Citation for published version:
Drugowitsch, J & Barry, AM 2006, *Towards convergence of learning classifier systems value iteration*. Computer Science Technical Reports, no. CSBU-2006-03, University of Bath, Department of Computer Science.

Publication date:
2006

Link to publication

©The Author April 2006

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 01. Dec. 2022

Department of
Computer Science

UNIVERSITY OF
BATH

Technical Report

Towards Convergence of Learning Classifier Systems Value Iteration

Jan Drugowitsch and Alwyn Barry

Technical Report 2006-03

April 2006

ISSN 1740-9497

Copyright ©April 2006 by the authors.

Contact Address:
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

Towards Convergence of Learning Classifier Systems Value Iteration

Jan Drugowitsch and Alwyn M Barry

April 2006

Abstract

In this paper we are extending our previous work on analysing Learning Classifier Systems (LCS) in the reinforcement learning framework [4] to deepen the theoretical analysis of Value Iteration with LCS function approximation. After introducing our formal framework and some mathematical preliminaries we demonstrate convergence of the algorithm for fixed classifier mixing weights, and show that if the weights are not fixed, the choice of the mixing function is significant. Furthermore, we discuss accuracy-based mixing and outline a proof that shows convergence of LCS Value Iteration with an accuracy-based classifier mixing. This work is a significant step towards convergence of accuracy-based LCS that use Q-Learning as the reinforcement learning component.

1 Introduction

In [4] we described how to model Learning Classifier Systems (LCS) in the reinforcement learning framework. Even though that work is restricted to constant classifier populations, it is a crucial milestone towards a unified theory of function approximation, reinforcement learning and classifier replacement in the context of LCS. In this paper we investigate some properties of the effects of using LCS function approximation in combination with Value Iteration, particularly w.r.t. convergence of the algorithm.

The question of convergence of Value Iteration is an important one, as Value Iteration is a deterministic iteration that Q-Learning stochastically approximates. XCS, the currently most used classifier system, uses Q-Learning as its reinforcement learning component and is therefore directly affected by our investigations. Even though we could attempt to model Q-Learning in LCS directly, it is more appropriate to first handle the deterministic case and then show that the stochastic approximation is appropriate in the sense of it converging to the deterministic iteration at infinity.

We start our investigations by firstly describing the reinforcement learning framework, the LCS function approximation, and the LCS Value Iteration algorithm. As most of our work is based on contraction and non-expansion, we continue by discussing vector fields, how they can form contraction maps, and some of their other properties that we will require. After showing properties of the Value Iteration update operator, we will study how single classifiers behave and how they can be mixed to form an overall value function approximation. For constant classifier mixing, we will prove convergence of the algorithm. An example that follows shows that we cannot arbitrarily mix the classifiers and still get converging behaviour. The rest of the paper is devoted to accuracy-based mixing and its analysis. We conclude with a proof that shows convergence of LCS Value Iteration with accuracy-based mixing, but relies on a conjecture that describes a certain property of the LCS function approximation.

2 LCS Value Iteration

This section gives the formal basis of reinforcement learning and Value Iteration and how Value Iteration can be applied in Learning Classifier Systems.

2.1 The Reinforcement Learning Framework

Let S be the finite set of states of size $N = |S|$, which we will map without loss of generality to the set of natural numbers $\mathbb{N}$. In every state $i \in S$ we can perform an action a from a set of actions A, leading to a transition to the next state $j \in S$ and a scalar reward. The probability of a transition from i to j by performing action a is given by $p_{ij}(a)$, which is the transition function $p : S \times S \times A \to [0, 1]$. Every such transition is mediated by the reward $r_{ij}(a)$, given by the reward function $r : S \times S \times A \to \mathbb{R}$. A policy $\mu : S \to A$ gives the behaviour of an agent in the problem domain, as it determines the action choice for every action. The aim is to find the policy that maximises the discounted reward in the long run; that is for state i

$$V^*(i) = \lim_{n \to \infty} \mathbb{E} \left(\sum_{t=0}^{n} \gamma_t r_{i_t i_{t+1}}(a_t) | i_0 = i, a_t = \mu^*(i_t) \right),$$

where $\gamma \in (0, 1]$ is the discount factor, and we assume the sequence of states $\{i_0, i_1, \dots\}$ and actions $\{a_0, a_1, \dots\}$ to be generated according to the optimal policy μ^*. $V^* : S \to \mathbb{R}$ denotes the optimal value function that returns the expected return for every state i. Knowing this value function allows us to derive the optimal policy by choosing the action that is expected to maximise the next value.

2.2 Value Iteration and Approximate Value Iteration

One way to find the optimal value function V^* is to solve Bellman's Equation

$$V^*(i) = \max_{a \in A} \sum_{j \in S} p_{ij}(a) \left(r_{ij}(a) + \gamma V^*(j) \right), \quad i = 1, \dots, N, \tag{1}$$

which relates the optimal value of a state to the maximum possible reward and discounted value of the next state.

Value Iteration is a method for finding the optimal value function by repeatedly applying the Dynamic Programming (DP) update T to a value vector $V \in \mathbb{R}^N$, holding the current value for every state in S. The update T applied to V is defined by (e.g. [2, Ch. 2.2.1])

$$(TV)(i) = \max_{a \in A} \sum_{j \in S} p_{ij}(a) \left(r_{ij}(a) + \gamma V(j) \right), \quad i = 1, \dots, N.$$

That gives the iteration

$$V_{t+1} = TV_t,$$

starting with some arbitrary initial $V_{-1} \in \mathbb{R}^N$. Due to the properties of T, this iteration is guaranteed to converge to the optimal value function V^* when it is applied an infinite number of times.

Given that the set of states is large, calculating the value for every state at every iteration is spatially and computationally prohibitive. Applying function approximation to the value function V is an approach to circumvent this problem. Let $\tilde{V} : S \to \mathbb{R}$ be a parametric function approximation of V. Even though we will for now ignore its parameters, consider that its properties are determined by a finite set of scalar values that is usually smaller than S. The aim at every iteration becomes to minimise the difference between the update according to Value Iteration and its current approximation, that is

$$\tilde{V}_{t+1} = \min_{\tilde{V}} \|T\tilde{V}_t - \tilde{V}\|,$$

where the minimum is restricted to the approximation space given by the approximation architecture. As discussed in [4, Sec. 3.2.1], this iteration might converge only for certain function approximation architectures. In other cases, such as for linear regression or neural networks, the update might not converge or even diverge. Therefore it is important to investigate whether it is compatible with the function approximation in use.

2.3 LCS Function Approximation

Learning Classifier Systems use a special case of function approximation by mixing the independent approximation of a finite set of classifiers to form the overall approximation. Let us consider a set of K classifiers, each identified by its index $k \in \{1, \dots, K\}$. Each classifier k *matches* a certain subset S_k of the state space S, which we will call the *matched states set* $S_k \subseteq S$. The objective of each classifier is to minimise the approximation error over its matched states. To account for a non-uniform sampling distribution, we will consider the function $\pi : S \to [0, 1]$ to represent the probability of sampling a particular state. Let $\tilde{V}_k$ be the approximation of classifier k. For a given value function V we want to minimise the mean squared error

$$\sum_{i \in S_k} \pi(i) \left(V(i) - \tilde{V}_k(i) \right)^2$$

To ease notation, let $I_{S_k} : S \rightarrow \{0, 1\}$ be the indicator function for S_k that returns $I_{S_k}(i) = 1$ if $i \in S_k$ and $I_{S_k}(i) = 0$ otherwise. We can then define the $N \times N$ non-negative diagonal matrix I_{S_k} (that can be distinguished for the symbol's use as a function from the context it is used in) by having $I_{S_k}(1), \dots I_{S_k}(N)$ along its diagonal. The sampling distribution can be given by the $N \times N$ non-negative diagonal matrix D with the sampling distribution $\pi(1), \dots, \pi(N)$ along its diagonal. The sampling w.r.t. classifier k is given by the $N \times N$ diagonal matrix $D_k = I_{S_k} D$. Using this notation, we want to minimise

$$\|V - \tilde{V}_k\|_{D_k}$$

where $\| \cdot \|_{D_k}$ is the weighted norm, given for any vector $z \in \mathbb{R}^N$ by

$$\|z\|_{D_k} = \sqrt{\sum_{i \in S} I_{S_k}(i)\pi(i)z(i)^2},$$

that is, weighted by the diagonal of the matrix D_k.

The approximation architecture of each classifier is linear, characterised by the independence of the approximation parameters and the state-dependent values. Such an architecture requires for each state a particular set of scalar features that characterise that state. Let $\{\phi_l : S \rightarrow \mathbb{R}\}_l \in \{1, \dots, L\}$ be a set of L basis functions, each of which gives one feature for a given state. We can then define the feature vector $\phi : S \rightarrow \mathbb{R}^L$ for state $i \in S$ by $\phi(i) = (\phi_1(i), \dots, \phi_L(i))'$. The classifier approximation is parameterised by a *weight vector* $w_k \in \mathbb{R}^L$ of the same size as the feature vector. That gives the classifier's approximation $\tilde{V}_k$ of state $i \in S$ by

$$\tilde{V}_k(i) = w_k' \phi(i),$$

which is the inner product of the weight vector and the feature vector for that state. If we combine the features of all states into a feature matrix Φ, that is

$$\Phi = \begin{pmatrix} - & \phi(1)' & - \\ & \dots & \\ - & \phi(N)' & - \end{pmatrix},$$

then we can define a classifier's approximation by $\tilde{V}_k = \Phi w_k$.

With the knowledge of the classifier approximation architecture we can be more specific about our objective, which is to minimise the distance between the function we want to approximate and its approximation, that is

$$\|V - \Phi w_k\|_{D_k}.$$

From linear algebra we know that the approximation that minimises this distance can be found by orthogonally projecting the function into the approximation space, given by $\{\sqrt{D_k}\Phi w_k : w_k \in \mathbb{R}^L\}$[1] for classifier k. This orthogonal projection is described by the $N \times N$ projection matrix Π_{D_k}, given by

$$\Pi_{D_k} = \Phi(\Phi' D_k \Phi)^{-1} \Phi' D_k,$$

which when applied to a vector $z \in \mathbb{R}^N$ gives the closest point $\Pi_{D_k} z$ in the approximation space.

Having described the approximation of one classifier, we will now discuss how these classifiers are mixed to give the overall approximation $\tilde{V}$. Let $\psi_k : S \rightarrow [0, 1]$ be the mixing weights for classifier k, satisfying $\psi_k(i) = I_{S_k}(i)\psi_k(i)$ and $\sum_{k=1}^K \psi_k(i) = 1$ for all $i \in S$. Let Ψ_k be the $N \times N$ non-negative diagonal mixing matrix with $\psi_k(1), \dots \psi_k(N)$ along its diagonal. Due to the properties of ψ_k, we have

$$\sum_{k=1}^K \Psi_k = I, \quad \text{and} \quad \Psi_k = I_{S_k} \Psi_k.$$

The overall approximation $\tilde{V}$ given the classifier's approximations $\{\tilde{V}_1, \dots \tilde{V}_K\}$ is then defined by

$$\tilde{V} = \sum_{k=1}^K \Psi_k \tilde{V}_k.$$

Hence, for each state it is given by the weighted average of all classifiers that match that state.

[1] Here we use the fact that for some weighted norm $\| \cdot \|_{D_k}$ and some vector $z \in \mathbb{R}^N$, $\|z\|_{D_k} = \|\sqrt{D_k} z\|$.

As derived in [3], mixing weights that conform to the Maximum Likelihood Estimate under some assumptions are

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

where ε_k is an estimate of the approximation error of classifier k, and $\nu \in \mathbb{R}^+_{\neq 0}$ is a mixing parameter that is usually set to $\nu = 1$. Hence, the classifiers are weighted inversely proportional to the quality of their approximation. As the approximation error estimate depends on the function to approximate and might change over time, the mixing weights will also change over time, which we will account for by denoting them by $\Psi_{k,t}$.

2.4 LCS Value Iteration

As described before, approximate Value Iteration is based on approximating each step of a Value Iteration. Each classifier maintains approximation $\tilde{V}_{k,t}$ at time t, which gives the overall approximation

$$\tilde{V}_t = \sum_{k=1}^K \Psi_{k,t}\tilde{V}_{k,t}.$$

On this approximation we perform one DP update, giving the non-approximated new value vector V_{t+1} by

$$V_{t+1} = T\tilde{V}_t.$$

At that point we want each classifier to approximate that value vector by minimising $\|V_{t+1} - \tilde{V}_k\|_{D_k}$. This minimum is given by

$$\tilde{V}_{k,t+1} = \Pi_{D_k}V_{t+1} = \Pi_{D_k}T\tilde{V}_t, \quad k = 1, \dots, K.$$

At the same time, each classifier keeps track of the approximation error, which at time $t+1$ is given by

$$\varepsilon_{k,t+1} = \frac{1}{\mathrm{Tr}(D_k)}\|V_{t+1} - \tilde{V}_{k,t+1}\|_{D_k} = \frac{1}{\mathrm{Tr}(D_k)}\|T\tilde{V}_t - \Pi_{D_k}T\tilde{V}_t\|_{D_k}.$$

That shows that given a certain problem that determines D_k and T, and a set of matched state sets S_k giving Π_{D_k}, the only time-dependent value of the classifier error $\varepsilon_{k,t+1}$ is the previous overall approximation[2] $\tilde{V}_t$. As classifier mixing is usually calculated from the classifier errors, the mixing weights $\Psi_{k,t+1}$ are subsequently also a function of $\tilde{V}_t$.

To completely describe the iteration, let us combine above step-wise instruction into a single update equation, given by

$$\tilde{V}_{t+1} = \sum_{k=1}^K \Psi_{k,t+1}\Pi_{D_k}T\tilde{V}_t. \tag{2}$$

Hence, the next overall approximation $\tilde{V}_{t+1}$ is completely determined by the current overall approximation $\tilde{V}_t$.

3 Convergence Considerations

We will describe some of the investigations that we can make regarding convergence of LCS Value Iteration when using averaging classifiers. For a more detailed discussion on the function approximation of averaging classifiers see [3]. The reasons why we restrict ourselves to averaging classifiers is given in [4, Sec. 4.2.1], but can be summarised by the possibility of divergence for other kinds of classifier approximation architectures.

[2] Note that $\tilde{V}_t$ is usually not explicitly represents but is recovered from the classifier approximations $\tilde{V}_{k,t}$. That requires knowledge of the mixing weights $\Psi_{k,t}$ that are computed from the classifier errors $\varepsilon_{k,t}$. As we use $\tilde{V}_t$ to calculate the next errors $\varepsilon_{k,t+1}$, we need to create a temporary copy of the current errors $\varepsilon_{k,t}$ to be able to calculate $\tilde{V}_t$.

3.1 Contraction Maps

The algorithm is based on a mapping from $\mathbb{R}^N$ into $\mathbb{R}^N$ that describes an N-dimensional vector field. We will firstly define some properties of such vector fields, and then describe how to combine these properties to get a contraction mapping.

Given two vectors $V, \bar{V} \in \mathbb{R}^N$ we will write $V \leq \bar{V}$ if

$$V(i) \leq \bar{V}(i), \qquad i = 1, \dots, N$$

holds. Contraction maps are defined by the change of their image w.r.t. the change of the relevant pre-image. Let us first define a general continuity property of functions (see, for example, [5, Ch. 9.3]):

Definition 3.1 (Lipschitz Continuity). Let f be a function on a metric space (M, d) from M to itself. Then f is *Lipschitz continuous* if for some non-negative constant $C \in \mathbb{R}$,

$$d(f(x), f(y)) \leq C d(x, y),$$

for all x and y in M. The constant C is called the *Lipschitz constant* of that function.

We can use the value of the Lipschitz constant to identify three cases:

Definition 3.2. Let f be a Lipschitz continuous function on the metric space (M, d) with Lipschitz constant C. The function is said to be

1. a *contraction* with *contraction modulus* C, if $C < 1$,
2. a *non-expansion* if $C = 1$, and
3. an *expansion* if $C > 1$.

Contraction mappings are a particularly interesting case because they exhibit useful properties when used in iterative algorithms, as expressed by the following theorem (see, for example, [5, Ch. 9.3]):

Theorem 3.1 (Contraction Mapping Theorem). *Let P be a closed real interval, that is P has one of the following forms: $[a, b]$, $[a, \infty)$, $(-\infty, b]$ or $(-\infty, \infty)$. Let $f : P \to P$ be a contraction mapping with contraction modulus $C \in (0, 1)$. Then*

1. *f has a unique fixed point s in P;*
2. *for any $x_0 \in P$, the simple iteration $x_{t+1} = f(x_t)$ gives a sequence converging to s.*

In our case the metric space (M, d) is given by $M = \mathbb{R}^N$. We will define the distance metric d as being the maximum norm for reasons that will become apparent once we investigate the properties of the DP update T. This maximum norm is for any two vectors $x, y \in \mathbb{R}^N$ defined by

$$d(x, y) = \|x - y\|_\infty = \max_{i=1,\dots,N} |x(i) - y(i)|.$$

To be more specific about the operators that define our algorithmic iteration, let us name some properties of the vector fields that they describe:

Definition 3.3 (Increasing Vector Field). Let f be a vector field from $f : \mathbb{R}^N \to \mathbb{R}^N$. Then f is *increasing*, if $x \leq y$ implies that $f(x) \leq f(y)$ for all x and y in $\mathbb{R}^N$.

Definition 3.4 (Scalar Shift Vector Field). Let f be a vector field from $f : \mathbb{R}^N \to \mathbb{R}^N$, let $\gamma \in \mathbb{R}$ be a scalar such that $\gamma \in [0, 1]$, let $m \in \mathbb{R}$ be a scalar, and let $e \in \mathbb{R}^N$ be a vector that is given by $e = (1, \dots, 1)'$. Then we call f a *scalar shift vector field* with scaling γ, if $f(x + me) = f(x) + \gamma m e$ for all x in $\mathbb{R}^N$.

Let us assume that we have an operator that describes a vector field that is both increasing and a scalar shift vector field. Then we can state the following:

Lemma 3.2. *Let $f : \mathbb{R}^N \to \mathbb{R}^N$ describe a vector field that is both increasing and a scalar shift vector field with scaling γ. Then f describes a non-expansion w.r.t. the maximum norm if $\gamma = 1$, and a contraction w.r.t. the maximum norm with contraction modulus γ otherwise.*

Proof. The proof is similar to the one showing the contraction of the DP update operator T in [2, Lemma 2.5]. Let $x, y \in \mathbb{R}^N$ be two vectors, and c be the maximum norm of $x - y$, that is

$$c = \max_{i=1,\dots,N} |x(i) - y(i)|.$$

Then we have

$$x(i) - c \leq y(i) \leq x(i) + c, \qquad i = 1, \dots, N.$$

Applying f, we can write, based on f's properties,

$$(f(x))(i) - \gamma c \leq (f(y))(i) \leq (f(x))(i) + \gamma c, \qquad i = 1, \dots, N.$$

Therefore,

$$|(f(x))(i) - (f(y))(i)| \leq \gamma c, \qquad i = 1, \dots, N.$$

Hence we have

$$\|f(x) - f(y)\|_\infty \leq \gamma \|x - y\|_\infty,$$

which for $\gamma = 1$ is a non-expansion, and for $\gamma < 1$ is a contraction with modulus γ. □

Therefore, given that we have an update function that describes an increasing and scalar shift vector space with scaling smaller than one, we have a contraction, and repeatedly applying this function will cause convergence to the fixed point of this function. We will proceed by showing that these properties hold for the DP update T, and then investigate if we can state the same for the DP update in combination with LCS function approximation using averaging classifiers.

3.2 The DP Update T

As the operator T is at the core of Value Iteration, we will discuss some of its properties. The DP update operator T maps from $\mathbb{R}^N$ to $\mathbb{R}^N$ and hence describes a vector field. A simple analysis (as given in [2, Sec. 2.3]) of this field reveals the following properties:

Lemma 3.3. *The vector field given by T is increasing and a scalar shift vector field with scaling γ. Hence, it is a contraction to the maximum norm with contraction modulus γ.*

Proof. The proof for the increasing property and scalar shift property of T is given in [2, Lemma 2.1] and [2, Lemma 2.1]. Its contraction follows from Lemma 3.2. □

Given an estimate V of the optimal value function, repeatedly applying the DP update T to this estimate will let the estimate converge to the optimal value function V^*, which is the unique fixed point of the update $V^* = TV^*$. As that fixed point expression is Bellman's Equation (1), we have found the solution to that equation.

3.3 Averaging Classifiers

Averaging Classifiers are classifiers that use the single feature $\phi(1) = 1$ for their approximation. This results in a $1 \times N$ feature matrix $\Phi = (1, \dots, 1)'$. For the projection Π_{D_k} of classifier k this gives

$$\begin{aligned} \Pi_{D_k} &= \Phi(\Phi D_k \Phi')^{-1}\Phi' D_k \\ &= \mathrm{Tr}(D_k)^{-1}\Phi\Phi' D_k. \end{aligned}$$

For any vector $V \in \mathbb{R}^N$ this gives the approximation

$$(\Pi_{D_k} V)(i) = \frac{\sum_{j \in S_k} \pi(j) V(j)}{\sum_{m \in S_k} \pi(m)}, \tag{3}$$

which is the distribution-weighted average of V over the matched states S_k. Like T, Π_{D_k} also describes a vector field $\Pi_{D_k} : \mathbb{R}^N \to \mathbb{R}^N$. Some helpful properties of this vector field are:

Lemma 3.4. *The vector field described by Π_{D_k} is increasing.*

Proof. Let $V, \bar{V} \in \mathbb{R}^N$ be two vectors such that $V \leq \bar{V}$, and let us denote their non-negative difference by $c = \bar{V} - V$. Using $V = \bar{V} - c$, we can derive for $\Pi_{D_k} V$ and a fixed state $i \in \{1, \dots, N\}$,

$$(\Pi_{D_k} V)(i) = \frac{\sum_{j \in S_k} \pi(j)\bar{V}(j)}{\sum_{m \in S_k} \pi(m)} - \frac{\sum_{j \in S_k} \pi(j)c(j)}{\sum_{m \in S_k} \pi(m)} \leq (\Pi_{D_k})(i),$$

which completes the proof. □

Lemma 3.5. *The operator Π_{D_k} describes a scalar shift vector field with scaling 1.*

Proof. The proof follows from expanding for $(\Pi_{D_k}(V + me))(i)$ for an arbitrary vector $V \in \mathbb{R}^N$, state $i \in \{1, \dots, N\}$, scalar $m \in \mathbb{R}$, and vector $e \in \mathbb{R}^N$ given by $e = (1, \dots, 1)'$. □

We can therefore say by Lemma 3.2 that the approximation Π_{D_k} performed by averaging classifiers gives a non-expansion w.r.t. the maximum norm. Hence, Π_{D_k} and T in combination would give a contraction on the same norm. However, we are not particularly interested in the approximation of a single classifier but want to consider all classifier in combination. To get more information about the behaviour of the overall approximation, we must, in addition to a single classifier's approximation, investigate how their mixed combination behaves. We will consider three cases: i) constant mixing, ii) arbitrary mixing, and iii) accuracy-based mixing.

3.4 Constant Mixing

Let us consider the case of constant mixing weights, that is $\Psi_{k,t} = \Psi_k$ for all $t = 0, 1, \dots$ and all $k \in \{1, \dots, K\}$. This allows us to show:

Lemma 3.6. *Let $\{\Psi_1, \dots \Psi_K\}$ be a set of time-invariant diagonal non-negative $N \times N$ mixing matrices, satisfying $\sum_{k=1}^K \Psi_k = I$, and $\Psi_k = I_{S_k}\Psi_k$, and let Π_{D_k} be the projection operator for averaging classifier k. Then $\sum_{k=1}^K \Psi_k \Pi_{D_k}$ is a non-expansion w.r.t. the maximum norm.*

Proof. We will show the validity of this lemma by demonstrating that the vector field described by our weighted classifier mix is increasing and a scalar shift vector field with scaling 1. From Lemma 3.2 it will follow that it is therefore a non-expansion w.r.t. the maximum norm.

Let us first show its increasing property by considering any two vectors $V, \bar{V} \in \mathbb{R}^N$ such that $V \leq \bar{V}$ and their non-negative difference $c = \bar{V} - V$. Using $V = \bar{V} - c$, we can derive for any state $i \in \{1, \dots, N\}$,

$$\left(\sum_{k=1}^K \Psi_k \Pi_{D_k} V\right)(i) = \left(\sum_{k=1}^K \Psi_k \Pi_{D_k} \bar{V}\right)(i) - \sum_{k=1}^K \Psi_k(i,i)(\Pi_{D_k} c)(i).$$

As the second sum on the right-hand side is non-negative, the vector field is increasing.

That the vector field is also a scalar shift vector field with scaling 1 can be shown by expanding

$$\left(\sum_{k=1}^K \Psi_k \Pi_{D_k}(V + me)\right)(i),$$

where $V \in \mathbb{R}^N$ is any vector, $i \in \{1, \dots N, \}$ is any state, m is a scalar, and $e \in \mathbb{R}^N$ is the vector $e = (1, \dots, 1)'$. □

This non-expansion leads to the result:

Theorem 3.7. *Learning Classifier System Value Iteration with averaging classifiers and fixed mixing weights converges to the unique fixed point of the iteration.*

Proof. The LCS Value Iteration update for fixed mixing weights is

$$\tilde{V}_{t+1} = \sum_{k=1}^K \Psi_k \Pi_{D_k} T \tilde{V}_t.$$

By Lemma 3.3, T is a contraction w.r.t. $\| \cdot \|_\infty$. By Lemma 3.6, $\sum_{k=1}^K \Psi_k \Pi_{D_k}$ is a non-expansion w.r.t. the same norm. Therefore, $\sum_{k=1}^K \Psi_k \Pi_{D_k} T$ is a contraction and by Theorem 3.1 the above update converges to its unique fixed point. □

3.5 Time-variant Arbitrary Mixing

Let us now consider what happens if we change the mixing weights at every iteration. Given that the mixing weights are a function of the previous overall value approximation, would it be possible to set them to arbitrary values and still have a contraction? If that is the case, then we can guarantee convergence independent of the nature of the function that determines the mixing weights.

Let $V, \bar{V} \in \mathbb{R}^N$ be two vectors, and $\{\Psi_1, \dots, \Psi_K\}$ and $\{\bar{\Psi}_1, \dots, \bar{\Psi}_K\}$ be the mixing weights. For the update according to Eq. (2) to be a contraction, we would require

$$\left\| \sum_{k=1}^K \Psi_k \Pi_{D_k} TV - \sum_{k=1}^K \bar{\Psi}_k \Pi_{D_k} T\bar{V} \right\|_\infty \leq \gamma \|V - \bar{V}\|_\infty$$

to hold. As before, we will separate the function approximation from the DP update and observe its properties. If it features non-expansion w.r.t. the maximum norm, then we will get an overall contraction. Non-expansion is satisfied if

$$\left\| \sum_{k=1}^K \Psi_k \Pi_{D_k} V - \sum_{k=1}^K \bar{\Psi}_k \Pi_{D_k} \bar{V} \right\|_\infty \leq \|V - \bar{V}\|_\infty$$

holds. However, due to the different mixing weights for the different vectors we cannot reduce the above to a linear system as we have previously done to prove Lemma 3.6.

Let us consider a simple example with 2 classifiers, a state space $S = \{1, 2\}$ and uniform sampling, that is $\pi(1) = \pi(2) = \frac{1}{2}$. The first classifier matches all states, and the second classifier only matches the second state, that is $S_1 = \{1, 2\}$ and $S_2 = \{2\}$. Let the two vectors to approximate be $V = (0, 1)'$ and $\bar{V} = (2, 4)$. Due to their averaging nature, the first classifier will give a value of $\frac{1}{2}$ for V, and a value of 3 for $\bar{V}$. The second classifier matches the values of its states and will therefore give 1 for V, and 4 for $\bar{V}$. As for state 2 we are mixing the approximations of both classifiers, and therefore its overall approximation will be in the range $[0, 1]$ for V, and in the range $[2, 4]$ for $\bar{V}$, depending on the mixing weights. Note that $\|V - \bar{V}\|_\infty = |V(2) - \bar{V}(2)| = 3$. As we can chose arbitrary mixing weights, let us fix the approximation of $\bar{V}(2)$ at 4. We can now observe that the difference between the approximations for $V(2)$ and $\bar{V}(2)$ is in the range $[3, 4]$ depending on the mixing weights for the approximation of V. Hence, it might be larger than $\|V - \bar{V}\|_\infty$ and therefore might violate our non-expansion property. This demonstrates that we cannot guarantee non-expansion of the LCS function approximation for arbitrary mixing weights.

Consequently, the choice of function that determines the classifier mixing weights is significant w.r.t. the convergence properties of LCS Value Iteration. That leads to the question of how it has to be formed to guarantee non-expansion of the function approximation? In the previous example we have used different weighting rules applied to different vectors to demonstrate the violation of non-expansion. The approximation of $\bar{V}(2)$ puts full weight on the second classifier. If we do the same for the approximation of $V(2)$, we conform to the non-expansion property. Equally, we could set the approximation of $V(2)$ to be some average of both classifiers. If a similar average is applied to the approximation of $\bar{V}(2)$ we can still preserve the non-expansion property. How can we generalise this observation?

3.6 Accuracy-based Mixing

To ease discussion over classifier mixing based on accuracy, we will introduce an operator $\mathcal{C}$ that describes the overall approximation given such a classifier mixing. As described before, the mixing weights are based on the matching classifiers' approximation errors $\varepsilon_k : \mathbb{R}^N \rightarrow \mathbb{R}^+$, which are given for classifier k as a function of the current overall value function estimate V by the mean squared error between the new estimate TV and its approximation $\Pi_{D_k} TV$ by classifier k, that is

$$\varepsilon_k(V) = \frac{1}{\mathrm{Tr}(D_k)} \|TV - \Pi_{D_k} TV\|^2_{D_k} = \frac{1}{\mathrm{Tr}(D_k)} \|(I - \Pi_{D_k}) TV\|^2_{D_k}.$$

The mixing weights for a classifier k are some inverse of the error of that classifier, weighted by the inverse error of all matching classifiers. Hence, we can define a set of functions $\psi_k : S \times \mathbb{R}^N \rightarrow [0, 1]$ that give the mixing weight for classifier k for a given state i and value function estimate V, by

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

where ν is a time-invariant positive scalar that determines the emphasis of accuracy in the mixing.

We will write $\mathcal{C}V$ for applying this mixing strategy to a set of averaging classifiers that approximate the vector V. Hence, $\mathcal{C}$ is defined by

$$(\mathcal{C}V)(i) = \sum_{k=1}^K \psi_k(i, V)(\Pi_{D_k} V)(i) \qquad i = 1, \dots, N.$$

If this averaging schema describes a non-expansion w.r.t. the maximum norm, then we can guarantee convergence for its use in combination with Value Iteration. Hence, our aim is to show that $\mathcal{C}$ is a non-expansion. As before, we will proceed by treating $\mathcal{C}$ as describing a vector field $\mathcal{C} : \mathbb{R}^N \to \mathbb{R}^N$.

To show that $\mathcal{C}$ is increasing, let us first investigate the following:

Lemma 3.8. *For any vector $V \in \mathbb{R}^N$, scalar $m \in \mathbb{R}$, and vector $e \in \mathbb{R}^N$ given by $e = (1, \dots, 1)'$ we have*

$$T(V + me) - \Pi_{D_k} T(V + me) = TV - \Pi_{D_k} TV, \qquad k = 1, \dots, K.$$

Proof. From Lemma 3.3 we know that T describes a scalar shift vector field with scaling γ. Hence we can write

$$T(V + me) - \Pi_{D_k} T(V + me) = TV - \Pi_{D_k} TV + \gamma(I - \Pi_{D_k})me.$$

Additionally, by Lemma 3.5, Π_{D_k} is a scalar shift vector field with scaling 1, and $\Pi_{D_k}(0e) = 0e$. Hence,

$$(I - \Pi_{D_k})me = me - \Pi_{D_k}(0e + me) = me - me = 0.$$

□

By our definition of the classifier error ε_k, Lemma 3.8 implies that the approximation error is independent of any scalar shift of the value estimate V, that is $\varepsilon_k(V + me) = \varepsilon_k(V)$. That seems intuitive, as the error refers to the difference between the new value estimate and its approximation, which by the scalar shift property of DP update T and the approximation Π_{D_k} is independent of the scalar shift. Hence, given that the relative differences between the values of the states that the classifier matches are correct, the error approximation is also correct. We hypothesis that therefore we can get good approximate error estimates even before the final value function is known.

The only elements in the mixing weight function that depend on the value function V are the errors of the classifiers. As these don't change with a scalar shift of the value function, the weight also remains the same, that is $\psi_k(i, V + me) = \psi_k(i, V)$. We will use this property to show that $\mathcal{C}$ is a scalar shift vector field.

Lemma 3.9. *The vector field given by $\mathcal{C}$ is a scalar shift vector field with scaling 1.*

Proof. By Lemma 3.8, $\varepsilon_k(V + me) = \varepsilon_k(V)$, where $V \in \mathbb{R}^N$ is any vector, $m \in \mathbb{R}$ is a scalar, and $e \in \mathbb{R}^N$ is the vector $e = (1, \dots, 1)'$. Hence, the same can be said for the mixing weight function ψ_k, that is for any state $i \in \{1, \dots, N\}$, $\psi_k(i, V + me) = \psi_k(i, V)$. Therefore,

$$\sum_{k=1}^{K} \psi_k(i, V + me)(\Pi_{D_k}(V + me))(i) = \sum_{k=1}^{K} \psi_k(i, V)(\Pi_{D_k}(V + me))(i).$$

Lemma 3.5 shows that $\Pi_{D_k}(V + me) = \Pi_{D_k} V + me$. Hence,

$$\sum_{k=1}^{K} \psi_k(i, V)(\Pi_{D_k}(V + me))(i) = \sum_{k=1}^{K} \psi_k(i, V)(\Pi_{D_k} V)(i) + \sum_{k=1}^{K} \psi_k(i, V)me.$$

As $\sum_{k=1}^{K} \psi_k(i, V) = 1$, the second sum on the right-hand side is simply a scalar shift vector me, resulting in an overall scalar shift with scaling 1. □

Having established the scalar shift property of $\mathcal{C}$, we will now investigate if it is increasing. From the definition of the mixing weights ψ_k we can see that for $\nu = 0$, ψ_k is independent of the current value function estimate and therefore time-invariant. Hence, we can apply Lemma 3.6 to show that $\mathcal{C}$ is increasing. However, as we have already discussed in [3], $\nu = 0$ is possibly the worst setting for this parameter. Thus, we are more interested in the properties of $\mathcal{C}$ for $\nu > 0$.

It is well known that a differentiable continuous function of a single variable is increasing if and only if its first gradient is non-negative (e.g. [1, Ch. 11.2]). As the vector field described by $\mathcal{C}$ is continuous and differentiable, we are interested in applying this principle to vector fields. As derived in Appendix A, a vector field is increasing if its Jacobian is non-negative in all its components. Hence, we will derive the Jacobian of $\mathcal{C}$ to determine if it is increasing.

As given in Appendix B, the components of the Jacobian of $\mathcal{C}$ are given by

$$\frac{\partial \mathcal{C}_i V}{\partial V(l)} = \sum_{k=1}^{K} \psi_k(i, V) \frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)} \left(1 + 2\nu\varepsilon_k(V)^{-1}(V(l) - V_k)(\mathcal{C}_i V - V_k)\right),$$

where $\mathcal{C}_i V = (\mathcal{C}V)(i)$ is the ith component of the result of $\mathcal{C}V$, and $V(l)$ is the lth component of V. Given that the above gives a non-negative result for all $i = 1, \dots, N$ and $l = 1, \dots, N$, the vector field described by $\mathcal{C}$ is increasing.

At present, our analysis has not identified whether the components of the Jacobian are non-negative and this part of the investigation is future work. If the components were found to be negative this does not necessarily mean that LCS Value Iteration diverges. The proof given in Appendix A is only a sufficient, but not a necessary condition for a vector field to be increasing. Thus, even if not all of the components of the Jacobian are non-negative, the vector field can still be increasing. Furthermore, our search for an increasing vector field is based on Lemma 3.2, which gives a sufficient but not necessary condition for non-expansion mappings. So, even if the vector field described by $\mathcal{C}$ is found to violate the increasing property, we cannot conclude that LCS Value Iteration is not guaranteed to converge.

As we have not yet been able to produce a proof that the vector field given by $\mathcal{C}$ is increasing, but neither were we able to find examples where it violates that property, we will state it as a conjecture, pending further investigation.

Conjecture 3.10. *The vector field given by $\mathcal{C}$ is increasing.*

This leads to the following result:

Theorem 3.11. *If Conjecture 3.10 holds, then LCS Value Iteration with accuracy-based mixing converges to its fixed point*

$$\tilde{V}^* = \mathcal{C}T\tilde{V}^*.$$

Proof. By Lemma 3.9, the vector field described by $\mathcal{C}$ is a scalar shift vector field with scaling 1. Combining this with its increasing property given by Conjecture 3.10, Lemma 3.2 shows that $\mathcal{C}$ is a non-expansion w.r.t. the maximum norm. As LCS Value Iteration is based on the iteration

$$\tilde{V}_{t+1} = \mathcal{C}T\tilde{V}_t,$$

and the DP update T is by Lemma 3.3 a contraction with contraction modulus γ, the operator conjunction $\mathcal{C}T$ describes a contraction to the maximum norm. Hence, by Theorem 3.1 the iteration converges to its unique fixed point. □

3.7 Handling Two-Step Iterations

In [4] the Value Iteration was described as a two-step iteration, even though we now see that it can be expressed as an iteration that only involves a single step. For completeness, we will explain how our analysis can be expanded to capture iterations of more than one step.

Such a modification will make the new value function estimate $\tilde{V}_{t+1}$ a function of both $\tilde{V}_t$ and $\tilde{V}_{t-1}$. To transform this method conceptually into a one-step iteration, we will introduce the variable U that at time t keeps the value of $\tilde{V}_{t-1}$, that is $U_t = \tilde{V}_{t-1}$. By concatenating vectors $\tilde{V}_t$ and U_t to vector $(\tilde{V}_t(1), \dots, \tilde{V}_t(N), U_t(1), \dots, U_t(N))'$ we can describe the 2-step iteration by

$$\begin{pmatrix} \tilde{V}_{t+1} \\ - \\ \tilde{V}_t \end{pmatrix} = \begin{pmatrix} \mathcal{C}T\tilde{V}_t \\ - \\ \tilde{V}_{t-1} \end{pmatrix}.$$

Note that $\mathcal{C}$ now depends on $\tilde{V}_t$ as well as $\tilde{V}_{t-1}$.

One step of this iteration is unlikely to lead to a contraction, as the new U_{t+1} is simply a copy of our current value function estimate $\tilde{V}_t$. As the upper half of the vector we are operating on is based on the original LCS Value Iteration, the iteration will not cause an expansion either. Applying the iteration twice gives the following update

$$\begin{pmatrix} \tilde{V}_{t+2} \\ - \\ \tilde{V}_{t+1} \end{pmatrix} = \begin{pmatrix} \mathcal{C}T\mathcal{C}T\tilde{V}_t \\ - \\ \mathcal{C}T\tilde{V}_t \end{pmatrix},$$

which is a contraction for both the upper and the lower half of the vector. Hence, we can still guarantee convergence of the iteration. Even though our argument is only informal, the same can be shown formally by defining a new update operator that performs an update on the concatenated vector and investigating the properties of the vector field that it defines. Additionally, the argument can be extended to any n-step iteration for a finite n, by concatenating the last n value function estimates into a single vector, and observing contraction after n iterations.

4 Conclusion

We have described the LCS Value Iteration algorithm and have given a proof of its convergence, based on the contraction of the DP update and the non-expansion of the LCS function approximation, and depending on a conjecture about the nature of accuracy-based mixing. Additionally we have shown convergence for fixed classifier mixing and have demonstrated that we cannot guarantee convergence for all kinds of classifier mixing function. To deal with accuracy-based mixing, we have introduced the operator $\mathcal{C}$ that described LCS function approximation with classifier mixing based on some inverse of the approximation error. Treating this operator as a vector field allowed us to show that it is a scalar shift vector field, and stated the conjecture that it is also increasing. Proving this conjecture is still an open question, but we have described a possible approach that is based on the non-negativity of the operator's Jacobian.

Convergence of LCS Value Iteration is an important property, as it is the first step in answering the question of whether we can safely use Q-Learning in LCS. In addition, it demonstrates approaches to investigating the stability of LCS function approximation in the reinforcement learning framework. Even though we are currently only dealing with constant populations of classifiers, showing convergence to a population-dependent fixed point allows us to use this work even when we are changing the population while we are performing the iteration. In that case, the fixed point would change, but every iteration after that change would bring us closer to the new fixed point. Naturally, we cannot ignore that the new population depends on the previous value function estimate, and analysing this interaction is a topic of future research.

Having guaranteed convergence to a unique solution is a strong property that makes classifier systems better candidates for real-world application, such as, for example, optimal control. Hence, following this track of research for LCS will be fruitful for a wide range of applications that have not previously been considered before due to the lack of theoretical guarantees.

Acknowledgements Thanks to Jonty Needham for being patient enough to listen to a large number of naïve math questions, and to answer some of them in an understandable, and sometimes not-so-understandable way. Additional thanks go to Prof Dmitri Vassiliev for hints on how to handle multi-step iterations.

References

[1] Howard Anton. *Calculus.* John Wiley & Sons, New York, 5th edition, 1995.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[3] Jan Drugowitsch and Alwyn M. Barry. A Formal Framework and Extensions for Function Approximation in Learning Classifier Systems. Technical Report CSBU2006-01, Dept. Computer Science, University of Bath, January 2006. ISSN 1740-9497.

[4] Jan Drugowitsch and Alwyn M. Barry. A Formal Framework for Reinforcement Learning with Function Approximation in Learning Classifier Systems. Technical Report CSBU2006-02, Dept. Computer Science, University of Bath, January 2006. ISSN 1740-9497.

[5] W. A. Sutherland. *Introduction to metric and topological spaces.* Clarendon Press, Oxford, UK, 1975.

A Increasing Vector Fields

As there is no default definition of what it means for a vector field to be increasing, there is neither a default monotonicity criterion for vector fields. Hence, in this section we will derive a monotonicity criterion for vector fields based on the ordering

$$x \leq y \quad \Leftrightarrow \quad x(i) \leq y(i), \quad i \in 1, \dots, N,$$

where x and y are real vectors with N components.

Let us first derive the mean-value theorem for functions of several variables:

Theorem A.1 (Mean-Value Theorem for Functions of Several Variables). *Let $f : \mathbb{R}^N \to \mathbb{R}$ be a function that is continuous on $[a, b]$ and is differentiable on (a, b). Then there exists a vector $c \in \mathbb{R}^N : a < c < b$ such that*

$$(b - a)' \nabla f(c) = f(b) - f(a).$$

Proof. Define $d = b - a$ and a continuous function $F : \mathbb{R} \to \mathbb{R}$, given by

$$F(t) = f(a + td).$$

Hence, $F(0) = f(a)$, and $F(1) = f(b)$. By the nature of f, F is continuous on $[0, 1]$ and differentiable on $(0, 1)$. By the mean-value theorem (e.g. [1, Ch. 4.9]) there exists a $t \in \mathbb{R} : 0 < t < 1$, such that

$$\frac{\partial F(t)}{\partial t} = \frac{F(1) - F(0)}{1 - 0} = f(b) - f(a).$$

The derivative of F w.r.t. t is by given by

$$\frac{\partial F(t)}{\partial t} = d' \nabla f(a + td).$$

Denoting $c = a + td$, we can substitute for $\frac{\partial F(t)}{\partial t}$ in above equation to get

$$(b - a)' \nabla f(c) = f(b) - f(a).$$

□

As a vector field $f : \mathbb{R}^N \to \mathbb{R}^N$ can be seen as a set of N functions $f_i : \mathbb{R}^N \to \mathbb{R}$ such that f_i defines the ith component of f, we can use the above theorem to get the following:

Theorem A.2 (Increasing Vector Fields). *Let $f : \mathbb{R}^N \to \mathbb{R}^N$ be a vector field with components f_i, $i = 1, \dots, N$, where f_i defines the ith component of f. Let f_i be continuous on $[a, b]$ and differentiable on (a, b), for all $i = 1, \dots, N$. Then, given that the Jacobian of f on the interval (a, b) is non-negative, the vector field is increasing.*

Proof. f is increasing if and only if all of its components f_i are increasing. Hence, we will demonstrate for an arbitrary i that f_i is increasing, given that all components of its gradient ∇f_i are non-negative, which is satisfied by the non-negative Jacobian of f.

Let $x_1 \in \mathbb{R}^N$ and $x_2 \in \mathbb{R}^N$ be two vectors in the interval $[a, b]$, such that $x_2 > x_1$, and let $x \in \mathbb{R}^N$ be a vector such that $x_1 < x < x_2$, that is $x \in (x_1, x_2)$. Hence, by Theorem A.1 we have

$$(x_2 - x_1)' \nabla f_i(x) = f_i(x_2) - f_i(x_1).$$

By $x_1 < x_2$ we know that all components of $x_2 - x_1$ are non-negative. Having a non-negative gradient implies by above equation that $f_i(x_1) \leq f_i(x_2)$. As this applies to all f_i, $i = 1, \dots, N$, the vector field given by f is increasing. □

B Jacobian of $\mathcal{C}$

In this section we will derive the Jacobian of our LCS approximation operator $\mathcal{C}$. As $\mathcal{C}$ is a function from $\mathbb{R}^N$ to $\mathbb{R}^N$, we will denote the ith component of $\mathcal{C}V$ by $\mathcal{C}_i V$ (which is the same as $(\mathcal{C}V)(i)$). To get the Jacobian of $\mathcal{C}$, we need to derive

$$\frac{\partial \mathcal{C}_i V}{\partial V(l)}, \qquad i = 1, \dots, N, \quad l = 1, \dots, N.$$

As $\mathcal{C}$ is a function of the approximation errors $\varepsilon_k : \mathbb{R}^N \rightarrow \mathbb{R}^+$, let us first derive the error's gradient. The error is given by

$$\varepsilon_k(V) = \frac{1}{\mathrm{Tr}(D_k)} \sum_{i \in S} I_{S_k}(i)\pi(i)\left(V(i) - V_k\right)^2,$$

where we write $V_k \in \mathbb{R}$ for the approximation of V by classifier k, given by $\Pi_{D_k} V$. Even though $\Pi_{D_k} V$ returns a vector, this vector's components are all the same, which is why we can represent them by the scalar V_k, independent of the state. Deriving the gradient of the error ε_k gives

$$\frac{\partial \varepsilon_k(V)}{\partial V(l)} = 2\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}\left(V(l) - V_k\right),$$

which is the difference between the value of state l and classifier k's approximation, weighted by the state distribution and classifier k's matching of that state. As we usually operate on the inverse $\varepsilon_k(V)^{-\nu}$ of the error, we require the gradient of this inverse, which is given by

$$\frac{\partial \varepsilon_k(V)^{-\nu}}{\partial V(l)} = -2\nu\varepsilon_k(V)^{-(\nu+1)}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k).$$

The next step is to derive the gradient of the mixing weights $\psi_k : S \times \mathbb{R}^N \rightarrow [0, 1]$. They are defined by

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

with a gradient of

$$\frac{\partial \psi_k(i, V)}{\partial V(l)} = \frac{I_{S_k}(i)\frac{\partial \varepsilon_k(V)^{-\nu}}{\partial V(l)}}{\sum_{p=1}^K I_{S_p}(i)\varepsilon_p(V)^{-\nu}} - \frac{I_{S_k}(i)\varepsilon_k(V)^{-\nu}\frac{\partial}{\partial V(l)}\left(\sum_{p=1}^K I_{S_p}(i)\varepsilon_p(V)^{-\nu}\right)}{\left(\sum_{p=1}^K I_{S_p}(i)\varepsilon_p(V)^{-\nu}\right)^2}.$$

Substituting for the error gradient and using $\varepsilon_k(V)^{-(\nu+1)} = \varepsilon_k(V)^{-\nu}\varepsilon_k(V)^{-1}$ results in

$$\begin{aligned}\frac{\partial \psi_k(V)^{-\nu}}{\partial V(l)} &= 2\nu\psi_k(i, V)\sum_{p=1}^K \psi_p(i, V)\varepsilon_p(V)^{-1}\frac{I_{S_p}(l)\pi(l)}{\mathrm{Tr}(D_p)}(V(l) - V_p) \\ &\quad -2\nu\psi_k(i, V)\varepsilon_k(V)^{-1}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k)\end{aligned}$$

To get the gradient of $\mathcal{C}$, which is defined by

$$\mathcal{C}_i(V) = \sum_{k=1}^K \psi_k(i, V)V_k,$$

we will combine

$$\sum_{k=1}^K \psi_k(i, V)\frac{\partial V_k}{\partial V(l)} = \frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)},$$

and

$$\begin{aligned}\sum_{k=1}^K \frac{\partial \psi_k(i, V)}{\partial V(l)}V_k &= 2\nu\sum_{k=1}^K \psi_k(i, V)\sum_{p=1}^K \psi_p(i, V)\varepsilon_p(V)^{-1}\frac{I_{S_p}(l)\pi(l)}{\mathrm{Tr}(D_p)}(V(l) - V_p)V_k \\ &\quad -2\nu\sum_{k=1}^K \psi_k(i, V)\varepsilon_k(V)^{-1}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k)V_k \\ &= 2\nu\sum_{k=1}^K \psi_k(i, V)\varepsilon_k(V)^{-1}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k)\sum_{p=1}^K \psi_p(i, V)V_p \\ &\quad -2\nu\sum_{k=1}^K \psi_k(i, V)\varepsilon_k(V)^{-1}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k)V_k \\ &= 2\nu\sum_{k=1}^K \psi_k(i, V)\varepsilon_k(V)^{-1}\frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)}(V(l) - V_k)(\mathcal{C}_i V - V_k),\end{aligned}$$

to get

$$\frac{\partial \mathcal{C}_i V}{\partial V(l)} = \sum_{k=1}^{K} \psi_k(i, V) \frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)} \left(1 + 2\nu\varepsilon_k(V)^{-1}(V(l) - V_k)(\mathcal{C}_i V - V_k)\right).$$

That defines all values of the Jacobian of $\mathcal{C}$.