Efficient Algorithms for Approximate Reasoning

P.M. van den Broek
Department of Computer Science,
University of Twente,
P.O.Box 217, 7500 AE Enschede,
the Netherlands
pimvdb@cs.utwente.nl

Abstract

We present algorithms for approximate reasoning computations, for some intersection and implication functions, which are as efficient as Mamdani’s interpolation method. Implementations of the algorithms are given in the functional language Miranda.

1. Introduction

In fuzzy reasoning with generalised modus ponens, one starts with a rule and a fact, and obtains a conclusion:

Rule     If X=A then Y=B
Fact     X=A’

----------------------------------------------

Conclusion    Y=B’

Here A and A’ are fuzzy sets on a universe U1, and B and B’ are fuzzy sets on a universe U2. We consider here the case where both U1 and U2 are finite sets. There are two approaches to define the fuzzy set B’ in terms of the fuzzy sets A, A’ and B. In the first approach, called approximate reasoning (cf Klir and Yuan [1], chapter 11), one chooses an intersection operator I and an implication operator J and defines B’ by

\[ B'(y) = \sup_x (I(A'(x),J(A(x),B(y)))) \]  \(1\)

In the second approach, called the interpolation method, which is originally due to Mamdani and Assilian [2], B’ is defined by

\[ B'(y) = \sup_x (\min(A'(x),\min(A(x),B(y)))) \]  \(2\)

Note the similarity between the two definitions of B’. However, eq. (2) is not a special instance of eq. (1), since the minimum operator is not an implication operator. In practice, the second approach seems to be used more often than the first approach, which, to our opinion, might be due to the fact that the second approach allows an efficient implementation, since eq. (2) can be written as

\[ B'(y) = \sup_x (\min(B(y),\min(A(x),A'(x)))) \]  \(3\)

Let n and m denote the cardinalities of the universes U1 and U2 respectively, then it is seen that the algorithms which are expressed by the eqs. (1) and (2) have complexity O(n*m), whereas the algorithm expressed by eq. (3) has complexity O(n+m).

The aim of this paper is to show that in the first approach there also exists, for some intersection and implication operators, an algorithm for the computation of B’ which has complexity O(n+m). The intersection and implication operators which we will treat, in the sections 2-5 respectively, are:

1. Kleene Dienes implication, standard intersection
2. Early Zadeh implication, standard intersection
3. Willmott implication, standard intersection
4. Kleene Dienes implication, bounded difference intersection

The algorithms which we derive are implemented in the functional language Miranda; full details are given in section 6. Their efficiency is measured by means of a small example.

2. Kleene Dienes implication, standard intersection

In this case equation (1) reads

\[ B'(y) = \sup_x (\min(A'(x),\max(1-A(x),B(y)))) \]  \(4\)

We have the following result: the definition of B’ in eq. (4) is equivalent to the following definition:
\[ B'(y) = p, \text{ if } B(y) \leq p \quad (5a) \]
\[ B'(y) = q, \text{ if } B(y) \geq q \quad (5b) \]
\[ B'(y) = B(y), \text{ if } p < B(y) < q \quad (5c) \]

where \( p \) and \( q \) are defined by
\[ p = \sup_x \min(A'(x), 1 - A(x)) \quad (6) \]
\[ q = \sup_x A'(x) \quad (7) \]

Proof: From eq. (4) we will prove eq. (5), and consider in turn the cases where \( B(y) \leq p \), \( B(y) \geq q \) and \( p < B(y) < q \).

Case 1: \( B(y) \leq p \); to be proved: \( B'(y) = p \).

From eq. (6) it follows that for all \( x \): \( \min(A'(x), 1 - A(x)) \leq p \). So either \( A'(x) \leq p \) or \( 1 - A(x) \leq p \). In the case where \( A'(x) \leq p \) we have:
\[ \min(A'(x), \max(1 - A(x), B(y))) \leq p. \quad (8) \]

In the case where \( 1 - A(x) \leq p \), we have \( \max(1 - A(x), B(y)) \leq p \), and so eq. (8) is valid in this case also. So eq. (8) holds for all \( x \), and we obtain, by taking the supremum over all \( x \), from eq. (4): \( B'(y) \leq p \).

On the other hand, from eq. (4) it follows, since \( B(y) \geq 0 \) and the functions \( \min \) and \( \max \) are nondecreasing in both arguments, that \( B'(y) \geq p \). So we obtain \( B'(y) = p \).

Case 2: \( B(y) \geq q \); to be proved: \( B'(y) = q \).

From eq. (7) it follows that \( A'(x) \leq q \) for all \( x \). So we have: \( B(y) \geq A'(x) \). Therefore \( \max(1 - A(x), B(y)) \geq A'(x) \) and so
\[ \min(A'(x), \max(1 - A(x), B(y))) \geq A'(x) \quad (9) \]

Since eq. (9) holds for all \( x \), we obtain, by taking the supremum over all \( x \), from eq. (4): \( B'(y) = q \).

Case 3: \( p < B(y) < q \); to be proved: \( B'(y) = B(y) \).

From \( p < B(y) \) and eq. (6) it follows that for each \( x \) we have: \( \min(A'(x), 1 - A(x)) < B(y) \). So either \( A'(x) < B(y) \) or \( 1 - A(x) < B(y) \). In the case where \( A'(x) < B(y) \) we have:
\[ \min(A'(x), \max(1 - A(x), B(y))) \leq B(y) \quad (10) \]

In the case where \( 1 - A(x) < B(y) \) we have max \( (1 - A(x), B(y)) \leq B(y) \), and so eq. (10) is valid in this case also. So eq. (10) holds for all \( x \), and we obtain, by taking the supremum over all \( x \): \( B'(y) \leq B(y) \).

On the other hand, from eq. (7) it follows, since \( B(y) \geq 0 \) and the functions \( \min \) and \( \max \) are nondecreasing in both arguments, that \( B'(y) \geq B(y) \). So we obtain \( B'(y) = B(y) \).

3. Early Zadeh implication, standard intersection

In this case equation (1) reads
\[ B'(y) = \sup_x \min(A'(x), \max(1 - A(x), \min(A(x), B(y)))) \quad (11) \]

We have the following result: the definition of \( B' \) in eq. (11) is equivalent to the following definition:
\[ B'(y) = p, \text{ if } B(y) \leq p \quad (12a) \]
\[ B'(y) = q, \text{ if } B(y) \geq q \quad (12b) \]
\[ B'(y) = B(y), \text{ if } p < B(y) < q \quad (12c) \]

where \( p \) and \( q \) are defined by
\[ p = \sup_x \min(A'(x), 1 - A(x)) \quad (13) \]
\[ q = \sup_x \min(A'(x), \max(1 - A(x), A(x))) \quad (14) \]

Proof: From eq. (11) we will prove eq. (12), and consider in turn the cases where \( B(y) \leq p \), \( B(y) \geq q \) and \( p < B(y) < q \).

Case 1: \( B(y) \leq p \); to be proved: \( B'(y) = p \).

From eq. (13) it follows that for all \( x \): \( \min(A'(x), 1 - A(x)) \leq p \). So either \( A'(x) \leq p \) or \( 1 - A(x) \leq p \). In the case where \( A'(x) \leq p \) we have:
\[ \min(A'(x), \max(1 - A(x), B(y))) \leq B(y) \quad (15) \]

In the case where \( 1 - A(x) \leq p \), we have max \( (1 - A(x), B(y)) \leq p \), and so eq. (15) is valid in this case also. So eq. (15) holds for all \( x \), and we
obtain, by taking the supremum over all \(x\), from eq. (11): \(B'(y) \leq p\).
On the other hand, from eq. (11) it follows, since \(B(y) \geq 0\) and the functions min and max are nondecreasing in both arguments, that \(B'(y) \geq p\). So we obtain \(B'(y) = p\).

**Case 2 :** \(B(y) \geq q\); to be proved: \(B'(y) = q\).

We will show that in this case the equation

\[
\min(A'(x), \max(1-A(x), \min(A(x), B(y)))) = \min(A'(x), \max(1-A(x), A(x)))
\]

holds for all \(x\). Then we obtain \(B'(y) = q\) by taking the supremum on both sides.

It is clear that eq. (16) holds if \(\min(A(x), B(y)) = A(x)\), i.e. if \(A(x) \leq B(y)\). Therefore suppose that \(A(x) > B(y)\). From \(B(y) \geq q\) and eq. (14) it follows that \(B(y) \geq \min(A'(x), \max(1-A(x), A(x)))\). Since \(\max(1-A(x), A(x)) > B(y)\) it follows that \(A'(x) \leq B(y)\). From \(A'(x) \leq B(y) < A(x)\) we deduce that both sides of eq. (16) are equal to \(A'(x)\).

**Case 3 :** \(p < B(y) < q\); to be proved: \(B'(y) = B(y)\).

From \(p < B(y)\) ans eq. (13) it follows that for each \(x\) we have: \(\min(A'(x), 1-A(x)) < B(y)\). So either \(A'(x) < B(y)\) or \(1-A(x) < B(y)\). In the case where \(A'(x) < B(y)\) we have

\[
\min(A'(x), \max(1-A(x), \min(A(x), B(y)))) \leq B(y)
\]

In the case where \(1-A(x) < B(y)\) we have

\[
\min(A'(x), \max(1-A(x), \min(A(x), B(y)))) = B(y)
\]

Since \(B'(y)\) is the supremum over all \(x\) of the left hand-side, we obtain \(B'(y) \geq B(y)\). Thus \(B'(y) = B(y)\).

4. **Willmott implication, standard intersection**

In this case equation (1) reads

\[
B'(y) = \sup_x \min(A'(x), \min(\max(1-A(x), \min(A(x), B(y))), \max(A(x), 1-A(x)), \max(B(y), 1-B(y))))
\]

We have the following result: the definition of \(B'\) in eq. (19) is equivalent to the following definition:

\[
\begin{align*}
B'(y) & = \min(r, p), \text{ if } B(y) \leq p \\
B'(y) & = \min(r, q), \text{ if } B(y) \geq q \\
B'(y) & = \min(r, B(y)), \text{ if } p < B(y) < q
\end{align*}
\]

where \(p\), \(q\) and \(r\) are defined by

\[
\begin{align*}
p & = \sup_x \min(A'(x), 1-A(x)) \\
q & = \sup_x \min(A'(x), \max(1-A(x), A(x))) \\
r & = \max(B(y), 1-B(y))
\end{align*}
\]

Proof: From eq. (19) we will prove eq. (20). It is straightforward to show from eq. (19) that

\[
B'(y) = \min(r, \sup_x \min(A'(x), \\max(1-A(x), A(x))), \\max(B(y), 1-B(y)))
\]

Comparing eq. (24) with eq. (11) we find that \(B'(y)\) in this case is the minimum of \(r\) and the value of \(B'(y)\) in the case of the Early-Zadeh implication. So eq. (20) follows from eq. (12).

5. **Kleene Dienes implication, bounded difference intersection**

In this case equation (1) reads

\[
B'(y) = \sup_x (0, A'(x) - 1 + \max(1-A(x), B(y)))
\]

We have the following result: the definition of \(B'\) in eq. (25) is equivalent to the following definition:
\[ B'(y) = p, \text{ if } B(y) \leq 1+p-q \]  
\[ B'(y) = B(y)+q-1, \text{ if } B(y) > 1+p-q \]  

where \( p \) and \( q \) are defined by

\[ p = \max \left( 0, \sup_x (A'(x)-A(x)) \right) \]  
\[ q = \sup_x A'(x), \]  

(27)  

(28)

Proof: From eq. (25) we will prove eq. (26). Eq. (25) can be rewritten as \( B'(y) = \max (0, \sup_x (A'(x)-1 + \max(1-A(x),B(y)))) \) = \( \max (0, \sup_x (A'(x)-A(x),A'(x)-1+B(y)))) \) = \( \max (0, \max(sup_x (A'(x)-A(x)),sup_x (A'(x)-1+B(y)))) \) = \( \max (p,q-1+B(y)). \) This implies the truth of eq. (26).

6. Implementations

For all four cases we give implementations of the straightforward algorithm and of the efficient algorithm in the functional programming language Miranda (Turner [3]), the evaluation result for some small example input, and the number of reductions performed during the evaluation, which is a measure of the efficiency of the algorithms. For comparison we also treat the method of interpolation in the same way.

In our small example, both universes U1 and U2 contain 11 elements. The functions A, A' and B are given by the lists \( a, a' \) and \( b \) (in Miranda notation) of their function values:

\[ a = [0,0.25,0.5,0.75,1,0.75,0.5,0.25,0,0,0] \]  
\[ a' = [0,0,0,0.25,0.5,0.75,1,0.75,0.5,0.25,0] \]  
\[ b = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1] \]

The implementation of the algorithms are functions which take lists like these as arguments, and return the corresponding list of function values of \( B' \). The result of the application of these functions to the lists given above is given for all cases, as well as the number of reductions of the computations.

**Kleene Dienes implication, standard intersection**

Straightforward implementation:

\[
\text{kl}_d \ a \ a' \ b = [\max [\min [p, \max [1-q, by]] | (p,q) \leftarrow \text{zip} (a',a)] | by\leftarrow b]
\]

Efficient implementation:

\[
\text{kl}_d' \ a \ a' \ b = [\text{kl}_d' by | by\leftarrow b]
\]

where

\[
\text{kl}_d' by = p, \text{ if } by \leq p
\]

\[
= by, \text{ if } by < q
\]

\[
= q, \text{ otherwise}
\]

\[
p = \max [\min [p,1-q] | (p,q) \leftarrow \text{zip} (a',a)]
\]

\[
q = \max [\min [p, max [1-q,q]] | (p,q) \leftarrow \text{zip} (a',a)]
\]

Result:

\[ [0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.6,0.7,0.8,0.9,1] \]

Number of reductions needed by \( \text{kl}_d \): 3965

Number of reductions needed by \( \text{kl}_d' \): 462

**Early Zadeh implication, standard intersection**

Straightforward implementation:

\[
\text{ez} \ a \ a' \ b = [\max [\min [p, \max [1-q, min[q,by]]]] | (p,q) \leftarrow \text{zip} (a',a)] | by\leftarrow b]
\]

Efficient implementation:

\[
\text{ez}' \ a \ a' \ b = [\text{ez}' by | by\leftarrow b]
\]

where

\[
\text{ez}' by = p, \text{ if } by \leq p
\]

\[
= by, \text{ if } by < q
\]

\[
= q, \text{ otherwise}
\]

\[
p = \max [\min [p,1-q] | (p,q) \leftarrow \text{zip} (a',a)]
\]

\[
q = \max [\min [p, max [1-q,q]] | (p,q) \leftarrow \text{zip} (a',a)]
\]

Result:

\[ [0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75] \]
Number of reductions needed by \( ez \): 6867
Number of reductions needed by \( ez2 \): 1046

Willmott implication, standard intersection

Straightforward implementation:

\[
\text{willmott} \ a \ a' \ b = \left[ \max \left( \min \left( \max \right. \left( 1 - q, by \right), \max \left( q, 1 - q \right), \max \left( by, 1 - by \right) \right) \right) \left| \left( p, q \right) \leftarrow \text{zip} \ (a', a) \right) \right| \text{by} \leftarrow b \right]
\]

Efficient implementation:

\[
\text{willmott} \ a \ a' \ b = \left[ \text{willmott} \ a' \ b \right] \left. \right| \text{by} \leftarrow b
\]

Result:

\[0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75\]

Number of reductions needed by \( \text{willmott} \): 9854
Number of reductions needed by \( \text{willmott} \) \(2\): 1421

Kleene Dienes implication, bounded difference intersection

Straightforward implementation:

\[
\text{kld} \ a \ a' \ b = \left[ \max \left( 0 : \left[ -w \right] \left( v, w \right) \left| \left( p, q \right) \leftarrow \text{zip} \ (a', a) \right) \right) \right| \text{by} \leftarrow b
\]

Efficient implementation:

\[
\text{kld} \ a \ a' \ b = \left[ \text{kld} \ a' \ b \right] \left. \right| \text{by} \leftarrow b
\]

Result:

\[0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75, 0.75\]

Number of reductions needed by \( \text{kld} \): 5044
Number of reductions needed by \( \text{kld} \) \(2\): 635

Method of interpolation

Straightforward implementation (eq. (2)):

\[
\text{mam} \ a \ a' \ b = \left[ \max \left( \min \left( \max \right. \left( p, \min \right. \left( q, by \right) \right) \right) \left| \left( p, q \right) \leftarrow \text{zip} \ (a', a) \right) \right) \left| \text{by} \leftarrow b \right. \right]
\]

Efficient implementation (eq. (3)):

\[
\text{mam} \ a \ a' \ b = \left[ \text{mam} \ a' \ b \right] \left. \right| \text{by} \leftarrow b
\]

Result:

\[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.75, 0.75\]

Number of reductions needed by \( \text{mam} \): 5056
Number of reductions needed by \( \text{mam} \) \(2\): 496

7. Conclusion

For several combinations of intersection and implication operators we have derived algorithms for the calculation of inference results in approximate reasoning. Their asymptotical complexity is equal to the asymptotical complexity of the interpolation method. Complete implementations have been given, which show that also in the case of a small example the efficiency of the algorithms
is comparable to the efficiency of the interpolation method.

8. References

