ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

# Similarities between policy gradient methods in reinforcement and supervised learning

Eric Benhamou[1,2] and David Saltiel[2,3]

1- Lamsade Dauphine, 2- AI Square Connect, 3- ULCO LISIC

**Abstract**. Reinforcement learning (RL) is about sequential decision making and is traditionally opposed to supervised learning (SL) and unsupervised learning (USL). In RL, given the current state, the agent makes a decision that may influence the next state as opposed to SL where the next state remains the same, regardless of decisions taken. Although this difference is fundamental, SL and RL are not so different. In particular, we emphasize in this paper that gradient policy methods can be cast as a SL problem where true label are replaced with discounted rewards. We provide a simple experiment where we interchange label and pseudo rewards to show that SL techniques can be directly translated into RL methods.

**keywords**: Policy gradient, cross-entropy, Kullback Leibler divergence.

## 1 Introduction

In RL, policy gradient methods (PGMs) are frequently used [1, 2, 3, 4]. PGMs are RL techniques that rely upon optimizing the parameters of policies to get the highest expected cumulative reward using gradient descent optimization. PGMs have been popularized in REINFORCE [1] and in [2] and have received wider attention with Actor-Critic methods [5, 6] in particular when using deep PGMs [7] that combine policy and value methods.

When looking in detail in REINFORCE [1], we can remark that the gradient term with respect to the policy can indeed be interpreted as the log term in the cross-entropy in SL. Moreover, if we notice that a RL problem can be reformulated as a SL problem where true labels are changed by expected discounted future rewards, and estimated probabilities by policy probabilities, the link between RL and SL becomes obvious. Besides, leveraging the tight relationship between cross-entropy and Kullback Leibler divergence, we can interpret the entropy regularization terms very naturally. This is precisely the objective of this short paper: call attention to the tight connection between RL and SL to give theoretical justifications of some techniques used in PGMs.

The paper is organized as follows. In section 3, we recall the various choices of functional losses in SL and exhibit that cross-entropy is one of the main possibilities for loss functions. We flaunt the relationship between cross-entropy and Kullback Leibler divergence, harping on the additional entropy term. In section 4, we present PGMs. We rub in the interpretation of RL problems as a modified cross-entropy SL problem. We conclude in section 5 with a financial numerical experience using deep PGM, stressing that in the specific case of actions that do not influence the environment, the difference between RL and SL is very tenuous.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

## 2 Related Work

Looking at similarities and synergies between RL and SL together was for a long time overlooked. This can be easily explained as the two research communities were different and thought they were laboring on incompatible or at least very different approaches. In an attempt to connect SL problem to RL, [8] and [9] noticed however some connections. [8] showed that any SL problem can be relatively easily recast into a one-step RL-learning problem. [9] suggested that in RL, any reward or value function can by explained by goals and that ultimately goals can embed some SL features. However, these connections are only weak and somehow do not show that RL can be recast as a SL problem, which is the purpose of this work. Another stream of research relates also to our approach and is Imitation Learning.

Imitation learning [10] is a classic technique for learning from a human demonstration. Imitation learning uses a supervised approach to imitate an expert's behaviors, hence doing a RL task to accomplish a SL one. DAGGER [11] is considered to be the mainstream imitation algorithm. It requests an action from the expert at each step. It uses an action sampled from a mixed distribution from the agent and the expert. This has led to numerous extensions of this algorithm and in particular a deep version of it (see [12] for a survey on these methods).

All of these works show that RL and SL are not as opposed as one may have thought. We argue here that, ignoring for a while the issue of feedback effect of the action on the next state environment, PGMs can be reformulated as a SL task where true labels are changed into future expected reward.

## 3 Supervised Learning

SL is quite general and encompasses both SL classification and SL regression. The goal of SL classification is to infer a function from labeled training data that maps inputs into labeled outputs. The finding of the function parameters is done traditionally through the optimization of a loss function. SL classifier's parameters are the ones of the optimal solution of the optimization program. To keep things simple, let us take a binary classification problem. Let us assume we observe $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ that are $n$ independent random copies of $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. Features $X$ live in some abstract space $\mathcal{X}$ ($\mathbf{R}^d$ for instance) while labels $Y$ is are binary: $\mathcal{Y} = \{-1, 1\}$. Naturally, one would like to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that best maps $X$ to $Y$. To assist in our goal, we take a loss function $: \{-1, 1\} \times \{-1, 1\} \mapsto \mathbf{R}$ that measures the error of a specific prediction. The loss function value at an arbitrary point $(Y, \hat{Y})$ reads as the cost incurred when predicting $\hat{Y}$ while true label is $Y$. In classification the loss function is often a zero-one loss, that is, $(Y, \hat{Y})$ is zero when the predicted label matches the true label $Y = \hat{Y}$ and one otherwise. To find our best classifier, we look for the classifier with the smallest expected loss. In other words, we look up for the function $f$ that minimizes the expected -risk, given by $\mathcal{R}(h) = \mathbf{E}_{X \times Y}[(Y, f(X))]$.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Typical loss functions are mean square, mean absolute, cross-entropy, logit, hinge and exponential loss. We will show shortly that cross-entropy plays a special role when reconciling SL and RL.

## 4 Reinforcement Learning Background

RL is usually modeled by an agent that interacts with an environment $\mathcal{E}$ over a number of discrete time steps. At each time step $t$, the agent levies a state $s_t$ and picks an action $a_t$ from a set of possible actions $\mathcal{A}$. This choice is made according to its policy $\pi$, where $\pi$ is a mapping from states $s_t$ to actions $a_t$. Once the action is decided and executed, the agent levies the next state $s_{t+1}$ and a scalar reward $r_t$. This goes on until the agent reaches a terminal state. The expected cumulated discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the sum of accumulated returns, where at each time step, future returns are discounted with the discount factor $\gamma \in (0, 1]$. At time $t$, a rational agent seeks to maximize its expected return given his current state $s_t$. We traditionally define

- the value of state $s$ under policy $\pi$ as $V^\pi(s) = \mathbb{E}[R_t|s_t = s]$. It is simply the expected returns for following policy $\pi$ from state $s$ ([13]).
- the action-value function under policy $\pi$ and initial action $a$ as $Q^\pi(s, a) = \mathbb{E}[R_t|s_t = s, a]$. It is simply the expected returns for selecting action $a$ in state $s$ and following policy $\pi$ ( [1]).

Both the optimal value function $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ and the optimal value of state $V^*(s) = \max_\pi V^\pi(s)$ satisfy Bellmann equations.

Whenever states and action are too large, we are forced to represent the action-value function with a function approximator, such as a neural network. Denoting the parameters $\theta$, the state action function is a function of $s, a$ and $\theta$ and is generally called the 'Q' function as follows: $Q(s, a; \theta)$

The updates to $\theta$ can be derived from a variety of reinforcement learning algorithms. In particular, in value-based methods, policy-based model-free methods directly parameterize the policy $\pi(a|s; \theta)$ and update the parameters $\theta$ by performing, typically approximate, gradient ascent on $\mathbb{E}[R_t]$.

An illustration of such a method is REINFORCE due to [1]. Standard REINFORCE updates the policy parameters $\theta$ in the direction $R_t \nabla_\theta \log \pi(a_t|s_t; \theta)$, which is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. It is possible to reduce the variance of this estimate while keeping it unbiased by subtracting a learned function of the state $b_t(s_t)$, known as a baseline [1], from the return. The resulting gradient is $\nabla_\theta \log \pi(a_t|s_t; \theta)(R_t - b_t(s_t))$. This approach can be viewed as an Actor Critic (AC) architecture where the policy $\pi$ is the actor and the baseline $b_t$ is the critic[14, 15]. The terms $\nabla_\theta \log \pi(a_t|s_t; \theta) R_t$ in REINFORCE and $\nabla_\theta \log \pi(a_t|s_t; \theta)(R_t - b_t(s_t))$ in AC method play a special role as is summarized in the proposition below:

**Proposition 4.0.1.** *Gradient descent in REINFORCE is computed by minimizing a cross-entropy term:*

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

$$\widetilde{J}(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{1}$$

*Likewise, in AC methods gradient descent minimizes a cross-entropy term:*

$$\widetilde{J}(\theta) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} A(s_{i,t}, a_{i,t}) \log \pi_\theta(a_{i,t} \mid s_{i,t})}{N} \tag{2}$$

This sheds light on the connection between RL and SL. These connections are summarized in table 1.

| Term | SL | RL (REINFORCE) | RL (A2C) |
|---|---|---|---|
| true label | $Y$ | expected future rewards: $R(\tau)$ | expected advantage: $A(s,a) = Q(s,a) - V(s)$ |
| log term | $\log(\hat{Y})$ | log of policy: $\log \pi_\theta(a_{i,t} \mid s_{i,t})$ | log of policy: $\log \pi_\theta(a_{i,t} \mid s_{i,t})$ |
| cross-entropy | $\frac{\sum_{i=1}^{N} Y_i \log \hat{Y}_i}{N}$ | Monte Carlo expectation: $\frac{\sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t}\|s_{i,t})}{N}$ | Monte Carlo expectation: $\frac{\sum_{i=1}^{N} \sum_{t=1}^{T} A(s_{i,t},a_{i,t}) \log \pi_\theta(a_{i,t}\|s_{i,t})}{N}$ |

Table 1: Comparing SL and RL for REINFORCE and AAC methods

## 5 Experiments

We will apply our remark to a very specific environment where actions do not influence the environment. In this specific case, computing labels as the expected reward entitles us to apply a typical SL method, namely gradient descent on the cross-entropy term to solve a RL problem. The considered reinforcement problem is a financial trading game concerning the Facebook stock (data were retrieved from https://finance.yahoo.com/quote/FB from May 18, 2012 (date of introduction of the stock to the stock market) to December 31, 2018. We denote by $(P_t)_{t=1,\dots}$ the daily closing price of the Facebook stock in sequential order. For each day, we compute the daily return as follows $r_t = \frac{P_t}{P_{t-1}} - 1$

The environment is composed of the last 10 daily returns. As returns are continuous, our state space is $\mathbb{R}_+^n$, which is very large by RL standard. Each day, our possible actions are threefold: do nothing, buy or sell the Facebook stock. If we enter a new position at time $t$, this is materialized only the next day. We will have an open position only at time $t+1$ initialized at the entering price $p_{t+1}$. Hence, if we only keep the position for one period, we will be facing the return $r_{t+2}$ as our position will be only closed at time $t+2$.

As for the reward, we take the Sharpe ratio of the trading strategy. To compute our Mark to Market (the value of our trading strategy), we mark any open position to last known price. We parametrize our policy with a deep network with 2 fully connected ReLU nodes layers. We use ReLU activation

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

function (as opposed to sigmoid) for two main reasons: sparsity and less chance to incur vanishing gradient. Recall a ReLU is defined as $h = max(0, a)$ where $a = Wx + b$. Vanishing gradient has less chance to vanish since the gradient is either zero or constant whenever the linear term is strictly positive ($a > 0$). In contrast, the gradient of sigmoid becomes increasingly small when the absolute value of $x$ tends to be large. This constant character of the ReLU's gradient results in faster learning. The other benefit of ReLUs is sparsity. Sparsity arises when the linear term is non positive: $a \leq 0$. The more units with non positive terms exist in a layer, the more sparse the resulting representation. In contrast, sigmoids always generate some non-zero value resulting in dense representations.

In this experience, we apply a typical SL method to solve this RL problem. Namely, we do gradient descent on a cross-entropy term with the labels computed as the expected reward. We compare various learning rates method with its value ranging from 5%, 25%, and 50% or with an annealing learning rate that decreases linearly from 0.5 to 0.1%. We do up to 250 iterations over a full episode consisting of 1664 daily returns points. We compare the methods with a traditional policy gradient method with an epsilon greedy exploration exploitation method where the parameter of exploration $\epsilon$ is set to 0.25. We provide the average Sharpe ratio of the various methods in figure 1. We see in the resulting experience that SL method achieves a decent performance using gradient descent method compared to the benchmark RL method. The best method is the one with an annealing learning rate decreasing linearly from 0.5 to 0.1%. The intuition of this better performance over other methods is that at first, it is important to have a high learning rate to speed convergence but as we make more and more iterations, it is better to use smaller learning to avoid oscillation in the gradient descent method. It is however important to notice that this SL version of an RL problem uses the fact that actions do not influence states. This is a particularity of our experiment and somehow a limitation of this interpretation of a RL problem as a SL problem.

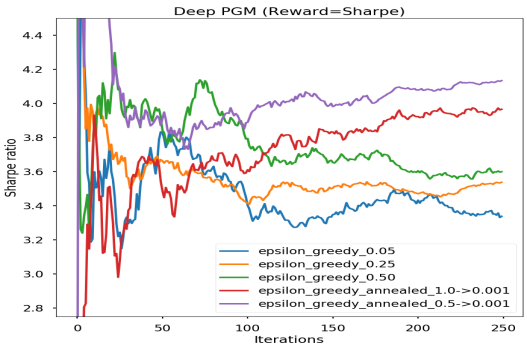


Fig. 1: Various learning rate strategies for our 'SL packaged' RL experiment.

We notice in figure 1 that the overall Sharpe ratio is above 3. This is very high by financial market standards. This could be explained by the fact that

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

Facebook stock has been incredibly rising over the last five years. Hence the algorithm has not much difficulty finding the optimal strategy that is to buy and hold the stock.

## 6 Conclusion

We show in this article that there are tight connections between SL and RL. PGMs in RL can be cast as cross-entropy minimization problems where true labels are replaced by expected future rewards or advantages while the log term is changed into the log policy term. This analogy takes its root from the minimization problem where we are looking for the parameters that maximize the expected future rewards or advantages. Should this optimization objective changed, we conjecture that we could make other analogies between SL and RL.

## References

[1] R. J. Williams. *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, volume 8. Springer, 1992.

[2] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[3] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, number 1 in Proceedings of Machine Learning Research, pages 387–395, Bejing, China, 22-24 Jun 2014. PMLR.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *ArXiv e-prints*, 2015.

[5] Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4):1143–1166, April 2003.

[6] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, March 2008.

[7] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48, pages 1928–1937, NY USA, 20-22 Jun 2016. PMLR.

[8] Alexander L. Strehl. *Associative Reinforcement Learning*, pages 49–51. Springer US, Boston, MA, 2010.

[9] Matthias Rolf and Minoru Asada. Where do goals come from? a generic approach to autonomous goal-system development. *ArXiv*, abs/1410.5557, 2014.

[10] S. Schaal. Learning from demonstration. In *NIPS, MIT Press*, pages 1040–1046. NIPS, MIT Press, 1996.

[11] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *In: AISTATS, JMLR.org, JMLR Proceedings*, 15:627–635, 2011.

[12] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[13] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[14] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[15] Thomas Degris, Patrick M. Pilarski, and Richard S. Sutton. Model-free reinforcement learning with continuous action in practice. *IEEE In ACC*, 2012:2177–2182, 2012.