TECHNISCHE UNIVERSITÄT WIEN

Fuzzy Clustering: eine Anwendung auf distributionelles Reinforcement Learning

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Mathematik

eingereicht von

Julien Malle
Matrikelnummer 11832435

an der Fakultät für Mathematik und Geoinformation

der Technischen Universität Wien

Betreuung: Prof. Dr.techn. Dipl.-Ing. Clemens Heitzinger

Wien, 27. Jänner 2021

Julien Malle	Clemens Heitzinger

TECHNISCHE UNIVERSITÄT WIEN

Fuzzy Clustering: an Application to Distributional Reinforcement Learning

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Technical Mathematics

by

Julien Malle
Registration Number 11832435

to the Faculty of Mathematics and Geoinformation

at the TU Wien

Advisor: Prof. Dr.techn. Dipl.-Ing. Clemens Heitzinger

Vienna, 27th January, 2021

Julien Malle	Clemens Heitzinger

Erklärung zur Verfassung der Arbeit

Julien Malle

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Jänner 2021

Julien Malle

Kurzfassung

Die meisten Reinforcement Learning-Algorithmen werden mit der Einschränkung eines diskreten (endlichen) Zustandsraums untersucht. Kompliziertere Zustandsräume werden normalerweise durch Funktionsnäherung behandelt, für die nur wenige theoretische Ergebnisse verfügbar sind. In dieser Arbeit wird eine clusterbasierte Näherung für kontinuierliche Zustandsräume untersucht. Die stückweise konstante Näherung, die durch (klassisches) hartes Clustering erhalten wird, wird empirisch unter Verwendung von Fuzzy-Menge und Zugehörigkeitsfunktionen verbessert. Wir untersuchen auch, wie das Clustering selbst mithilfe von Zugehörigkeitsfunktionen automatisiert werden kann, die auf das bekannte MNIST-Problem angewendet werden.

Abstract

Most Reinforcement Learning algorithms are studied with the restriction of a discrete (finite) state space. More complicated state spaces are usually handled through function approximation, for which few theoretical results are available. In this paper, a clustering-based approximation for continuous state spaces is studied. The piecewise constant approximation obtained by (classical) hard clustering is enhanced empirically using fuzzy membership functions. We also look at how the clustering itself could be automated, using membership functions applied to the well-known MNIST digit-recognition problem.

Contents

CHAPTER 1

Introduction

Reinforcement Learning (RL) is defined in [21] as "learning what to do—how to map situations to actions—so as to maximize a numerical reward signal". "Situations" are modeled as Markov Decision Processes (MDP), featuring a *state space* $\mathcal{X}$ and an *action space* $\mathcal{A}$. The goal is to model an *agent* that will take actions at given time-steps in order to maximize some reward function. An agent is characterized by a particular mapping from states to actions, called a *policy* and often noted π. For some state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$, $\pi(a|x)$ represents the probability that action a is taken while in state x.

In practice, RL algorithm have as a goal to estimate *value functions*, defined with respect to some policy π:

$$v_\pi(x) := \mathbb{E}_\pi \left[\sum_k \gamma^k R_{t+k+1} \middle| X_t = x \right], \quad x \in \mathcal{X}, \tag{1.1}$$

where R_t (resp. X_t) is the reward (resp. state) at time-step t, and γ is the *discount factor*. $v_\pi(x)$ represents the expected sum of (discounted) rewards achieved from x by following policy π. Alternatively, algorithms such as Q-learning aim to estimate

$$q_\pi(x, a) := \mathbb{E}_\pi \left[\sum_k \gamma^k R_{t+k+1} \middle| X_t = x, A_t = a \right], \quad (x, a) \in \mathcal{X} \times \mathcal{A}, \tag{1.2}$$

where A_t is the action taken at time-step t. v_π is called *state-value function* and q_π *action-value function*.

In *Distributional* Reinforcement Learning, which the following focuses on, distributions of value functions are modelled rather than their expected values. This arguably allows for a better modelling of the process [3] and was shown to produce state-of-the-art results

on the Arcade Learning Environment [2].

Most RL algorithm rely on the assumption that the state-action space $\mathcal{X} \times \mathcal{A}$ is finite (this is often referred to as the *tabular* case; see e.g. [21] Chap. 3.7). In this simplifying assumption, value function are easily modelled and convergence can be proved for several algorithm. The handling of more complicated state spaces, e.g. $\mathcal{X} \subset \mathbb{R}^d$ continuous, can be done through function approximation. However interaction between function approximation and (distributional) RL, as well as convergence of the resulting algorithms, is still an open question.

The main goal of this work is to introduce a fuzzy clustering based approach to function approximation, that we translate to an adapted distributional RL algorithm that explicitly handles a continuous state space. It is organized in the following way: in Section 2, we first briefly review clustering solutions that later allow for a discretization of the state space, and we further translate this to function approximation in Section 3. In Section 4, we introduce an adapted version of the CDRL algorithm, and we investigate in Section 5 how the proposed function approximation could be used in an optimized state representation. Lastly, we introduce in Section 6 a theoretical framework to study the proposed algorithm and provide proofs for the hard-clustering case.

CHAPTER 2

K-Means and Fuzzy C-Means

In this section, we introduce the well-known clustering algorithms K-Means (KM) and Fuzzy C-Means (FCM), that may serve as discretization schemes. In the case of KM (subsection 2.1), the discretization is very straight-forward and corresponds to the clustering produced. Similarly, FCM produces a *fuzzy* clustering, relying on fuzzy sets and membership functions, that we may call a *fuzzy discretization* (subsection 2.2). Finally, in subsection 2.3, we point to a problem in the initialization of the FCM algorithm and thus argue for the use of KM for all initializations in this work.

2.1 K-Means

The K-Means (KM) algorithm (see e.g. [15]), aims at producing a set of cluster centers $\mathcal{Z} = \{z_1, ..., z_K\}$ that, for a given dataset $\mathcal{X} \subset \mathbb{R}^d$, minimizes the objective function

$$\Phi = \sum_{z \in \mathcal{Z}} \sum_{x \in \mathcal{X}(z)} \|x - z\|^2 \tag{2.1}$$

where $\mathcal{X}(z) = \{x \in \mathcal{X} : z = \arg\min_{\hat{z} \in \mathcal{Z}} \|x - \hat{z}\|^2\}$. The algorithm consists, after initialization, in the repetition of the following two steps:

- update each $z \in \mathcal{Z}$ to be the centre of $\mathcal{X}(z)$:

$$\forall z \in \mathcal{Z}: \quad z := \frac{1}{|\mathcal{X}(z)|} \sum_{x \in \mathcal{X}(z)} x, \tag{2.2}$$

- update each $\mathcal{X}(z)$ using their definitions.

The proof of convergence towards a (local) minimum of the objective function relies on the fact that, for any subset (or *cluster*) $C \subset \mathcal{X}$, the element $c \in \mathbb{R}^d$ minimizing

$\sum_{x \in C} \|x - c\|^2$ is precisely the center of C. For a uniformly random initialization (that is, K elements of $\mathcal{X}$ are picked as initial cluster centers uniformly at random), there exists no guarantee with respect to closeness to the actual global optimum of the objective function ; however when the initialization is not done *uniformly* at random but using the K-Means++ initialization [1], then the expected outcome is upper-bounded by $8(\ln K + 2) \times \Phi_{\text{opt}}$, where Φ_{opt} is the global minimum of Φ.

2.2 Fuzzy sets and the Fuzzy C-Means algorithm

The Fuzzy C-Means (FCM) algorithm, first introduced in [8] and later improved in [5], provides a similar procedure than that of KM, applied to fuzzy sets. We choose to look at these sets through the concept of membership functions. *Hard* sets correspond to the usual idea of sets; they can be represented by indicator functions

$$\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

Similarly, *fuzzy* sets are defined through indicator functions, that can however take (continuous) values in $[0, 1]$. In this case, the indicator function is often called *membership* function.

The FCM algorithm, given $C \in \mathbb{N}$, $m \in \mathbb{R}, m > 1$ and a given dataset $\mathcal{X} \subset \mathbb{R}^d$, aims at producing a set of centers, or *representatives* $\mathcal{Z} = \{z_1, ..., z_C\}$ and corresponding membership functions $\mu_1, ...\mu_C$ that minimize the objective function

$$\Psi(\mu_1, \ldots, \mu_C, z_1, \ldots, z_C) = \sum_{x \in \mathcal{X}} \sum_{j=1}^{C} \mu_j(x)^m \|x - z_j\|^2 \tag{2.3}$$

under the condition that $\sum_j \mu_j = 1$. Using a Lagrange multiplier with this condition, it is found (e.g. in [5]) that the repetition of the following redefinition of the representatives and memberships provides as outcome a local minimum of Ψ:

- update each $z_j, j \in \{1, \ldots, C\}$ to be the μ_j-weighted center of $\mathcal{X}$:

$$\forall j \in \{1, ..., C\} : \quad z_j := \frac{\sum_{x \in \mathcal{X}} \mu_j(x)^m x}{\sum_{x \in \mathcal{X}} \mu_j(x)^m} \tag{2.4}$$

- update each $\mu_j, j \in \{1, \ldots, C\}$ with the following expression:

$$\mu_j(x) := \left[\sum_{k=1}^{C} \left(\frac{\|x - z_j\|}{\|x - z_k\|} \right)^{\frac{2}{m-1}} \right]^{-1}. \tag{2.5}$$

The hyper-parameter m is often called *fuzzifier*, in the sense that it controls "how fuzzy" the clustering will be; especially we have the following asymptotic results:

- For $m \longrightarrow \infty$, all μ_j are equal and the representatives coincide at the centre of $\mathcal{X}$.
- For $m \longrightarrow 1$, the FCM algorithm degenerates into KM [5].

On the second point, given that all representatives are distinct, note that when $m \longrightarrow 1$, the expressions for the membership functions give

$$\mu_j(x) = \begin{cases} 1 \text{ if } z_j \text{ is the representative closest to } x, \\ 0 \text{ else.} \end{cases} \tag{2.6}$$

2.3 Limits of the FCM algorithm

A major limitation of the FCM algorithm stems from the fact that representatives can coincide with one another; it is often the case that the algorithm outputs C times the centre of the data with all-equal memberships, especially when considering high dimensional space for the dataset [23] or high number of representatives.
We find a possible explanation for the latter issue in the details of implementations (see [7, 12] for example), and especially in the initialization: the algorithm in [5] initializes "uniformly" the memberships and obtains the representatives using (2.4). However, a uniform initialization of the membership functions is not so trivial; in [7, 12], it is initialized as

$$\mu_j = \frac{\nu_j}{\sum_k \nu_k}, \quad \text{where } \nu_k \sim \mathcal{U}(0,1) \text{ i.i.d.} \tag{2.7}$$

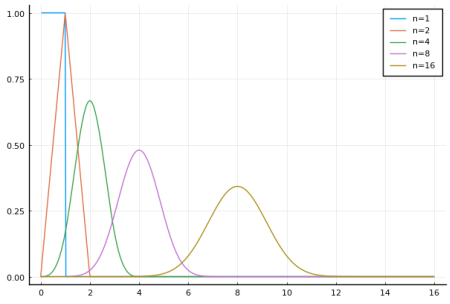
Although this is quite intuitive to satisfy the $\sum_j \mu_j = 1$ condition, the resulting distribution for μ_j is far from uniform. Its density function f_{μ_j} can be written as

$$f_{\mu_j}(u) = \frac{1}{(1-u)^2} \int_0^{\frac{1-u}{u}} v f_{\hat{\nu}_j}(v) dv, \quad u \in (0,1), \tag{2.8}$$

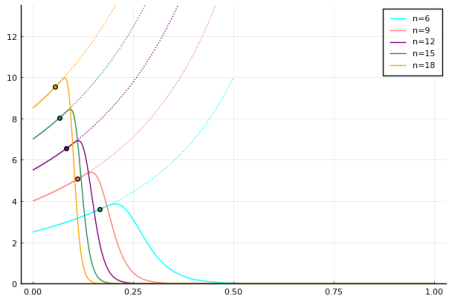
where $\hat{\nu}_j = \sum_{k \neq j} \nu_k$ and $f_{\hat{\nu}_j}$ is the density of the Irwin-Hall distribution of parameter $C-1$ [11, 10]. This density consists in a piecewise polynomial function, but its full expression holds little or no insight; it is however represented together with f_{μ_j} in Fig 2.1a and 2.1b.

Fig 2.1c and 2.1d are generated using a 2 dimensional dataset of $n = 5000$ samples generated by a uniform distribution in $(0,1)^2$. We apply FCM and k-means++ initializations for $C = 50$. We represent in Fig 2.1c an histogram of the maximal value across the obtained membership functions for every sample: no one exceeds $5e-4$. The resulting cluster representatives can be seen in red in Fig 2.1d, where we also plot the result of the k-means++ initialization in green. This explains why it is usual to obtain C times the center of the data when running the FCM algorithm (note that this situation is a fixed point of the algorithm).

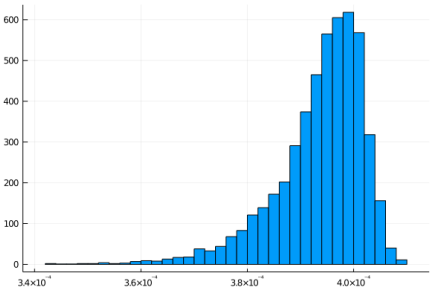
Overall, it is impractical to use the FCM algorithm, especially in our context of discretization where we do not seek actual clusters (that is, "compact and well separated" as defined by Dunn in [9]). On these grounds, we shall prefer the KM algorithm (together with the k-means++ initialization) to perform *a priori* discretization.



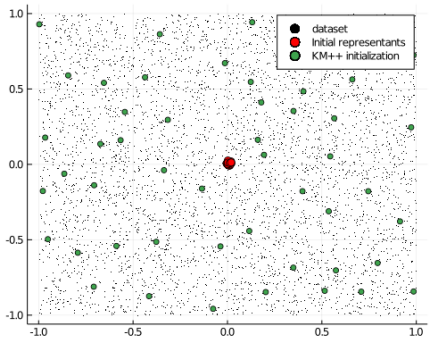
(a) Irwin-Hall distribution



(b) Distribution for the FCM initialization



(c) Test results for FCM initialization



(d) k-means++ vs FCM initialization

Figure 2.1: Problems in the initialization of FCM

CHAPTER 3

From Clustering to Function Approximation

In this section, we consider a function $f\colon \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ and a set of cluster centers $\mathcal{Z}$, that is associated to a (fuzzy) clustering of $\mathcal{X}$. In subsection 3.1, we find an optimal piece-wise constant approximation of f with respect to the hard clustering of $\mathcal{X}$ associated with $\mathcal{Z}$. In subsection 3.2, we introduce several ways to perform a *fuzzy approximation*, based on a fuzzy clustering of $\mathcal{X}$ associated with $\mathcal{Z}$. Finally, we test in subsection 3.3 these approximations on several classical functions.

3.1 K-Means and Piecewise Constant Approximation

The (hard) clustering provided by the K-Means algorithm can naturally be used to perform a discretization of the state space $\mathcal{X}$. Let us consider $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ a function to be approximated, together with a set of cluster centers $\mathcal{Z} = \{z_1, \dots, z_K\}$ which naturally gives a Voronoï partition $\mathcal{X} = \bigcup_{z\in\mathcal{Z}} \mathcal{X}(z)$. Considering that $\mathcal{X}$ is bounded, then each $\mathcal{X}(z)$, $z \in \mathcal{Z}$, is bounded aswell. We seek the optimal (constant) value $t_0 \in \mathbb{R}$ for the constant approximation of f over $\mathcal{X}(z)$, for any $z \in \mathcal{Z}$; for this we consider a scalar product for function defined on $\mathcal{X}(z)$ $\langle ., . \rangle_{\mathcal{X}(z)}$ together with its associated norm $\|.\|_{\mathcal{X}(z)}$, and for any $t \in \mathbb{R}$ use the same letter to denote the scalar constant and the function which value is constant equal to t. We have classically that

$$\|f - t\|^2_{\mathcal{X}(z)} = \|f - t_0\|^2_{\mathcal{X}(z)} + 2\langle f - t_0, t_0 - t\rangle_{\mathcal{X}(z)} + \|t_0 - t\|^2_{\mathcal{X}(z)}. \tag{3.1}$$

Since $\langle f - t_0, t_0 - t\rangle_{\mathcal{X}(z)} = (t_0 - t)\Big(\langle f, 1\rangle_{\mathcal{X}(z)} - t_0\langle 1, 1\rangle_{\mathcal{X}(z)}\Big)$, we find the optimal value t_0 (with respect to the chosen scalar product) as

$$t_0 = \frac{\langle f, 1\rangle_{\mathcal{X}(z)}}{\langle 1, 1\rangle_{\mathcal{X}(z)}}. \tag{3.2}$$

Choosing the euclidean scalar product yields an optimal value equal to the mean of f over $\mathcal{X}(z)$:

$$t_0 = \frac{1}{\lambda(\mathcal{X}(z))} \int_{\mathcal{X}(z)} f d\lambda. \tag{3.3}$$

3.2 Fuzzy Approximation

We now consider our function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$ to be approximated together with a set of cluster representatives $\mathcal{Z} = \{z_1, \ldots, z_K\}$, and membership functions $\mu_1, \ldots, \mu_K$ corresponding to a fuzzifier $m > 1$ with expressions defined in (2.5). We note that this expression relates to Shepard's Inverse Distance Weighting [19], and use the interpolation

$$\tilde{f}(x) = \sum_{j=1}^{C} \mu_j(x)\alpha_j \tag{3.4}$$

accordingly, where $\alpha_j \in \mathbb{R}, forall j \in \{1, \ldots, C\}$. It should be noted that, contrary to the previous section, a (weighted) mean of f does not yield an "optimal" value for $\alpha_1, \ldots, \alpha_C$ in general. The Inverse Distance Weighting was originally developed to interpolate values out of observation at irregularly scattered points; in that setting, α_j is set to the value observed at point z_j. This is supported by the fact that, given (3.4), $\tilde{f}(z_j) = \alpha_j$ for all j. In the rest of this paper, we may use α_j or $f(z_j)$ indistinctly. In our setting, observations do not coincide with cluster representatives: we shall intuitively project values observed at points $x \in \mathcal{X}$ to $\tilde{f}(z_j) = \alpha_j$ at a rate proportional to $\mu_j(x)$.
As (3.4) is precisely the Inverse Distance Weighted interpolation described in [19], we can also recover improvement to this formula suggested in the same paper: specifically, the expression of the membership functions implies that the interpolated function $\tilde{f}$ has zero-gradient at each representative z_j, which is quite clearly an undesired condition. We define the alternative interpolation

$$\tilde{f}(x) := \sum_j \mu_j(x) \left[f(z_j) + \Delta f(z_j, x) \right], \quad \text{where } \begin{cases} \lim\limits_{x \to z_j} \Delta f(z_j, x) = 0, \\ \lim\limits_{x \to z_j} \nabla_x (\Delta f(z_j, x)) = \nabla f(z_j), \\ \lim\limits_{\|x - z_j\| \to \infty} \Delta f(z_j, x) = 0, \end{cases} \tag{3.5}$$

where $\nabla f(z_j)$ is the desired gradient at z_j defined by Shepard as

$$\nabla f(z_j) := \sum_{k \neq j} \nu_j(z_k) \frac{f(z_k) - f(z_j)}{\|z_k - z_j\|^2} (z_k - z_j), \quad \text{with } \nu_j(z_k) = \left[\sum_{l \neq k} \left(\frac{\|z_k - z_j\|}{\|z_l - z_j\|} \right)^{\frac{2}{m-1}} \right]^{-1}. \tag{3.6}$$

3.3 Experiments

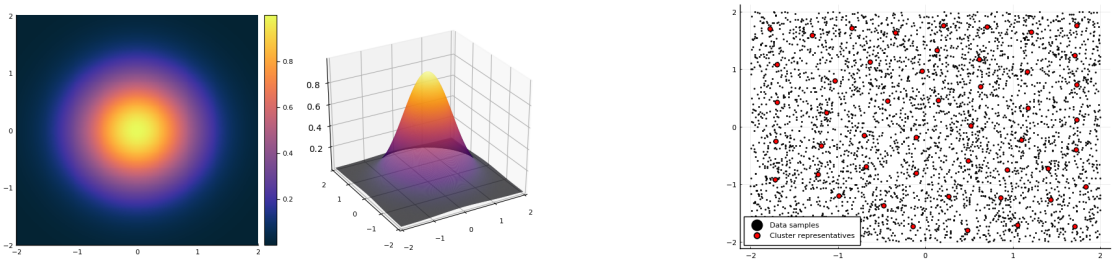
In this subsection, we show results of the three approximation methods presented above (piece-wise constant approximation, fuzzy approximation with and without gradients).

The function to approximate is $f(x) = e^{-\|x\|^2}$ (represented in Fig 3.1a), where $\|.\|$ is the euclidean norm and $x \in [-2, 2]^2$. The process is as follows:

- We create a dataset $\mathcal{X}$ consisting of 4000 points drawn uniformly in $[-2, 2]^2$;
- we perform an *a priori* discretization of $[-2, 2]^2$ by running the K-Means algorithm on this dataset ($K = 50$, see Fig 3.1b);
- for each cluster representative obtained, we approximate the cluster representative's value (this depends on the approximation method);
- we compute the approximated function $\tilde{f}$ and measure the error E by the following approximation:

$$E(f, \tilde{f}) = \sqrt{\int_{[-2,2]^2} (f - \tilde{f})^2 d\lambda} \approx \sqrt{\frac{1}{|I|}\frac{1}{|J|} \sum_{i \in I, j \in J} (f(x_i, y_j) - \tilde{f}(x_i, y_j))^2}, \quad (3.7)$$

  where $\{(x_i, y_j)\}_{(i,j) \in I \times J}$ are evenly spaced across $[-2, 2]^2$.



(a) $f(x) = e^{-\|x\|^2}$ in 2 dimensions (b) Discretization from K-Means

Figure 3.1: A test case for cluster-based function approximation

For the piece-wise constant approximation (see Fig 3.2), we assign as value for each cluster the mean of all samples that fall into said cluster (it is optimal with respect to the considered error, see section 3.1). There is no surprise for the obtained result, which has the highest error among the three methods.

For the fuzzy approximation (without additional gradients), we choose a fuzzifier $m = 1.5$ and assign as value for each fuzzy cluster the membership-weighted mean of the observed values

$$\alpha_j := \frac{\sum_{x \in \mathcal{X}} \mu_j(x) f(x)}{\sum_{x \in \mathcal{X}} \mu_j(x)}. \quad (3.8)$$

The resulting interpolated function is shown in Fig 3.3 and achieves a lower error than the piece-wise constant approximation: it is essentially a smoothed version of it.

We also show in Fig 3.4 the interpolated function for more value of the fuzzifier. It is interesting to see that, as mentioned in section 2.2, the behavior of the fuzzy approximation

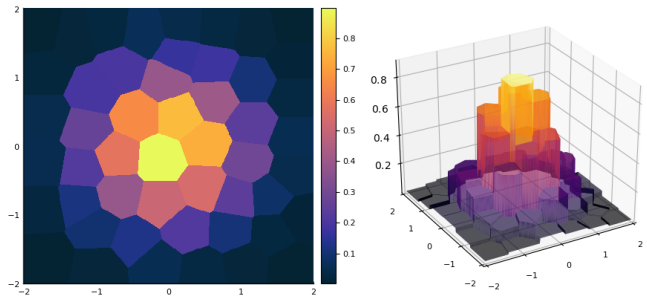


Figure 3.2: Piece-wise constant approximation ($E \approx 0.072$)

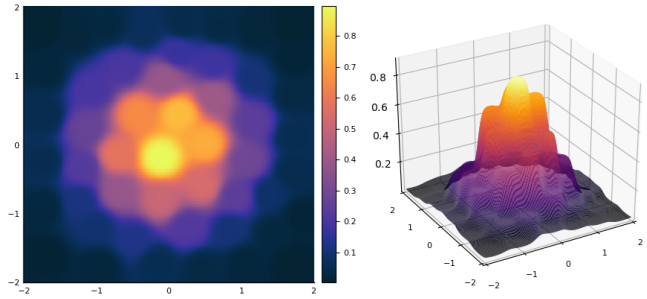


Figure 3.3: Fuzzy approximation ($E \approx 0.053$)

tends to be that of the piece-wise constant approximation as the fuzzifier tends to 1. Additionally, when the fuzzifier is too high, the interpolation becomes noisy as values from points further away are given important weight.

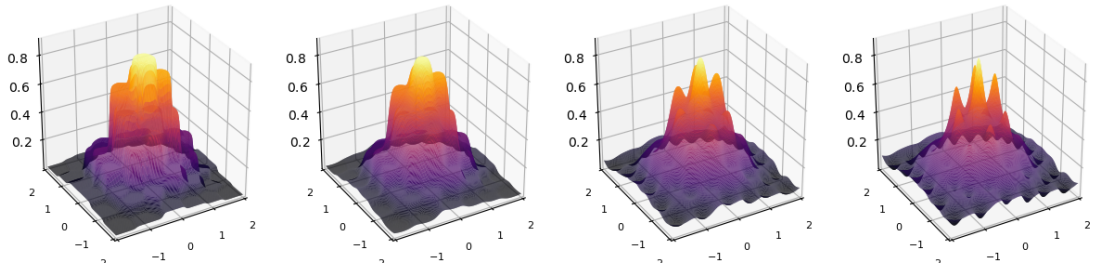


Figure 3.4: Fuzzy approximation for different values of the fuzzifier (from left to right: 1.2, 1.5, 1.8, 2)

For the fuzzy approximation with gradients, we remain in the same setting as for the one without gradients, and use the following expression for Δf, which satisfies all prerequisites of (3.5):

$$\Delta f(z_j, x) := e^{-\|x-z_j\|^2}(\nabla f(z_j) \cdot (x - z_j)). \tag{3.9}$$

This method yields the lowest error among all three approximation schemes and its shape and the shape of f look more alike; it is however more expansive to compute.

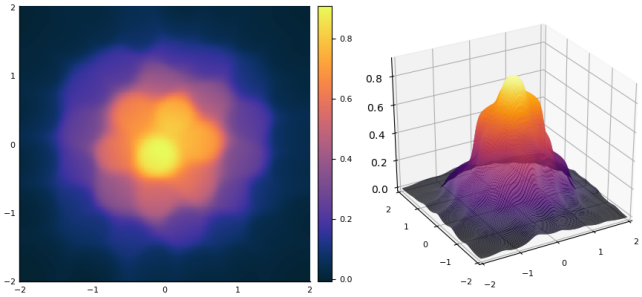


Figure 3.5: Fuzzy approximation with gradients ($E \approx 0.037$)

We also provide in Table 3.1 a comparison of errors when approximating other functions with the above 3 schemes (piece-wise constant approximation, fuzzy approximation, fuzzy approximation with gradients). For all considered functions to approximate, the fuzzy approximation with gradients provided the best results.

Table 3.1: Comparison of approximation errors for multiple functions.

	Piece-wise constant	**Fuzzy**	**Fuzzy with gradients**
$x \mapsto e^{-\|x\|^2}$	0.072	0.053	**0.037**
$x \mapsto \tanh \|x\|^2$	0.076	0.054	**0.036**
$x \mapsto \frac{1}{1+\|x\|^2}$	0.056	0.038	**0.024**
$x \mapsto \sin \|x\|^2$	0.36	0.32	**0.28**
$x \mapsto \|x\|^2$	0.54	0.41	**0.28**

3.4 Choosing the Fuzzifier

Choosing a correct value for the fuzzifier is an ongoing research problem (see e.g. [13] and [23]). In Fig 3.6, we represent the approximation error as a function of the data samples' dimension (the function to approximate remains $x \mapsto e^{-\|x\|^2}$) and of the fuzzifier. The errors are scaled by their optimal value for each dimension.

We can see that as dimensions go higher, the optimal value for the fuzzifier decreases. We can fit here $m = 1 + \frac{1}{d}$ as a rule of thumb for the choice of the fuzzifier, which is similar to the one found in [23]. However, perhaps especially for these small dimensions, the optimal value of the fuzzifier is quite sensitive to the function to be approximated.

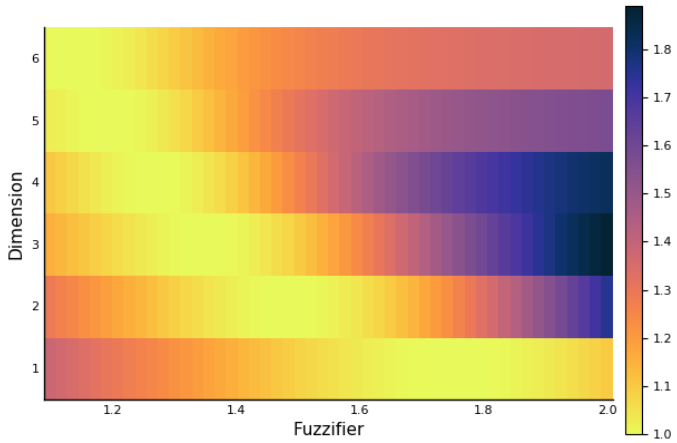


Figure 3.6: Approximation error as a function of dimension and of the fuzzifier

CHAPTER 4

The Fuzzy Approximation in a RL context

4.1 Distributional Reinforcement Learning

In Distributional Reinforcement Learning, distributions of the value functions are estimated, rather than their expected values. In the CDRL mixture update algorithm ([17], see Algorithm 1), the action-value function's distribution is estimated at iteration t by a categorical distribution

$$\eta_t^{(x,a)} = \sum_{i=1}^{N} p_{t,i}^{(x,a)} \delta_{c_i} \tag{4.1}$$

for fixed atoms $c_1, \ldots, c_N$. The next iteration is obtained using a stochastic distributional Bellman operator. The convergence of η_t, with respect to the Cramér distance, is proved in [17].

Algorithm 1 CDRL mixture update [17]

Require: $\eta_t^{(x,a)} = \sum_{i=1}^N p_{t,i}^{(x,a)} \delta_{c_i}$ for fixed atoms $c_1, \dots, c_N$

Sample transition (x_t, a_t, r_t, x_{t+1})
if Categorical policy evaluation **then**
    $a^* \sim \pi(\cdot|x_{t+1})$
else if Categorical Q-learning **then**
    $a^* \leftarrow \arg\max_a \mathbb{E}_{R \sim \eta_t^{(x_{t+1},a)}}[R]$
end if
$\hat{\eta}_*^{(x_t,a_t)} \leftarrow (f_{r_t,\gamma})_\# \eta_t^{(x_{t+1},a^*)}$ ▷ Distributional Bellman target
$\hat{\eta}_t^{(x_t,a_t)} \leftarrow \Pi_{\mathcal{C}} \hat{\eta}_*^{(x_t,a_t)}$ ▷ Projection onto the support
$\eta_{t+1}^{(x_t,a_t)} \leftarrow (1 - \alpha_t(x_t, a_t))\eta_t^{(x_t,a_t)} + \alpha_t(x_t, a_t)\hat{\eta}_t^{(x_t,a_t)}$ ▷ Mixture update
return η_{t+1}

4.2 From a Discretized State Space to a Fuzzy Approximation

As the CDRL algorithm from the previous section, RL algorithms often rely on the assumption that the state space of the considered problem is discrete. When it is not the case, one has to rely on function approximation. The piece-wise constant function approximation from subsection 3.1 allows a direct use of RL algorithms, since states are assimilated to finitely many clusters; neural networks, that can model more complicated functions, have also drawn a lot of interest in the recent years, with for example Deep Q-Learning [16]. Function approximation is also used when the state space in discrete but has too many states for tabular methods to be practical.

We propose to aim for the middle and, using a fuzzy approximation, provide

- a general approach to be adapt tabular methods with a (hopefully) better approximation than the piece-wise constant one;
- an example of this approach adapting the CDRL algorithm (Algorithm 1) in the following subsections.

The general approach is an intuitive one, mimicking the ideas behind tabular methods, in which values are stored independently for each state-actions pairs. It consists of two steps, which we shall respectively call *interpolation* and *projection*:

1. the access to the value of a given (continuous) state $x \in \mathcal{X}$ is through the fuzzy interpolation (3.4);
2. values computed at an observed state $x \in \mathcal{X}$, where $\mathcal{X}$ is continuous, are projected on the value of every representatives $z_j, j \in \{1, \dots, C\}$, proportionally to the

membership functions $\mu_j, j \in \{1, \dots, C\}$ (examples for such updates rules are given in the next subsection).

It is worth mentioning that this approach can be viewed as some generalization of tabular methods, in the sense that, as mentioned in section 2.2, when $m \longrightarrow 1$ in the expression of the membership functions (2.5), this degenerates to the piece-wise constant approximation described in section 3.1.

The rest of this section is dedicated to the application of this approach to the CDRL algorithm, featuring some modified algorithms in subsection 4.3 and their performance on a test case, described in subsection 4.4.

4.3 Fuzzy CDRL

The change in the CDRL is straightforward to obtain the Fuzzy CDRL (F-CDRL) algorithm, and can be found in details in the appendix (Section 8). We detail the two changes below, keeping the notations from Algorithm 1. The first change consist, as presented in the previous subsection, in the expression of the distributions for some state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$:

$$\eta_t^{(x,a)} = \sum_j \mu_j(x) \eta_t^{(z_j,a)}. \tag{4.2}$$

Note that there is no abuse of notation here, as we do have $\eta_t^{(x,a)} \in \mathcal{P}_C$ for all $(x, a) \in \mathcal{X} \times \mathcal{A}$. Indeed, using the expression of $\eta_t^{(z_j,a)}$ from (1.2), we get

$$\eta_t^{(x,a)} = \sum_i \underbrace{\left[\sum_j \mu_j(x) p_{t,i}^{(z_j,a)} \right]}_{=: p_{t,i}^{(x,a)}} \delta_{c_i}, \tag{4.3}$$

where $\sum_i p_{t,i}^{(x,a)} = 1$ follows from $\sum_i p_{t,i}^{(z_j,a)} = 1$ $(\forall j)$ and $\sum_j \mu_j(x) = 1$.

The second change consists of a projection step: values are stored at representatives, that is we update values for each representatives during the mixture update step, with a learning rate proportional to membership functions. This new update rule reads

$$\forall j : \quad \eta_{t+1}^{(z_j,a_t)} \leftarrow (1 - \alpha_t(z_j, a)\mu_j(x_t))\eta_t^{(z_j,a_t)} + \alpha_t(z_j, a)\mu_j(x_t)\hat{\eta}_t^{(x_t,a_t)}, \tag{4.4}$$

where $\alpha_t(z_j, a) = 0$ if $a \neq a_t$. This can again be seen as some generalization of the mixture update of Algorithm 1. However, if we look at the updated value at the sampled

state-action pair,

$$\eta_{t+1}^{(x_t,a_t)} \overset{(4.2)}{=} \sum_j \mu_j(x_t)\eta_{t+1}^{(z_j,a_t)} \overset{(4.4)}{=} \eta_t^{(x_t,a_t)} + \alpha_t \left[\underbrace{\left(\sum_j \mu_j(x_t)^2 \right)}_{\leq 1} \hat{\eta}_t^{(x_t,a_t)} - \sum_j \mu_j(x_t)^2 \eta_t^{(z_j,a_t)} \right], \tag{4.5}$$

we see that the successive projection and interpolation steps decrease the overall amount updated for the distribution at x_t when x_t is in between representatives (that is, when $\sum_j \mu_j(x_t)^2$ is low). Given this, we propose the following modified version of the update rule

$$\forall j: \quad \eta_{t+1}^{(z_j,a_t)} \leftarrow \left(1 - \alpha_t \frac{\mu_j(x_t)}{\sum_k \mu_k(x_t)^2}\right) \eta_t^{(z_j,a_t)} + \alpha_t \frac{\mu_j(x_t)}{\sum_k \mu_k(x_t)^2} \hat{\eta}_t^{(x_t,a_t)}. \tag{4.6}$$

Lastly, we propose one last idea that mixes (4.2) with (3.5). Using the writing from (4.3), this version of the algorithm uses the interpolation rule

$$p_{t,i}^{(x,a)} := \sum_j \mu_j(x) \left(p_{t,i}^{(z_j,a)} + \Delta p_{t,i}(z_j, x) \right) \tag{4.7}$$

Note that for this to be well-defined (that is, the resulting $\eta_t^{(x,a)}$ is a categorical distribution), we need that $\sum_i \sum_j \mu_j(x) \Delta p_{t,i}(z_j, x) = 0$. It so happens that the implementation of section 3 (see (3.9)) verifies this condition :

$$(\nabla p_{t,i})_j = \sum_{k \neq j} \nu_j(z_k) \frac{p_{t,i}^{(z_k,a)} - p_{t,i}^{(z_j,a)}}{\|z_k - z_j\|^2} (z_k - z_j) \tag{4.8}$$

(thus $\sum_i (\nabla p_{t,i})_j = 0$ is immediate from $\sum_i p_{t,i}^{(z_k,a)} = \sum_i p_{t,i}^{(z_j,a)} = 1$). This leaves us with 3 different algorithms: F-CDRL (Alg 2), Gradients FCDRL (GF-CDRL, Alg 3), and F-CDRL with the "fast update rule" (Fast F-CDRL, Alg 4). Performances of these algorithms are compared with each other and with the CDRL algorithm used with state discretization (i.e. piece-wise constant approximation) in the next subsection.

4.4 Results

To test the algorithm described in section 4.3, we build a test environment, in which the agent monitors an object moving on a (fixed) surface with obstacles, and must find a way from a starting position to a goal area. We adopt a simplified physical model where the object lies on the surface at all time and is subject to the weight force together with fluid friction. Fig 4.1 gives a representation of the chosen environment. Note that it is impossible to get out of the center hole, so the main task of the agent will be to avoid falling into it. A reward of $+100$ is gotten when reaching the goal area, while each timestep gives a reward of -1 (up to -200).

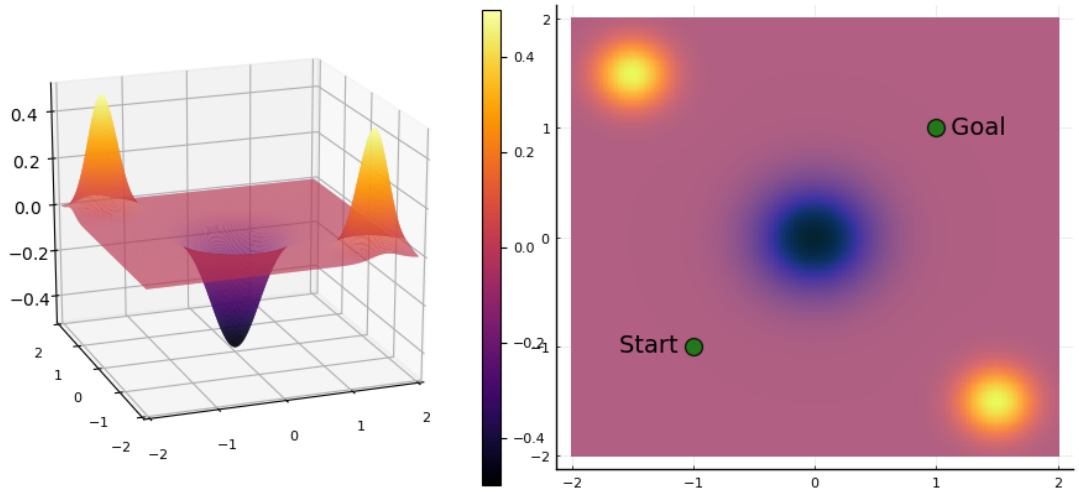


Figure 4.1: The test environment (left: the surface on which the object evolves; right: heatmap representation)

To test the differences between piece-wise constant approximation and the fuzzy approximation, position and velocity of the object are described by continuous features. The agent acts on the environment through 9 possible actions (including a "no-action" one), by which it adds a force to the physical ones, at a constant intensity and in 8 possible directions (at angles $\theta \in \{k\frac{\pi}{4}, k \in \{0, \dots, 7\}\}$). Since the object lies on a surface (by hypothesis), we only need to considered 2D-projected position and velocity for the problem to be a MDP.

Note that this environment is purely deterministic; however, we include two sources of uncertainty in our modeling:

- we choose to make only the position available to the agent, for both reduced run-time and improved readability of the results;
- the chosen models can only represent a restricted class of functions that may not render the exact situation of the environment (consider for instance the piece-wise constant approximation, where state-aliasing can happen in the neighborhood of obstacles).

Agents are trained according to 5 different processes : the CDRL algorithm with discretized states, with a piece-wise constant approximation through binning (Bins PC) or K-Means (K-Means PC), F-CDRL, GF-CDRL, Fast F-CDRL. Data from the training of the "Bins PC" agent is collected to compute representatives for the 4 other processes using the K-Means algorithm (following the remarks of section 2.3, we do not use FCM for the fuzzy representatives; also this provides a fair comparison between the different processes). The training is done over 15 000 trajectories, each computed according to

an ϵ-greedy policy ($\epsilon = 0.05$) based on the latest update of the agent. We compare results obtained using two different learning rates ($\alpha = 0.1$ and $\alpha = 0.01$). For all fuzzy algorithms, we use a fuzzifier $m = 1.5$ based on Fig 3.6, since the dimension observed by the agent is 2 (Note that it is perhaps sub-optimal). We collect rewards from the last 10 000 trajectories and an overview of results is shown in Tables 4.1 and 4.2. Fuzzy versions of CDRL achieve the best scores, whereas the K-Means discretized tabular CDRL generates steadier policies (the lower standard deviation is here correlated with fewer "lost games", i.e. the agent falls in the central hole fewer times).

Table 4.1: Performance comparison of the different algorithms ($\alpha = 0.1$).

	Bins PC	**K-Means PC**	**F-CDRL**	**GF-CDRL**	**Fast F-CDRL**
Mean reward	47.73	52.21	**64.30**	60.83	57.27
Standard deviation	50.26	**26.04**	39.95	44.19	46.37
Max reward	78	74	83	**84**	**84**
Ratio of lost games	3.3%	**0.6%**	2.1%	2.6%	2.8%

Table 4.2: Performance comparison of the different algorithms ($\alpha = 0.01$).

	Bins PC	**K-Means PC**	**F-CDRL**	**GF-CDRL**	**Fast F-CDRL**
Mean reward	55.98	61.83	**67.60**	66.09	62.43
Standard deviation	24.12	**18.95**	19.10	20.47	50.57
Max reward	73	75	**84**	**84**	83
Ratio of lost games	4.9‰	**2.2‰**	2.3‰	4.2‰	3.33%

To complete this section, we propose a finer analysis based on the training with $\alpha = 0.01$, that provided the best results. We note that Fast F-CDRL provided poorer results than F-CDRL, and will not be part of the latter figures; the fast update is perhaps unsuitable for the considered discretization and would probably provide better result with a finer grid of representatives. Rewards (averaged over 300 trajectories) are shown in Fig 4.2. Note that the amount of runs needed to find a wining policy is (at least partially) chance-based, since we only implement the ϵ-greedy scheme to encourage exploration.

Fig 4.3 shows, for the 4 considered algorithms, the maximal expected values of the learned action-value distributions across the state space, that is, the expected value for the greedy action taken at each position of the state space. We can note that tabular algorithms explored only the bottom part of the state space while fuzzy algorithms explored the top part of it, but it is irrelevant and relies only on chance and on the first winning trajectories found during the early steps of learning. The main and awaited difference between tabular and fuzzy algorithm relies on that fact that the latter are of continuous nature, and thus represent the present continuous problem in a neater way. This is reflected by the better performance of the fuzzy algorithms at this task.

We finally take a look at action-value distributions at the initial state for each algorithm, which are represented in Fig 4.4. As expected from Fig 4.3, the action with highest

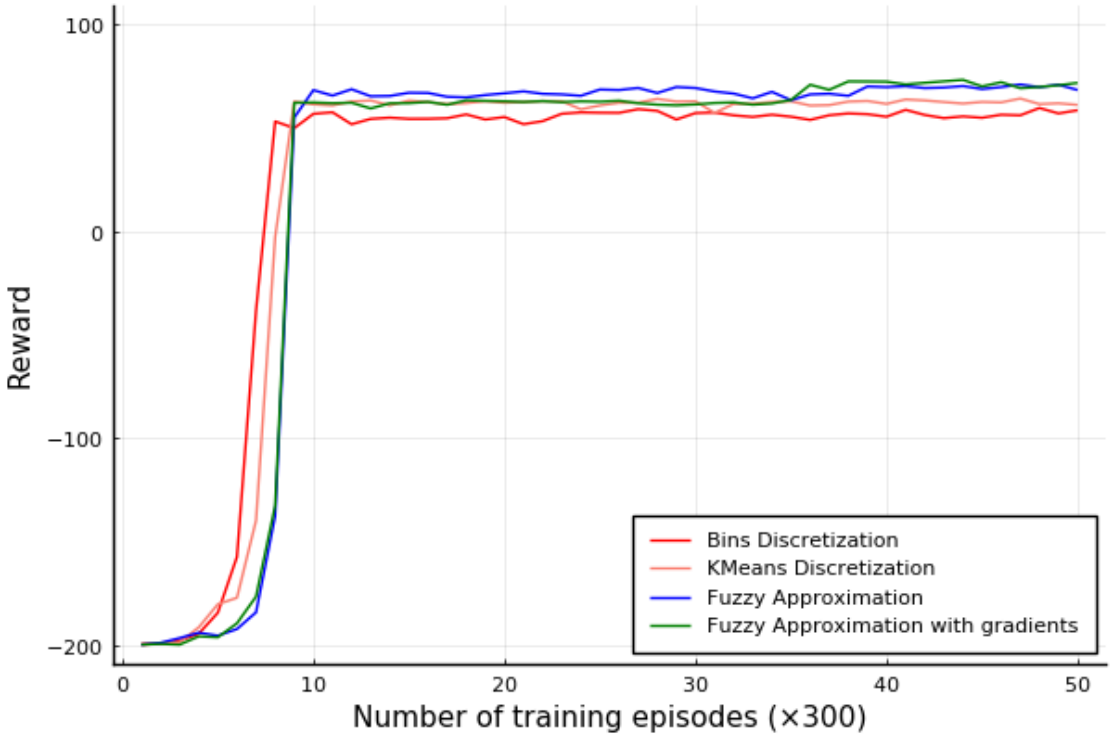
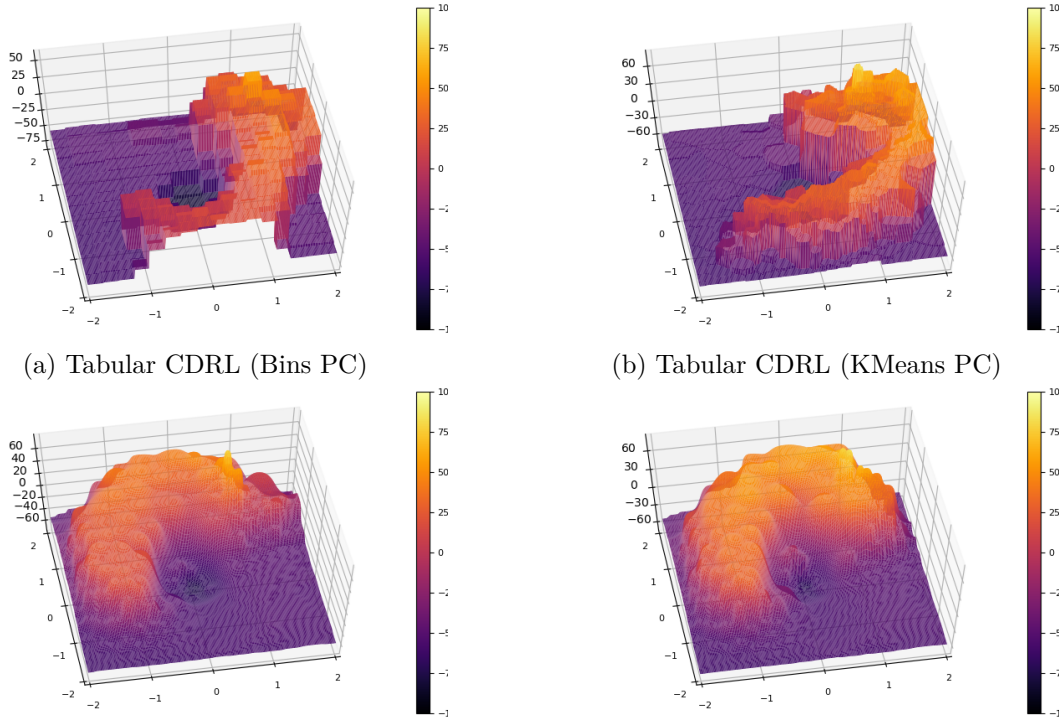


Figure 4.2: Evolution of rewards obtained through training for 4 algorithms (results are averaged over 300 runs)

expected reward for tabular algorithms (resp. fuzzy algorithms) is “down-right” (resp. “up”). We do not expect to see a huge difference between actions for the initial state, given the significant fluid friction set; we hypothesize that the difference here is mainly due to some over-fitting to the learned policy, whereas other actions were not much undertaken and thus had their values less updated.



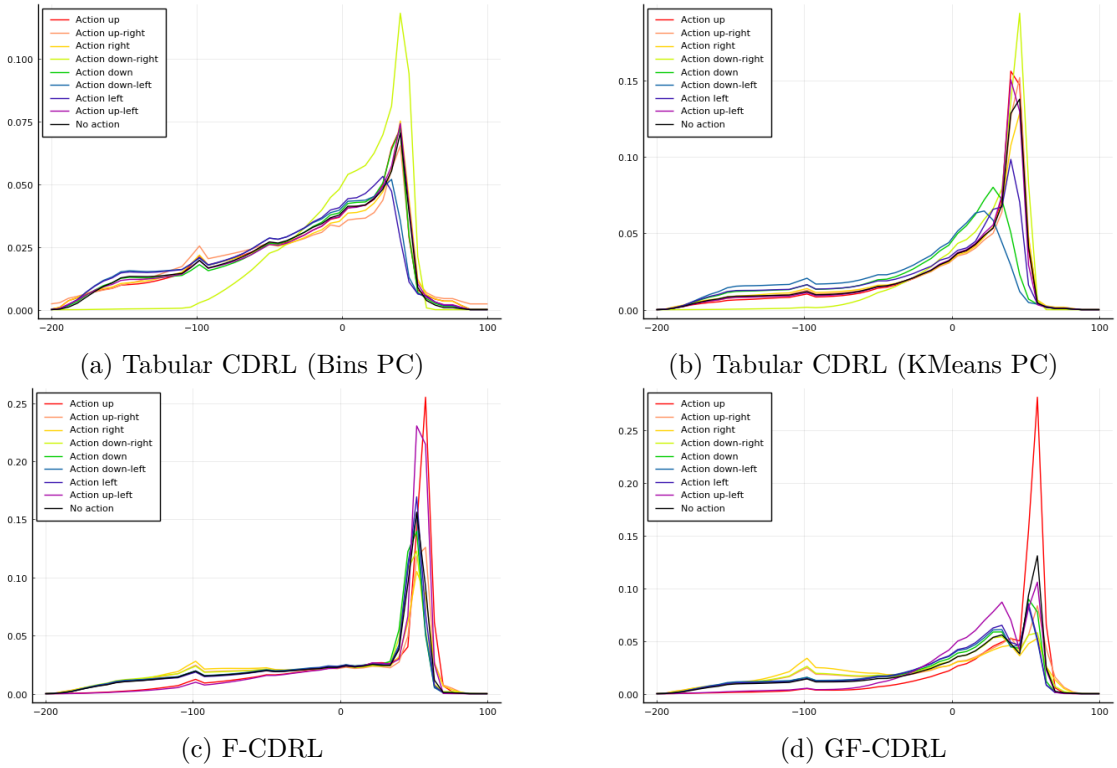
(a) Tabular CDRL (Bins PC)

(b) Tabular CDRL (KMeans PC)

(c) F-CDRL

(d) GF-CDRL

Figure 4.3: Maximal expected value of action-value distribution for the different algorithms.



(a) Tabular CDRL (Bins PC)

(b) Tabular CDRL (KMeans PC)

(c) F-CDRL

(d) GF-CDRL

Figure 4.4: Learned action-value distribution at the initial state for the different algorithms.

CHAPTER 5

Fuzzy Approximation in a Classification Context

In this section, we build upon the infamous MNIST handwritten digits recognition problem [14] to investigate how representatives can be searched automatically. Note that the goal is to better understand how representatives are modified when subject to some optimization, and not to find any state-of-the-art model. In this setting, there was no optimization of the hyperparameters of the models. Note finally that we stick to the euclidean distance to compare objects; although there are distances that were developed to specifically compare images (see e.g. [4, 20]), their complexity makes it impractical to have membership functions based on them.

5.1 The Search for Adequate Representatives

We use a simple model consisting of what we shall call a *memberships layer*, that computes the membership functions (as defined in subsection 2.2, Eq (2.5)) of flattened images with respect to a set of representatives that are parameters of the model (and thus subject to optimization). We further process the obtained vector through a basic neural network. The last layer of the network is then composed with a softmax operation, and trained against the cross-entropy loss. This is represented in Fig 5.1. The models trained feature 25 representatives followed by 3 layers of 50 neurons each with Rectified Linear Units as activation functions, and an output layer of 10 neurons with a softmax operation; we use ADAM as optimizer.

We test this type of network in two setups: in the first, we initialize representatives of the memberships layer to all-zeros to inspect the results of a blind search for representatives; in the second, we initialize the representatives using the K-Means++ initialization algorithm [1], that is with images from the dataset that are roughly spread across the (flattened) data.

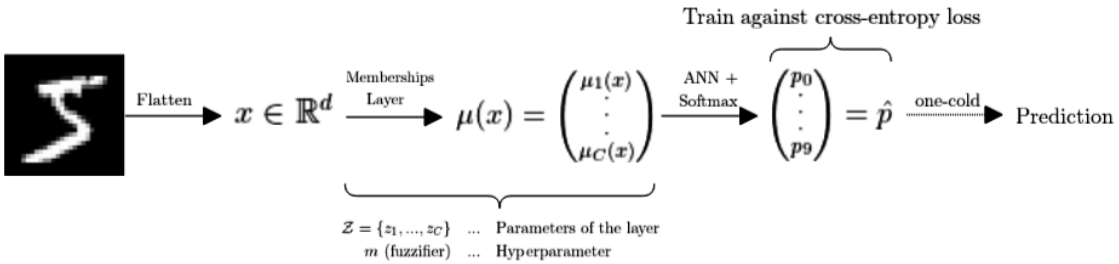
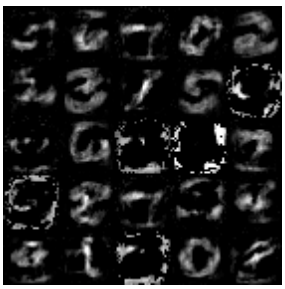


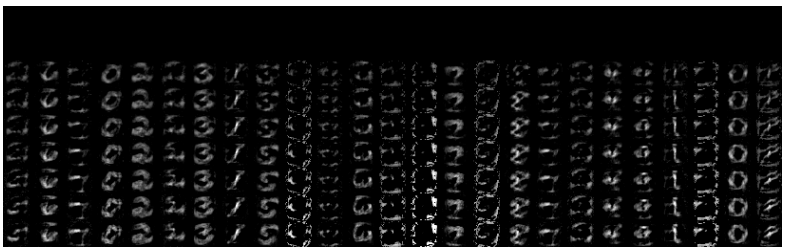
Figure 5.1: The considered network architecture

Generating representatives

The perhaps most striking result is that, when initializing the representatives uniformly at 0, the network is able to produce digit-like representatives. Representatives produced after 1 epoch of training are represented in Fig 5.2a, while their evolution throughout training is represented in Fig 5.2b.



(a) Representatives produced



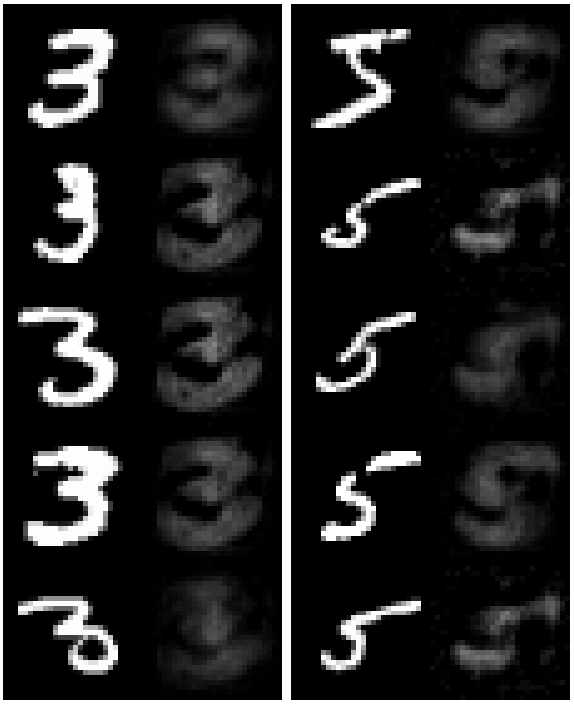
(b) Evolution of the representatives while training

Figure 5.2: Representatives produced after 1 epoch of training, when initializing uniformly at 0.

Some representatives appear to be noisy, but it appears not to be too detrimental for the network; to better understand what is going on, we perform a fuzzy interpolation based on the learned representatives $\mathcal{Z} = \{z_j, j = 1, \ldots, 25\}$. For some data sample x, we compute the interpolated image $\hat{x}$ by

$$\hat{x} := \sum_j \mu_j(x) z_j. \tag{5.1}$$

In Fig 5.3, we show the results for a few images. We can see that despite the noisiness of the representatives, the interpolated images are consistent with the original number; they are even remarkably close to each other (see Fig 5.3a) even if original digit have dissimilarities.



(a) Interpolated 3's (b) Interpolated 5's

Figure 5.3: Results of the fuzzy interpolation of some images by the learned memberships layer. For each pair of digit, the left one is the actual data sample and the right one is the interpolated one.

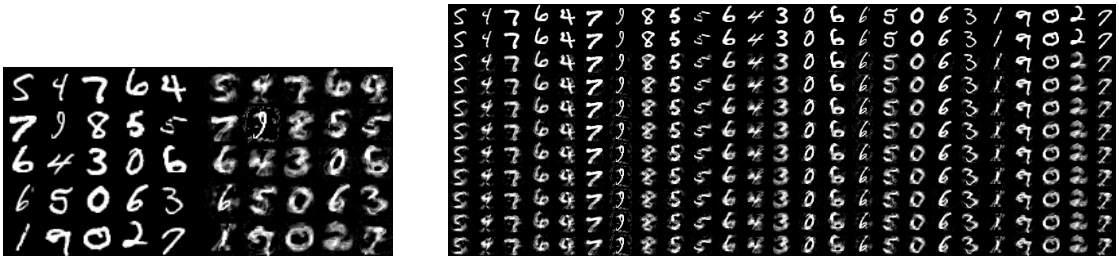
Updating Representatives

We now investigate what happens when representatives are initialized by real images, using the K-Means++ initialization scheme. The results from 1 epoch of training are represented in Fig 5.4. We observe that some digits are noticeably changed during training: see the fifth column in Fig 5.4b, where a 4 seem to evolve towards a 9.

In Fig 5.5, we represent the same interpolation process as (5.1). Fig 5.5b uses the initial representatives while Fig 5.5c uses the learned representatives; we use heatmaps instead of gray images for readability, since the scale of interpolated images is different. We can see in Fig 5.5c that the original digit is turned into a very prototypical 5.

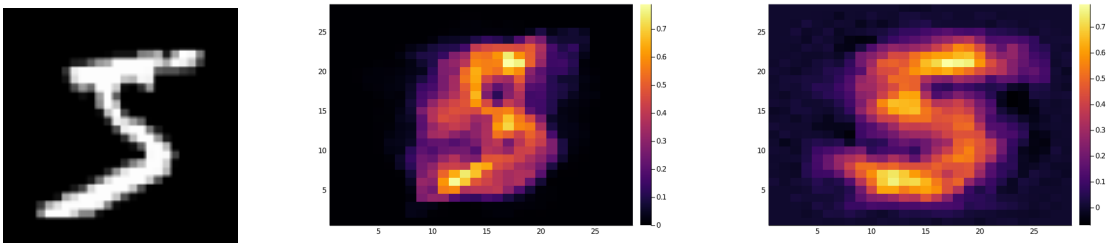
5.2 Details on the Backpropagation Algorithm

Finally, in order to provide an explanation onto why digit-like representatives can be found by the model, we investigate further the details of the backpropagation (see e.g. [18] for explanation of the BP Algorithm) in our network, that is we seek a simplified expression of the gradient of the loss with respect to representatives. We use the following



(a) Representatives produced (b) Evolution of the representatives while training

Figure 5.4: Representatives produced after 1 epoch of training, when initializing with k-means++. (a) Initial representatives (left) compared with learned representatives (right); (b) some representatives change shapes while training.



(a) The interpolated image (b) Interpolation using the initial representatives (c) Interpolation using the learned representatives

Figure 5.5: Fuzzy interpolation of a digit. Interpolated images are representated through heatmaps for scale and readability.

notation:

- each neuron layer consists in an aggregation function h composed with an activation function g. In our case, the aggregation function is a scalar product between the weights of the layer and the values at the entry layer:
- the forward pass from layer $l-1$ to layer l is thus defined by

$$x_i^{(l)} = g^{(l)}(h_i^{(l)}) = g^{(l)}\left(\sum_k w_{ik}^{(l)} x_k^{(l-1)}\right), \tag{5.2}$$

  where $x_i^{(l)}$ is the i-th neuron of the l-th layer and $w_{ik}^{(l)}$ is the weight from $x_k^{(l-1)}$ to $x_i^{(l)}$;
- errors are backpropagated through dense layers using

$$e_k^{(l-1)} = g'^{(l-1)}(h_k^{(l-1)}) \sum_i w_{ik}^{(l)} e_i^{(l)}, \tag{5.3}$$

where $e_i^{(l)}$ is the error backpropagated to $x_i^{(l)}$, initialized as the cross-entropy loss between the output layer and the observed value, $e_i^{(\text{out})} = g'^{(\text{out})}(h_i^{(\text{out})}) \cdot \frac{\partial E_i}{\partial p_i}$.

Note that (5.3) is obtained by using successive chain rules; taking the gradient of the error E with respect to some representative z_j of the membership layer, we have for the last neuron layer (numbered n)

$$\begin{aligned} \frac{\partial E}{\partial z_j} &= \sum_i \frac{\partial E_i}{\partial p_i} \overbrace{g'^{(n)}(h_i^{(n)})}^{\frac{\partial p_i}{\partial h_i}} \overbrace{\sum_k w_{ik}^{(n)} \frac{\partial x_k^{(n-1)}}{\partial z_j}}^{\frac{\partial h_i}{\partial z_j}} \\ &= \sum_k \frac{\partial x_k^{(n-1)}}{\partial z_j} \sum_i w_{ik}^{(n)} e_i^{(n)}. \end{aligned} \tag{5.4}$$

Eq (5.3) is then obtained (inductively) by applying the chain rule to $\frac{\partial x_k^{(n-1)}}{\partial z_j}$. We are then interested in the error backpropagated to the memberships layer:

$$\begin{aligned} \frac{\partial E}{\partial z_j} &= \sum_k \frac{\partial \mu_k(x)}{\partial z_j} \sum_i w_{ik}^{(1)} e_i^{(1)} \\ &= \frac{2}{m-1} \mu_j(x) \left[\sum_{k \neq j} \mu_k(x) \sum_i (w_{ik}^{(1)} - w_{ij}^{(1)}) e_i^{(1)} \right] \frac{z_j - x}{\|z_j - x\|^2}, \end{aligned} \tag{5.5}$$

where we use the following expression for $\frac{\partial \mu_k(x)}{\partial z_j}$

$$\frac{\partial \mu_k(x)}{\partial z_j} = \begin{cases} \dfrac{2}{m-1} \cdot \mu_j(x) \mu_k(x) \cdot \dfrac{z_j - x}{\|z_j - x\|^2} & \text{if } j \neq k, \\ -\dfrac{2}{m-1} \cdot \mu_j(x)(1 - \mu_j(x)) \cdot \dfrac{z_j - x}{\|z_j - x\|^2} & \text{if } j = k. \end{cases} \tag{5.6}$$

We now wish to compare this with the principle of the FCM algorithm. For this we introduce a modified version of the objective function defined in Eq (2.3), focusing on the influence of some data point $x \in \mathcal{X}$:

$$\Psi(x) := \sum_k \mu_k(x)^m \|z_k - x\|^2. \tag{5.7}$$

Taking the gradient of (5.7) with respect to some representative z_j, we ultimately find

$$\frac{\partial \Psi(x)}{\partial z_j} = 2\mu_j(x)^m (z_j - x). \tag{5.8}$$

Note that (5.5) and (5.8) are both in the same direction $(z_j - x)$, which is of most importance in a gradient descent scheme (where scaling is impacted by the learning rate anyway). We can also rewrite (5.5) to make (5.8) and additional scaling factors appear:

$$\frac{\partial E}{\partial z_j} = \underbrace{\left(\frac{1}{m-1} S_m(x)^{m-1}\right)}_{\text{position term}} \cdot \underbrace{\left(\sum_{k \neq j} \mu_k(x) \sum_i (w_{ik}^{(1)} - w_{ij}^{(1)}) e_i^{(1)}\right)}_{\text{backpropagation term}} \cdot \underbrace{\frac{\partial \Psi(x)}{\partial z_j}}_{\text{FCM}}, \tag{5.9}$$

where we write $S_m(x) = \left(\sum_k \left(\frac{1}{\|z_k - x\|}\right)^{\frac{2}{m-1}}\right)^{-1}$ for the scaling term of membership functions. We put together the representative search and this factoring in the following way:

- Representatives are "searched" by the algorithm in a way similar to the one of FCM: gradient updates have the same direction for the backpropagation through the network and for the FCM (partial) objective function;
- the "backpropagation term" links the update of the representatives to the rest of the network, and thus provides some kind of "objective based fuzzy clustering";
- the "position term" is a scaling term depending on the sample's position with respect to all representatives, and is high when the sample is far away (in term of the euclidean distance) from all representatives. The gradient is scaled up, and thus the representatives moved towards the sample, when the clustering cannot represent the sample accurately.

CHAPTER 6

Operator View on the Interpolation Function Approximation

For a policy π, the *distributional Bellman operator* $\mathcal{T}^\pi \colon \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}} \longrightarrow \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ is defined by

$$(\mathcal{T}^\pi \eta)^{(x,a)} := \int_{\mathbb{R}} \int_{\mathcal{X}} \sum_{a' \in \mathcal{A}} (f_{r,\gamma})_{\#} \, \eta^{(x',a')} \pi(a'|x') p(dr, dx'|x, a). \tag{6.1}$$

Note that here, contrary to [17], we do not consider $\mathcal{X}$ to be finite. However, this does not change contraction properties for $\mathcal{T}^\pi$ with respect to the Wasserstein metric (see [3]) and for $\Pi_{\mathcal{C}}\mathcal{T}^\pi$ with respect to the Cramér metric (see [17]). Note that since $\mathcal{X}$ is not finite, $\mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ associated with a supremum metric (namely the supremum p-Wasserstein metric $\overline{d}_p$ [3] or the supremum Cramér metric $\bar{l}_2$ [17]) need not be complete and we must restrict ourselves explicitly to the study of *bounded* value distribution functions. In this setting, the Banach fixed point theorem gives the existence of a unique value distribution function $\eta_\pi \in \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ satisfying $\eta_\pi = \mathcal{T}^\pi \eta_\pi$.

We also define operators $\mathcal{I} \colon \mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}} \longrightarrow \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ and $\Pi^{\mathcal{Z}} \colon \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}} \longrightarrow \mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}}$, for the interpolation and projection steps respectively. For example,

$$(\mathcal{I}\eta)^{(x,a)} = \sum_j \mathbf{1}_{\mathcal{X}_j}(x) \eta^{(z_j,a)}, \tag{6.2}$$

$$\left(\Pi^{\mathcal{Z}}\eta\right)^{(z_j,a)} = \frac{1}{|\mathcal{X}_j|} \int_{\mathcal{X}_j} \eta^{(x,a)} dx \tag{6.3}$$

correspond the hard-clustering case described in Section 3.1, where $\mathcal{X} = \bigcup_j \mathcal{X}_j$ and $\mathcal{Z} = \{z_1, \dots, z_C\}$ with z_j associated with $\mathcal{X}_j$ for all j.

In particular, these operator are non-expansions with respect to the supremum Cramér metric, on $\mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ for $\mathcal{I}$ and on $\mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}}$ for $\Pi^{\mathcal{Z}}$.

Theorem 1. *Let two interpolation and projection operators $\mathcal{I}\colon \mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}} \to \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ and $\Pi^{\mathcal{Z}}\colon \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}} \to \mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}}$ be non-expansions with respect to the supremum-Cramér metric, that commute with the categorical projection $\Pi_{\mathcal{C}}$. Then the operator $\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\mathcal{I}$ is a $\sqrt{\gamma}$-contraction in the supremum-Cramér metric $\bar{l}_2$. Furthermore, there exists a unique fix-point $\eta_* \in \mathcal{P}(\mathbb{R})^{\mathcal{Z}\times\mathcal{A}}$ of $\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\mathcal{I}$ found as the $\bar{l}_2$ limit of $\left\{\left(\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\mathcal{I}\right)^m \eta_0\right\}_{m\in\mathbb{N}}$ for any initial distribution η_0. We obtain the following inequality between η_* and η_π:*

$$(1-\sqrt{\gamma})\bar{l}_2(\mathcal{I}\eta_*, \eta_\pi) \leq \bar{l}_2(\Pi_{\mathcal{C}}\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) + \bar{l}_2(\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \eta_\pi). \tag{6.4}$$

Proof. The contraction property for $\mathcal{T}^{\pi}$ does not rely in any way on $\mathcal{X}$ being discrete (see [17]). Since all other operators involved are non-expansions with respect to $\bar{l}_2$, $\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\mathcal{I}$ is indeed a $\sqrt{\gamma}$-contraction. We can then use Banach's fixed-point theorem to find the desired fix-point η_*. To prove the inequality (6.4), we first use the triangle inequality for $\bar{l}_2$ to find

$$\bar{l}_2(\mathcal{I}\eta_*, \eta_\pi) \leq \bar{l}_2(\mathcal{I}\eta_*, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) + \bar{l}_2(\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \eta_\pi). \tag{6.5}$$

Furthermore, we use the pythagorean like theorem proved in [17] and proceed using the fixed-point properties for η_π, η_* to obtain

$$\begin{aligned}
\bar{l}_2^2(\mathcal{I}\eta_*, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) &\leq \bar{l}_2^2(\mathcal{I}\eta_*, \Pi_{\mathcal{C}}\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) && +\bar{l}_2^2(\Pi_{\mathcal{C}}\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) \\
&= \bar{l}_2^2(\mathcal{I}\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\mathcal{I}\eta_*, \mathcal{I}\Pi^{\mathcal{Z}}\Pi_{\mathcal{C}}\mathcal{T}^{\pi}\eta_\pi) && +\bar{l}_2^2(\Pi_{\mathcal{C}}\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi) \\
&\leq \gamma\bar{l}_2^2(\mathcal{I}\eta_*, \eta_\pi) && +\bar{l}_2^2(\Pi_{\mathcal{C}}\mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi, \mathcal{I}\Pi^{\mathcal{Z}}\eta_\pi).
\end{aligned} \tag{6.6}$$

We get the desired inequality by combining (6.5) and (6.6) and using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. □

Note that we can interpret the two terms of the right-hand side of inequality (6.4) as the modelling power of categorical distributions, $\bar{l}_2(\Pi_{\mathcal{C}}\eta, \eta)$, and the modelling power of piece-wise constant distribution functions, $\bar{l}_2(\mathcal{I}\Pi^{\mathcal{Z}}\eta, \eta)$. It is proved in [17] that there exists an upper-bound independent of η for $\bar{l}_2(\Pi_{\mathcal{C}}\eta, \eta)$:

$$\bar{l}_2^2(\Pi_{\mathcal{C}}\eta, \eta) \leq \max_{1\leq i\leq K-1}(c_{i+1} - c_i). \tag{6.7}$$

We now restrict ourselves to the hard-clustering case. In particular for the operators defined in (6.2) and (6.3), we have the following result:

Theorem 2. *We consider the metric spaces $(\mathcal{X}, d_2)$, where d_2 is the euclidean metric, and $(\mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}, \bar{l}_2)$, where $\bar{l}_2$ is the supremum-Cramér metric. Let $\eta \in \mathcal{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$.*

- *if $x \mapsto \eta^{(x,a)}$ is uniformly continuous for all $a \in \mathcal{A}$, then we can choose $\mathcal{Z}$ in order for $\bar{l}_2(\mathcal{I}\Pi^{\mathcal{Z}}\eta, \eta)$ to be arbitrarily small;*

- *if we (additionally) have that, for all $a \in \mathcal{A}$, there exists some constant $M(a) \in \mathbb{R}_+$, such that*

$$\bar{l}_2(\eta^{(x,a)}, \eta^{(y,a)}) \leq M(a) d_2(x, y) \quad \forall x, y \in \mathcal{X}, \tag{6.8}$$

then we have the explicit upperbound

$$\bar{l}_2(\mathcal{I}\Pi^{\mathcal{Z}}\eta, \eta) \leq M \cdot \delta, \tag{6.9}$$

where $M := \sup_{a \in \mathcal{A}} M(a)$ and $\delta := \sup_j \{d_2(x, y) \colon x, y \in \mathcal{X}_j\}$. In particular δ can be arbitrarily small, depending on the representatives $\mathcal{Z}$.

Finally, we can also follow the very same steps as in [17] to prove the convergence of the policy evaluation algorithm that uses the stochastic operator

$$\left(\hat{\mathcal{P}}\eta\right)^{(z_j,a)} = \begin{cases} (f_{r,\gamma})_{\#} \left(\mathcal{I}\eta\right)^{(x',a')} & \text{if } x \in \mathcal{X}_j, \\ \eta^{(z_j,a)} & \text{if } x \notin \mathcal{X}_j, \end{cases} \tag{6.10}$$

where $\hat{\mathcal{P}}$ is the stochastic counterpart of the operator $\mathcal{P} = \Pi^{\mathcal{Z}}\mathcal{T}^{\pi}\mathcal{I}$, together with the mixture update

$$\begin{aligned} \eta_{t+1}^{(z_j,a)} &= (1 - \alpha_t(z_j, a))\eta_t^{(z_j,a)} + \alpha_t(z_j, a) \left(\Pi_{\mathcal{C}}\hat{\mathcal{P}}\eta_t\right)^{(z_j,a)} \\ &= (1 - \alpha_t(z_j, a))\eta_t^{(z_j,a)} + \alpha_t(z_j, a)\Pi_{\mathcal{C}}(f_{r_t,\gamma})_{\#} \left(\mathcal{I}\eta\right)^{(x_{t+1},a*)}, \end{aligned} \tag{6.11}$$

where the second equality holds since $x_t \notin \mathcal{X}_| \implies \alpha_t(z_j, a) = 0$. We further detail only the main difference with [17]. A major point of the proof relies in the decomposition

$$\begin{aligned} \eta_{t+1}^{(z_j,a)} = \eta_t^{(z_j,a)} &+ \alpha_t(z_j, a) \left[(\Pi_{\mathcal{C}}\mathcal{P}\eta_t)^{(z_j,a)} - \eta_t^{(z_j,a)}\right] \\ &+ \alpha_t(z_j, a) \left[\Pi_{\mathcal{C}}(f_{r_t,\gamma})_{\#} \left(\mathcal{I}\eta_t\right)^{(x',a')} - (\Pi_{\mathcal{C}}\mathcal{P}\eta_t)^{(z_j,a)}\right], \end{aligned} \tag{6.12}$$

where the last term is a random signed measure. As in [17] and [22] we want that

$$\mathbb{E}\left[\Pi_{\mathcal{C}}(f_{r_t,\gamma})_{\#} \left(\mathcal{I}\eta_t\right)^{(x',a')} - (\Pi_{\mathcal{C}}\mathcal{P}\eta_t)^{(z_j,a)}\right]((-\infty, y]) = 0 \quad \forall y \in \mathbb{R}. \tag{6.13}$$

The only notable change is that, since $\mathcal{X}$ is no longer finite but partitioned into finitely many $\mathcal{X}_j, 1 \leq j \leq C$, we have to take a distribution for x' into account. The given distribution in (6.3) matches that of a uniform distribution for states x' in $\mathcal{X}_j$. This is however not a requirement on the model, for we can use any distribution for $x' \in \mathcal{X}_j$, given that the true operators $\Pi^{\mathcal{Z}}, \mathcal{T}^{\pi}$ are never actually computed and that this distribution is only implicit in the stochastic operator $\hat{\mathcal{P}}$.

Finally, we further restrict the policy π to only depend on the clusters corresponding to states, i.e.

$$\forall a \in \mathcal{A} : \forall x \in \mathcal{X}_j : \quad \pi(a|x) = \pi(a|z_j). \tag{6.14}$$

In this setting, we can rewrite the operator $\mathcal{P}$ as a Bellman operator in a discrete setting as

$$\left(\Pi^{\mathcal{Z}} \mathcal{T}^{\pi} \mathcal{I} \eta\right)^{(z_j,a)} = \int_{\mathbb{R}} \sum_k \sum_{a' \in \mathcal{A}} (f_{r,\gamma})_{\#} \eta^{(z_k,a')} \pi(a'|z_k) \tilde{p}(dr, z_k|z_j, a), \tag{6.15}$$

where $\tilde{p}(dr, z_k|z_j, a) := \frac{1}{|\mathcal{X}_j|} \int_{x \in \mathcal{X}_j} \int_{x' \in \mathcal{X}_k} p(dr, dx'|x, a) dx$. In this setting, all properties of the discrete setting are hence inherited.

CHAPTER 7

Conclusion

In this work, a clear connection was drawn from (fuzzy) clustering to function approximation and RL using the fuzzy interpolation. Empirical results on a test problem were found to be encouraging. It was also shown how the fuzzy interpolation could be used to perform an optimized search for state representatives. An interesting direction for further work would be to bridge these two approaches to allow for an online searched for representatives in the reinforcement learning process. Additionally, the empirical approach can be improved in multiple ways, for example by implementing multiple forms for the membership functions, for example following the approach in [13]. In [6], although they follow a different point of view on fuzzy sets and do not consider *distributional* RL, a fuzzy interpolation is also applied to the action space.

From the theoretical point of view, we showed that the concept of interpolation and projection operators is well-suited to the study of this type of function approximation for RL. In the hard-clustering case, we can follow the same steps as in [17] to prove the results established for the discrete setting.

Bibliography

[1] David Arthur and Sergei Vassilvitskii. "K-means++: the advantages of careful seeding". In: *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. 2007.

[2] M. G. Bellemare et al. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. ISSN: 1076-9757. DOI: 10.1613/jair.3912. URL: http://dx.doi.org/10.1613/jair.3912.

[3] Marc G. Bellemare, Will Dabney, and Rémi Munos. *A Distributional Perspective on Reinforcement Learning*. 2017. arXiv: 1707.06887 [cs.LG].

[4] S. Belongie, J. Malik, and J. Puzicha. "Shape matching and object recognition using shape contexts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.4 (2002), pp. 509–522.

[5] James C. Bezdek. "Pattern Recognition with Fuzzy Objective Function Algorithms". In: USA: Kluwer Academic Publishers, 1981. Chap. 3 (S11). ISBN: 0306406713.

[6] Andrea Bonarini et al. "Reinforcement distribution in fuzzy Q-learning". In: *Fuzzy Sets and Systems* 160.10 (2009). Special Issue: Fuzzy Sets in Interdisciplinary Perception and Intelligence, pp. 1420–1443. ISSN: 0165-0114. DOI: https://doi.org/10.1016/j.fss.2008.11.026. URL: http://www.sciencedirect.com/science/article/pii/S0165011408005319.

[7] Madson Dias. *fuzzy-c-means: Python module implementing the Fuzzy C-means clustering algorithm*. Python module v0.0.6. 2019. URL: https://pypi.org/project/fuzzy-c-means/.

[8] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of Cybernetics* 3.3 (1973), pp. 32–57. DOI: 10.1080/01969727308546046. eprint: https://doi.org/10.1080/01969727308546046. URL: https://doi.org/10.1080/01969727308546046.

[9] J. C. Dunn. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". In: *Journal of Cybernetics* 3.3 (1973), pp. 32–57. DOI: 10.1080/01969727308546046. eprint: https://doi.org/10.1080/01969727308546046. URL: https://doi.org/10.1080/01969727308546046.

[10] Philip Hall. "The Distribution of Means for Samples of Size N Drawn from a Population in which the Variate Takes Values between 0 and 1, all such Values Being Equally Probable". In: *Biometrika* 19.3-4 (Dec. 1927), pp. 240–244. ISSN: 0006-3444. DOI: 10.1093/biomet/19.3-4.240. eprint: https://academic.oup.com/biomet/article-pdf/19/3-4/240/742325/19-3-4-240.pdf. URL: https://doi.org/10.1093/biomet/19.3-4.240.

[11] J. O. Irwin. "On the Frequency Distribution of the Means of Samples from a Population Having any Law of Frequence with Finite Moments, with Special Reference to Peason's Type II". In: *Biometrika* 19.3-4 (Dec. 1927), pp. 225–239. ISSN: 0006-3444. DOI: 10.1093/biomet/19.3-4.225. eprint: https://academic.oup.com/biomet/article-pdf/19/3-4/225/742314/19-3-4-225.pdf. URL: https://doi.org/10.1093/biomet/19.3-4.225.

[12] JuliaStats. *Clustering.jl: Methods for data clustering and evaluation of clustering quality*. Julia package v0.14.1. 2020. URL: https://github.com/JuliaStats/Clustering.jl.

[13] Frank Klawonn and Frank Höppner. "What Is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier". In: Aug. 2003, pp. 254–264. DOI: 10.1007/978-3-540-45231-7_24.

[14] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST Database of handwritten digits*. http://yann.lecun.com/exdb/mnist/. Oct. 2020.

[15] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

[16] Volodymyr Mnih, Koray Kavukcuoglu, and et al. Silver David. "Human-level control through deep reinforcement learning". In: *Nature* 518 (Feb. 2015), pp. 529–33. DOI: 10.1038/nature14236.

[17] Mark Rowland et al. *An Analysis of Categorical Distributional Reinforcement Learning*. 2018. arXiv: 1802.08163 [stat.ML].

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: MIT Press, 1986. Chap. 8.

[19] Donald Shepard. "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data". In: *Proceedings of the 1968 23rd ACM National Conference*. ACM '68. New York, NY, USA: Association for Computing Machinery, 1968, pp. 517–524. ISBN: 9781450374866. DOI: 10.1145/800186.810616. URL: https://doi.org/10.1145/800186.810616.

[20] P. Y. Simard et al. "Transformation Invariance in Pattern Recognition – Tangent Distance and Tangent Propagation". In: *International Journal of Imaging Systems and Technology* 11.3 (2001).

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[22] John N. Tsitsiklis. "Asynchronous Stochastic Approximation and Q-Learning". In: *Mach. Learn.* 16.3 (Sept. 1994), pp. 185–202. ISSN: 0885-6125. DOI: 10.1023/A:1022689125041. URL: https://doi.org/10.1023/A:1022689125041.

[23] Roland Winkler, Frank Klawonn, and Rudolf Kruse. "Problems of Fuzzy c-Means Clustering and Similar Algorithms with High Dimensional Data Sets". In: July 2010. DOI: 10.1007/978-3-642-24466-7-9.

CHAPTER 8

Appendix

We detail below the proposed algorithms. Changes from the CDRL algorithm (Alg 1) are highlighted in red.

Algorithm 2 F-CDRL mixture update

Require: $\eta_t^{(z_j,a)} = \sum_j p_{t,i}^{(z_j,a)} \delta_{c_i}$ for fixed atoms $c_1, \dots c_N$ and representatives $z_1, \dots z_C$

$\eta_t^{(x,a)} \leftarrow \sum_j \mu_j(x) \eta_t^{(z_j,a)}$ ▷ Interpolation

Sample transition (x_t, a_t, r_t, x_{t+1})

if Categorical policy evaluation **then**

  $a^* \sim \pi(\cdot|x_{t+1})$

else if Categorical Q-learning **then**

  $a^* \leftarrow \arg\max_a \mathbb{E}_{R \sim \eta_t^{(x_{t+1},a)}}[R]$

end if

$\hat{\eta}_*^{(x_t,a_t)} \leftarrow (f_{r_t,\gamma})_{\#} \eta_t^{(x_{t+1},a^*)}$ ▷ Distributional Bellman target

$\hat{\eta}_t^{(x_t,a_t)} \leftarrow \Pi_{\mathcal{C}} \hat{\eta}_*^{(x_t,a_t)}$ ▷ Projection onto the support

$\eta_{t+1}^{(z_j,a_t)} \leftarrow (1 - \alpha_t(z_j, a_t)\mu_j(x_t)) \eta_t^{(z_j,a_t)} + \alpha_t(z_j, a_t)\mu_j(x_t) \hat{\eta}_t^{(x_t,a_t)}$ ▷ Mixture update, projection on representatives

return η_{t+1}

Algorithm 3 GF-CDRL mixture update

Require: $\eta_t^{(z_j,a)} = \sum_j p_{t,i}^{(z_j,a)} \delta_{c_i}$ for fixed atoms $c_1, \dots c_N$ and representatives $z_1, \dots z_C$

$\eta_t^{(x,a)} \leftarrow \sum_j \mu_j(x) \left(\eta_t^{(z_j,a)} + \Delta \eta_t^{(z_j,a)} \right)$ ▷ Interpolation with gradients
Sample transition (x_t, a_t, r_t, x_{t+1})
if Categorical policy evaluation **then**
    $a^* \sim \pi(\cdot | x_{t+1})$
else if Categorical Q-learning **then**
    $a^* \leftarrow \arg\max_a \mathbb{E}_{R \sim \eta_t^{(x_{t+1},a)}}[R]$
end if
$\hat{\eta}_*^{(x_t,a_t)} \leftarrow (f_{r_t,\gamma})_{\#} \eta_t^{(x_{t+1},a^*)}$ ▷ Distributional Bellman target
$\hat{\eta}_t^{(x_t,a_t)} \leftarrow \Pi_{\mathcal{C}} \hat{\eta}_*^{(x_t,a_t)}$ ▷ Projection onto the support
$\eta_{t+1}^{(z_j,a_t)} \leftarrow (1 - \alpha_t(z_j, a_t)\mu_j(x_t)) \eta_t^{(z_j,a_t)} + \alpha_t(z_j, a_t)\mu_j(x_t) \hat{\eta}_t^{(x_t,a_t)}$ ▷ Mixture update, projection on representatives
return η_{t+1}

Algorithm 4 Fast F-CDRL mixture update

Require: $\eta_t^{(z_j,a)} = \sum_j p_{t,i}^{(z_j,a)} \delta_{c_i}$ for fixed atoms $c_1, \dots c_N$ and representatives $z_1, \dots z_C$

$\eta_t^{(x,a)} \leftarrow \sum_j \mu_j(x) \eta_t^{(z_j,a)}$ ▷ Interpolation
Sample transition (x_t, a_t, r_t, x_{t+1})
if Categorical policy evaluation **then**
    $a^* \sim \pi(\cdot | x_{t+1})$
else if Categorical Q-learning **then**
    $a^* \leftarrow \arg\max_a \mathbb{E}_{R \sim \eta_t^{(x_{t+1},a)}}[R]$
end if
$\hat{\eta}_*^{(x_t,a_t)} \leftarrow (f_{r_t,\gamma})_{\#} \eta_t^{(x_{t+1},a^*)}$ ▷ Distributional Bellman target
$\hat{\eta}_t^{(x_t,a_t)} \leftarrow \Pi_{\mathcal{C}} \hat{\eta}_*^{(x_t,a_t)}$ ▷ Projection onto the support
$\eta_{t+1}^{(z_j,a_t)} \leftarrow (1 - \alpha_t(z_j, a_t) \frac{\mu_j(x_t)}{\sum_k \mu_k(x_t)^2}) \eta_t^{(z_j,a_t)} + \alpha_t(z_j, a_t) \frac{\mu_j(x_t)}{\sum_k \mu_k(x_t)^2} \hat{\eta}_t^{(x_t,a_t)}$ ▷ Fast mixture update, projection on representatives
return η_{t+1}

List of Figures

List of Tables

List of Algorithms