Cerebellar Model Articulation Controller (CMAC)

Jacky Baltes

University of Auckland

Email: j.baltes@auckland.ac.nz

Web: www.tcs.auckland.ac.nz/~jacky

March 7, 2003

Cerebellum

- If we try and do AI, why don’t we use the human brain as a model.
  - Many apsects are not well understood. Too complex.
  - How do I walk?
- But, some AI/Robotics methodologies are inspired by biology.
- Some parts of the brain are reasonably well understood. The cerebellum has been (sort of) understood since 1992.
- It is responsible for coordination, motion control.
- How do we know? People that have a damaged cerebellum (a) think normally, (b) move clumsily, and (c) know that they are having problems.

Cerebellum

- Cerebellum monitors motor commands from the brain and modifies them (Timing, grouping, fine tuning).
- Then sends them to the spinal cord to control the muscles.
- Input: Emotional context, sensory context, motor commands.
- Output: Modifications to input motor commands.

Cerebellum

- Motor programs are sequences of motor commands that can be executed without feedback. (Running, Golf swing, typing).
- The motor commands may be adapted (injured finger). Adjusting parameters of the commands.

Cerebellar Learning and Adaptation

- Cerebellum is responsible for motor programs, reflexes, voluntary movements.
- Cerebellum must (a) learn sequences of motor commands, and (b) adjust the parameters of learned sequences.
- Although exact working unknown, some constructive theories of the cerebellum have been proposed. Cerebellum has a very regular structure.
- David Marr: A theory of cerebellar cortex
- James Albus: A theory of cerebellar function.
- Cerebellar learning is context driven (emotional, sensor). Through repeated rehearsal, the cerebellar cortex learns the associated context for a motor command. After sufficient training, the cerebellum will trigger a motion command if the context occurs.
- Cerebellum must perform pattern matching (to recognize context). Space requirements in the brain require a coarse pattern matching algorithm.

- Therefore, we also require pattern separation. (To determine the exact pattern).
- Small pattern sizes because each pattern recognizer only has a limited learning capacity. Need to have active regulation of patterns.
- Some of the features suggested by this theory have been verified experimentally (Modifiability of Purkinje cells).

Albus' CMAC Algorithm

- Motivation:
  - Robot motion requires complicated non-linear feedback. Hard problem to solve using a mathematical model. Fragile solutions.
  - Robots and the world are very hard to model.
  - Robots carry out the same task over and over again.
  - Controller must be robust in the presence of noise.
- CMAC
  - Table based computation is very fast
  - Learning
  - Generalization

CMAC Algorithm

- Table lookup y = f(x), where y is a scalar or vector and x is the input vector.
- But, lookup is done on a set of table entries
- Output is the sum of all table cell elements.

CMAC Learning

- A CMAC learns by filling in the values for the table cells (Initially 0).
- By repeatedly rehearsing a motion, the cell values are adjusted.
- Learning is controlled by comparing actual and desired output (Supervised learning).

Generalization

- Can not show all possible input patterns.
- Generalization means generate *similar* outputs for *smilar* inputs.
- How do we determine what is similar?
- This increases robustness.

Mapping Function

- $X = (x_1, x_2, ...x_n)$ is an input vector. A^* is the set of table cells for X.
- The mapping algorithm determines A^* for each X.
- Why do we use several table cells and not just one? For generalization.

Generalization

- Mapping: $X_1 \rightarrow A_1^*$ and $X_1 \rightarrow A_2^*$.
- CMAC Output: Sum of A_1^* and Sum of A_2^* respectively.
- $A_1^* Y A_2^* \rightarrow$ implies X_1 and X_2 are independent, otherwise they are coupled.
- If X_1 and X_2 are similar, then we want them to be coupled, otherwise, we want them to be different.
- We want a mapping function that makes similar inputs (based on difference of features) have a non-zero overlap (so they are strongly coupled). The further two inputs are apart, the less coupling they should have, and if they are apart by more than a certain threshold, they should be independent.

Design of the Mapping Function

- Increase in size of A^* increases the generalization.
- Low resolution of the mapping function leads to much generalization.
- Size of the cell table is roughly R^N, where R is the resolution of the mapping function, and N is the input space dimension.
- Only a small fraction of the table is being used, so people often implement the cell table as a hash.

Learning

- Compute error signal $e = f - f'$.
- CMAC output is $\sum A^*$.
- Add $(f - f')/|A^*|$ to each cell in A^* (Albus)
- Time inversion. $f = \sum A^*$. f is the correct output for situation X'. Instead of changing the output for vector X, change the output for X', to compute the correct output. Add $(f - f')/|A'^*|$ (Hirzinger)
- Weighted learning. Use a learning rate $0 < \beta \leq 1$ to modify table cells. For example, add $\beta * (f - f')/|A^*|$