Current Research

Tim Arnett

Summer 2017

Contents

1 Hierarchical Fuzzy Inference Systems (hFIS) vs. Single Fuzzy Inference System (sFIS)

- **Efficacy of hFIS compared to sFIS**

  In order to show the differences between an hFIS and sFIS, a method was developed during the summer of 2016 to convert an hFIS to an sFIS (ref). This method was based on an analysis of the current active mode within the FISs and equating the output membership functions. The membership functions of the sFIS could then be described as a function of the output membership functions of the hFIS. The final representation of the sFIS equivalent to an hFIS is shown in Eq. (1)-(5).

$$\begin{aligned} y = \mu_k \begin{bmatrix} \mu_j & \mu_{j+1} \end{bmatrix} K_{u_1} RB_1 \begin{bmatrix} \mu_i \\ \mu_{i+1} \end{bmatrix} + \\ \mu_{k+1} \begin{bmatrix} \mu_j & \mu_{j+1} \end{bmatrix} K_{u_2} RB_2 \begin{bmatrix} \mu_i \\ \mu_{i+1} \end{bmatrix} \end{aligned} \tag{1}$$

$$
\begin{aligned}
&RB_1 = \\
&\begin{bmatrix} (U_{l+1,j} - U_{l,j}) \begin{bmatrix} U_i & U_{i+1} \end{bmatrix} \\ (U_{l+1,j+1} - U_{l,j+1}) \begin{bmatrix} U_i & U_{i+1} \end{bmatrix} \end{bmatrix} + \\
&\begin{bmatrix} \left(y1_{l+1} U_{l,j} - y1_l U_{l+1,j}\right) \begin{bmatrix} 1 & 1 \end{bmatrix} \\ \left(y1_{l+1} U_{l,j+1} y1_l U_{l+1,j+1}\right) \begin{bmatrix} 1 & 1 \end{bmatrix} \end{bmatrix} + \\
&\left(y1_{l+1} - y1_l\right) \left(y2_{m+1} U_{m,k} - y2_m U_{m+1,k}\right) \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
&RB_2 = \\
&\begin{bmatrix} (U_{l+1,j} - U_{l,j}) \begin{bmatrix} U_i & U_{i+1} \end{bmatrix} \\ (U_{l+1,j+1} - U_{l,j+1}) \begin{bmatrix} U_i & U_{i+1} \end{bmatrix} \end{bmatrix} + \\
&\begin{bmatrix} \left(y1_{l+1} U_{l,j} - y1_l U_{l+1,j}\right) \begin{bmatrix} 1 & 1 \end{bmatrix} \\ \left(y1_{l+1} U_{l,j+1} - y1_l U_{l+1,j+1}\right) \begin{bmatrix} 1 & 1 \end{bmatrix} \end{bmatrix} + \\
&\left(y1_{l+1} - y1_l\right) \left(y2_{m+1} U_{m,k+1} - y2_m U_{m+1,k+1}\right) \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}
\end{aligned} \tag{3}
$$

$$
K_{u_1} = \left(\frac{1}{y1_{l+1} - y1_l}\right) \left(\frac{U_{m+1,k} - U_{m,k}}{y2_{m+1} - y2_m}\right) \tag{4}
$$

$$
K_{u_2} = \left(\frac{1}{y1_{l+1} - y1_l}\right) \left(\frac{U_{m+1,k+1} - U_{m,k+1}}{y2_{m+1} - y2_m}\right) \tag{5}
$$

As complex as this appears, a closed form for a more general hFIS to sFIS representation may be possible. This transformation may be useful after dynamic hFIS structure learning to then convert back to an sFIS to more closely approximate ideal control mappings. However, this transformation does not explore the limitations of the approximation abilities of the hFIS as compared to the sFIS. The constraints on the FISs used during this work were found to be called Ruspini Partitioning (ref ruspini 1969). A paper showing universal approximation abilities of an sFIS with these properties was found (ref) but did not explore the

-

-

2 Dynamic hFIS structure

3 Weekly Record

3.1 5/15/17-5/19/17

4 Stream of Consciousness

4.1 Learning underlying target function structure during reinforcement learning

- Current criteria for identifying when an sFIS should attempt to split into a particular hFIS are suboptimal (understatement). Currently utilizing the coefficients of the polynomials for each mode of the current best set of FIS parameters (see (ref)).
- What other methods can be used to quickly identify portions (or all) of the underlying optimal control function? This is starting to blur lines between Machine Learning, optimal control, and system/parameter ID.
- Can Neural Nets be used in conjunction with FISs (maybe as the activation functions) to produce good results? Is that useful? Can Convolutional Neural Nets be used for feature extraction in the solution space during reinforcement learning? If so can that be used to adjust the structure of the hFIS in a fast and meaningful way during reinforcement learning?
- There's some work on learning Bayesian network structure. Can we use those methods or perhaps Bayesian methods themselves to refine hypotheses (solutions of desirable parameter sets) on the fly during reinforcement learning? Possibly major issues involving establishing probabilities of better reward/fitness for particular actions.