1 Aggregators

1.1 Belief Measure Aggregation

Let $p, q \in [0, 1]$ be a probability, possibility, belief measure.

- Minimum/maximum: $\min\{p, q\}, \max\{p, q\}$
- Lukasiewicz: $L_{AND}(p, q) = \max\{p + q - 1, 0\}, L_{OR}(p, q) = \min\{p + q, 1\}$
- Probabilistic: $Pr_{AND}(p, q) = pq, Pr_{OR}(p, q) = p + q - pq$
- Weak: $W(p, q) = \min\{p, q\}$ if $\max\{p, q\} = 1$; 0 otherwise
- Strong: $S(p, q) = \max\{p, q\}$ if $\min\{p, q\} = 0$; 1 otherwise
- Hamacher (AND): $H^{\gamma}_{AND}(p, q) = \frac{pq}{\gamma+(1-\gamma)(p+q-pq)}, \gamma \geq 0$
- Hamacher (OR): $H^{\gamma}_{OR}(p, q) = \frac{p+q-(2-\gamma)pq}{1-(1-\gamma)pq}, \gamma \geq 0$
- Yager (AND): $Y^{\gamma}_{AND} = 1 - \min\{1, \sqrt[\gamma]{(1-p)^{\gamma} + (1-q)^{\gamma}}\}$
- Yager (OR): $Y^{\gamma}_{OR} = \min\{1, \sqrt[\gamma]{p^{\gamma} + q^{\gamma}}\}$

1.2 Ordered Weighted Averaging Operators

An OWA operator is the mapping

$$F(x_1, \ldots, x_n) = \sum_{k=1}^{n} w_k a_k$$

where $W = \{w_1, \ldots, w_n\}$ are weights lying in the unit interval and summing to one and a_k is the k-th largest of the x_k. Notable OWA operators are:

- If $w_1 = 1, w_k = 0, k \neq 1$, then $F(x_1, \ldots, x_n) = \max_{k=1,\ldots,n}\{x_1, \ldots, x_n\}$
- If $w_n = 1, w_k = 0, k \neq n$, then $F(x_1, \ldots, x_n) = \min_{k=1,\ldots,n}\{x_1, \ldots, x_n\}$
- If $w_k = \frac{1}{n}, \forall k$, then $F(x_1, \ldots, x_n) = \frac{x_1+\cdots+x_n}{n}$ (mean value)

Example: Assume $W = [0.4, 0.3, 0.2, 0.1]$. Then $F(0.7, 1, 0.2, 0.6) = 0.4 \cdot 1 + 0.3 \cdot 0.7 + 0.2 \cdot 0.6 + 0.1 \cdot 0.2 = 0.75$.

A fundamental aspect of the OWA operator is the re-ordering step, in particular an aggregate x_i is not associated with a particular weight w_i but rather a weight is associated with a particular *ordered* position of aggregate. When we view the OWA weights as a column vector we shall find it convenient to refer to the weights with the low indices as weights at the top and those with the higher indices with weights at the bottom.

1.2.1 Window type OWA

A window type OWA operator takes the average of the m arguments about the center. For this class of operators we have:

- $w_i = 0, \text{ if } i < k$
- $w_i = \frac{1}{m}, \text{ if } k \leq i < k + m$
- $w_i = 0, \text{ if } i \geq k + m$

This operator takes the arithmetic mean of all but the best and the worst scores of an alternative.

Example: For example, let $m = 3$ and $k = 2$. Then the weights of this window type OWA operator are calculated as $w_1 = 0, w_2 = w_3 = w_4 = 1/3, w_5 = 0$.

2 Probabilistic Combination of Multiple Evidence

2.1 Dempster-Shafer Theory

The Dempster-Shafer theory (DS) can be viewed as a method for reasoning under epistemic uncertainty. Reasoning under epistemic uncertainty refers to logically arriving at decisions based on available knowledge. The most important part of this theory is Dempster's rule of combination which combines evidence from two or more sources to form inferences.

Consider k experts and a set Θ of subsets of evidences $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$. Expert i, i.e., sensor i, contributes its observation by assigning its beliefs over Θ. This assignment function is called the 'probability mass function' of the sensor S_i, denoted by $m_i \in [0, 1]$. For each possible piece of evidence θ_m DS theory gives a rule of combining sensor S_i observation m_i and sensor S_j observation m_j:

$$m_i \oplus m_j(\theta_m) = \frac{\sum_{\theta_a \cap \theta_b = \theta_m} m_i(\theta_a) m_j(\theta_b)}{1 - \sum_{\theta_a \cap \theta_b = \emptyset} m_i(\theta_a) m_j(\theta_b)} \tag{1}$$

This combining rule can be generalized by iteration: we treat m_j not as sensor S_j observation, but rather as the already combined (using DS combining rule) observation of sensor S_x and sensor S_y.

2.2 Linear & Symmetric Opinion Pool

The linear opinion pool is the most common way of combining the probabilities of different n agents/experts/sensors to produce a social probability. For a set of probability measures $\{p_i\}_{i=1}^n$, say that a probability measure p_0 is a linear opinion pool with respect to $\{p_i\}_{i=1}^n$ if there exists some $\lambda \in \mathbb{R}^n$, such that $\sum_{i=1}^n \lambda_i = 1$ and thus

$$p_0 = \sum_{i=1}^n \lambda_i p_i \tag{2}$$

Perhaps the most commonly used linear opinion pool is the one in which each experts probability assessment is weighted equally. Motivating this on representational form, we would claim that there is no reason to favour one expert over another, so that their opinions should be treated equally. A probability measure p_0 is a symmetric linear opinion pool with respect to $\{p_i\}_{i=1}^n$ if

$$p_0 = \sum_{i=1}^{n} \frac{1}{n} p_i \tag{3}$$

3 Voting Algorithms

3.1 Voting on aggregation level

Consider a conjunction of n arithmetic conditions of the form $x_i \bowtie_i \theta_i$, $i = 1, \ldots, n$ and $\bowtie_i$ is operand. Then,

$$f(n) = x_1 \bowtie_1 \theta_1 \wedge \ldots x_n \wedge \bowtie_n \theta_n$$

We quantify the conjunction $f(n)$ through either *hard* or *soft* quantifiers.

3.2 Hard Conjunction Quantifier

In this case, the conjunction $f(n) \in \{0, 1\}$. Specifically, $f(n) = 0$ iff there is a condition $i, 1 \leq i, \leq n$, such that the term $x_i \not\bowtie_i \theta_i$ holds true. Once all n conditional terms in $f(n)$ hold true, i.e., $x_i \bowtie_i \theta_i, \forall i$, then $f(n) = 1$.

Example $f(2) = x_1 > 10 \wedge x_2 \leq 2$. If $x_1 = 12$ and $x_2 = 1$ then $f(2) = 1$. If $x_1 = 10$ and $x_2 = 1$ then the first terms does not hold true, thus, $f(2) = 0$.

3.3 Soft Conjunction Quantifier

In this case, we can quantify each conditional term with a number in $\{0\} \cup [z, 1]$, with $z \in (0, 1)$. The z parameter is the *base* of the decision between holds-true and holds-false. In our case we can simple assume that $z = 0.5$, thus, a quantification $f(n) \leq z$ indicates $f(n) = 0$. We present how to quantify a quantification $f(n) > z$. Specifically, consider the term $x_i \bowtie_i \theta_i$. The soft quantification of this term is the weighted support of the distance of x_i from the corresponding threshold θ_i w.r.t. z. Hence, in case where $z = 0.5$ we have that the soft quantification of the considered term is

$$f_i = z + z\frac{g(v_i, \theta_i)}{\theta_i}$$

where v_i is the current value of the x_i variable and the function $g(u, v) = u - v$ if $\bowtie_i \in \{\geqslant, >\}$ or else $g(u, v) = v - u$. The g function denotes a linear soft quantification. Hence, if $\ell = \min\{f_i, f_2, \ldots, f_n\}$, then

$$f(n) = \ell, \text{ if } \ell \geq z$$

or

$$f(n) = 0, \text{ if } \ell < z$$

Example $f(2) = x_1 > 10 \wedge x_2 \leq 2$. If $x_1 = 12$ and $x_2 = 1$ then, $f_1 = 0.5 + 0.5\frac{12-10}{10} = 0.5 + 0.1 = 0.6$ and $f_2 = 0.5 + 0.5\frac{3-1}{3} = 0.5 + 0.33 = 0.88$. Hence, $\ell = \min\{0.6, 0.88\} = 0.6$ and then we obtain $f(2) = 0.6$.

Now if $x_1 = 1$ and $x_2 = 1$ then $f_1 = 0.5 + 0.5\frac{1-10}{10} = 0.5 - 0.9 = -0.4$ and $f_2 = 0.5 + 0.5\frac{3-1}{3} = 0.5 + 0.33 = 0.88$. Hence, $\ell = \min\{-0.4, 0.88\} = -0.4$ and then we obtain $f(2) = 0$.

3.4 Voting on detection level

Consider that we have a set of $f(n_i) \in \{0\} \cup [z, 1] \subseteq [0, 1], i = 1, \ldots, m$ hard or soft quantifications each with n_i terms. The simple voting $F(m)$of such quantifications w.r.t. base z is

$$F(m) = 1 \text{ if } \frac{1}{m}\sum_{i=1}^{m} f(n_i) \geq z$$

or

$$F(m) = 0 \text{ if } \frac{1}{m}\sum_{i=1}^{m} f(n_i) < z$$

In our case we can have $z = 0.5$ for a consensus.

4 Missing Value Substitution Algorithm

The problem

Let time t a value x_t is missing for a Streamer. Let assume that the Streamer maintains a sliding-window of the last m values $x_{t-m}, \ldots, x_t$. The problem is to estimate the x_t missing value w.r.t. reservoir $m-1$ values. It is worth noting that an estimated missing value x_t is stored in the m reservoir for further implementations of the missing value events. This obviously increases the *unknown* error in data accuracy.

4.1 Naive Missing Value Algorithm

The value at t, x_t, is the immediate previous value, i.e.,

$$x_t = x_{t-1}$$

and for any time $k \geq t$, in which any value is missing, $x_k = x_{t-1}$ until m values are substituted in the future.

4.2 Current Mean Value Missing Value Algorithm

The value at t, x_t, is the mean value of the most recent m stored values, i.e.,

$$x_t = \frac{1}{m-1}\sum_{k=t-m}^{t-1} x_k$$

and for any time $k \geq t$, in which any value is missing, the x_k value is estimated by the previous estimated missing values through the mean value of the last m measurements.

4.3 Polynomial Extrapolation Missing Value Algorithm

We adopt the Lagrange Polynomial method. We assume the series of $n+1$ mappings $(x_0, t_0), (x_1, t_1), \ldots, (x_n, t_n)$ and search for a polynomial passing through all $n+1$ mappings. The polynomial of Lagrange is the nth degree and has the form:

$$f_n(t) = \sum_{i=0}^{n} \ell_i(t) x_i$$

with

$$\ell_i(t) = \prod_{j=0, j \neq i}^{n} \frac{t - t_j}{t_i - t_j}$$

The $\ell_i(t)$ is a weighting function that includes a product of $n-1$ terms with terms of $j = i$ omitted. In our case we have $n = m-2$, i.e., $m-1$ known mappings and search for the m-th mapping (x_t, t) with $(x_0, t_0) = (x_{t-m+1}, t-m+1), (x_1, t_1) = (x_{t-m+2}, t-m+2), \ldots, (x_n, t_n) = (x_{t-1}, t-1)$. Hence,

$$x_t = f_{m-2}(t)$$

5 Novelty Detection Algorithm

The problem

In this problem a series of values are coming and try to determine which value at t, x_t, is candidate to be novel or outlier. This implies that the system is on-lined trained and forms a set of clusters C. A value that is not classified to any of the up-to-now clusters then it is candidate for outlier. If a cluster $c \in C$ has a significant number of hits then it is a novel cluster. A cluster with a low number of hits and any value got classified in such cluster then it is a candidate outlier.

5.1 On-line Clustering Novelty Detection Algorithm

Consider at time t an incoming value x_t. Consider also a knowledge base $\mathcal{C}$ with a set of clusters (defined below). This algorithm has a parameter, say vigilance $r \in [0, 1]$. Through this parameter, we construct a cluster c with all values which belong to the interval $[x(1-r), x(1+r)]$. For each new value x_t, if x_t can be classified to one cluster

$$c^* = \arg\min_{c \in \mathcal{C}} |c - x_t|$$

i.e., if $|c^* - x_t| \leq rc^*$ (c^* is the cluster head, then x_t belongs to c^*. Then a counter, $f(c^*)$ increases (a hit). Otherwise, a new cluster is constructed with cluster head as x_t and the corresponding counter is $f(x_t) = 1$. This implies that $\mathcal{C} = \mathcal{C} \cup \{x_t\}$.

This algorithm returns the probability of x_t being 'outlier'. This probability is defines as follows:

$$P(x_t) = \frac{\sum_{c \in \mathcal{C} f(c)}}{\sum_{c \in \mathcal{C} f(c)} + 1}$$

The outlier detection gives feedback to the Voting algorithms for take into consideration the x_t value for voting or not.

5.2 The CUMSUM Detection Algorithm

This algorithm attempts to detect a change on the distribution of a time series $x_t \in \mathbb{R}$ w.r.t. a target value. The detection is reported for one-side and two-side detection. In the one-side detection, the time series x_t is detected when it deviates above the target h^+, while in the two-side detection the x_t is detected whether it deviates above h^+ and below h^-.

The input parameters for the CUMSUM algorithm are:

- the target value $\mu \in \mathbb{R}$
- the above-tolerance value k^+
- the below-tolerance value k^-
- the above-threshold value h^+
- the below-threshold value h^-

The output parameters for the CUMSUM algorithm are:

- the above-detection signal $s^+ \in \{0, 1\}$
- the bellow-detection signal $s^- \in \{0, 1\}$

The algorithm detects a change and (i) sets the $s^+ = 1$ if the time series deviates above the target value or (ii) sets the $s^- = 1$ if the time series deviates below the target value. The algorithm has as follows:

Algorithm

```
$R \leftarrow 0$
$Q \leftarrow 0$
$t \leftarrow 1$
while(true)
  $s^+ = 0$                          slack
  $s^- = 0$
  $R = \max(0, x_t - (\mu + k^+) + R)$
  $Q = \min(0, x_t - (\mu - k^-) + Q)$
 if($R > h$)
  $s^+ = 1$
 end-if
 if($Q < -h$)
  $s^- = 1$
 end-if
end-while
End
```

5.3 Shewhart Controller

In the Shewhart control chart, a variable x_k is detected to deviate at time k from its normality denoted by two control limits: the Upper Control Limit (UCL) and Lower Control Limit (LCL). The control limits are defined as the distance from the current mean value of the process $x_1, \ldots, x_k$ which is:

- $\bar{x}_k + a\sigma_k$ for UCL, and

- $\bar{x}_k$ - $a\sigma_k$ for LCL

That is, x_k is detected to fire an alarm if $x_k > UCL_k$ or $x_k < LCL_k$. The UCL_k and the LCL_k values are defined as follows (through incremental methods):

$$\bar{x}_k = \bar{x}_{k-1} + \frac{x_k - \bar{x}_{k-1}}{k}$$

and

$$\sigma_k^2 = \frac{1}{k}((k-1)\sigma_{k-1}^2 + (x_k - \bar{x}_k)(x_k - \bar{x}_{k-1}))$$

The controller returns +1 if $x_k > UCL_k$, -1 if $x_k < LCL_k$ and normality, i.e., 0 if $x_k \in (LCL, UCL)$. The parameter a is defined to be $a = 3$.

6 Crisp Value Bayesian Network

In this version, the Bayesian Network Contextor (BNC) has input n crisp evidences $e_1, \ldots, e_n$ and output a vector of m crisp hypotheses $h_1, \ldots, h_m$. The BCN needs the conditional probability matrix $M_{n\times m}$ and the hypothesis vector $H_{m\times 1}$.

The output $\mathbf{P} = [P_1, P_2, \ldots, P_m]^\top$ is the probabilities of the hypotheses w.r.t. pieces of evidences. The output can also be calculated in a sequential way. The type of the probabilities vectors $\mathbf{P} = [P_1, P_2, \ldots, P_m]^\top$ is as follows: The probability of the k-th hypothesis, $k = 1, \ldots, m$ is:

$$P_k = P(h_k|e_1, \ldots, e_n) = \frac{P(e_1|h_k)P(e_2|h_k)\cdots P(e_n|h_k)}{\sum_{i=1}^m P(e_1|h_i)P(e_2|h_i)\cdots P(e_n|h_i)} \tag{4}$$

The matrix $M_{n\times m} = [a_{ik}], i = 1, \ldots, n, k = 1, \ldots, m$ such that $a_{ik} = P(e_i|h_k)$ and the vector $H_{m\times 1} = [P(h_1), P(h_2), \ldots, P(h_m)]^\top$.

Obviously, one can calculate the $P(h_k|e_1)$, then the $P(h_k|e_1, e_2)$, then the $P(h_k|e_1, e_2, e_3)$ and so on, in a sequential way. The evidence with the highest probability is the winner of the BCN, i.e., most probable hypothesis is $h_{k^*} = \arg\max\{P_k\}_{k=1,\ldots,m}$.