Towards a Theoretical Framework for LCS

Jan Drugowitsch
Dr Alwyn Barry
Overview

Big Question: Why use LCS?

1. Aims and Approach
2. Function Approximation
3. Dynamic Programming
4. Classifier Replacement
5. Future Work
Research Aims

- To establish a formal model for Accuracy-based LCS
- To use established models, methods and notation
- To transfer method knowledge (analysis / improvement)
- To create links to other ML approaches and disciplines
Approach

- Reduce LCS to components:
  - Function Approximation
  - Dynamic Programming
  - Classifier Replacement

- Model the components
- Model their interactions
- Verify models by empirical studies
- Extend models from related domains
Function Approximation

- **Architecture Outline:**
  - Set of \( K \) classifiers, *matching* states \( S_k \subseteq S \)
  - Every classifier \( k \) approximates the value function \( V \) over \( S_k \) independently.

- **Assumptions**
  - Function \( V : S \rightarrow \mathbb{R} \) is time-invariant
  - Time-invariant set of classifiers \( \{S_J, \ldots, S_K\} \)
  - Fixed sampling dist. \( \Pi(S) \) given by \( \pi : S \rightarrow [0, 1] \)
  - Minimising MSE: \( \sum_{i \in S_k} \pi(i)(V(i) - \tilde{V}_k(i))^2 \)
Function Approximation

- Linear approx. architecture

- Feature vector \( \phi : S \rightarrow \mathbb{R}^L \)
- Approximation parameters vector \( \omega_k \in \mathbb{R}^L \)
- Approximation \( \bar{V}_k = \phi(i)^T \omega_k \)

- Averaging classifiers, \( \phi(i) = (1), \forall i \in S \)
- Linear estimators, \( \phi(i) = (1, i)', \forall i \in S \)
Function Approximation

- Approximation to MSE for \( k \) at time \( t \) :

\[
\varepsilon_{k,t} = \frac{1}{c_{k,t} - 1} \sum_{m=0}^{t} I_{S_k}(i_m)(V(i_m) - \omega'_{k,t}\phi(i_m))^2
\]

- Matching: \( I_{S_k} : S \rightarrow \{0,1\} \)

- Match count: \( c_{k,t} = \sum_{m=0}^{t} I_{S_k}(i_m) \)

- Since the sequence of states \( i_m \) is dependant on \( \pi \), the MSE is automatically weighted by this distribution.
Function Approximation

- Mixing classifiers
  - A classifier $k$ covers $S_k \subseteq S$
  - Need to *mix* estimates to recover $\tilde{V}(i)$
  - Requires mixing parameter (will be ‘accuracy’ ... see later!)

\[
\tilde{V}(i) = \sum_{k=1}^{K} \psi_k(i)\omega_k\phi(i) \quad \text{where} \quad \psi_k : S \rightarrow \{0,1\}, \sum_{k=1}^{K} \psi_k(i) = 1
\]

- $\psi_k(i) = 0$ where $i \notin S_k$ ... classifiers that do not match do not contribute
Function Approximation

Matrix notation

\[ V = (V(1), \cdots, V(N))' \]
\[ \Phi = \begin{pmatrix} \cdots \phi(1)' \cdots \\ \cdots \\ \cdots \phi(N)' \cdots \end{pmatrix} \]
\[ D = \begin{pmatrix} \pi(1), \cdots, 0 \\ \vdots \\ 0, \cdots, \pi(N) \end{pmatrix} \]
\[ I_{S_k} = \begin{pmatrix} I_{S_k}(1), \cdots, 0 \\ \vdots \\ 0, \cdots, I_{S_k}(N) \end{pmatrix} \]

\[ D_k = I_{S_k} D \]
\[ \widetilde{V}_k = \Phi \omega_k \]
\[ \Psi_k = \begin{pmatrix} \psi_k(1), \cdots, 0 \\ \vdots \\ 0, \cdots, \psi_k(N) \end{pmatrix} \]

\[ \widetilde{V} = \sum_{k=1}^{K} \Psi_k \widetilde{V}_k = \sum_{k=1}^{K} \Psi_k \Phi \omega_k \]
Function Approximation

Achievements

- Proof that the MSE of a classifier $k$ at time $t$ is the normalised sample variance of the value functions over the states matched until time $t$
- Proof that the expression developed for sample-based approximated MSE is a suitable approximation for the actual MSE, for finite state spaces
- Proof of optimality of approximated MSE from the vector-space viewpoint (relates classifier error to the Principle of Orthogonality)
Function Approximation

Achievements (cont’d)
- Elaboration of algorithms using the model:
  - Steepest Gradient Descent
  - Least Mean Square
    - Show sensitivity to step size and input range hold in LCS
  - Normalised LMS
  - Kalman Filter
    - ... and its inverse co-variance form
    - Equivalence to Recursive Least Squares
    - Derivation of computationally cheap implementation
- For each, derivation of:
  - Stability criteria
    - For LMS, identification of the excess mean squared estimation error
  - Time constant bounds

May 2006
© A.M. Barry and J. Drugowitsch, 2006
Function Approximation

- Achievements (cont’d)
  - Investigation:

![Graph showing mean and variance over sample size](image-url)
Function Approximation

- How to set the mixing weights $\psi_k(i)$?
  - Accuracy-based mixing (YCS-like!)
  - Divorce mixing for fn. approx. from fitness

$$\psi_k(i) = \frac{I_{S_k}(i)\varepsilon_k^{-\nu}}{\sum_{p=1}^{K} I_{S_p}(i)\varepsilon_p^{-\nu}}$$

- Investigating power factor $\nu \in \mathbb{R}^+$
- Demonstrate, using the model, that we obtain the MLE of the mixed classifiers when $\nu = 1$
Function Approximation

- **Mixing - Empirical Results**
  - Higher $\nu$, more importance to low error
  - Found $\nu = 1$ best compromise

Classifiers: $0..\pi/2$, $\pi/2..\pi$, $0..\pi$

Classifiers: $0..0.5$, $0.5..1$, $0..1$
Dynamic Programming

**Dynamic Programming:**

\[
V^*(i) = \max_{\mu} \mathbb{E} \left( \sum_{t=0}^{\infty} \gamma^t r(i_t, a_t, i_{t+1}) \mid i_0 = i, a_t = \mu(i_t) \right)
\]

**Bellman’s Equation:**

\[
V^*(i) = \max_{a \in A} \mathbb{E} \left( r(i, a, j) + \gamma V^*(j) \mid j = p(i, a) \right)
\]

**Applying the DP operator \( T \) to value est.**

\[
(TV)(i) = \max_{a \in A} \mathbb{E} \left( r(i, a, j) + \gamma W(j) \mid j = p(i, a) \right)
\]
Dynamic Programming

- $T$ is a contraction w.r.t. maximum norm
  - Converges to unique fixed point $V^* = TV^*$
- With function approximation:
  - Approximation after every update:
    $$\tilde{V}_{t+1} = f(T\tilde{V}_t)$$
  - Might not converge for more complex approximation architectures
- Is LCS function approximation a non-expansion w.r.t. maximum norm?
Dynamic Programming

- **LCS Value Iteration**
  - Assume:
    - constant population
    - averaging classifiers, \( \phi (i) = (1), \forall i \in S \)
  - Classifier is approximating: \( V_{t+1} = T\tilde{V}_t \)
  - Based on minimising MSE:
    \[
    \sum_{i \in S_k} \left( V_{t+1}(i) - \tilde{V}_{k,t+1}(i) \right)^2 = \left\| V_{t+1} - \tilde{V}_{k,t+1} \right\|_{S_k}^2 = \left\| T\tilde{V}_t - \tilde{V}_{k,t+1} \right\|_{S_k}^2
    \]
Dynamic Programming

- LCS Value Iteration (cont’d)
  - Minimum is orthog. projection on approx:
    \[ \tilde{V}_{k,t+1} = \Pi_{I_{S_k}} V_{t+1} = \Pi_{I_{S_k}} TV_t \]
  - Need also to determine new mixing weight by evaluating the new approx. error:
    \[ \varepsilon_{k,t+1} = \frac{1}{\text{Tr}(I_{S_k})} \left\| V_{t+1} - \tilde{V}_{k,t+1} \right\|_{I_{S_k}}^2 = \frac{1}{\text{Tr}(I_{S_k})} \left\| (I - \Pi_{D_k}) TV_t \right\|_{I_{S_k}}^2 \]
  - \( \therefore \) the only time dependency is on \( \tilde{V}_t \)
  - Thus:
    \[ \tilde{V}_{t+1} = \sum_{k=1}^{K} \Psi_{k,t+1} \Pi_{I_{S_k}} TV_t \]
Dynamic Programming

- LCS Asynchronous Value Iteration
  - Only update $V_k$ for the matched state
  - Minimise: $\sum_{m=0}^{t} I_{S_k} (i_m) \left((T\tilde{V}_m)(i_m) - \omega'_k \phi(i_m)\right)^2$

- Thus, approx. costs weighted by state dist

$$\sum_{i \in S_k} \pi(i) \left((T\tilde{V})(i) - \omega'_k \phi(i)\right)^2 = \left\|T\tilde{V} - \Phi \omega_k\right\|^2_{D_k}$$

$\therefore \tilde{V}_{k,t+1} = \Pi_{D_k} T\tilde{V}_t$

$$\tilde{V}_{t+1} = \sum_{k=1}^{K} \Psi_{k,t+1} \Pi_{D_k} T\tilde{V}_t$$
Dynamic Programming

- Also derived:
  - LCS Q-Learning
    - LMS Implementation
    - Kalman Filter Implementation
    - Demonstrate that the weight update uses the normalised form of local gradient descent
  - Model-based Policy Iteration
    \[ \tilde{V}_{t+1} = \sum_{k=1}^{K} \psi_k \prod_{D_k} T_{\mu} \tilde{V}_t \]
  - Step-wise Policy Iteration
  - TD(\(\lambda\)) is not possible in LCS due to re-evaluation of mixing weights
Dynamic Programming

- Is LCS function approximation a non-expansion w.r.t. maximum norm?
  - We derive that:
    - Constant Mixing: Yes
    - Arbitrary Mixing: Not necessarily
    - Accuracy-based Mixing: non-expansion (dependant upon a conjecture)

- Is LCS function approximation a non-expansion w.r.t. weighted norm?
  - We know $T_{\mu}$ is a contraction
  - We show $II_{DK}$ is a non-expansion
  - We demonstrate:
    - Single Classifier: convergence to a fixed pt
    - Disjoint Classifiers: convergence to a fixed pt
    - Arbitrary Mixing: spectral radius can be $\geq 1$ even in some fixed weightings
Classifier Replacement

- How can we define the optimal population formally?
  - What do we want to reach?
  - How does the *global* fitness of one classifier differ from its *local* fitness (i.e. accuracy)?
  - What is the quality of a population?

- Methodology
  - Examine methods to define the optimal population
  - Relate to generalised hill-climbing
  - Map to other search criteria
**Future Work**

- Formalising and analysing classifier replacement
- Interaction of replacement with function approximation
- Interaction of replacement with DP
- Handling stochastic environments
- Error bounds and rates of convergence
- ...

Big Answer: LCS is an integrative framework, ...

... But we need the framework definition!