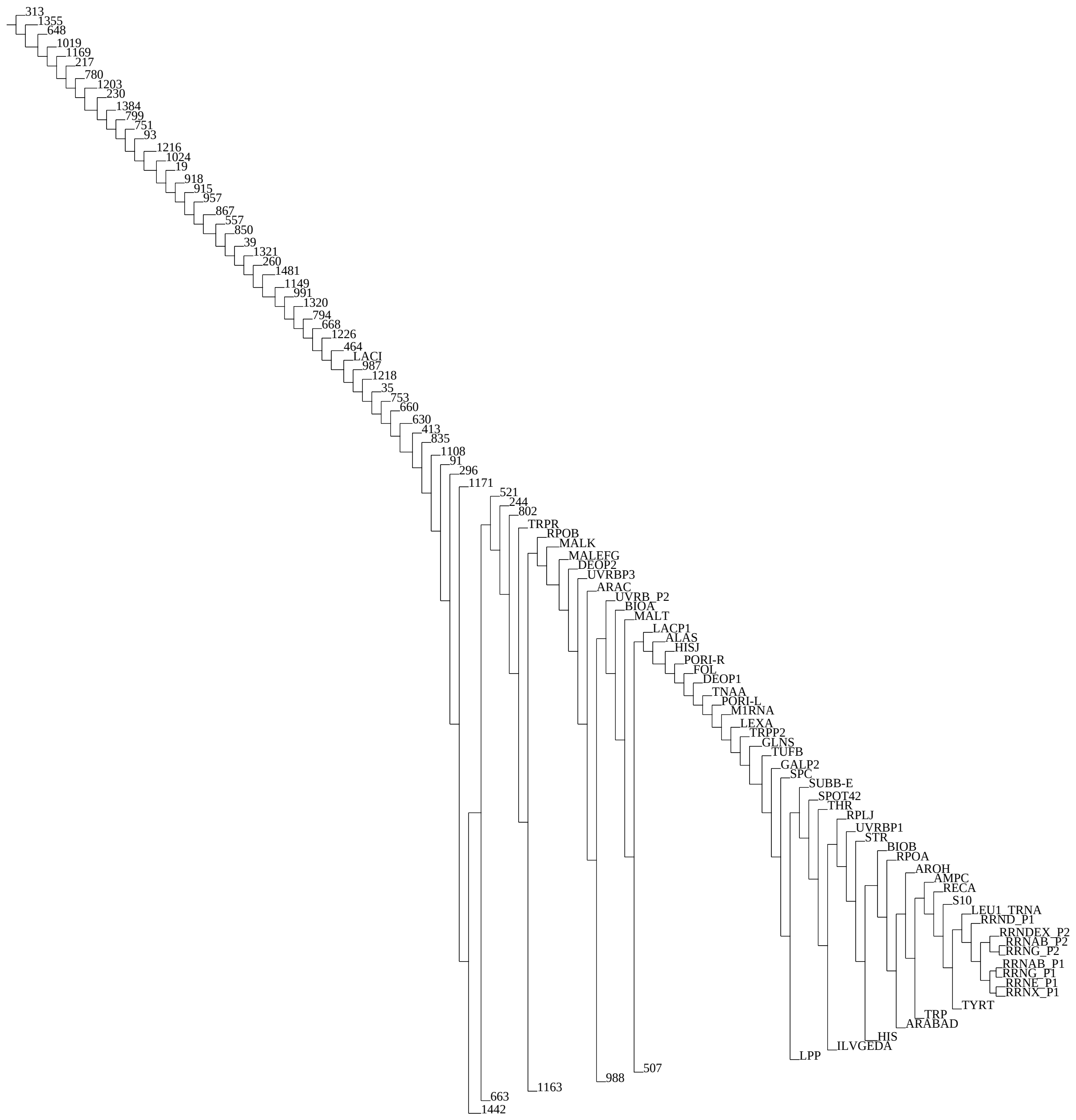

Figure D.2: Cluster analysis of the 106 example promoter set.

Table D.3: Results for every promoter example — part 1

	learning algorithms													
Example	KBANN		Standard ANN		PEBLS		Nearest Neighbor		ID3		Perceptron		Cobweb	
Name	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold
0019		0		0		0		0		3		0.0		3
0035		1		0		2		4		**8**		2.0		1
0039		2		0		0		0		4		0.5		0
0091		0		2	*	**11**	*	**11**	*	**7**		2.0		0
0093		0		0		0		4		0		0.0		0
0217		0		0		0		2		0		0.0		2
0230		0		0		0		4		4		0.5		0
0244		0		0		0	*	**11**	*	**6**		2.5		1
0260		0		2		0		**6**	*	5		2.0		0
0296	*	**8**		5		0		2		2	*	5.0		1
0313		0		0		0		0	*	1		0.0		0
0413		0		0		1		0		1		0.5		0
0464		0		5	*	**7**	*	**10**	*	**7**		**6.5**		2
0507		4	*	**11**	*	**6**	*	**11**	*	**11**	*	**9.0**	*	**11**
0521		0		0		0	*	**10**		**6**		3.5		2
0557		0		0		0		4		0		0.5		0
0630		0		0		0		0		2		1.0		1
0648		0		0		0		0		0		0.5		0
0660		0		0		0		3	*	0		0.0		1
0663		0		0		0		5		5		2.5		2
0668		2		3	*	**8**		1		3		3.5		1
0751		0		0		0	*	**7**	*	**6**		0.0		1
0753		0		0		2		0	*	3		0.5		1
0780		0		0		0		2	*	0		0.0		0
0794		0		0		0		1		0		1.5		2
0799		0		0		0		0		1		0.0		1
0802		0		3		4		0		4		5.0		0
0835		2		2		1	*	**11**		3		5.0		5
0850		0		0		0		0		5		0.5		2
0867		0		0		0		0		0		0.0		1
0915		0		0		0		0		0		0.0		1
0918		0		0		0		3		1		0.0		1
0957		0		0		0		4	*	**6**		1.0		1
0987		0		0		0		0	*	2		1.5		1
0988		0		3	*	**9**		**11**	*	**9**	*	5.0	*	**7**
0991		0		0		1	*	**11**		0		0.0		0

Notes: 'L1' indicates that the leaving-one-out methodology is used. '*' in a 'L1' column indicates an incorrect classification. Numbers in '10fold' columns indicate the number of incorrect classifications eleven trials using eleven orderings of the examples. Numeric names indicate examples that are non-promoters.

Table D.4: Results for every promoter example — part 2

	learning algorithms													
Example	KBANN		Standard ANN		PEBLS		Nearest Neighbor		ID3		Perceptron		Cobweb	
Name	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold
1019		0		1		0		0		4		4.0		0
1024		0		0		0		0		1		0.0		0
1108		0	*	**11**		3		3		1	*	**6.0**		5
1149		0		0		2		0		**9**		1.0		3
1163		0		0		2	*	**10**		3		2.5		2
1169		0		0		0		0		0		0.5		0
1171		0		0		0		0		3		0.0		1
1203		0		0		0		0		4		0.5		1
1216		0		0		0		5		1		2.0		1
1218		0		0		0		0	*	**6**		0.0		0
1226		0		0		0		2		0		0.0		0
1320		0		0		0		0		0		0.0		1
1321		0		0		0		0		5		0.0		0
1355		0		0		0		0		0		0.0		0
1384		0		0		0		0		0		0.0		0
1442		0		0		0	*	**11**		0		0.0		2
1481		0		0		0		1		1		0.0		1
alas		0		0		1		0		2		0.5		3
ampc		0		0		0		0		1		1.0		3
arabad		0		0		1		0		0		2.5		2
arac		0		0		0		0		**8**		0.0		1
aroh		0		0		0		0	*	0		0.0		0
bioa		0		0		3		0	*	2		1.0		0
biob		0		0		0		0		0		5.5		1
deop1		0		0		4		0		2		0.0		5
deop2	*	**8**		0		4		0		5		1.0		3
fol		0		0		0		0		3		2.0		3
galp2		0		0		0		0	*	1		0.0		4
glns		0		0		0		0		0		0.0		1
his		0		0		0		0		0		0.0		0
hisj		0		0		0		0		3		0.5		0
ilvgeda		0		0		0		0		1		0.0		0
laci	*	**11**	*	**11**	*	**9**	*	**8**		**11**		3.5	*	**9**
lacp1		0		5		2		1		3	*	3.5	*	**7**
lev1-trna		0		0		0		0	*	**9**		0.0		0
lexa		0		0		0		0		2		0.5		3

Notes: 'L1' indicates that the leaving-one-out methodology is used. '*' in a 'L1' column indicates an incorrect classification. Numbers in '10fold' columns indicate the number of incorrect classifications eleven trials using eleven orderings of the examples. Numeric names indicate examples that are non-promoters.

Table D.5: Results for every promoter example — part 3

Example	learning algorithms													
	KBANN		Standard ANN		PEBLS		Nearest Neighbor		ID3		Perceptron		Cobweb	
Name	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold	L1	10fold
lpp		0		0		0		0		1		0.0		1
m1rna		0		0		0		0		1		2.0		3
malefg		0		0		0		0		**8**		0.0		**6**
malk		5		0		4		0		**8**		1.5		2
malt		2	*	**10**		0		0		2	*	**7.0**		4
pori-l		0		0		1		0		0		0.0		2
pori-r		0		1		2		0		**9**	*	5.0		5
reca		0		0		0		0		0		0.0		0
rplj		0		0		0		0		1		0.0		2
rpoa		0		0		0		0		0		0.0		0
rpob		0	*	**11**		0		0		**11**	*	**7.0**	*	**7**
rrnab-p1		0		0		0		0		0		0.0		0
rrnab-p2		0		0		0		0		0		0.0		0
rrnd-p1		0		0		0		0		0		0.0		0
rrndex-p2		0		0		0		0		0		0.0		0
rrne-p1		0		0		0		0		0		0.0		0
rrng-p1		0		0		0		0		0		0.0		0
rrng-p2		0		0		0		0		1		0.0		0
rrnx-p1		0		0		0		0		0		0.5		0
s10		0		0		0		0		0		2.5		1
spc		0		0		0		0		**7**		0.0		1
spot42		0		0		0		0		1		0.0		0
str		0		0		0		0		0		0.0		0
subb-e		0		0		0		0		0		0.0		0
thr		0		0		0		0		0		0.0		0
tnaa		2		0		0		0		2		0.0		5
trp		0		0		0		0		0		0.0		0
trpp2		0		0		0		3		0		0.0		1
trpr	*	**9**	*	**11**		2	*	**9**		0	*	**10.0**	*	**10**
tufb		0		0		0		0		1		1.5		2
tyrt		0		0		0		0		**7**		0.0		0
uvrbp1		0		0		0		0		1		0.0		0
uvrbp2		0		4		1	*	**11**	*	0		0.5		5
uvrbp3		0		2	*	**7**		0		3		**6.0**		4
totals	4	5.1	6	9.4	7	9.1	14	18.4	19	24.5	9	13.0	6	15.9

Notes: 'L1' indicates that the leaving-one-out methodology is used. '*' in a 'L1' column indicates an incorrect classification. Numbers in '10fold' columns indicate the number of incorrect classifications eleven trials using eleven orderings of the examples. Numeric names indicate examples that are non-promoters.

Appendix E

BASE MODEL FOR GEOMETRY LEARNING

Tables E.1 - E.5 present the complete set of rules used by the KBANN-based model of geometry learning to simulate the knowledge of children prior to formal instruction in geometry. That is, these rule reproduce the geometric reasoning of children who have not had formal instruction in geometry.

Table E.6 defines the features used to describe each of the geometric figures used during the training and testing of the model.

Table E.1: Shape-naming rules — part I.

name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,somewhat), point-direction(?obj,up), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,very), point-direction(?obj,up), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,little), pointy(?obj,somewhat), point-direction(?obj,right), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,big), pointy(?obj,somewhat), point-direction(?obj,up), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,very), point-direction(?obj,down), shape(?obj,skinny), 2-long-and-2-short-sides(?obj,no).
name(triangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,little), pointy(?obj,somewhat), point-direction(?obj,down), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(pentagon, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(pentagon, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,large), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(hexagon, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(hexagon, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,large), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(quadrilateral, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,big), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(quadrilateral, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).

Table E.2: Shape-naming rules — part II.

name(square, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(square, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,big), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(rectangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,medium), 2-long-and-2-short-sides(?obj,yes).
name(rectangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,skinny), 2-long-and-2-short-sides(?obj,yes).
name(rectangle, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,no), area(?obj,medium), pointy(?obj,not), point-direction(?obj,none), shape(?obj,fat), 2-long-and-2-short-sides(?obj,yes).
name(rhombus, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,somewhat), point-direction(?obj,right), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(rhombus, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,little), pointy(?obj,somewhat), point-direction(?obj,up), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(parallelogram, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,somewhat), point-direction(?obj,left), shape(?obj,medium), 2-long-and-2-short-sides(?obj,yes).
name(parallelogram, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,big), pointy(?obj,somewhat), point-direction(?obj,right), shape(?obj,fat), 2-long-and-2-short-sides(?obj,yes).
name(parallelogram, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,little), pointy(?obj,very), point-direction(?obj,left), shape(?obj,skinny), 2-long-and-2-short-sides(?obj,yes).
name(trapezoid, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,very), point-direction(?obj,right), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(trapezoid, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,medium), pointy(?obj,somewhat), point-direction(?obj,up), shape(?obj,medium), 2-long-and-2-short-sides(?obj,no).
name(trapezoid, ?obj)	:-	tilted(?obj,0), slanty-lines(?obj,yes), area(?obj,big), pointy(?obj,somewhat), point-direction(?obj,left), shape(?obj,fat), 2-long-and-2-short-sides(?obj,no).

Table E.3: Feature-counting rules.

same-name(?obj1, ?obj2)	:-	name(?nme, ?obj1), name(?nme, ?obj2).
same-tilted(?obj1, ?obj2)	:-	tilted(?obj1, ?tlt), tilted(?obj2, ?tlt).
same-slanty-lines(?obj1, ?obj2)	:-	slanty-lines(?obj1, ?slant), slanty-lines(?obj2, ?slany).
same-area(?obj1, ?obj2)	:-	area(?obj1, ?area), (?obj2, ?area).
same-pointy(?obj1, ?obj2)	:-	pointy(?obj1, ?pty), pointy(?obj2, ?pty).
same-point-direction(?obj1, ?obj2)	:-	point-direction(?obj1, ?pd), point-direction(?obj2, ?pd)
same-shape(?obj1, ?obj2)	:-	shape(?obj1, ?shp), shape(?obj2, ?shp).
same-2-2(?obj1, ?obj2)	:-	2-long-and-2-short-sides(?obj1, ?two), 2-long-and-2-short-sides(?obj2, ?two).

Table E.4: Similarity-recognition rules.

Let *8-Antecedents* represent the following set of eight antecedents:
same-2-2(?obj1, ?obj2), same-shape(?obj1, ?obj2),
same-point-direction(?obj1,mkvobj2), same-pointy(?obj1, ?obj2),
same-area(?obj1, ?obj2), same-slanty-lines(?obj1, ?obj2),
same-tilted(?obj1, ?obj2), same-name(?obj1, ?obj2)

similarity(?obj1, ?obj2,very)	:-	n-true-antecedents(7, *8-Antecedents*).
similarity(?obj1, ?obj2,quite)	:-	n-true-antecedents(6, *8-Antecedents*).
similarity(?obj1, ?obj2,mostly)	:-	n-true-antecedents(5, *8-Antecedents*).
similarity(?obj1, ?obj2,fairly)	:-	n-true-antecedents(4, *8-Antecedents*).
similarity(?obj1, ?obj2,sort-of)	:-	n-true-antecedents(3, *8-Antecedents*).
similarity(?obj1, ?obj2,not-very)	:-	n-true-antecedents(2, *8-Antecedents*).

Table E.5: Combining rules.

most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,very), not similarity(?obj1, ?obj3,very), not similarity(?obj3, ?obj2,very).
most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,quite), not similarity(?obj1, ?obj3,quite), not similarity(?obj3, ?obj2,quite).
most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,mostly), not similarity(?obj1, ?obj3,mostly), not similarity(?obj3, ?obj2,mostly).
most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,fairly), not similarity(?obj1, ?obj3,fairly), not similarity(?obj3, ?obj2,fairly).
most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,sort-of), not similarity(?obj1, ?obj3,sort-of), not similarity(?obj3, ?obj2,sort-of).
most-similar-pair(?obj1, ?obj2)	:-	similarity(?obj1, ?obj2,not-very), not similarity(?obj1, ?obj3,not-very), not similarity(?obj3, ?obj2,not-very).

Table E.6: Features and their possible values.

Feature Name	*Possible Values*
Visual Features	
Tilted	{0 10 20 30 40}
Slanty	{Yes No}
Area	{Little Medium Big}
Shape	{Skinny Fat Medium}
Pointy	{Yes No}
Point Direction	{None Up Down Right Left}
2 long and 2 short sides	{Yes No}
Symbolic Features	
Convex	{Yes No}
Number of Sides	{3 4 5 6 8}
Number of Angles	{3 4 5 6 8}
Number of Right Angles	{0 1 2 3 4}
Number of Pairs of Parallel Sides	{0 1 2 3 4}
Number of Pairs of Equal Opposite Angles	{0 1 2 3 4}
Adajacent Angles Sum to 180	{Yes No}
Number of Pairs of Opposite Sides Equal	{0 1 2 3 4}
Number of Lines of Symmetry	{0 1 2 3 4 5 6 8}
All Sides Equal	{Yes No}
All Angles Equal	{Yes No}
Number of Equal Sides	{0 2 3 4 5 6 8}
Number of Equal Angles	{0 2 3 4 5 6 8}

Bibliography

[Aha91] Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Ahmad88] Ahmad, S. (1988). A study of scaling and generalization in neural networks. Technical Report CCSR-88-13, University of Illinois, Center for Complex Systems Research.

[Alberts88] Alberts, B. M. (1988). *Mapping and Sequencing the Human Genome.* National Academy Press, Washington, D.C.

[Atlas89] Atlas, L., Cole, R., Connor, J., El-Sharkawi, M., Marks II, R. J., Muthusamy, Y., and Barnard, E. (1989). Performance comparisons between backpropagation networks and classification trees on three real-world applications. In *Advances in Neural Information Processing Systems*, volume 2, pages 622–629, Denver, CO. Morgan Kaufmann.

[Barnard89] Barnard, E. and Cole, R. A. (1989). A neural-net training program based on conjugate-gradient optimization. Technical Report CSE 89-014, Oregon Graduate Institute, Beaverton, OR.

[Bennett88] Bennett, S. (1988). Real world EBL: Learning error tolerant plans in the robotics domain. In *Proceedings of the AAAI Explanation-Based Learning Symposium*, pages 122–126, Stanford, CA.

[Berenji91] Berenji, H. R. (1991). Refinement of approximate reasoning-based controllers by reinforcement learning. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 475–479, Chicago, IL.

[Blum88] Blum, A. and Rivest, R. L. (1988). Training a 3-node neural network is NP-complete. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 9–18, Cambridge, MA.

[Breiman84] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees.* Wadsworth, Belmont, CA.

[Brunak91] Brunak, S., Engelbrecht, J., and Knudsen, S. (1991). Prediction of human mRNA donor and acceptor sites from the DNA sequence. Available in the *neuroprose* archive at archive.cis.ohio-state.edu as brunak.netgene.ps.Z.

[Brunner56] Brunner, J. S., Goodnow, J. J., and Austin, G. A. (1956). *A Study of Thinking.* Wiley, New York.

[Buntine91] Buntine, W. L. and Weigand, A. S. (1991). Bayesian back-propagation.

[Chauvin88] Chauvin, Y. (1988). A back-propagation algorithm with optimal use of hidden units. In *Advances in Neural Information Processing Systems*, volume 1, pages 519–525, Denver, CO. Morgan Kaufmann.

[Cost90] Cost, S. and Salzberg, S. (1990). A weighted nearest neighbor algorith for learning with symbolic features. Technical Report JHU-90/11, Johns Hopkins, Baltimore, MD.

[Craven90] Craven, M. W. (1990). Experiments in optimal brain damage. Unpublished manuscript.

[Craven92] Craven, M. W. and Shavlik, J. W. (1992). Visualizing learning and computation in artificial neural networks. *International Journal on Artificial Intelligence Tools*, 1(2). (Forthcoming).

[Danyluk89] Danyluk, A. P. (1989). Finding new rules for incomplete theories: Explicit biases for induction with contextual information. In *Proceedings of the Sixth International Machine Learning Workshop*, pages 34–36, Ithaca, NY.

[DeJong86] DeJong, G. F. and Mooney, R. F. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176.

[Dennis91] Dennis, S. and Phillips, S. (1991). Analysis tools for neural networks.

[Diederich88] Diederich, J. (1988). Knowledge-intensive recruitment learning. Technical Report ICSI-TR-88-010, International Computer Science Institute, Berkeley, CA.

[Dietterich86] Dietterich, T. G. (1986). Learning at the knowledge level. *Machine Learning*, 1:287–316.

[Dietterich90] Dietterich, T. G., Hild, H., and Bakiri, G. (1990). A comparative study of ID3 and backpropagation for English text-to-speech mapping. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 24–31, Austin, TX.

[Duda79] Duda, R., Gaschnig, J., and Hart, P. (1979). Model design in the prospector consultant system. In Michie, D., editor, *Expert Systems in the Microelectronic Age*. Edinburgh University Press, Edinburgh, Scotland.

[Fahlman88] Fahlman, S. E. (1988). Faster learning variations on back-propagation: An empirical study. In Touretsky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51. Morgan Kaufmann, San Mateo, CA.

[Fahlman89] Fahlman, S. E. and Lebiere, C. (1989). The cascade-correlation learning architecture. In *Advances in Neural Information Processions Systems*, volume 2, pages 524–532, Denver, CO. Morgan Kaufmann.

[Fisher87] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

[Fisher89] Fisher, D. H. and McKusick, K. B. (1989). An empirical comparison of ID3 and back-propagation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 788–793, Detroit, MI.

[Flann89] Flann, N. S. and Dietterich, T. G. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4:187–226.

[Fozzard88] Fozzard, R., Bradshaw, G., and Ceci, L. (1988). A connectionist expert system that actually works. In *Advances in Neural Information Processing Systems*, volume 1, pages 248–255, Denver, CO. Morgan Kaufmann.

[Franzini87] Franzini, M. A. (1987). Speech recognition with back propagation. In *IEEE Ninth Annual Conference of the Engineering in Medicine and Biology Society*, pages 1702–1703.

[Fu89] Fu, L. M. (1989). Integration of neural heuristics into knowledge-based inference. *Connection Science*, 1:325–340.

[Fu91] Fu, L. M. (1991). Rule learning by searching on adapted nets. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 590–595, Anaheim, CA.

[Fuys88] Fuys, D., Geddes, D., and Tischler, R. (1988). The van Hiele model of thinking in geometry among adolescents. In *Journal for Research in Mathematics Education, Monograph No. 3.* National Council of Teachers of Mathematics, Reston, VA.

[Gallant88] Gallant, S. I. (1988). Connectionist expert systems. *Communications of the ACM*, 31:152–169.

[Goodman83] Goodman, N. (1983). *Fact, Fiction and Forecast.* Harvard University Press, Cambridge, MA.

[Hall88] Hall, R. J. (1988). Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning*, 3:45–77.

[Hanson90] Hanson, S. J. and Burr, D. J. (1990). What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences*, 13:471–518.

[Hanson88] Hanson, S. J. and Pratt, L. Y. (1988). Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems*, volume 1, pages 177–185, Denver, CO. Morgan Kaufmann.

[Harley87] Harley, C. B. and Reynolds, R. P. (1987). Analysis of *E. coli* promoter sequences. *Nucleic Acids Research*, 15:2343–2361.

[Hartigan75] Hartigan, J. A. (1975). *Clustering Algorithms.* Wiley, New York.

[Hawley83] Hawley, D. K. and McClure, W. R. (1983). Compilation and analysis of *Escherichia Coli* promoter DNA sequences. *Nucleic Acids Research*, 11:2237–2255.

[Hayashi90] Hayashi, Y. (1990). A neural expert system with automated extraction of fuzzy if-then rules. In *Advances in Neural Information Processing Systems*, volume 3, pages 578–584, Denver, CO. Morgan Kaufmann.

[Hebb49] Hebb, D. O. (1949). *The Organization of Behavior.* Wiley, New York.

[Hinton89] Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40:185–234.

[Hinton91] Hinton, G. E. (1991). Personal communication.

[Hinton86] Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition; Vol. 1: Foundations*, pages 77–109. MIT Press, Cambridge, MA.

[Hirsh89] Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. In *Proceedings of the Sixth International Machine Learning Workshop*, pages 29–33, Ithaca, NY.

[Hoehfeld91] Hoehfeld, M. and Fahlman, S. E. (1991). Learning with limited numerical precision using the cascade-correlation architecture. Technical Report CMU-CS-91-130, Carnegie-Mellon, Pittsburgh, PA.

[Holland86a] Holland, J. H. (1986a). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: An AI Approach*, volume 2, pages 593–624. Morgan Kaufmann, San Mateo, CA.

[Holland86b] Holland, J. H., Holyoak, R. J., Nisbitt, R. E., and Thagard, P. (1986b). *Induction: Processes of Inference, Learning and Discovery.* MIT Press, Cambridge, MA.

[Holley89] Holley, L. and Karplus, M. (1989). Protein secondary structure prediction with a neural network. *Proceedings of the National Academy of Science*, 56:152–156.

[Hollis90] Hollis, P. W., Harper, J. S., and Paulos, J. J. (1990). The effects of precision constraints in a backpropagation learning network. *Neural Computation*, 2:363–373.

[Holte89] Holte, R. C., Acker, L. E., and Porter, B. W. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–819, Detroit, MI.

[Honavar88] Honavar, V. and Uhr, L. (1988). A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In Hinton, G. E., Sejnowski, T. J., and Touretzky, D. S., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 472–484. Morgan Kaufmann, San Mateo, CA.

[Hunter91] Hunter, L. (1991). Classifying for prediction: A multistrategy approach to predicting protein secondary structure. In *Proceedings of the First International Workshop on Multistrategy Learning*, pages 394–402, Harpers Ferry, WV.

[IUB Nomenclature Committee85] IUB Nomenclature Committee (1985). Ambiguity codes. *European Journal of Biochemistry*, 150:1–5.

[Jones89] Jones, M. A. and Story, G. A. (1989). Inheritance reasoning in connectionist networks. In *International Conference on Neural Networks.*

[Judd88] Judd, S. (1988). On the complexity of loading shallow neural networks. *Journal of Complexity*, 4:177–192. Also appears in *Readings in Machine Learning.*

[Katz89] Katz, B. F. (1989). EBL and SBL: A neural network synthesis. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 683–689, Ann Arbor, MI.

[Kleene56] Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In Shannon, C. E. and McCarthy, J., editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ.

[Koedinger90] Koedinger, K. R. and Anderson, J. R. (1990). Theoretical and empirical motivations for the design of ANGLE: A new geometry learning environment. In *Proceedings of the AAAI Symposium on Knowledge-Based Environments for Learning and Teaching*, Stanford, CA.

[Kolen90] Kolen, J. F. and Pollack, J. B. (1990). Back-propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, volume 3, Denver, CO. Morgan Kaufmann.

[Kononenko91] Kononenko, I. and Bratko, I. (1991). Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80.

[Koudelka87] Koudelka, G. B., Harrison, S. C., and Ptashne, M. (1987). Effect of non-contacted bases on the affinity of 434 operator for 434 repressor and Cro. *Nature*, 326:886–888.

[Kruschke88] Kruschke, J. K. (1988). Creating local and distributed bottlenecks in hidden layers of back-propagation networks. In Hinton, G. E., Sejnowski, T. J., and Touretzky, D. S., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 357–370. Morgan Kaufmann, San Mateo, CA.

[Kruschke91] Kruschke, J. K. and Movellan, J. R. (1991). Benefits of gain: Speeded neural learning and minimal hidden layers in back-propagation networks. *IEEE Transactions on Systems, Man and Cybernetics*, 21(1).

[Langley89] Langley, P. (1989). Editorial: Toward a unified science of machine learning. *Machine Learning*, 3:253–259.

[Lapedes89] Lapedes, A., Barnes, C., Burkes, C., Farber, R., and Sirotkin, K. (1989). Application of neural networks and other machine learning algorithms to DNA sequence analysis. In *Computers and DNA, SFI Studies in the Science of Complexity VII.* Addison-Wesley, Reading, MA.

[Le Cun89] Le Cun, Y., Denker, J. S., and Solla, S. A. (1989). Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, Denver, CO. Morgan Kaufmann.

[Lebowitz86] Lebowitz, M. (1986). Integrated learning: Controlling explanation. *Cognitive Science*, 10:219–240.

[Lehrer89] Lehrer, R., Knight, W., Love, M., and Sancilio, L. (1989). Software to link action and description in pre-proof geometry. Presented at the Annual Meeting of the American Educational Research Association.

[Litzkow88] Litzkow, M., Livny, M., and Mutka, M. W. (1988). Condor — a hunter of idle workstations. In *Proceedings of the Eighth International Conference on Distributed Computing Systems.*

[Lukashin89] Lukashin, A. V., Anshelevich, V. V., Amirikyan, B. R., Gragerov, A. I., and Frank-Kamenetskii, M. D. (1989). Neural network models of promoter recognition. *Journal of Biomolecular Structure and Dynamics*, 6:1123–1133.

[Maclin91] Maclin, R. and Shavlik, J. W. (1991). Refining domain theories expressed as finite-state automata. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 524–528, Chicago, IL.

[Masuoka90] Masuoka, R., Watanabe, N., Kawamura, A., Owada, Y., and Asakawa, K. (1990). Neurofuzzy system — fuzzy inference using a structured neural network. In *Proceedings of the International Conference on Fuzzy Logic & Neural Networks*, pages 173–177.

[Matheus90] Matheus, C. J. (1990). *Feature Construction: An Analytic Framework and an Application to Decision Trees.* PhD thesis, University of Illinois at Urbana-Champaign.

[McCulloch43] McCulloch, W. S. and Pitts, W. A. (1943). A logical calculus of ideas immanent in nervous activity. *Bulliten of Mathematical Biophysics*, 5:115–133.

[McDermott82] McDermott, J. (1982). R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 19.

[McMillan91] McMillan, C., Mozer, M. C., and Smolensky, P. (1991). The connectionist scientist game: Rule extraction and refinement in a neural network. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL.

[Michalski83] Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161.

[Miller56] Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

[Minsky63] Minsky, M. (1963). Steps towards artificial intelligence. In Figenbaum, E. A. and Feldman, J., editors, *Computers and Thought.* McGraw-Hill, New York.

[Minsky88] Minsky, M. L. and Papert, S. (1988). *Perceptrons: Expanded Edition.* MIT Press, Cambridge, MA. Original edition published in 1969.

[Mitchell82] Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18:203–226.

[Mitchell86] Mitchell, T. M., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80.

[Moody88] Moody, J. and Darken, C. (1988). Learning with localized receptive fields. In Hinton, G. E., Sejnowski, T. J., and Touretzky, D. S., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 133–143. Morgan Kaufmann, San Mateo, CA.

[Mooney89] Mooney, R. J. and Ourston, D. (1989). Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 5–8, Ithaca, NY.

[Mooney91a] Mooney, R. J. and Ourston, D. (1991a). Personal communication.

[Mooney91b] Mooney, R. J. and Ourston, D. (1991b). Constructive induction in theory refinement. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 178–182, Evanston, IL.

[Mozer88] Mozer, M. C. and Smolensky, P. (1988). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, volume 1, pages 107–115, Denver, CO. Morgan Kaufmann.

[Murphy85] Murphy, G. L. and Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, 91:289–316.

[Murphy91] Murphy, P. M. and Pazzani, M. J. (1991). ID2-of-3: Constructive induction of N-of-M concepts for discriminators in decision trees. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 183–187, Evanston, IL.

[Nessier62] Nessier, U. and Weene, P. (1962). Hierarchies in concept attainment. *Journal of Experimental Psychology*, 64:640–645.

[Ng90] Ng, K. and Lippmann, R. P. (1990). A comparative study of the practical characteristics of neural networks and conventional pattern classifiers. In *Advances in Neural Information Processing Systems*, volume 3, pages 970–976, Denver, CO. Morgan Kaufmann.

[Noordewier91] Noordewier, M. O., Towell, G. G., and Shavlik, J. W. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems*, volume 3, Denver, CO. Morgan Kaufmann.

[Nowlan91] Nowlan, S. J. and Hinton, G. E. (1991). Simplifying neural networks by soft weight-sharing. In *Advances in Neural Information Processing Systems*, volume 4, Denver, CO. Morgan Kaufmann.

[Oliver88] Oliver, W. L. and Schneider, W. (1988). Using rules and task division to augment connectionist learning. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 55–61, Montreal, Canada.

[O'Neill89] O'Neill, M. C. (1989). *Escherichia coli* promoters: I. Consensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organzation. *Journal of Biological Chemistry*, 264:5522–5530.

[O'Neill89] O'Neill, M. C. and Chiafari, F. (1989). *Eserichia Coli* promoters II: A spacing-class dependent promoter search protocol. *Journal of Biological Chemistry*, 264:5531–5534.

[O'Rorke82] O'Rorke, P. (1982). A comparative study of inductive learning systems AQ15 and ID3 using a chess endgame test problem. Technical Report UIUCDCS-F-82-899, University of Illinois, Department of Computer Science, Urbana, IL.

[Ourston90] Ourston, D. and Mooney, R. J. (1990). Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 815–820, Boston, MA.

[Ourston91] Ourston, D. and Mooney, R. J. (1991). Improving shared rules in multiple category domain theories. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 534–538, Evanston, IL.

[Pazzani88] Pazzani, M. J. (1988). *Learning Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods.* PhD thesis, Computer Science Department, University of California, Los Angles.

[Pazzani89] Pazzani, M. J. and D, S. (1989). The influence of prior theories on the ease of concept acquisition. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 844–851, Ann Arbor, MI.

[Pineda87] Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physics Review Letters*, 59:2229–2232.

[Pratt91] Pratt, L. Y. and Kamm, C. A. (1991). Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 584–589, Anaheim, CA.

[Qian88] Qian, N. and Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins unsing neural networks models. *Journal of Molecular Biology*, 202:856–884.

[Quinlan86] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[Quinlan91] Quinlan, J. R. (1991). Improved estimates for the accuracy of small disjuncts. *Machine Learning*, 6:93–98.

[Rajamoney87] Rajamoney, S. A. and DeJong, G. F. (1987). The classification, detection and handling of imperfect theory problems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 205–207, Milan, Italy.

[Record89] Record, T. (1989). Personal communication.

[Redmond89] Redmond, M. (1989). Combining case-based reasoning, explanation-based learning, and learning from instruction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 20–22, Ithaca, NY.

[Rendell90] Rendell, L. and Cho, H. (1990). Empirical learning as a function of concept character. *Machine Learing*, 5:267–298.

[Rendell89] Rendell, L. A., Cho, H. H., and Seshu, R. (1989). Improving the design of similarity-based rule-learning systems. *International Journal of Expert Systems*, 2:97–133.

[Rosenblatt62] Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan, New York.

[Rueckl88] Rueckl, J. G., Cave, K. R., and Kosslyn, S. M. (1988). Why are "what" and "where" processed by separate cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience*, 1(2).

[Rumelhart91] Rumelhart, D. (1991). Personal communication.

[Rumelhart86] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, pages 318–363. MIT Press, Cambridge, MA.

[Saito88] Saito, K. and Nakano, R. (1988). Medical diagnostic expert system based on PDP model. In *Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 255–262.

[Schank86] Schank, R., Collins, G. C., and Hunter, L. E. (1986). Transcending inductive category formation in learning. *Behavioral and Brain Sciences*, 9:639–686.

[Scott91] Scott, G. M., Shavlik, J. W., and Ray, W. H. (1991). Refining PID controllers using neural networks. In *Advances in Neural Information Processing Systems*, volume 4, Denver, CO. Morgan Kaufmann.

[Sejnowski87] Sejnowski, T. J. and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

[Sestito90] Sestito, S. and Dillon, T. (1990). Using multi-layered neural networks for learning symbolic knowledge. In *Proceedings of the 1990 Australian Artificial Intelligence Conference*, Perth, Australia.

[Shavlik90a] Shavlik, J. W. (1990a). Case-based reasoning with noisy case boundaries: An application in molecular biology. Technical Report 988, University of Wisconsin, Madison, WI.

[Shavlik90b] Shavlik, J. W. (1990b). *Extending Explanation-Based Learning by Generalizing Explanation Structures.* Pitman, London.

[Shavlik91] Shavlik, J. W., Mooney, R. J., and Towell, G. G. (1991). Symbolic and neural net learning algorithms: An empirical comparison. *Machine Learning*, 6:111–143.

[Shavlik89] Shavlik, J. W. and Towell, G. G. (1989). An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1:233–255.

[Shortliffe84] Shortliffe, E. H. and Buchanan, B. G. (1984). A model of inexact reasoning in medicine. In Buchanan, B. G. and Shortliffe, E. H., editors, *Rule-Based Expert Systems*, pages 233–262. Addison-Wesley, Reading, MA.

[Smolensky87] Smolensky, P. (1987). On variable binding and the representation of symbolic structures in connectionist systems. Technical Report CU-CS-355-87, University of Colorado - Boulder.

[Smolensky88] Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–23.

[Sobel87] Sobel, M. A., editor (1987). *Mathematics.* McGraw-Hill, New York.

[Squires91] Squires, C. S. and Shavlik, J. W. (1991). Experimental analysis of aspects of the cascade-correlation learning architecture. Machine Learning Research Group Working Paper 91-1. University of Wisonsin – Madison.

[Stormo90] Stormo, G. D. (1990). Consensus patterns in DNA. In *Methods in Enzymology*, volume 183, pages 211–221. Academic Press, Orlando, FL.

[Tesauro89] Tesauro, G. and Sejnowski, T. J. (1989). A parallel network that learns to play backgammon. *Artificial Intelligence*, 39:357–390.

[Thompson91] Thompson, K., Langley, P., and Iba, W. (1991). Using background knowledge in concept formation. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 554–558, Evanston, IL.

[Touretsky88] Touretsky, D. S. and Hinton, G. E. (1988). A distributed connectionist production system. *Cognitive Science*, 12:423–466.

[Towell91] Towell, G. G. and Shavlik, J. W. (1991). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In *Advances in Neural Information Processing Systems*, volume 4, Denver, CO. Morgan Kaufmann.

[Towell90] Towell, G. G., Shavlik, J. W., and Noordewier, M. O. (1990). Refinement of approximately correct domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA.

[Tsoi90] Tsoi, A. C. and Pearson, R. A. (1990). Comparison of three classification techniques, CART, C4.5, and multi-layer perceptrons. In *Advances in Neural Information Processing Systems*, volume 3, pages 963–969, Denver, CO. Morgan Kaufmann.

[van Hiele86] van Hiele, P. M. (1986). *Structure and Insight.* Academic Press, New York.

[van Hiele-Geldof57] van Hiele-Geldof, D. (1957). *De didaktick van de Meetkunde in de eerste klass van het (The Didactics of Geometry in the Lowest Class of Secondary School).* PhD thesis, University of Utretch. English translation by M. Verdonck.

[von Heijne87] von Heijne, G. (1987). *Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit.* Academic Press, San Deigo, CA.

[Wantanabe69] Wantanabe, S. (1969). *Knowing and Guessing: A Formal and Quantatative Study.* Wiley, New York.

[Waterman86] Waterman, D. A. (1986). *A Guide to Expert Systems.* Addison Wesley, Reading, MA.

[Watson87] Watson, J. D., Hopkins, H. H., Roberts, J. W., Steitz, J. A., and Weiner, A. M. (1987). *The Molecular Biology of the Gene.* Benjamin-Cummings, Menlo Park, CA.

[Wattenmaker87] Wattenmaker, W. D., Nakamura, G. L., and Medin, D. L. (1987). Relationships between similarity-based and explanation-based categorization. In Hilton, D., editor, *Contemporary Science and Natural Explanations: Common Sense Concepts of Causality*, pages 205–41. Harvester Press, Sussex, England.

[Weigand90] Weigand, A. S., Rumelhart, D. E., and Huberman, B. A. (1990). Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems*, volume 3, pages 875–882, Denver, CO. Morgan Kaufmann.

[Weiss89] Weiss, S. M. and Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 688–693, Detroit, MI.

[Weiss90] Weiss, S. M. and Kulikowski, C. A. (1990). *Computer Systems that Learn.* Morgan Kaufmann, San Mateo, CA.

[Wejchert89] Wejchert, J. and Tesauro, G. (1989). Neural network visualization. In *Advances in Neural Information Processing Systems*, volume 2, pages 465–472, Denver, CO. Morgan Kaufmann.

[Whewell89] Whewell, W. (1989). *Theory of the Scientific Method.* Hackett, Indianapolis. Originally published in 1840.

[Wieland87] Wieland, A. and Leighton, R. (1987). Geometric analysis of neural network capabilities. In *Neural Networks Conference*, San Diego, CA.

[Wisniewski89] Wisniewski, E. J. (1989). Learning from examples: The effect of different conceptual roles. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 844–851, Ann Arbor, MI.

[Wisniewski91] Wisniewski, E. J. and Medin, D. L. (1991). Is it a pucket or a purse? Tightly coupled theory and data driven learning. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 564–569, Evanston, IL.

[Zadeh83] Zadeh, L. A. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11:199–227.