Université de Montréal

Learning and Planning with Noise in Optimization and Reinforcement Learning

par

Valentin Thomas

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

May 24, 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Learning and Planning with Noise in Optimization and Reinforcement Learning

présentée par

Valentin Thomas

a été évaluée par un jury composé des personnes suivantes :

Nom du président du jury
(président-rapporteur)

Yoshua Bengio
(directeur de recherche)

Nicolas Le Roux
(codirecteur)

Pierre-Luc Bacon
(membre du jury)

Philip S. Thomas
(examinateur externe)

Nom du représentant du doyen
(représentant du doyen de la FESP)

Résumé

La plupart des algorithmes modernes d'apprentissage automatique intègrent un certain degré d'aléatoire dans leurs processus, que nous appellerons le *bruit*, qui peut finalement avoir un impact sur les prédictions du modèle. Dans cette thèse, nous examinons de plus près l'apprentissage et la planification en présence de bruit pour les algorithmes d'apprentissage par renforcement et d'optimisation.

Les deux premiers articles présentés dans ce document se concentrent sur l'apprentissage par renforcement dans un environnement inconnu, et plus précisément sur la façon dont nous pouvons concevoir des algorithmes qui utilisent la stochasticité de leur politique et de l'environnement à leur avantage. Notre première contribution présentée dans ce document se concentre sur le cadre de l'apprentissage par renforcement non supervisé. Nous montrons comment un agent laissé seul dans un monde inconnu sans but précis peut apprendre quels aspects de l'environnement il peut contrôler indépendamment les uns des autres, ainsi qu'apprendre conjointement une représentation latente démêlée de ces aspects que nous appellerons *facteurs de variation*. La deuxième contribution se concentre sur la planification dans les tâches de contrôle continu. En présentant l'apprentissage par renforcement comme un problème d'inférence, nous empruntons des outils provenant de la littérature sur les méthodes de Monte Carlo séquentiel pour concevoir un algorithme efficace et théoriquement motivé pour la planification probabiliste en utilisant un modèle appris du monde. Nous montrons comment l'agent peut tirer parti de note objectif probabiliste pour imaginer divers ensembles de solutions.

Les deux contributions suivantes analysent l'impact du bruit de gradient dû à l'échantillonnage dans les algorithmes d'optimisation. La troisième contribution examine le rôle du bruit de l'estimateur du gradient dans l'estimation par maximum de vraisemblance avec descente de gradient stochastique, en explorant la relation entre la structure du bruit du gradient et la courbure locale sur la généralisation et la vitesse de convergence du modèle. Notre quatrième contribution revient sur le sujet de l'apprentissage par renforcement pour analyser l'impact du bruit d'échantillonnage sur l'algorithme d'optimisation de la politique par ascension du gradient. Nous constatons que

le bruit d'échantillonnage peut avoir un impact significatif sur la dynamique d'optimisation et les politiques découvertes en apprentissage par renforcement.

Mots clés: Apprentissage de représentations, Contrôle par Inférence Probabiliste, Apprentisage Profond par Renforcement, Planification, Optimisation stochastique, Généralisation.

Abstract

Most modern machine learning algorithms incorporate a degree of randomness in their processes, which we will refer to as noise, which can ultimately impact the model's predictions. In this thesis, we take a closer look at learning and planning in the presence of noise for reinforcement learning and optimization algorithms.

The first two articles presented in this document focus on reinforcement learning in an unknown environment, specifically how we can design algorithms that use the stochasticity of their policy and of the environment to their advantage. Our first contribution presented in this document focuses on the unsupervised reinforcement learning setting. We show how an agent left alone in an unknown world without any specified goal can learn which aspects of the environment it can control independently from each other as well as jointly learning a disentangled latent representation of these aspects, or factors of variation. The second contribution focuses on planning in continuous control tasks. By framing reinforcement learning as an inference problem, we borrow tools from Sequential Monte Carlo literature to design a theoretically grounded and efficient algorithm for probabilistic planning using a learned model of the world. We show how the agent can leverage the uncertainty of the model to imagine a diverse set of solutions.

The following two contributions analyze the impact of gradient noise due to sampling in optimization algorithms. The third contribution examines the role of gradient noise in maximum likelihood estimation with stochastic gradient descent, exploring the relationship between the structure of the gradient noise and local curvature on the generalization and convergence speed of the model. Our fourth contribution returns to the topic of reinforcement learning to analyze the impact of sampling noise on the policy gradient algorithm. We find that sampling noise can significantly impact the optimization dynamics and policies discovered in on-policy reinforcement learning.

Keywords: Representation Learning, Control as Inference, Deep Reinforcement Learning, Planning, Stochastic Optimization, Generalization.

Table des matières

Liste des tableaux

Liste des figures

Other non included works

---

not sure there should be a section for it or what name it should have

- **On the role of overparameterization in off-policy Temporal Difference learning with linear function approximation** (**Valentin Thomas***). https://openreview.net/forum?id=g-H3oNARs2, *In NeurIPS 2022*.
- **The Role of Baselines in Policy Optimization** (with Jincheng Mei*, Wesley Chung, **Valentin Thomas**, Bo Dai, Csaba Szepesvari and Dale Schuurmans). https://arxiv.org/abs/2301.06276, *In NeurIPS 2022*.
- - **Bridging the Gap Between Target Networks and Functional Regularization** (Alexandre Piché*, **Valentin Thomas***, Joseph Marino, Rafael Pardiñas, Gian Maria Marconi, Christopher Pal, Mohammad Emtiyaz Khan), *https://arxiv.org/abs/2210.12282*.
- **Planning with Latent Simulated Trajectories** (Alexandre Piché*, **Valentin Thomas**, Cyril Ibrahim, Yoshua Bengio, Julien Cornebise and Chris Pal), *In ICLR 2019 Workshop on Structure & Priors in Reinforcement Learning*.
- **Independently Controllable Features** (Emmanuel Bengio*, **Valentin Thomas**, Joelle Pineau, Doina Precup and Yoshua Bengio), *In RLDM 2017*.

Notation and acronyms

In this thesis we will use the convention of denoting matrices with capital bold letters. Vectors and scalars will be denoted by lowercase letters and the distinction will be made clear from context.

Notation

Information theory		
X	$\triangleq$	Random variable
$\mathbb{E}$	$\triangleq$	Expectation
$H(X)$	$\triangleq$	Entropy of X
$\mathcal{I}(X,Y)$	$\triangleq$	Mutual information between X and Y
$\mathcal{D}_{\text{KL}}$	$\triangleq$	Kullback-Leibler Divergence
Machine learning and reinforcement learning		
θ or ϕ	$\triangleq$	Learnable parameters of a model
$\mathcal{L}$	$\triangleq$	Loss function
$\mathcal{J}$	$\triangleq$	Objective function to maximize
s	$\triangleq$	State (vector)
a	$\triangleq$	Action (scalar or vector)
r	$\triangleq$	Reward (scalar)
R	$\triangleq$	Return (scalar)
$\tau_{1:T}$	$\triangleq$	$\{s_i, a_i\}_{i=1}^{T}$ Trajectory: sequence of state-action pairs
π	$\triangleq$	A policy, i.e a distrubtion over actions given a state
γ	$\triangleq$	Discount factor
λ	$\triangleq$	Trace-decay parameter for TD(λ)
V^{π}	$\triangleq$	Value function associated to the policy π
Q^{π}	$\triangleq$	State-action value function associated to the policy π
$\mathcal{O}_t$	$\triangleq$	Optimality variable used in control as inference
p_{env}	$\triangleq$	Transition probability of the environment
$\mathbf{P}$	$\triangleq$	Transition matrix of the environment
p_{model}	$\triangleq$	Model of the environment
w_t	$\triangleq$	Importance sampling weight

Acronyms

Acronym	Full name
ML	Machine Learning
DL	Deep Learning
MLE	Maximum Likelihood Estimation
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MDP	Markov Decision Process
TD	Temporal Difference
ICF	Independently Controllable Factors
IS	Importance Sampling
MPC	Model Predictive Control
CEM	Cross Entropy Method
HMM	Hidden Markov Model
MCTS	Monte Carlo Tree Search
MCMC	Markov Chain Monte Carlo
SMC	Sequential Monte Carlo
RS	Random Shooting
SAC	Soft Actor Critic
SG	Stochastic Gradient
SGD	Stochastic Gradient Descent
PG	Policy Gradient
NPG	Natural Policy Gradient

Remerciements

Je dédicace cette thèse tout d'abord à ma famille, à ma mère, mon père et ma soeur pour avoir toujours été lá, m'avoir soutenu dans mes choix et pour avoir toujours encouragé ma curiosité dès mon plus jeune âge.

Je voudrais ensuite remercier mes directeurs de thèse, Yoshua et Nicolas. Leur soutien et leurs conseils ont été essentiels pour mener à bien cette thése. Je remercie également les membres de mon jury de thèse,

These: Yoshua, Nicolas

Stage: Philippe, Marlos, Bilal, Remi, Theophane

Je remercie es amis pour les beaux moments que nous avons partageés pendant cet aventure. Á mes amis de France, Pierrick, Marc et ceux rencontrés à Montréal, Florian, Meta, Simon, Thomas, Jules, Veronica, Melissa, Linda et Audrey.

Amis: Alexandre, Stephanie, Gauthier, Salem, Victor, Ahmed, Thomas, Jules, Marc, Pierrick, Akram Wesley Melissa Veronica Linda Florian Strub Simon Guiroy Gabriel Tristan

Audrey Tess

Pets: Idefix, Yuki

Partner: Melodie

Chapitre 1

Introduction

Learning through interaction is a fundamental aspect of natural intelligence: humans and animals learn from their experience in order to develop complex behaviors. For this reason, a primary long-term goal in the field of artificial intelligence is to create machines capable of emulating this process, able to interact with our world to perform specific tasks or achieve general objectives. Reinforcement learning (RL) is a key paradigm for learning how to interact; it is a subfield of machine learning in which an agent learns to act through trial and error. This approach has demonstrated promising results in various applications, such as games (Tesauro, 1994; Mnih et al., 2013; Silver et al., 2016) and robotics (Levine et al., 2018).

Many concepts in reinforcement learning share connections with other fields. For instance, the concept of rewards and punishments in RL is closely related to the dopamine system in the brain (Schultz et al., 1997). This analogy later inspired the Temporal Difference algorithm (Sutton, 1988), which we will explain in Section 2.3. Some researchers explore the intersection between psychology and reinforcement learning, as there are parallels between animal learning through conditioning and the learning process of RL agents (Sutton & Barto, 1981). Reinforcement learning is also strongly linked to fundamental notions in optimal control, such as Bellman equations (Bellman et al., 1954; Bellman & Kalaba, 1959), which eventually led to the development of highly successful algorithms like Q-Learning (Watkins & Dayan, 1992).

In optimal control, most algorithms are *planning algorithms*. They aim to determine the optimal course of action by considering hypothetical situations, often without direct interaction with the environment. In planning, an agent uses a model of the environment, either learned or provided, to simulate potential state transitions and rewards, using it as a policy or something thereby improving its policy for better decision-making. Planning methods hold significant importance in RL, with notable examples like Monte

Carlo Tree Search (MCTS), an algorithm that constructs a search tree of potential state-action trajectories to determine the optimal action for the current state.

However these methods rely on a model of the world which is often unknown, and estimating it can be a challenging task. *Learning algorithms* often refer to methods that do not need extra simulated experience but only learn from direct trial and error. Through this process, the agent incrementally updates its knowledge about the environment, often represented as value functions (e.g., state-value or action-value functions) or policy parameters. Over time, as the agent accumulates more experience, it can refine its policy to better navigate the environment and achieve higher rewards. Some popular learning algorithms include Q-learning, SARSA, and various policy gradient methods.

In this thesis, we are interested in learning and planning algorithms in the presence of stochasticity. First we will present briefly some notions of information theory, optimization and reinforcement learning useful to understand the four contributions presented in the subsequent chapters.

Our first contribution in "learning disentangled independently controllable factors of variation" (Bengio et al., 2017; Thomas et al., 2017, 2018) addresses the issue of how an agent, without any specified goal, can comprehend its environment by interacting with it, discovering controllable elements, and constructing useful internal representations of these aspects of the world. We propose and investigate a direct mechanism, inspired from how children learn, that explicitly connects an agent's control over its environment to its internal feature representations. We then show how these jointly learned policies and *independently controllable factors* lead to learning a disentangled latent representation that can be used for planning.

However, even when the agent is able to build an appropriate representation or model of the world, how to use it to decide which action to take is another challenge. We tackle this one with our second work "Probabilistic planning with sequential Monte Carlo Methods" by designing a novel planning algorithm. By interpreting the set of solutions to a task as a distribution as in *control as inference* (Toussaint & Storkey, 2006; Toussaint, 2009; Levine, 2018), we show how we can use a particle filter method to estimate this distribution. This yield a new, intuitive and theoretically grounded algorithm for planning in stochastic continuous domains.

Our next two contributions are concerned with the process of learning itself and how it can be affected by noise. The third paper we present examines the role that gradient noise and local curvature can have on the optimization speed and the generalization of a model. We show how a simple metric can measure the capacity of a model more effectively that the raw number of parameters. In our fourth and last contribution, we investigate the role of gradient noise for policy gradient methods in bandits and RL. More specifically we take a closer look at *baselines*, an ubiquitous algorithmic choice in policy

gradient methods motivated by variance reduction purposes. In accordance with classical optimization theorems, the prevailing view among researchers is that gradient noise leads to slower convergence in RL as well. However, in this paper, we show that this view is flawed and that there is an important interplay between the gradient noise and the propensity of the agent to try new actions, which can ultimately lead to discovering better policies.

Chapitre 2

Background

In this background section, we present the key topics necessary for understanding the contributions of this PhD thesis. We give brief overviews of information theory, fundamentals of machine learning and reinforcement learning.

2.1. Information theory

How to quantify information and discrepancies between probability distributions is at the heart of machine learning and an important prerequisite for understanding our contributions of Chapter 3, Chapter 4 and Chapter 5.

2.1.1. Probability distribution and density/mass function

For simplicity, in this section, we will not introduce the notion of probability distribution in its most general form using measure theory, but only using probability density functions. For a more complete treatment, please refer to Kolmogorov & Bharucha-Reid (2018); Billingsley (2008).

Definition 1 (Probability Mass Function (pmf))**.** *Let us consider a discrete random variable $X \in \mathcal{X}$ where $\mathcal{X}$ is discrete and a function $p : \mathcal{X} \mapsto \mathbb{R}$. We say that p is the probability mass function of X if*

(1) $p(x) \geq 0, \forall x \in \mathcal{X}$

(2) $\sum_{x \in \mathcal{X}} p(x) = 1$

(3) $\mathbb{P}(X = x) = p(x)$

In the context of continuous random variables, we can define the notion of *probability density function* which has a very similar definition.

Definition 2 (Probability Density Function (pdf))**.** *Let us consider a random variable $X \in \mathcal{X}$ and a function $p : \mathcal{X} \mapsto \mathbb{R}$. We say that p is the probability density function of X if*

(1) $p(x) \geq 0, \forall x \in \mathcal{X}$

(2) $\int_{\mathcal{X}} p(x)dx = 1$

(3) $\mathbb{P}(X \in A) = \int_{x \in A} p(x)dx$

We will make use of this concept extensively as most machine learning methods can be understood as trying to approximate a distribution p (for instance the data distribution) with an approximate distribution q.

2.1.2. Divergences

While distances are the typical notion used for comparing mathematical objects, a weaker notion called "divergences" are often used to compare probability measures.

Definition 3 (Distance)**.** *A function d on a set $\mathcal{X}$ between two objects x and $y \in \mathcal{X}$ must satisfy four properties to be a distance*

(1) non-negativity: $d(x,y) \geq 0$

(2) identity of indiscernibles: $d(x,y) = 0$ *if and only if* $x = y$

(3) symmetry: $d(x,y) = d(y,x)$

(4) triangle inequality: $\forall z \in \mathcal{X},\ d(x,y) \leq d(x,z) + d(z,y)$

A divergence, on the other hand, only requires the *non-negativity* and *identity of the indiscernibles* properties and is defined for probability distributions.

Definition 4 (Divergence)**.** *A function $\mathcal{D}(\cdot||\cdot)$ between two distributions p and q must satisfy two properties to be a divergence*

(1) non-negativity: $\mathcal{D}(p||q) \geq 0$

(2) identity of indiscernibles: $\mathcal{D}(p||q) = 0$ *if and only if* $p = q$

The most emblematic divergence, especially in machine learning and generally Maximum Likelihood Estimation (MLE) is certainly the Kullback-Leibler divergence (Kullback & Leibler, 1951; Kullback, 1997).

Example 1 (Kullback-Leibler divergence)**.** *The Kullback-Leibler divergence is defined by*

$$\mathcal{D}_{\mathrm{KL}}(p||q) = \int_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx \tag{2.1.1}$$

The Kullback-Leibler divergence contains the term $-\int_{x \in \mathcal{X}} p(x) \log q(x) dx$ which is the negative log-likelihood of q under the distribution p, which is also known as the *cross-entropy*. This divergence is very common in machine learning as we will see in Section 2.2.

2.1.3. Measures of information

Now that we have defined the fundamental notions of probabilities and divergences, we can make us of them to build useful concepts from information theory such as *entropy* or *mutual information*.

The notion of entropy is fundamental in physics where it was first introduced by Boltzmann. Shannon (1948) transcribed the concept to communication theory and it

is now used widely in statistics and machine learning. For a discrete random variable X with mass function p, the entropy $H(X)$ of X is

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x). \tag{2.1.2}$$

Entropy should be understood as the *information* necessary to describe p. As such, the entropy is always positive $H(X) \geq 0$: it is equal to 0 only when p is a Dirac function; in that case p does not contain any uncertainty. $H(X)$ is also bounded by the number of values X can take: $H(X) \leq \log|\mathcal{X}|$ which is achieved when p is the uniform distribution, the distribution with maximal uncertainty. Note that entropy can be defined in the same way for continuous variables, we call this the *differential entropy* $h(X) = -\int_{\mathcal{X}} p(x)\log(x)dx$ but we lose the upper and lower bounds described above.

Using the notion of Kullback-Leibler divergence notion defined above and by defining the uniform distribution whose probability mass function is $u(x) = \frac{1}{|\mathcal{X}|},\ \forall x \in \mathcal{X}$, we have

$$\mathcal{D}_{\mathrm{KL}}(p||u) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{u(x)} = \log|\mathcal{X}| - H(X)$$

Thus, entropy can also be interpreted as a measure of how close we are to the uniform distribution.

If we consider the case where we have two random variables X and Y with joint probability mass function $p(x,y)$ and marginals $p(x)$ and $p(y)$ we can write the joint entropy as

$$\begin{aligned} H(X,Y) &= -\sum_{x,y} p(x,y)\log p(x,y) \\ &= -\sum_{x,y} p(x,y)\big(\log p(y|x) - \log p(x)\big) \\ &= -\sum_{x} p(x)\sum_{y} p(y|x)\log p(y|x) - \sum_{x} p(x)\log p(x) \\ &= H(Y|X) + H(X) \end{aligned}$$

where we defined $H(Y|X) = -\sum_x p(x)\sum_y p(y|x)\log p(y|x)$, the *conditional entropy*. $H(Y|X)$ should be understood as the amount of information necessary to describe Y given that we know everything about X.

Finally we can introduce the idea of *mutual information, i.e* how much information is shared between X and Y. The mutual information $\mathcal{I}(X,Y)$ can be written in several different manners

$$\begin{aligned}\mathcal{I}(X,Y) &= H(X)+H(Y)-H(X,Y)\\ &= H(X)-H(X|Y)\\ &= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\ &= \mathcal{D}_{\mathrm{KL}}\big(p(x,y)||p(x)p(y)\big)\end{aligned}$$

see Figure 1 for a schematic view.

Another useful interpretation of mutual information is to see it as the divergence between the joint probability and the product of the marginals. When X and Y and conditional independent, the divergence, thus mutual information, is 0. The concept of mutual information is especially relevant in reinforcement learning where it has been used (in many different ways) as an intrinsic reward signal (Still & Precup, 2012; Mohamed & Rezende, 2015; Gregor et al., 2016; Thomas et al., 2018; Eysenbach et al., 2018; Eslami et al., 2018).

2.2. Fundamentals of machine learning and optimization

2.2.1. Setting and maximum likelihood estimation

At a high level, *machine learning* is the science concerned with *learning from data*. This discipline is at the crossroads between different fields such as statistics, computer science and optimization. *Data* is represented in the form of a dataset $\mathcal{D}_{\text{train}} = \{x_i\}_{i=1\ldots N}$

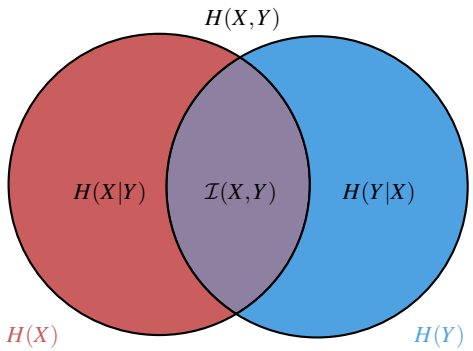


Fig. 1. Mutual information is the information shared between X and Y. From this Venn diagram, we recover the first two definitions of the mutual information in term of the joint and conditional entropy. For instance the red circle represents the entropy associated to X, but as the purple intersection is the mutual information between X and Y, the red circle minus the purple intersection represents the uncertainy of X conditioned on us knowing Y, i.e $H(X|Y)$.

where x_i are the N *training examples* assumed to be sampled independently and identically distributed (i.i.d) from a distribution p, the true data distribution. In this context *learning* means finding a *function* or *model* that "fits" the data according to some loss function $\mathcal{L}$.

Mathematically speaking, this is predominantly framed as a parametric optimization problem

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\theta}(x_i)$$

where $\mathcal{L}$ is our loss function and $\theta \in \mathbb{R}^d$ is the parameter vector we optimize over. With the advent of large neural networks, θ can be a very high dimensional vector, up to hundreds of billions/ a few trillions of parameters as of late 2022 (Chowdhery et al., 2022; Fedus et al., 2022), in opposition with classical methods for which θ was often low dimensional (compared to the number of training examples).

Commonly, our *model* is a parametric function q_{θ} (for instance a neural network) whose goal is to approximate p. A natural objective to ensure q_{θ} becomes closer to p is to maximize the likelihood of the training examples x_i sampled from p under q_{θ}, i.e maximizing $q_{\theta}(\mathcal{D}_{\text{train}}) = \Pi_{i=1}^{N} q_{\theta}(x_i)$ as per the i.i.d assumption. As the logarithm is a non-decreasing function and N is constant, we can choose to maximize instead

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log q_{\theta}(x_i).$$

It appears clearly here that maximizing the likelihood of the data is equivalent to minimizing the negative log-likelihood $\mathcal{L}_{\theta}(\cdot) = -\log q_{\theta}(\cdot)$ averaged over $\mathcal{D}_{\text{train}}$.

Note that this **maximum likelihood** (or minimum negative log-likelihood) objective can be related to a Kullback-Leibler divergence. By calling $\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}(x)$ the empirical distribution over the training samples, we have that

$$\begin{aligned} \mathcal{D}_{\text{KL}}(\hat{p} || q_{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} \log \hat{p}(x_i) - \log q_{\theta}(x_i) \\ &= -H(\hat{p}) + \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\theta}(x_i) \end{aligned}$$

As the entropy of $\hat{p}$ does not depend on θ, it appears clearly that our objective is equivalent to minimizing a divergence between the empirical distribution of p and q_{θ}.

2.2.2. Generalization

Even when maximizing likelihood, one may learn a model q_θ which is accurate on the training data but inaccurate on unseen data. The discrepancy between the loss on the true data distribution and the training loss is called the generalization gap

$$\mathcal{G} = \mathbb{E}_{x\sim p}[\mathcal{L}_\theta(x)] - \mathbb{E}_{x\sim \hat{p}}[\mathcal{L}_\theta(x)]$$

The generalization gap and how to estimate it is at the core of Chapter 5: in our article we show how one can build estimators of $\mathcal{G}$ from the local curvature and variance of our model.

2.2.3. Stochastic gradient descent

Now we turn back our attention to the original optimization problem $\min_\theta \frac{1}{N}\sum_{i=1}^N \mathcal{L}_\theta(x_i)$. When using neural networks, we can efficiently estimate $\nabla_\theta \mathcal{L}_\theta$, the gradient of $\mathcal{L}_\theta$, using the backpropagation algorithm (Rumelhart et al., 1986). Thus, we can use the *gradient descent* algorithm to minimize our loss

$$\theta_{t+1} = \theta_t - \eta \frac{1}{N}\sum_{i=1}^N \nabla_\theta \mathcal{L}_{\theta_t}(x_i) \tag{2.2.1}$$

where η is a positive scalar called the learning rate. Under some smoothness assumptions on $\mathcal{L}$ and for a small enough η, we can show that this algorithm will converge to a local minimum of our objective function.

This algorithm is however expensive when N is very large as is common in modern machine learning. An alternative to regular gradient descent is to use **stochastic gradient descent** (Robbins & Monro, 1951), i.e, we only use a sample, or more generally, a mini-batch of B samples to compute a noisy estimate of the gradient. This trade-off between noise and complexity of computing a gradient estimate is well-understood and favorable in the regime where N is large (Bottou & Bousquet, 2007).

$$\theta_{t+1} = \theta_t - \eta_t \frac{1}{B}\sum_{i=1}^B \nabla_\theta \mathcal{L}_{\theta_t}(x_i), \quad x_1, \dots x_B \overset{\text{i.i.d}}{\sim} \mathcal{D}_{\text{train}} \tag{2.2.2}$$

Where η_t is a time-dependent learning rate usually annealed to 0 to ensure convergence. This algorithm is the central idea behind all popular optimization algorithms used for training deep neural networks, such as Adam (Kingma & Ba, 2014) which uses momentum and where the gradient is preconditioned by a data-dependent "normalization" matrix.

2.3. Reinforcement Learning

- Add figure for RL interaction either here or in intro

2.3.1. General setting and Markov Decision Processes

Reinforcement learning (RL) is a sequential decision making problem where an agent can take *decisions* or *actions* in a world, called *environment*, in order to maximize some signal called the *reward*. This is formalized as a Markov Decision Process (MDP) as described in Bellman et al. (1954) and Puterman (2014). An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, p_{\text{env}}, r, \mu \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions that the agent can take, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, a function that maps a state and action to a scalar and $p_{\text{env}}(s'|s,a), \quad s', s \in \mathcal{S} \times \mathcal{S}, a \in \mathcal{A}$ is the transition function, a probability distribution over states given the current state and action. The assumption that the next state only depends only on the current state and the action taken by the agent, and not on previous states or actions is referred to as *the Markov assumption*.

The goal of the agent is to learn a probability distribution over actions called a policy $\pi(a_t|s_t)$ that maximizes the discounted sum of the reward

$$\mathcal{J}(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p_{\text{env}}(\cdot|s_t,a_t)}[\sum_t \gamma^t r(s_t, a_t)]$$

where γ is called the *discount factor*. In the episodic case, where sequences of states and actions ultimately reach an ending state, γ has to be positive and smaller or equal to 1. In the continuing setting, where an agent interacts with the world indefinitely, we need $\gamma \in [0,1[$ to ensure the objective remains bounded. In Deep Reinforcement Learning, the policy π is usually obtained using a neural network. In policy gradient methods (Section 2.3.3.1) the policy is directly parametrized by a neural network with weights θ so we will refer to this parametrized policy network as π_θ and the objective will be denoted either $\mathcal{J}(\pi_\theta)$ or $\mathcal{J}(\theta)$.

In practice, the agent will interact with the environment in the following manner

Algorithm 1 Sample a trajectory

// Initialize starting state
$s_0 \sim \mu(\cdot)$
$\tau_{0:0} = \{\}$
for t in $\{0, \ldots, \infty\}$ **do**
    // Sample and execute action
    $a_t \sim \pi(a_t|s_t)$
    $s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t, a_t)$
    // Update trajectory and return
    $\tau_{0:t} \leftarrow \tau_{0:t-1} \cup \{s_t, a_t\}$
    $R_t \leftarrow R_{t-1} + \gamma^t r_t$
    If s_{t+1} is terminal, **break**
end for

Where a trajectory $\tau_{0:t}$ is the sequence of state-action pairs $\tau_{0:t} = \{(s_0,a_0), \ldots, (s_t,a_t)\}$ encountered and R_t is the empirical discounted return, i.e the sum of discounted rewards for this trajectory. It appears clearly that this process is highly stochastic as it requires sampling actions from our policy (which could have a high entropy) as well as sampling the next state and reward from the environment transition dynamics p_{env}, which is a priori unknown and could be highly unpredictable. Therefore, even for a given policy π, the trajectory sampled τ and its associated discounted return R_t can be vastly different every time Algorithm 1 is run. This *noise* arising from the interaction between the agent and the environment at the core of most contributions presented in this thesis.

Furthermore, under some conditions[1] the MDP admits a unique *stationary distribution* d^π. Algorithm 1, if ran with enough times, would eventually sample states and actions according to d^π. If we are interested in the discounted return, we can adapt Algorithm 1 by adding a probability $1-\gamma$ of restarting from the initial distribution μ at every step and never break the interaction loop. The stationary distribution of this discounted MDP will be referred to as d^π_γ.

While in traditional optimization we assume we can evaluate our objective function immediately, in reinforcement learning it needs to be evaluated through this noisy interactive process. Thus, it is not surprising that many of the algorithms used in reinforcement learning have the following structure

[1] For a unique stationary distribution to exist, the MDP must be *irreducible* and *aperiodic*.

Algorithm 2 Alternating policy evaluation and policy improvement

1: Choose an initial policy π_0
2: **for** t in $\{0,\dots,T\}$ **do**
3: *// Policy evaluation*
4: Estimate $\mathcal{J}(\pi_t)$ though interaction (real or simulated)
5: *// Policy improvement*
6: Improve π_t to π_{t+1} based on the evaluation of π_t
7: **end for**

The next two subsections will thus respectively be concerned with the *policy evaluation* and *policy improvement* problems.

2.3.2. Value functions in reinforcement learning

Alongside with the policy π, one of the most important concept in reinforcement learning is the notion of *value function*. Intuitively, it is a measure of how "good" a current situation is for the agent, or more precisely "how much discounted return" we can expect to gather from a given situation.

2.3.2.1. Value and Q functions

We first define the value and Q functions. Both are expectations of the future discounted return, but while the value is conditional on a state s, for the Q function we condition on both a state and an action s, a.

Thus, for the value function V^π

$$V^\pi(s_t) = \mathbb{E}\big[\sum_{t'\geq t} \gamma^{t'-t} r_{t'} | a_t \sim \pi(\cdot|s_t), s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t,a_t), \dots\big] \tag{2.3.1}$$

The Q function is the expected discounted return under the current policy conditioned on the current state and action

$$Q^\pi(s_t,a_t) = \mathbb{E}\big[\sum_{t'\geq t} \gamma^{t'-t} r_{t'} | s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t,a_t), a_{t+1} \sim \pi(\cdot|s_{t+1}), \dots\big] \tag{2.3.2}$$

These two functions can be easily related to our original objective

$$\mathcal{J}(\pi) = \mathbb{E}_{s_0\sim\mu}[V^\pi(s_0)] = \mathbb{E}_{s_0\sim\mu, a_0\sim\pi(\cdot|s_0)}[Q^\pi(s_0,a_0)]$$

Thus the value function averaged over the initial state is the objective function we aim at evaluating.

2.3.2.2. Bellman equations for V^π and Q^π

Because of the Markov assumption, i.e that the distribution of s_{t+1} and r_t only depends on s_t and a_t, the value of a state can be expressed in function of the value of its successor states. Indeed, the value and Q functions verify a recursion known as the Bellman equation

$$\begin{aligned}
V^\pi(s_t) &= \mathbb{E}_{a_t,s_{t+1}\dots}\Big[\sum_{t'\geq t}\gamma^{t'-t}r_{t'}|s_t\Big] \\
&= \mathbb{E}_{a_t,s_{t+1}\dots}\Big[r_t+\gamma\sum_{t'\geq t+1}\gamma^{t'-(t+1)}r_{t'}|s_t\Big] \\
&= \mathbb{E}_{a_t,s_{t+1}}\Big[r_t+\gamma\mathbb{E}_{a_{t+1},s_{t+2}\dots}[\sum_{t'\geq t+1}\gamma^{t'-(t+1)}r_{t'}|s_t,a_t,s_{t+1}\Big] \quad \text{(Law of Total Expectation)} \\
&= \mathbb{E}_{a_t,s_{t+1}}\Big[r_t+\gamma\mathbb{E}_{a_{t+1},s_{t+2}\dots}[\sum_{t'\geq t+1}\gamma^{t'-(t+1)}r_{t'}|s_{t+1}\Big] \quad \text{(Markov property)} \\
&= \mathbb{E}[r_t+\gamma V^\pi(s_{t+1})] \qquad (2.3.3)
\end{aligned}$$

In the same manner

$$Q^\pi(s_t,a_t)=\mathbb{E}[r_t+\gamma Q^\pi(s_{t+1},a_{t+1})] \qquad (2.3.4)$$

More generally, by unrolling the equation for n steps instead of just one, we can get

$$V^\pi(s)=\mathbb{E}[\sum_{t'=t}^{t+n-1}r_{t'}+\gamma^n V^\pi(s_{t+n})] \qquad (2.3.5)$$

And we will refer to $R_t^n \triangleq \sum_{t'=t}^{t+n-1} r_{t'}+\gamma^n V^\pi(s_{t+n})$ as the *n-step return*, which is a random variable conditioned on state s_t. The expected 0-step return is simply the value function of s_t while the ∞-step return (i.e where we unroll until the episode stops) is the empirical discounted return R_t. This circles back to the definition of $V^\pi(s_t)$ as being the expected empirical return from s_t. As all the n-step returns are unbiased estimates of the value function, it is also possible to mix them in order to obtain new estimators such as the λ-return which weights the n-step returns according to a geometric distribution of parameter λ

$$R_t^\lambda \triangleq (1-\lambda)\sum_{n\geq 1}\lambda^{n-1}\sum_{t'=t}^{t+n-1}r_{t'}+\gamma^n V^\pi(s_{t+n}) \qquad (2.3.6)$$

For $\lambda=0$, we get back the 1-step return $r_t+\gamma V^\pi(s_t)$ while for $\lambda\to 1$, R_t^λ is the Monte Carlo return, i.e the empirical discounted return until the end of the episode.

2.3.2.3. Learning parametric value functions

Now we will see how to use the properties of the true value function V^π to build a parametric estimator of it. While there are non-parametric methods, often referred to

as *memory-based* methods (Atkeson et al., 1997), nowadays in most settings the value function is parametrized via a neural network. Let us call ϕ the parameters of the value (respectively Q) function approximator V_ϕ (resp Q_ϕ).

As the value function satisfies the Bellman equation eq. (2.3.4), we can learn a function that satisfies the same equation

$$\min_\phi \tfrac{1}{2}\mathbb{E}_{s,a}\left[\mathbb{E}_{s',r}[\big(r(s,a)+\gamma V_\phi(s')] - V_\phi(s)\big)^2\right] \tag{2.3.7}$$

where the expectations are taken over transitions s', r, s, a encountered during a trajectory. This loss is referred to as the Mean Square Bellman Error (MSBE). However, taking the gradient of this loss directly poses some practical challenges

$$\nabla_\phi \text{MSBE}(\phi) = \mathbb{E}_{s,a}\left[\mathbb{E}_{s',r}[\big(r(s,a)+\gamma V_\phi(s')] - V_\phi(s)\big)\cdot\big(\mathbb{E}_{s',r}[\gamma\nabla_\phi V_\phi(s')] - \nabla_\phi V_\phi(s)\big)\right]$$

While we used the notation s' in the two expectations $\mathbb{E}_{s',r}$ we need to have access to two independent samples of $s' \sim p_{\text{env}}(\cdot|s,a)$ for this gradient to be unbiased, and it is not something we can do easily without access to a simulator of the environment. This is known as the *double sampling* problem (Baird, 1995). In order to circumvent this issue, we can "fix" the *target* $r(s,a)+\gamma V_\phi(s')$ and not differentiate it. This lead to the *pseudo-gradient*

$$-\mathbb{E}_{s,a,s',r}\left[\big(r(s,a)+\gamma V_\phi(s') - V_\phi(s)\big)\cdot\nabla_\phi V_\phi(s)\right]$$

This update rule using this pseudo-gradient is known as the TD(0) algorithm, which stands for **Temporal Difference** learning (Sutton, 1988) and it can be expressed as an expectation over a transition (s,a,r,s') is suitable for use with stochastic gradient descent and deep neural networks. Therefore, it remains one of the most popular methods for learning value functions to this day.

Furthermore we can extend TD(0) to TD(λ) by using a λ-return for the target, i.e R_t^λ instead of $r_t + \gamma V_\phi(s_t)$.

2.3.3. Policy optimization

Ultimately, as mentioned previously, our goal is to improve $\mathcal{J}$, given our current policy π we aim at finding a new one π' yielding a higher expected return, i.e $\mathcal{J}(\pi') \geq \mathcal{J}(\pi)$. In the next subsections, we will present *Policy gradient* and *policy greedification*, the two main families of policy improvement methods used in modern reinforcement learning.

2.3.3.1. Policy gradient and actor critic methods

To learn a new policy via gradient ascent, we need to differentiate through the objective $\mathcal{J}(\pi_\theta)$ with respect to the policy parameter θ. We have (Williams, 1992; Sutton et al.,

1999)

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{s,a\sim d^\pi_\gamma(s,a)}[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)] \tag{2.3.8}$$

Note that this requires knowledge of the true Q-function and an expectation over all states to be exact. In practice, we use an estimator for $Q^\pi(s,a)$ and perform a *stochastic gradient* update by sampling $s,a \sim d^\pi$ by rolling out trajectories in the environment[2].

When we use the Monte Carlo estimate $R(s,a)$ instead of $Q^{\pi_\theta}(s,a)$, we usually refer to this method simply as "vanilla policy gradient" as we don't necessarily need to learn a parametric value function to improve our policy.

This stochastic gradient can be broken into two parts: the actor and the critic

$$\underbrace{\Psi(s,a)}_{\text{"critic"}} \nabla_\theta \log \underbrace{\pi_\theta(a|s)}_{\text{"actor"}}, \quad s,a \sim d^\pi$$

While the "actor" is simply our policy which samples actions, the "critic" part is a scalar quantity that appears in front the gradient of the log probabilities $\nabla_\theta \log \pi_\theta(a|s)$ and evaluates the desirability of action a in state s. As we saw previously, the true Q-function $Q^{\pi_\theta}(s,a)$ leads to a valid gradient estimator, but there is a variety of choices used in practice for the critic. As we may not have access to the true Q-function, one can use an approximation of it, Q_ϕ, learned with TD methods for instance. Alternatively we might employ a bootstrapped estimator of the value function as presented in Section 2.3.2.2 such as $r(s,a) + \gamma V_\phi(s')$ or a λ-return instead of Q_ϕ. This class of algorithms where we learn one set of parameters for the policy to take actions and another set of parameters for the critic to judge the value of the action is called **actor-critic algorithms.**

The most popular choices for critics are called **advantage functions**. These critics are *centered* estimates of the Q function. As $\mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)] = V^\pi(s)$ advantage functions can be of the form $A_\phi(s,a) = Q_\phi(s,a) - V_\phi(s)$, $A_\phi(s,a) = r(s,a) + \gamma V_\phi(s') - V_\phi(s)$ or generally using a n-step or λ-return in lieu of the Q function estimator.

[2]Note here that sampling from d^π instead of d^π_γ, while theoretically incorrect is widely used in practice. See (Nota & Thomas, 2019) for a more in-depth discussion.

Historically, advantage functions have been motivated by variance reduction arguments. Indeed, for any function dependent on state only $b(s)$ (such as the value function), which we will refer to as a *baseline*, we have

$$\begin{aligned}
\mathbb{E}_{s,a\sim d_\gamma^\pi(s,a)}[b(s)\nabla_\theta \log \pi_\theta(a|s)] &= \mathbb{E}_{s\sim d_\gamma^\pi(s)}\big[b(s)\mathbb{E}_{a\sim\pi_\theta(a|s)}[\nabla_\theta \log \pi_\theta(a|s)]\big] \\
&= \mathbb{E}_{s\sim d_\gamma^\pi(s)}\big[b(s)\mathbb{E}_{a\sim\pi_\theta(a|s)}[\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}]\big] \\
&= \mathbb{E}_{s\sim d_\gamma^\pi(s)}\big[b(s)\sum_a \nabla_\theta \pi_\theta(a|s)\big] \\
&= \mathbb{E}_{s\sim d_\gamma^\pi(s)}\big[b(s)\nabla_\theta \underbrace{\sum_a \pi_\theta(a|s)}_{=1}\big] \\
&= 0 \quad \textit{(gradient of a constant is zero)}
\end{aligned}$$

Thus, adding or removing any state-dependent baseline $b(s)$ (such as the value function) to the Q function does not bias the policy gradient. Conceptually, as the baseline does not depend on the action a, it does not affect the relative preference of one action over another under policy $\pi_\theta(a|s)$. Because of the apparent link with the notion of *control variates* in statistics —these are functions with zero mean that, when chosen carefully, can reduce variance— the role of the baseline $b(s)$ is often motivated by variance reduction arguments. However, in Chapter 6 we will take a closer look at the role of baselines in policy gradient methods and show that they have an important impact on the optimization process, beyond their variance reduction property.

We now illustrate here how a simple actor-critic algorithm using a 1-step return advantage could be implemented

2.3.3.2. Policy greedification and value-based methods

Greedification is a totally different method that allows us to derive policies directly from Q-functions and as such we do not need to parameterize π by θ. We define the *greedy* policy π' with respect to Q^π as

$$\pi'(a|s) = \mathbf{1}_{\{a=\arg\max_\alpha Q^\pi(s,\alpha)\}}(a) \tag{2.3.9}$$

Thus the greedy policy is a deterministic policy which places all of its probability on the best action according to Q^π. It can be shown (Sutton & Barto, 2018) that the greedy policy is an improvement over π, we have $V^{\pi'}(s) \geq V^\pi(s)\ \forall s$, in particular $\mathcal{J}(\pi') \geq \mathcal{J}(\pi)$.

SARSA and Q-Learning (Watkins & Dayan, 1992) are examples of well-known *value-based* algorithms: both learn an approximate value function using Temporal Difference learning and then perform a greedification step using the current Q function estimate. While SARSA performs the TD step on the current policy π (i.e its 1-step return target

Algorithm 3 Simple actor critic algorithm with 1-step return

Choose initial θ, ϕ.
for episode = 1 to M **do**
  Initialize state $s_0 \sim \mu$
  for t = 0 to T **do**
    // Interact with the environment
    $a_t \sim \pi_{\theta_t}(\cdot|s_t)$
    $s_{t+1}, r_t \sim p_{\text{env}}(\cdot,\cdot|s_t, a_t)$
    $A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
    If s_{t+1} is terminal **then** $A_t = r_t - V_\phi(s_t)$
    // Policy evaluation
    $\phi \leftarrow \phi + \eta_\phi \cdot A_t \nabla_\phi V_\phi(s)$
    // Policy improvement
    $\theta \leftarrow \theta + \eta_\theta \cdot A_t \nabla_\theta \log \pi_\theta(a|s)$
    // Update state
    If s_{t+1} is terminal **then break**
  end for
end for

would be $r(s_t, a_t) + \gamma Q_\phi(s_{t+1}, a_{t+1})$ where $a_{t+1} \sim \pi(\cdot|s_{t+1})$), on the other hand Q-Learning uses a target with a look-ahead step as the next action is sampled from π', thus the target is $r(s_t, a_t) + \gamma Q_\phi(s_{t+1}, a')$, $a' \sim \pi'(\cdot|s_{t+1})$ or equivalently $r(s_t, a_t) + \gamma \max_a Q_\phi(s_{t+1}, a)$ as π' is the greedy policy.

Algorithm 4 Q-learning

// Initialize Q network
Choose initial ϕ.
for episode = 1 to M **do**
  Initialize state $s_0 \sim \mu$
  for t = 0 to T **do**
    // Sample and execute action
    With probability ε select a random action a, otherwise $a \leftarrow \text{argmax}_a Q_\phi(s_t, a)$
    $s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t, a_t)$
    $\delta_t = r_t + \gamma \max_{a'} Q_\phi(s_{t+1}, a') - Q_\phi(s_t, a_t)$
    If s_{t+1} is terminal **then** $\delta_t = r_t - Q_\phi(s_t, a_t)$
    // Temporal Difference update on the greedy policy
    $\phi \leftarrow \phi + \eta_\phi \cdot \delta_t \nabla_\phi Q_\phi(s_t, a_t)$
    If s_{t+1} is terminal **then break**
  end for
end for

Note here that in the algorithm we sampled actions from an $\varepsilon-$greedy policy. While we could sample greedily from Q by taking its argmax, it is common practice to use a more random ε-greedy policy that has a ε probability of sampling other actions as well.

This enables us to *explore* actions and states we might not have encountered otherwise. We will discuss briefly the exploration problem in Section 2.3.3.4.

This algorithm has been very successful, and Deep Q-Learning (Mnih et al., 2013), a slightly modified version of this algorithm for deep reinforcement learning, was the first algorithm to reach human level performance on the ALE benchmark (Bellemare et al., 2013).

2.3.3.3. General conditions for improvement

While policy gradient-based methods and greedification-based ones are conceptually different, it is still possible to understand them as optimizing the same objective for our policy.

We can write a more general, non-local, version of the policy gradient theorem to compare more generally two policies π' and π using a variant of the performance difference lemma (Kakade & Langford, 2002)

$$\mathcal{J}(\pi') - \mathcal{J}(\pi) = \mathbb{E}_{s\sim d_\gamma^{\pi'}}\big[\sum_a \big(\pi'(a|s) - \pi(a|s)\big)Q^\pi(s,a)\big] \tag{2.3.10}$$

This equation is a generalization of the policy gradient theorem as taking the limit $\lim_{t\to 0}\frac{\mathcal{J}(\theta+t\delta\theta)-\mathcal{J}(\theta)}{t}$ would lead back to the policy gradient. Alternatively, as the discounted stationary distribution $d_\gamma^{\pi'}$ is always positive, we could look for the policy π' that maximizes $\sum_a \big(\pi'(a|s) - \pi(a|s)\big)Q^\pi(s,a)$, which is achieved for the greedy policy of Equation (2.3.9). Thus with one unified objective we can understand how the two main methods for policy improvement are related. While policy gradient chooses infinitesimal step to increase the return, greedification performs more drastic updates by directly transitioning to the greedy policy.

2.3.3.4. Exploration

The drastic updates of the greedification exemplify one of the main challenges of reinforcement learning. Let's say we are in a bandit setting, *id est* there is only one state and we have to find the action that leads to the best reward. It may be that our initial guess is wrong because of the stochasticity of the reward function. For instance if our action is to choose which restaurant we would like to eat at, we may not be able to identify the best restaurant overall by only tasting one item from the menu. Therefore to be able to identify the best action, we need to *explore* enough each possibility in order to find the optimal policy. This is at odds with greedification which purely *exploits* based on our current guess of what the value of each action is. This tradeoff between *exploration*, taking suboptimal actions in order to potentially discover better ones, and *exploitation*, taking the

best action we know in order accumulate reward, is at the heart of reinforcement learning. While exploration is an active research topic, we will only mention here the two simplest and most widely used schemes for policy gradient and Q-learning.

For actor-critic and policy gradient methods, it is common to add an *entropy bonus* $H(a|s)$, which measures the randomness of the action distribution, to the reward in order to encourage our policy to try out different actions. For value-based methods using a greedification step, the most common strategy for exploration is using an ε-greedy policy, ie, we select the argmax with probability $1-\varepsilon$ and another action at random with probability ε. ε is typically decayed over time so that the policy ultimately becomes the greedy policy.

Chapitre 3

Independently Controllable Factors

Article details

Thomas V*, Bengio E*, Fedus W*, Pondard J, Beaudoin P, Larochelle H, Pineau J, Precup D, Bengio Y. "Disentangling the independently controllable factors of variation by interacting with the world". Presented at the *NeurIPS 2017 workshop on Learning Disentangled Representations: from Perception to Control* as an oral talk.

Previous iterations of this paper have been presented at *Reinforcement Learning and Decision Making (RLDM) 2017* and at the *Montreal AI Symposium*.

Foreword

This project began at first in January 2017 and led to several short papers and involved many authors from different institutions. The first one, Bengio et al. (2017) was published at RLDM 2017, a subsequent and longer paper, Thomas et al. (2017) was presented at the Montreal AI Symposium 2017, and finally, a latter and more theoretically sound version, Thomas et al. (2018) was presented as a spotlight paper in the NeurIPS 2017 workshop on Learning Disentangled Features: from Perception to Control.

The original motivation for this project was the way young children spontaneously learn to discover what they can do, how they can affect the world and the surrounding objects in a totally unsupervised manner (Berlyne, 1966; Gopnik et al., 1999). To do so, they associate aspects of the world they can control to a representation of such aspect -or object- in their brain. In this line of work, we looked at, in particular, aspects of the world that can be modified and represented independently from each other: we call them **independently controllable factors of variations**. This combines two objectives into one: (1) the agent has to discover without supervision a diverse set of policies it can execute, and (2) each policy must be mapped to a representation in the latent space.

The first point has been the focus of several works where, as in our work, the objective is similar to a mutual information criterion between the observed states and the label of the policy (or option/skill/context used) (Still & Precup, 2012; Mohamed & Rezende, 2015; Gregor et al., 2016; Florensa et al., 2017; Eysenbach et al., 2018; Achiam et al., 2018).

The second point, learning disentangled representation for reinforcement learning, has been investigated by Anand et al. (2019) where they annotate by hand *attributes* of the world and learn a representation that shares a high mutual information with those attributes. In a more complex 3D world, Eslami et al. (2018) learn a representation of a scene by encoding the information about different viewpoints at once. Works combining both the idea of exploring and learning a good representation of the world are more rare. We can cite Kim et al. (2019), where they use a mutual information objective to help exploration and learning of representation (this is however not unsupervised) and Li & Mandt (2018) a follow-up work on the contribution presented here where they propose a simple mechanism to discover factors that cannot be controlled by the agent.

A very challenging aspect of this work was to be able to learn a diversity of meaningful factors. In the end, we always observed what we called a *factor collapse* where a few interesting factors would be learned but many would remain undiscovered no matter the amount of training. While we were able to characterize and understand this problem very well, we did not manage at the time to solve it. Recently Strouse et al. (2021) proposed a solution to this issue by discriminating between *aleatoric* and *epistemic* uncertainties using an ensemble of neural networks, thus encouraging the system to be more curious about learning new factors rather than exploiting the ones already discovered.

Personal contribution

- Theoretical understanding of what objective ICF is optimizing. Making the link with (causal) mutual information
- Understanding and showcasing how our structured representation can be used for planning and inference
- Empirical validation (code + visualization) for most experiments presented in this paper

Abstract

It has been postulated that a good representation is one that disentangles the underlying explanatory factors of variation. However, it remains an open question what kind of training framework could potentially achieve that. Whereas most previous work focuses on the static setting (e.g., with images), we postulate that some of the causal factors could be discovered if the learner is allowed to interact with its environment. The agent can experiment with different actions and observe their effects. More specifically, we hypothesize that some of these factors correspond to aspects of the environment which are independently controllable, i.e., that there exists a policy and a learnable feature for each such aspect of the environment, such that this policy can yield changes in that feature with minimal changes to other features that explain the statistical variations in the observed data. We propose a specific objective function to find such factors, and verify experimentally that it can indeed disentangle independently controllable aspects of the environment without any extrinsic reward signal.

3.1. Introduction

When solving Reinforcement Learning problems, what separates great results from random policies is often having the right feature representation. Even with function approximation, learning the right features can lead to faster convergence than blindly attempting to solve given problems (Jaderberg et al., 2016).

The idea that learning good representations is vital for solving most kinds of real-world problems is not new, both in the supervised learning literature (Bengio, 2009; Goodfellow et al., 2016), and in the RL literature (Dayan, 1993; Precup, 2000b). An alternate idea is that these representations do not need to be learned explicitly, and that learning can be guided through internal mechanisms of reward, usually called intrinsic motivation (Barto et al.; Oudeyer & Kaplan, 2009; Salge et al., 2013; Gregor et al., 2017).

We build on a previously studied (Thomas et al., 2017) mechanism for representation learning that has close ties to intrinsic motivation mechanisms and causality. This mechanism explicitly links the agent's control over its environment to the representation of the environment that is learned by the agent. More specifically, this mechanism's hypothesis is that most of the underlying factors of variation in the environment can be controlled by the agent independently of one another.

We propose a general and easily computable objective for this mechanism, that can be used in any RL algorithm that uses function approximation to learn a latent space. We show that our mechanism can push a model to learn to disentangle its input in a

meaningful way, and learn to represent factors which take multiple actions to change and show that these representations make it possible to perform model-based predictions in the learned latent space, rather than in a low-level input space (e.g. pixels).

3.2. Learning disentangled representations

The canonical deep learning framework to learn representations is the autoencoder framework (Hinton & Salakhutdinov, 2006). There, an encoder $f : S \to H$ and a decoder $g : H \to S$ are trained to minimize the *reconstruction error*, $\|s - g(f(s))\|_2^2$. H is called the latent (or representation) space, and is usually constrained in order to push the autoencoder towards more desirable solutions. For example, imposing that $H \in \mathbb{R}^K, S \in \mathbb{R}^N$, $K \ll N$ pushes f to learn to compress the input; there the bottleneck often forces f to extract the principal factors of variation from S. However, this does not necessarily imply that the learned latent space disentangles the different factors of variations. Such a problem motivates the approach presented in this work.

Other authors have proposed mechanisms to disentangle underlying factors of variation. Many deep generative models, including variational autoencoders (Kingma & Welling, 2014) , generative adversarial networks (Goodfellow et al., 2014) or non-linear versions of ICA (Dinh et al., 2014; Hyvarinen & Morioka, 2016) attempt to disentangle the underlying factors of variation by assuming that their joint distribution (marginalizing out the observed s) factorizes, i.e., that they are marginally independent.

Here we explore another direction, trying to exploit the ability of a learning agent to act in the world in order to impose a further constraint on the representation. We hypothesize that interactions can be the key to learning how to disentangle the various causal factors of the stream of observations that an agent is faced with, and that such learning can be done in an unsupervised way.

3.3. The selectivity objective

We consider the classical reinforcement learning setting but in the case where extrinsic rewards are not available. We introduce the notion of **controllable factors of variation** $\phi \in \mathbb{R}^K$ which are generated from a neural network $\Phi(h,z), z \sim \mathcal{N}(0,1)^m$ where $h = f(s)$ is the current latent state. The factor ϕ represents an embedding of a policy π_ϕ whose goal is to realize the variation ϕ in the environment.

To discover meaningful factors of variation ϕ and their associated policies π_ϕ, we consider the following general quantity $\mathcal{S}$ which we refer to as selectivity and that is used as

a reward signal for π_ϕ:

$$\mathcal{S}(h,\phi) = \mathbb{E}\left[\log \frac{A(h',h,\phi)}{\mathbb{E}_{p(\varphi|h)}[A(h',h,\varphi)]} \;\middle|\; s' \sim \mathbf{P}_{ss'}^{\pi_\phi}\right] \tag{3.3.1}$$

Here $h = f(s)$ is the encoded initial state before executing π_ϕ and $h' = f(s')$ is the encoded terminal state. ϕ and φ represent factors of variation a *factor*. $A(h',h,\phi)$ should be understood as a score describing how close ϕ is to the variation it caused in (h',h). For example in the experiments of section 4.1, we choose A to be a gaussian kernel between $h'-h$ and ϕ, while in the experiments of section 4.2, we choose $A(h',h,\phi) = \max\{0,\langle h'-h,\phi\rangle\}$. The intuition behind these objectives is that in expectation, a factor ϕ should be close to the variation it caused (h',h) when following π_ϕ compared to other factors φ that could have been sampled and followed thus encouraging **independence** within the factors.

Conditioned on a scene representation h, a distribution of policies are feasible. Samples from this distribution represent ways to modify the scene and thus may trigger an internal selectivity reward signal. For instance, h might represent a room with objects such as a light switch. $\phi = \phi(h,z)$ can be thought of as the distributed representation for the "name" of an underlying factor, to which is associated a policy and a value. In this setting, the light in a room could be a factor that could be either on or off. It could be associated with a policy to turn it on, and a binary value referring to its state, called an attribute or a feature value.We wish to jointly learn the policy $\pi_\phi(\cdot|s)$ that modifies the scene, so as to control the corresponding value of the attribute in the scene, whose variation is computed by a scoring function $A(h',h,\phi) \in \mathbb{R}$. In order to get a distribution of such embeddings, we compute $\phi(h,z)$ as a function of h and some random noise z.

The goal of a selectivity-maximizing model is to find the density of factors $p(\phi|h)$, the latent representation h, as well as the policies π_ϕ that maximize $\mathbb{E}_{p(\phi|h)}[\mathcal{S}(h,\phi)]$.

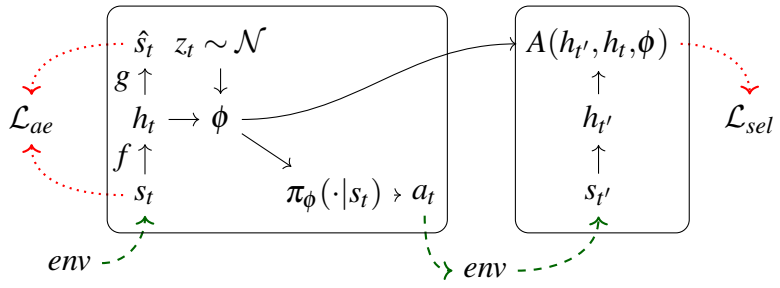


Fig. 1. The computational model of our architecture. s_t is the first state, from its encoding h_t and a noise distribution z, ϕ is generated. ϕ is used to compute the policy π_ϕ, which is used to act in the world. The sequence h_t,$h_{t'}$ is used to update our model through the selectivity loss, as well as an optional autoencoder loss on h_t.

3.3.1. Link with mutual information and causality

The selectivity objective, while intuitive, can also be related to information theoretical quantities defined in the latent space. From (Donsker & Varadhan, 1975; Ruderman et al., 2012) we have $\mathcal{D}_{\mathrm{KL}}\big(p||q\big) = \sup_{A\in\mathcal{L}^{\infty}(q)} \mathbb{E}_p[\log A] - \log \mathbb{E}_q[A]$. Applying this equality to the mutual information $\mathcal{I}_p(\phi, h'|h) = \mathbb{E}_{p(h'|h)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(\phi|h',h)||p(\phi|h)\big)\big]$ gives

$$\mathcal{I}_p(\phi, h'|h) \geq \sup_{\theta} \mathbb{E}_{p(\phi|h)}\big[\mathcal{S}(h,\phi)\big]$$

where θ is the set of weights shared by the factor generator, the policy network and the encoder.

Thus, our total objective along entire trajectories is a lower bound on the causal (Ziebart, 2010) or directed (Massey, 1990) information $\mathcal{I}_p(\phi \mapsto h) = \sum_t \mathcal{I}_p(\phi_{1:t}, h_t|h_{t-1})$ which is a measure of the **causality** the process ϕ exercises on the process h. See Appendix A.3 for details.

3.4. Experiments

We use MazeBase (Sukhbaatar et al., 2015) to assess the performance of our approach. We do not aim to solve the game. In this setting, the agent (a red circle) can move in a small environment (64×64 pixels) and perform the actions `down, left, right, up`. The agent can go anywhere except on the orange blocks.

3.4.1. Learned representations

After jointly training the reconstruction and selectivity losses, our algorithm disentangles four directed factors of variations as seen in Figure 2: $\pm x$-position and $\pm y$-position of the agent. For visualization purposes we chose the bottleneck of the autoencoder to be of size $K = 2$. To complicate the disentanglement task, we added the redundant action `up` as well as the action `down+left` in this experiment.

The disentanglement appears clearly as the latent features corresponding to the x and y position are orthogonal in the latent space. Moreover, we notice that our algorithm assigns both actions `up` (white and pink dots in Figure 2.a) to the same feature. It also does not create a significant mode for the feature corresponding to the action `down+left` (light blue dots in Figure 2.a) as this feature is already explained by features `down` and `left`.

3.4.2. Towards planning and policy inference

This disentangled structure could be used to address many challenging issues in reinforcement learning. We give two examples in figure 2:

[1] pink and white for `up`, light blue for `down+left`, green for `right`, purple black `down` and night blue for `left`.

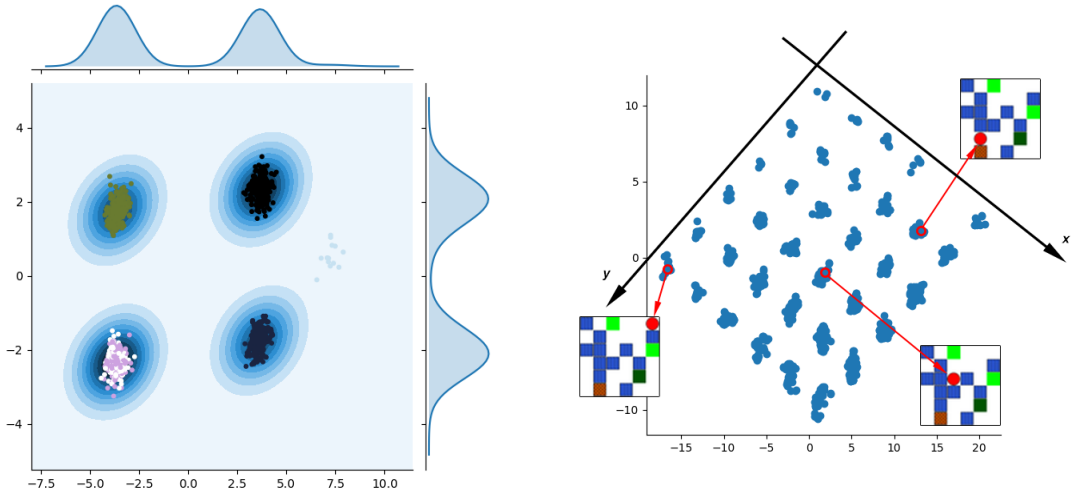


Fig. 2. (a) Sampling of 1000 variations $h' - h$ and its kernel density estimation encountered when sampling random controllable factors ϕ. We observe that our algorithm disentangles these representations on 4 main modes, each corresponding to the action that was actually taken by the agent.[1] (b) The disentangled structure in the latent space. The x and y axis are disentangled such that we can recover the x and y position of the agent in any observation s simply by looking at its latent encoding $h = f(s)$. The missing point on this grid is the only position the agent cannot reach as it lies on an orange block.

- Model-based predictions: Given an initial state, s_0, and an action sequence $a_{\{0:T-1\}}$, we want to predict the resulting state s_T.
- A simplified deterministic policy inference problem: Given an initial state s_{start} and a terminal state s_{goal}, we aim to find a suitable action sequence $a_{\{0:T-1\}}$ such that s_{goal} can be reached from s_{start} by following it.

Because of the *tanh* activation on the last layer of $\phi(h,z)$, the different factors of variation $dh = h' - h$ are placed on the vertices of a hypercube of dimension K, and we can think of the the policy inference problem as finding a path in that simpler space, where the starting point is h_{start} and the goal is h_{goal}. We believe this could prove to be a much easier problem to solve.

However, this disentangled representation alone cannot solve completely these two issues in an arbitrary environment. Indeed, the only factors we are able to disentangle are the factors directly *controllable* by the agent, thus, we are not able to account for the ambient dynamics or other agents' influence.

3.4.3. Multistep embedding of policies

In this experiment, ϕ are embeddings of 3-steps policies π_ϕ. We add a model-based loss $\mathcal{L}_{MB} = ||h_{t+3} - T_\theta(h_t, \phi)||^2$ defined only in the latent space, and jointly train a decoder alongside with the encoder. Notice that we never train our model-based cost at pixel

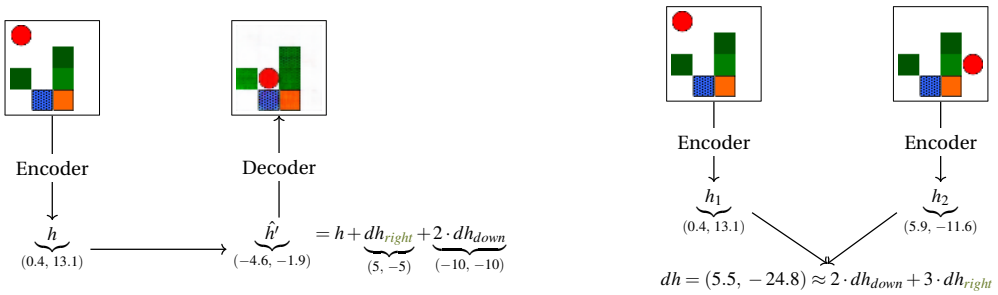


Fig. 3. (left) Predicting the effect of a cause on Mazebase. The leftmost image is the visual input of the environment, where the agent is the round circle, and the switch states are represented by shades of green. After the training, we are able to distinguish one cluster per dh (Figure 2), that is to say per variation obtained after performing an action, independently from the position h. Therefore, we are able to move the agent just by adding the corresponding dh to our latent representation h. The second image is just the reconstruction obtained by feeding the resulting h' into the decoder. (right) Given a starting state and a goal state, we are able to decompose the difference of the two representations dh into a (non-directed) sequence of movements.

level. While we currently suffer from mode collapsing of some factors of variations, we show that we are successfully able to do predictions in latent space, reconstruct the latent prediction with the decoder, and that our factor space disentangles several types of variations.

3.5. Conclusion, success and limitations

Pushing representations to model independently controllable features currently yields some encouraging success. Visualizing our features clearly shows the different controllable aspects of simple environments, yet, our learning algorithm is unstable.

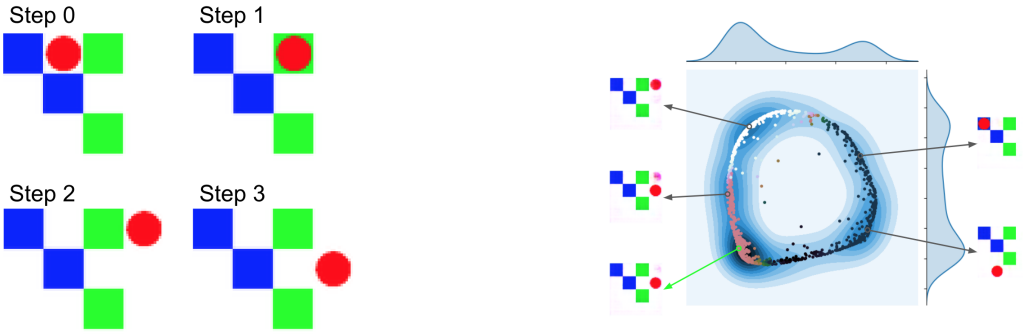


Fig. 4. (a) The actual 3-step trajectory done by the agent. (b) PCA view of the space $\phi(h_0, z), z \sim \mathcal{N}(0,1)$. Each arrow points to the reconstruction of the prediction $T_\theta(h_0, \phi)$ made by different ϕ. The ϕ at the start of the green arrow is the one used by the policy in (a). Notice how its prediction accurately predicts the actual final state.

What seems to be the strength of our approach could also be its weakness, as the independence prior forces a very strict separation of concerns in the learned representation, and should maybe be relaxed.

Some sources of instability also seem to slow our progress: learning a conditional distribution on controllable aspects that often collapses to fewer modes than desired, learning stochastic policies that often optimistically converge to a single action, tuning many hyperparameters due to the multiple parts of our model. Nonetheless, we are hopeful in the steps that we are now taking. Disentangling happens, but understanding our optimization process as well as our current objective function will be key to further progress.

Chapitre 4

Probabilistic Planning with Sequential Monte Carlo Methods

Article details

Thomas, V.*, Piché, A.*, , Ibrahim, C., Bengio, Y. and Pal, C. "Probabilistic Planning with Sequential Monte Carlo methods". In *International Conference on Learning Representations (ICLR) 2019*.

This article was also presented as a contributed talk at the NeurIPS 2018 workshop on Infer to Control.

This work was done jointly with Alexandre Piché during the summer 2018 at Mila and ElementAI and was presented at ICLR 2019.

Foreword

The original intuition for this work was that it should be possible to have a tree search planning algorithm where interesting (*i.e* might get high return) branches were reinforced and less interesting branches cut off. Using control as inference was a natural way to associate the return a branch might get to the probability it would have to be reinforced or cut.

To the best of our knowledge, there is only one article that framed planning as an inference problem (Attias, 2003) and it was in a very specific setting with strong assumptions. We realized that our idea was intimately linked with sequential Monte Carlo methods and that our algorithm could be framed as an instance of a particle filter. In control theory, there is a duality between estimating the current state and controlling the dynamics to the desired goal. While particle filters are typically used for estimation, here, within the context of control as inference, we are able to design a particle filter algorithm for control.

Impact since publication

This paper already has some citations and follow-up works using our formulation of planning as inference and our algorithm (Wang et al., 2019; Lioutas et al., 2022). An evalution paper, (Byravan et al., 2022), found that SMCP performed favorably compared to CEM, both on performance and computational complexity "[...] on the **harder GTTP tasks SMC slightly outperforms CEM**. We use SMC throughout the paper as it makes better use of the proposal [distribution] compared to CEM [...] **CEM uses a significantly larger computational budget than our SMC planner** which is non-iterative; in spite of this SMC is still quite competitive with CEM across all tasks [...]"

A book (Belousov et al., 2021) cites SMCP as a promising direction for planning: "Recent research in probabilistic dynamic models and planning with sequential Monte Carlo methods viewing control as an inference problem demonstrate the advantages of probabilistic planning in MPC and may be **one of the most promising directions**[to improve MPC in a black box environment]."

Personal contribution

- Major contribution on the theoretical understanding of the method. Making the link with the two-filter formula (Bresler, 1986) and smoothing/forward-backward algorithms
- Determining the expression for the update (maximum entropy advantage) and writing the proofs in the appendix
- Designing and performing the toy experiment Figure 4b and Figure 4a
- Creating Figure 1, Figure 3 and Figure 2
- Writing of the paper alongside with Alexandre

Abstract

In this work, we propose a novel formulation of planning which views it as a probabilistic inference problem over future optimal trajectories. This enables us to use sampling methods, and thus, tackle planning in continuous domains using a fixed computational budget. We design a new algorithm, Sequential Monte Carlo Planning, by leveraging classical methods in Sequential Monte Carlo and Bayesian smoothing in the context of *control as inference*. Furthermore, we show that Sequential Monte Carlo Planning can capture multimodal policies and can quickly learn continuous control tasks.

4.1. Introduction

To exhibit intelligent behaviour machine learning agents must be able to learn quickly, predict the consequences of their actions, and explain how they will react in a given situation. These abilities are best achieved when the agent efficiently uses a model of the world to plan future actions. To date, planning algorithms have yielded very impressive results. For instance, Alpha Go (Silver et al., 2017) relied on Monte Carlo Tree Search (MCTS) (Kearns et al., 2002) to achieve super human performances. Cross entropy methods (CEM) (Rubinstein & Kroese, 2004) have enabled robots to perform complex nonprehensile manipulations (Finn & Levine, 2017) and algorithms to play successfully Tetris (Szita & Lörincz, 2006). In addition, iterative linear quadratic regulator (iLQR) (Kalman et al., 1960; Kalman, 1964; Todorov & Li, 2005) enabled humanoid robots tasks to get up from an arbitrary seated pose (Tassa et al., 2012).

Despite these successes, these algorithms make strong underlying assumptions about the environment. First, MCTS requires a discrete setting, limiting most of its successes to discrete games with known dynamics. Second, CEM assumes the distribution over future trajectories to be Gaussian, i.e. unimodal. Third, iLQR assumes that the dynamics are locally linear-Gaussian, which is a strong assumption on the dynamics and would also assume the distribution over future optimal trajectories to be Gaussian. For these reasons, planning remains an open problem in environments with continuous actions and complex dynamics. In this paper, we address the limitations of the aforementioned planning algorithms by creating a more general view of planning that can leverage advances in deep learning (DL) and probabilistic inference methods. This allows us to approximate arbitrary complicated distributions over trajectories with non-linear dynamics.

We frame planning as density estimation problem over optimal future trajectories in the context of *control as inference* (Dayan & Hinton, 1997; Toussaint & Storkey, 2006; Toussaint, 2009; Rawlik et al., 2010, 2012; Ziebart, 2010; Levine & Koltun, 2013). This perspective allows us to make use of tools from the inference research community and, as previously mentioned, model any distribution over future trajectories. The planning distribution is complex since trajectories consist of an intertwined sequence of states and actions. Sequential Monte Carlo (SMC) (Stewart & McCarty, 1992; Gordon et al., 1993; Kitagawa, 1996) methods are flexible and efficient to model such a distribution by sequentially drawing from a simpler proposal distribution. From the SMC perspective, the policy can be seen as the proposal and a learned model of the world as the propagation distribution. This provides a natural way to combine model-free and model-based RL.

Contribution. We depict the problem of planning as one of density estimation that can be estimated using SMC methods. We introduce a novel planning strategy based on the SMC class of algorithms, in which we treat the policy as the proposed distribution to be learned. We investigate how our method empirically compares with existing model-based methods and a strong model-free baseline on the standard benchmark Mujoco (Todorov et al., 2012).

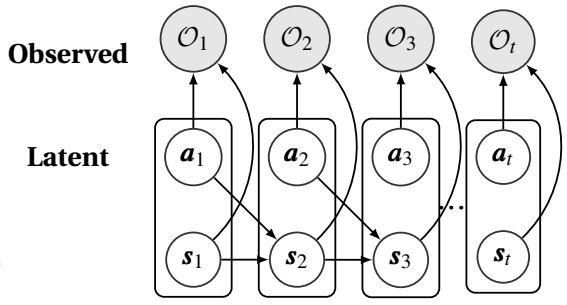
4.2. Background

4.2.1. Control as inference

We consider the general case of a Markov Decision Process (MDP) $\{\mathcal{S}, \mathcal{A}, p_{\text{env}}, r, \gamma, \mu\}$ where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces respectively. We use the letters s and a to denote states and actions, which we consider to be continuous vectors. Further notations include: $p_{\text{env}}(s'|s,a)$ as the state transition probability of the environment, $r(s,a)$ as the reward function, and $\gamma \in [0,1)$ as the discount factor. μ denotes the probability distribution over initial states.

This work focuses on an episodic formulation, with a fixed end-time of T. We define a trajectory as a sequence of state-action pairs $\tau_{t:T} = \{(s_t,a_t),\ldots,(s_T,a_T)\}$, and we use the notation π for a policy which represents a distribution over actions conditioned on a state. Here π is parametrized by a neural network with parameters θ. The notation $q_\theta(\tau_{1:T}) = \mu(s_1)\prod_{t\geq 1}^{T-1} p_{\text{env}}(s_{t+1}|s_t,a_t)\prod_{t\geq 1}^{T}\pi_\theta(a_t|s_t)$ denotes the probability of a trajectory $\tau_{1:T}$ under policy π_θ.

Traditionally, in reinforcement learning (RL) problems, the goal is to find the optimal policy that maximizes the expected return $\mathbb{E}_{q_\theta}[\sum_{t=1}^{T}\gamma^t r_t]$. However,



it is useful to frame RL as an inference problem within a probabilistic graphical framework (Rawlik et al., 2012; Toussaint & Storkey, 2006; Levine, 2018). First, we introduce an auxiliary binary random variable $\mathcal{O}_t$ denoting the "optimality" of a pair (s_t, a_t) at time t and define its probability[1] as $p(\mathcal{O}_t = 1|s_t, a_t) = \exp(r(s_t, a_t))$. $\mathcal{O}$ is a convenience variable only here for the sake of modeling. By considering the variables (s_t, a_t) as latent and $\mathcal{O}_t$ as observed, we can construct a Hidden Markov Model (HMM) as depicted in figure 1. Notice that the link $s \to a$ is not present in figure 1 as the dependency of the optimal action on the state depends on the future observations. In this graphical model, the optimal policy is expressed as $p(a_t|s_t, \mathcal{O}_{t:T})$.

The posterior probability of this graphical model can be written as[2]:

$$p(\tau_{1:T}|\mathcal{O}_{1:T}) \propto p(\tau_{1:T}, \mathcal{O}_{1:T}) = \mu(s_1) \prod_{t=1}^{T-1} p_{\text{env}}(s_{t+1}|a_t, s_t) \exp\Big(\sum_{t=1}^{T} r(s_t, a_t) + \log p(a_t)\Big). \quad (4.2.1)$$

It appears clearly that finding optimal trajectories is equivalent to finding plausible trajectories yielding a high return.

Many *control as inference* methods can be seen as approximating the density by optimizing its variational lower bound: $\log p(\mathcal{O}_{1:T}) \geq \mathbb{E}_{\tau_{1:T} \sim q_\theta}[\sum_{t=1}^{T} r(s_t, a_t) - \log \pi_\theta(a_t|s_t)]$ (Rawlik et al., 2012; Toussaint, 2009). Instead of directly differentiating the variational lower bound for the whole trajectory, it is possible to take a message passing approach such as the one used in Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and directly estimate the optimal policy $p(a_t|s_t, \mathcal{O}_{t:T})$ using the backward message, i.e a soft Q function instead of the Monte Carlo return.

4.2.2. Sequential Monte Carlo methods

Since distributions over trajectories are complex, it is often difficult or impossible to directly draw samples from them. Fortunately in statistics, there are successful strategies for drawing samples from complex sequential distributions, such as SMC methods.

For simplicity, in the remainder of this section we will overload the notation and refer to the target distribution as $p(\tau)$ and the proposal distribution as $q(\tau)$. We wish to draw samples from p but we only know its unnormalized density. We will use the proposal q to

[1] as in Levine (2018), if the rewards are bounded above, we can always remove a constant so that the probability is well defined.

[2] Notice that in the rest of the paper, we will abusively remove the product of the action priors $\prod_{t=1}^{T} p(a_t) = \exp\big(\sum_{t=1}^{T} \log p(a_t)\big)$ from the joint as in Levine (2018). We typically consider this term either constant or already included in the reward function. See Appendix B.1.2 for details.

draw samples and estimate p. In the next section, we will define the distributions p and q in the context of planning.

Importance sampling (IS):. When τ can be efficiently sampled from another simpler distribution q i.e. the proposal distribution, we can estimate the likelihood of any point τ under p straightforwardly by computing the *unnormalized importance sampling weights* $w(\tau) \propto \frac{p(\tau)}{q(\tau)}$ and using the identity $p(\tau) = \bar{w}(\tau)q(\tau)$ where $\bar{w}(\tau) = \frac{w(\tau)}{\int w(\tau)q(\tau)d\tau}$ is defined as the *normalized importance sampling weights*. In practice, one draws N samples from q: $\{\tau^{(n)}\}_{n=1}^{N} \sim q$; these are referred to as *particles*. The set of particles $\{\tau^{(n)}\}_{n=1}^{N}$ associated with their weights $\{w^{(n)}\}_{n=1}^{N}$ are simulations of samples from p. That is, we approximate the density p with a weighted sum of diracs from samples of q:

$$p(\tau) \approx \sum_{n=1}^{N} \bar{w}^{(n)} \delta_{\tau^{(n)}}(\tau), \text{ with } \tau^{(n)} \text{ sampled from } q$$

where $\delta_{\tau_0}(\tau)$ denotes the Dirac delta mass located as τ_0.

Sequential Importance Sampling (SIS):. When our problem is sequential in nature $\tau = \tau_{1:T}$, sampling $\tau_{1:T}$ at once can be a challenging or even intractable task. By exploiting the sequential structure, the unnormalized weights can be updated iteratively in an efficient manner: $w_t(\tau_{1:t}) = w_{t-1}(\tau_{1:t-1})\frac{p(\tau_t|\tau_{1:t-1})}{q(\tau_t|\tau_{1:t-1})}$. We call this the **update step**. This enables us to sample sequentially $\tau_t \sim q(\tau_t|\tau_{1:t-1})$ to finally obtain the set of particles $\{\tau_{1:T}^{(n)}\}$ and their weights $\{w_T^{(n)}\}$ linearly in the horizon T.

Sequential Importance Resampling (SIR):. When the horizon T is long, samples from q usually have a low likelihood under p, and thus the quality of our approximation decreases exponentially with T. More concretely, the unnormalized weights $w_t^{(n)}$ converge to 0 with $t \to \infty$. This usually causes the normalized weight distribution to degenerate, with one weight having a mass of 1 and the others a mass of 0. This phenomenon is known as *weight impoverishment*.

One way to address weight impoverishment is to add a **resampling step** where each particle is stochastically resampled to higher likelihood regions at each time step. This can typically reduce the variance of the estimation from growing *exponentially* with t to growing *linearly*.

4.3. Sequential Monte Carlo Planning

In the context of *control as inference*, it is natural to see planning as the act of approximating a distribution of optimal future trajectories via simulation. In order to plan, an agent must possess a model of the world that can accurately capture the consequences of its actions. In cases where multiple trajectories have the potential of being optimal,

the agent must rationally partition its computational resources to explore each possibility. Given finite time, the agent must limit its planning to a finite horizon h. We, therefore, define *planning* as the act of approximating the optimal distribution over trajectories of length h. In the control-as-inference framework, this distribution is naturally expressed as $p(a_1, s_2, \dots s_h, a_h | \mathcal{O}_{1:T}, s_1)$, where s_1 represents our current state.

4.3.1. Planning and Bayesian smoothing

As we consider the current state s_1 given, it is equivalent and convenient to focus on the planning distribution with horizon h: $p(\tau_{1:h}|\mathcal{O}_{1:T})$. Bayesian smoothing is an approach to the problem of estimating the distribution of a latent variable conditioned on all past and future observations. One method to perform smoothing is to decompose the posterior with the *two-filter formula* (Bresler, 1986; Kitagawa, 1994):

$$p(\tau_{1:h}|\mathcal{O}_{1:T}) \propto \underline{p(\tau_{1:h}|\mathcal{O}_{1:h})} \cdot \underline{p(\mathcal{O}_{h+1:T}|\tau_h)} \tag{4.3.1}$$

This corresponds to a forward-backward messages factorization in a Hidden Markov Model as depicted in Figure 2. We broadly underline in orange forward variables and in blue backward variables in the rest of this section.

Filtering is the task of estimating $p(\tau_{1:t}|\mathcal{O}_{1:t})$: the probability of a latent variable conditioned on all past observations. In contrast, **smoothing** estimates $p(\tau_{1:t}|\mathcal{O}_{1:T})$: the density of a latent variable conditioned on all the past and future measurements.

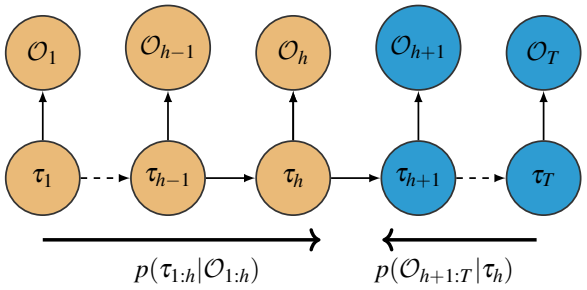


Fig. 2. Factorization of the HMM into forward (orange) and backward (blue) messages. Estimating the forward message is filtering, estimating the value of the latent knowing all the observations is smoothing.

In the belief propagation algorithm for HMMs, these probabilities correspond to the forward message $\alpha_h(\tau_h) = p(\tau_{1:h}|\mathcal{O}_{1:h})$ and backward message $\beta_h(\tau_h) = p(\mathcal{O}_{h+1:T}|\tau_h)$, both of which are computed recursively. While in discrete spaces these forward and backward messages can be estimated using the sum-product algorithm, its complexity scales with the square of the space dimension making it unsuitable for continuous tasks. We will now devise efficient strategies for estimating reliably the full posterior using the SMC methods covered in section 4.2.2.

4.3.2. The Backward Message and the Value Function

The backward message $p(\mathcal{O}_{h+1:T}|\tau_h)$ can be understood as the answer to: *What is the probability of following an optimal trajectory from the next time step on until the end of the episode, given my current state?*. Importantly, this term is closely related to the notion of *value function* in RL. Indeed, in the control-as-inference framework, the state- and action-value functions are defined as $V(s_h) \triangleq \log p(\mathcal{O}_{h:T}|s_h)$ and $Q(s_h,a_h) \triangleq \log p(\mathcal{O}_{h:T}|s_h,a_h)$ respectively. They are solutions of a soft-Bellman equation that differs a little from the traditional Bellman equation (O'Donoghue et al., 2016; Nachum et al., 2017; Schulman et al., 2017a; Abdolmaleki et al., 2018). A more in depth explanation can be found in (Levine, 2018). We can show subsequently that:

$$p(\mathcal{O}_{h+1:T}|\tau_h) = \underline{\mathbb{E}_{s_{h+1}|\tau_h}\left[\exp\left(V(s_{h+1})\right)\right]} \tag{4.3.2}$$

Full details can be found in Appendix B.1.3. Estimating the backward message is then equivalent to learning a value function. This value function as defined here is the same one used in Maximum Entropy RL (Ziebart, 2010).

4.3.3. Sequential Weight Update

Using the results of the previous subsections we can now derive the full update of the sequential importance sampling weights. To be consistent with the terminology of section 4.2.2, we call $p(\tau_{1:h}|\mathcal{O}_{1:T})$ the target distribution and $q_\theta(\tau_{1:h})$ the proposal distribution. The sequential weight update formula is in our case:

$$\begin{aligned}
w_t &= w_{t-1} \cdot \frac{p(\tau_t|\tau_{1:t-1},\mathcal{O}_{1:T})}{q_\theta(\tau_t|\tau_{1:t-1})} \\
&\propto w_{t-1} \frac{1}{q_\theta(\tau_t|\tau_{1:t-1})} \frac{\underline{p(\tau_{1:t}|\mathcal{O}_{1:t})}}{\underline{p(\tau_{1:t-1}|\mathcal{O}_{1:t-1})}} \frac{\underline{p(\mathcal{O}_{t+1:T}|\tau_t)}}{\underline{p(\mathcal{O}_{t:T}|\tau_{t-1})}} \\
&\propto w_{t-1} \cdot \frac{p_{\text{env}}(s_t|s_{t-1},a_{t-1})}{p_{\text{model}}(s_t|s_{t-1},a_{t-1})} \cdot \mathbb{E}_{s_{t+1}|s_t,a_t}\left[\exp\left(A(s_t,a_t,s_{t+1})\right)\right]
\end{aligned}$$

Where

$$A(s_t,a_t,s_{t+1}) = r_t - \log \pi_\theta(a_t|s_t) + V(s_{t+1}) - \log \mathbb{E}_{s_t|s_{t-1},a_{t-1}}\left[\exp\left(V(s_t)\right)\right] \tag{4.3.3}$$

is akin to a maximum entropy advantage function. The change in weight can be interpreted as sequentially correcting our expectation of the return of a trajectory.
The full derivation is available in Appendix B.1.4. Our algorithm is similar to the Auxiliary Particle Filter (Pitt & Shephard, 1999) which uses a one look ahead simulation step to update the weights. Note that in practice we do not have access to the ratio $\frac{p_{\text{env}}(s_t|s_{t-1},a_{t-1})}{p_{\text{model}}(s_t|s_{t-1},a_{t-1})}$, as

it would be equivalent to having access to a perfect model of the world otherwise. Therefore, we will use the simplified weight update:

$$w_t \propto w_{t-1} \cdot \mathbb{E}_{s_{t+1}|s_t,a_t}[\exp\left(A(s_t,a_t,s_{t+1})\right)]$$

by assuming our model of the environment is perfect to obtain this slightly simplified form.

This assumption is implicitly made by most planning algorithms (LQR, CEM ...): it entails that our plan is only as good as our model is. A typical way to mitigate this issue and be more robust to model errors is to re-plan at each time step; this technique is called Model Predictive Control (MPC) and is commonplace in control theory.

4.3.4. Sequential Monte Carlo Planning Algorithm

We can now use the computations of previous subsections to derive the full algorithm. We consider the root state of the planning to be the current state s_t. We aim at building a set of particles $\{\tau_{t:t+h}^{(n)}\}_{n=1}^N$ and their weights $\{w_{t+h}^{(n)}\}_{n=1}^N$ representative of the planning density $p(\tau_{t:t+h}|\mathcal{O}_{1:T})$ over optimal trajectories. We use SAC (Haarnoja et al., 2018) for the policy and value function, but any other Maximum Entropy policy can be used for the proposal distribution. Note that we used the value function estimated by SAC as a proxy the optimal one as it is usually done by actor critic methods.

Algorithm 5 SMC Planning using SIR

1: **for** t in $\{1,\dots,T\}$ **do**
2: $\quad\{s_t^{(n)} = s_t\}_{n=1}^N$
3: $\quad\{w_t^{(n)} = 1\}_{n=1}^N$
4: $\quad$**for** i in $\{t,\dots,t+h\}$ **do**
5: $\quad\quad$*// Update*
6: $\quad\quad\{a_i^{(n)} \sim \pi(a_i^{(n)}|s_i^{(n)})\}_{n=1}^N$
7: $\quad\quad\{s_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot|s_i^{(n)}, a_i^{(n)})\}_{n=1}^N$
8: $\quad\quad\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(s_i^{(n)}, a_i^{(n)}, s_{i+1}^{(n)})\right)\}_{n=1}^N$
9: $\quad\quad$*// Resampling*
10: $\quad\quad\{\tau_{1:i}^{(n)}\}_{n=1}^N \sim \text{Mult}(n; w_i^{(1)}, \dots, w_i^{(N)})$
11: $\quad\quad\{w_i^{(n)} = 1\}_{n=1}^N$
12: $\quad$**end for**
13: $\quad$Sample $n \sim \text{Uniform}(1, N)$.
14: $\quad$*// Model Predictive Control*
15: $\quad$Select a_t, first action of $\tau_{t:t+h}^{(n)}$
16: $\quad s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t, a_t)$
17: $\quad$Add (s_t, a_t, r_t, s_{t+1}) to buffer $\mathcal{B}$
18: $\quad$Update π, V and p_{model} with $\mathcal{B}$
19: **end for**

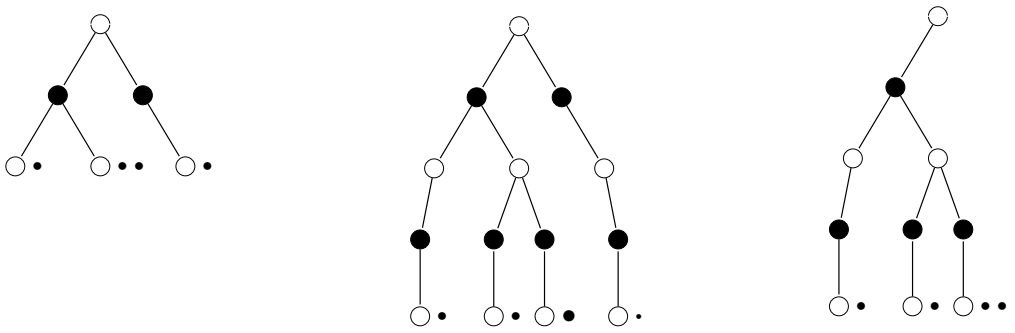
We summarize the proposed algorithm in Algorithm 5. At each step, we sample from the proposal distribution or model-free agent (**line 6**) and use our learned model to sample the next state and reward (**line 7**). We then update the weights (**line 8**). In practice we only use one sample to estimate the expectations, thus we may incur a small bias. The resampling step is then performed (**line 10-11**) by resampling the trajectories according to their weight. After the planning horizon is reached, we sample one of our trajectories (**line 13**) and execute its first action into the environment (**line 15-16**). The observations (s_t, a_t, r_t, s_{t+1}) are then collected and added to a buffer (**line 17**) used to train the model as well as the policy and value function of the model-free agent. An alternative algorithm that does not use the resampling step (SIS) is highlighted in Algorithm 6 in Appendix B.1.6.

A schematic view of the algorithm can also be found on figure 3.

4.3.5. Optimism Bias and Control as Inference

We now discuss shortcomings our approach to planning as inference may suffer from, namely encouraging risk seeking policies.

Bias in the objective: Trajectories having a high likelihood under the posterior defined in Equation 4.2.1 are not necessarily trajectories yielding a high *mean* return. Indeed,



(a) Configuration at time $t-1$**:** we have the root white node s_{t-1}, the actions $a_{t-1}^{(n)}$ are black nodes and the leaf nodes are the $s_t^{(n)}$. We have one particle on the leftmost branch, two on the central branch and one on the rightmost branch.

(b) Update: New actions and states are sampled from the proposal distribution and model. The particle sizes are proportional to their importance weight w_t.

(c) Resampling: after sampling with replacement the particles relatively to their weight, the less promising branch was cut while the most promising has now two particles.

Fig. 3. Schematic view of Sequential Monte Carlo planning. In each tree, the white nodes represent states and black nodes represent actions. Each bullet point near a state represents a particle, meaning that this particle contains the total trajectory of the branch. The root of the tree represents the root planning state, we expand the tree downward when planning.

as $\log \mathbb{E}_p\big[\exp R(\tau)\big] \geq \mathbb{E}_p\big[R(\tau)\big]$ we can see that the objective function we maximize is an *upper bound* on the quantity of interest: the mean return. This can lead to risk-seeking trajectories as one very good outcome in $\log \mathbb{E} \exp$ could dominate all the other potentially very low outcomes, even if they might happen more frequently. This fact is alleviated when the dynamics of the environment are close to deterministic (Levine, 2018). Thus, this bias does not appear to be very detrimental to us in our experiments 4.4.2 as our environments are fairly close to deterministic. The bias in the objective also appears in many control as inference works such as Particle Value Functions (Maddison et al., 2017) and the probabilistic version of LQR proposed in Toussaint (2009).

Bias in the model: A distinct but closely related problem arises when one trains jointly the policy π_θ and the model p_{model}, i.e if $q(\tau_{1:T})$ is directly trained to approximate $p(\tau_{1:T}|\mathcal{O}_{1:T})$. In that case, $p_{\text{model}}(s_{t+1}|s_t,a_t)$ will not approximate $p_{\text{env}}(s_{t+1}|s_t,a_t)$ but $p_{\text{env}}(s_{t+1}|s_t,a_t,\mathcal{O}_{t:T})$ (Levine, 2018). This means the model we learn has an optimism bias and learns transitions that are overly optimistic and do no match the environment's behavior. This issue is simply solved by training the model separately from the policy, on transition data contained in a buffer as seen on line 18 of Algorithm 5.

4.4. Experiments

4.4.1. Toy example

In this section, we show how SMCP can deal with multimodal policies when planning. We believe multimodality is useful for exploring since it allows us to keep a distribution over many promising trajectories and also allows us to adapt to changes in the environment e.g. if a path is suddenly blocked.

We applied two version of SMCP: i) with a resampling step (SIR) ii) without a resampling step (SIS) and compare it to CEM on a simple 2D point mass environment 4. Here, the agent can control the displacement on (x,y) within the square $[0,1]^2$, $a = (\Delta x, \Delta y)$ with maximum magnitude $||a|| = 0.05$. The starting position ($\bullet$) of the agent is $(x = 0, y = 0.5)$, while the goal ($\star$) is at $g = (x = 1, y = 0.5)$. The reward is the agent's relative closeness increment to the goal: $r_t = 1 - \frac{||s_{t+1}-g||^2}{||s_t-g||^2}$. However, there is a partial wall at the centre of the square leading to two optimal trajectories, one choosing the path below the wall and one choosing the path above.

The proposal is an isotropic normal distribution for each planning algorithm, and since the environment's dynamics are known, there is no need for learning: the only difference between the three methods is how they handle planning. We also set the value function to 0 for SIR and SIS as we do not wish to perform any learning. We used 1500 particles for each method, and updated the parameters of CEM until convergence. Our experiment 4 shows how having particles can deal with multimodality and how the resampling step can help to focus on the most promising trajectories.

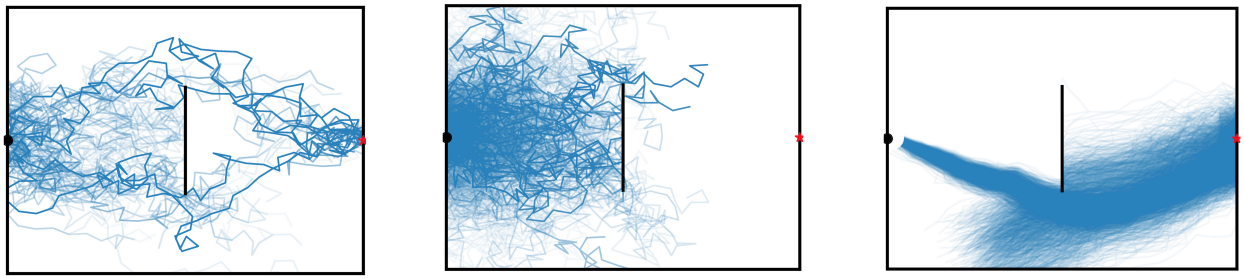
4.4.2. Continuous Control Benchmark

The experiments were conducted on the Open AI Gym Mujoco benchmark suite (Brockman et al., 2016; Todorov et al., 2012). To understand how planning can increase the learning speed of RL agents we focus on the 250000 first time steps. The Mujoco environments provide a complex benchmark with continuous states and actions that requires exploration in order to achieve state-of-the-art performances.

The environment model used for our planning algorithm is the same as the probabilistic neural network used by Chua et al. (2018), it minimizes a gaussian negative log-likelihood model:

$$\mathcal{L}_{\text{Gauss}}(\theta) = \frac{1}{2}\sum_{n=1}^{N}[\mu_\theta(s_n,a_n)-(s_{n+1}-s_n)]^\top\Sigma_\theta^{-1}(s_n,a_n)[\mu_\theta(s_n,a_n)-(s_{n+1}-s_n)]+\log\det\Sigma_\theta(s_n,a_n),$$

where Σ_θ is diagonal and the transitions (s_n,a_n,s_{n+1}) are obtained from the environment.



(a) Sequential Importance Resampling (SIR): when resampling the trajectories at each time step, the agent is able to focus on the promising trajectories and does not collapse on a single mode.

(b) Sequential Importance Sampling (SIS): if we do not perform the resampling step the agent spends most of its computation on uninteresting trajectories and was not able to explore as well.

(c) CEM: here the agent samples all the actions at once from a Gaussian with learned mean and covariance. We needed to update the parameters 50 times for the agent to find one solution, but it forgot the other one.

Fig. 4. Comparison of three methods on the toy environment. The agent (•) must go to the goal ($\star$) while avoiding the wall (|) in the center. The proposal distribution is taken to be an isotropic gaussian. Here we plot the planning distribution imagined at $t = 0$ for three different agents. A darker shade of blue indicates a higher likelihood of the trajectory. Only the agent using Sequential Importance Resampling was able to find good trajectories while not collapsing on a single mode.

We included two popular planning algorithms on Mujoco as baselines: CEM (Chua et al., 2018) and Random Shooting (RS) (Nagabandi et al., 2017). Furthermore, we included SAC (Haarnoja et al., 2018), a model free RL algorithm, since i) it has currently one of the highest performances on Mujoco tasks, which make it a very strong baseline, and ii) it is a component of our algorithm, as we use it as a proposal distribution in the planning phase.

Our results suggest that SMCP does not learn as fast as CEM and RS initially as it heavily relies on estimating a good value function. However, SMCP quickly achieves higher performances than CEM and RS. SMCP also learns faster than SAC because it was able to leverage information from the model early in training. We hypothesize that the lack of performance gain of SMCP over SAC in the Hopper environment is due to the low quality of its model and the complexity of the task.

Note that our results differ slightly from the results usually found in the model-based RL literature. This is because we are tackling a more difficult problem: estimating the transitions and the reward function. We are using unmodified versions of the environments which introduces many hurdles. For instance, the reward function is challenging to learn from the state and very noisy.

As in Henderson et al. (2017), we assess the significance of our results by running each algorithm with multiple seeds (10 random seeds in our case, from seed 0 to seed 9).

4.5. Conclusion and Future Work

In this work, we have introduced a connection between planning and inference and showed how we can exploit advances in deep learning and probabilistic inference to design a new efficient and theoretically grounded planning algorithm. We additionally proposed a natural way to combine model-free and model-based reinforcement learning for planning based on the SMC perspective. We empirically demonstrated that our method achieves state of the art results on Mujoco. Our result suggest that planning can lead to faster learning in control tasks.

However, our particle-based inference method suffers some several shortcomings. First, we need many particles to build a good approximation of the posterior, and this can be computationally expensive since it requires to perform a forward pass of the policy, the value function and the model for every particle. Second, resampling can also have adverse effects, for instance all the particles could be resampled on the most likely particle, leading to a particle degeneracy. More advanced SMC methods dealing with this issue such as backward simulation (Lindsten et al., 2013) or Particle Gibbs with Ancestor Sampling (PGAS) (Lindsten et al., 2014) have been proposed and using them would certainly improve our results.

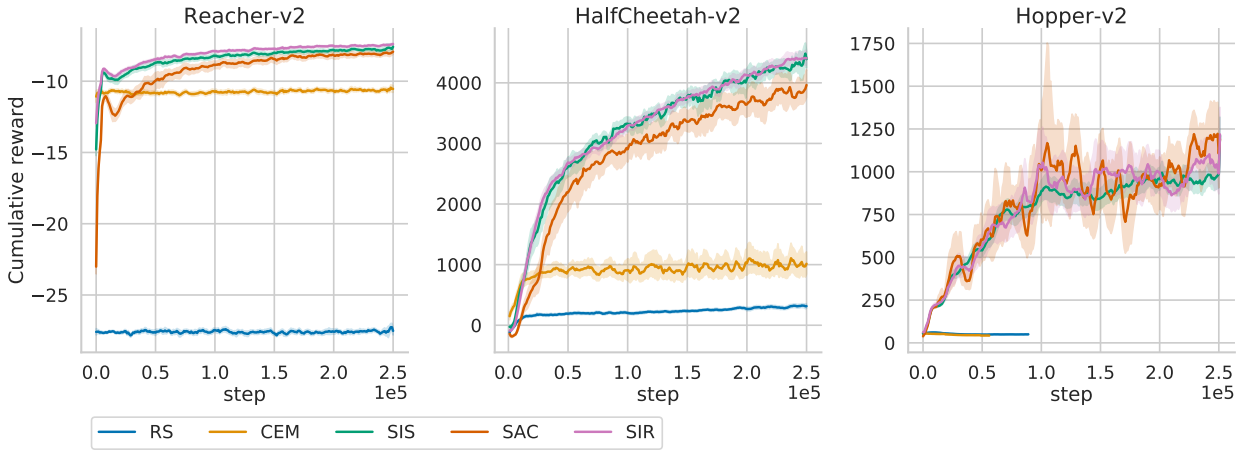


Fig. 5. Training curves on the Mujoco continuous control benchmarks. Sequential Monte Carlo Planning both with resampling (SIR) (pink) and without (SIS) (orange) learns faster than the Soft Actor-Critic model-free baseline (blue) and achieves higher asymptotic performances than the planning methods (Cross Entropy Methods and Random Shooting). The shaded area represents the standard deviation estimated by bootstrap over 10 seeds as implemented by the Seaborn package.

Another issue we did not tackle in our work is the use of models of the environment learned from data. Imperfect model are known to result in compounding errors for prediction over long sequences. We chose to re-plan at each time step (Model Predictive Control) as it is often done in control to be more robust to model errors. More powerful models or uncertainty modeling techniques can also be used to improve the accuracy of our planning algorithm. While the inference and modeling techniques used here could be improved in multiple ways, SMCP achieved impressive learning speed on complex control tasks. The planning as inference framework proposed in this work is general and could serve as a stepping stone for further work combining probabilistic inference and deep reinforcement learning.

Chapitre 5

On the Interplay between Noise and Curvature and its Effect on Optimization and Generalization

Article details

Thomas V, Pedregosa F, Merriënboer B, Manzagol PA, Bengio Y, Le Roux N. "On the interplay between noise and curvature and its effect on optimization and generalization". In *International Conference on Artificial Intelligence and Statistics (AISTATS) 2020. PMLR.*

Foreword

This project started while I was thinking about the role of the stochasticity in SGD and its influence on generalization. At the same time, Nicolas Le Roux gave a talk at Mila where he mentioned a link between generalization, curvature and gradient noise in supervised learning. This started a collaboration which ultimately led to this paper published at AISTATS 2020.

Impact since publication

This work has inspired others such as Naganuma et al. (2022) which validated our results by performing experiments at a larger scale than we did, Rame et al. (2022) used our empirical results on the similarity between the empirical and the true Fisher matrix and Schneider et al. (2021) implemented the Takeuchi Information Criterion and approximations we developed for it as part of their package.

Personal contribution

- Making the link between our original idea and the already existing paper Takeuchi (1976) which introduced the estimator first, was only published in Japanese and even reinvented later (Murata et al., 1994).

- Performing all the large-scale experiments: training hundreds of neural networks with different parameters and architectures and datasets and recording useful statistics
- Theory and experiments for the link between the Hessian **H**, the Fisher matrix **F** and the uncentered gradient covariance matrix **C**
- Optimization theory for SGD and the interplay between **H** and **C** in the quadratic case
- Writing of the paper with Nicolas and created the figures

Abstract

The speed at which one can minimize an expected loss using stochastic methods depends on two properties: the curvature of the loss and the variance of the gradients. While most previous works focus on one or the other of these properties, we explore how their interaction affects optimization speed. Further, as the ultimate goal is good generalization performance, we clarify how both curvature and noise are relevant to properly estimate the generalization gap. Realizing that the limitations of some existing works stems from a confusion between these matrices, we also clarify the distinction between the Fisher matrix, the Hessian, and the covariance matrix of the gradients.

5.1. Introduction

Training a machine learning model is often cast as the minimization of a smooth function f over parameters θ in $\mathbb{R}^d$. More precisely, we aim at finding a minimum of an expected loss, i.e.

$$\theta^* \in \arg\min_\theta \mathbb{E}_p[\mathcal{L}(\theta, x)] \,, \tag{5.1.1}$$

where the expectation is under the data distribution $x \sim p$. In practice, we only have access to an empirical distribution $\hat{p}$ over x and minimize the training loss

$$\hat{\theta}^* \in \arg\min_\theta \mathbb{E}_{\hat{p}}[\mathcal{L}(\theta, x)] \tag{5.1.2}$$

$$= \arg\min_\theta f(\theta) \,. \tag{5.1.3}$$

To minimize this function, we assume access to an oracle which, for every value of θ and x, returns both $\mathcal{L}(\theta, x)$ and its derivative with respect to θ, i.e., $\nabla_\theta \mathcal{L}(\theta, x)$. Given this oracle, stochastic gradient iteratively performs the following update: $\theta_{t+1} = \theta_t - \alpha_t \nabla \mathcal{L}(\theta_t, x)$ [1] where $\{\alpha_t\}_{t \geq 0}$ is a sequence of stepsizes.

Two questions arise: First, how quickly do we converge to $\hat{\theta}^*$ and how is this speed affected by properties of $\mathcal{L}$ and $\hat{p}$? Second, what is $\mathbb{E}_p[\mathcal{L}(\hat{\theta}^*, x)]$?

It is known that the former is influenced by two quantities: the curvature of the function, either measured through its smoothness constant or its condition number, and the noise on the gradients, usually measured through a bound on $\mathbb{E}_{\hat{p}}[\|\nabla \mathcal{L}(\theta, x)\|^2]$. For instance, when f is μ-strongly convex, L-smooth, and the noise is bounded, i.e. $\mathbb{E}_{\hat{p}}[\|\nabla \mathcal{L}(\theta, x)\|^2] \leq c$, then stochastic gradient with a constant stepsize α will converge linearly to a ball (Schmidt, 2014). Calling Δ the suboptimality, i.e. $\Delta_k = f(\theta_k) - f(\hat{\theta}^*)$, we have

$$\mathbb{E}[\Delta_k] \leq (1 - 2\alpha\mu)^k \Delta_0 + \frac{L\alpha c}{4\mu} . \tag{5.1.4}$$

This implies that as $k \to \infty$, the expected suboptimality depends on both the curvature through μ and L, and on the noise through c. However, bounding curvature and noise using constants rather than full matrices hides the dependencies between these two quantities. We also observed that, because existing works replace the full noise matrix with a constant when deriving convergence rates, that matrix is poorly understood and is often confused with a curvature matrix. This confusion remains when discussing the generalization properties of a model. Indeed, the generalization gap stems from a discrepancy between the empirical and the true data distribution. An estimator of this gap must thus include an estimate of that discrepancy in addition to an estimate of the impact of an infinitesimal discrepancy on the loss. The former can be characterized as noise and the latter as curvature. Hence, attempts at estimating the generalization gap using only the curvature (Keskar et al., 2017; Novak et al., 2018) are bound to fail as do not characterize the size or geometry of the discrepancy.

In this work, we make the following contributions:

- We provide theoretical and empirical evidence of the similarities and differences surrounding the curvatures matrices; the Fisher $\mathbf{F}$ and the Hessian $\mathbf{H}$, and the noise matrix, $\mathbf{C}$;
- We briefly expand the convergence results of Schmidt (2014), theoretically and empirically highlighting the importance of the relationship between noise and curvature for strongly convex functions and quadratics;
- We make the connection with an old estimator of the generalization gap, the Takeuchi Information Criterion, and show how its use of both curvature and noise

[1] We omit the subscripts when clear from context.

yields a superior estimator to other commonly used ones, such as flatness or sensitivity for neural networks.

5.2. Information matrices: definitions, similarities, and differences

Before delving into the impact of the information matrices for optimization and generalization, we start by recalling their definitions. We shall see that, despite having similar formulations, they encode different information. We then provide insights on their similarities and differences.

We discuss here two information matrices associated with curvature, the Fisher matrix $\mathbf{F}$ and the Hessian $\mathbf{H}$, and one associated with noise, the gradients' uncentered covariance $\mathbf{C}$. In particular, while $\mathbf{F}$ and $\mathbf{H}$ are well understood, $\mathbf{C}$ is often misinterpreted. For instance, it is often called "empirical Fisher" (Martens, 2014) despite bearing no relationship to $\mathbf{F}$, the true Fisher. This confusion can have dire consequences and optimizers using $\mathbf{C}$ as approximation to $\mathbf{F}$ can have arbitrarily poor performance (Kunstner et al., 2019).

To present these matrices, we consider the case of maximum likelihood estimation (MLE). We have access to a set of samples $(x,y) \in \mathcal{X} \times \mathcal{Y}$ where x is the input and y the target. We define $p : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ as the **data distribution** and $q_\theta : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ such that $q_\theta(x,y) = p(x)q_\theta(y|x)$ as the **model distribution**[2]. For each sample $(x,y) \sim p$, our loss is the negative log-likelihood $\mathcal{L}(\theta,y,x) = -\log q_\theta(y|x)$. Note that all the definitions and results in this section are valid whether we use the true data distribution p or the empirical $\hat{p}$.

Matrices $\mathbf{H}$, $\mathbf{F}$ and $\mathbf{C}$ are then defined as:

$$\mathbf{H}(\theta) = \mathbb{E}_p\left[\frac{\partial^2}{\partial\theta\partial\theta^\top}\mathcal{L}(\theta,y,x)\right] \tag{5.2.1}$$

$$\mathbf{C}(\theta) = \mathbb{E}_p\left[\frac{\partial}{\partial\theta}\mathcal{L}(\theta,y,x)\frac{\partial}{\partial\theta}\mathcal{L}(\theta,y,x)^\top\right] \tag{5.2.2}$$

$$\mathbf{F}(\theta) = \mathbb{E}_{q_\theta}\left[\frac{\partial}{\partial\theta}\mathcal{L}(\theta,y,x)\frac{\partial}{\partial\theta}\mathcal{L}(\theta,y,x)^\top\right] \tag{5.2.3}$$

$$= \mathbb{E}_{q_\theta}\left[\frac{\partial^2}{\partial\theta\partial\theta^\top}\mathcal{L}(\theta,y,x)\right]. \tag{5.2.4}$$

We observe the following: a) The definition of $\mathbf{H}$ and $\mathbf{C}$ involves the data distribution, in contrast with the definition of $\mathbf{F}$, which involves the model distribution; b) If $q_\theta = p$, all matrices are equal. Furthermore, as noted by Martens (2014), $\mathbf{H} = \mathbf{F}$ whenever the

[2] $q_\theta(y|x)$ are the softmax activations of a neural network in the classification setting.

matrix of second derivatives does not depend on y, a property shared in particular by all generalized linear models.

As said above, **H**, **F**, and **C** characterize different properties of the optimization problem. **H** and **F** are curvature matrices and describe the geometry of the space around the current point. **C**, on the other hand, is a "noise matrix" and represents the sensitivity of the gradient to the particular sample.[3]

We now explore in more details their similarities and differences.

5.2.1. Bounds between H, F and C

The following proposition bounds the distance between the information matrices:

Proposition 5.2.1 (Distance between **H**, **F** and **C**)**.** *Assuming the second moments of the Fisher are bounded above, i.e.* $\mathbb{E}_{q_\theta}[||\nabla^2_\theta \mathcal{L}(\theta,x,y)||^2] \leq \beta_1$ *and* $\mathbb{E}_{q_\theta}[||\nabla_\theta \mathcal{L}(\theta,x,y)\nabla_\theta \mathcal{L}(\theta,x,y)^\top||^2] \leq \beta_2$*, we have*

$$\begin{aligned} ||\mathbf{F}-\mathbf{H}||^2 &\leq \beta_1 \, \mathcal{D}_{\chi^2}(p||q_\theta) \,, \\ ||\mathbf{F}-\mathbf{C}||^2 &\leq \beta_2 \, \mathcal{D}_{\chi^2}(p||q_\theta) \,, \\ ||\mathbf{C}-\mathbf{H}||^2 &\leq \left(\beta_1+\beta_2\right) \mathcal{D}_{\chi^2}(p||q_\theta) \,. \end{aligned}$$

where $\mathcal{D}_{\chi^2}(p||q_\theta) = \iint \frac{(p(x,y)-q_\theta(x,y))^2}{q_\theta(x,y)} dydx$ *is the* χ^2 *divergence and* $||\cdot||$ *is the Frobenius norm.*

All the proofs are in the appendix.

In particular, when $p = q_\theta$ we recover that $\mathbf{F} = \mathbf{H} = \mathbf{C}$. At first glance, one could assume that, as the model is trained and q_θ approaches p, these matrices become more similar. The χ^2 divergence is however poorly correlated with the loss of the model. One sample where the prediction of the model is much smaller than the true distribution can dramatically impact the χ^2 divergence and thus the distance between the information matrices. We show in Section 5.5.3 how these distances evolve when training a deep network.

5.2.2. C does not approximate F

C is often referred to as the "empirical Fisher" matrix, implying that it is an approximation to the true Fisher matrix **F** (Martens, 2014). In some recent works (Liang et al., 2017; George et al., 2018) the empirical Fisher matrix is used instead of the Fisher matrix. However, in the general case, there is no guarantee that **C** will approximate **F**, even in the limit of infinite samples. We now give a simple example highlighting their different roles:

[3] Technically, it is **S**, the centered covariance matrix, rather than **C** which plays that role but the two are similar close to a stationary point.

Example 2 (Mean regression)**.** *Let $X = (x_i)_{i=1,\dots,N}$ be an i.i.d sequence of random variables. The task is to estimate $\mu = \mathbb{E}[x]$ by minimizing the loss $\mathcal{L}(\theta) = \frac{1}{2N}\sum_{n=1}^{N} ||x_n - \theta||^2$. The minimum is attained at $\theta^{MLE} = \frac{1}{N}\sum_{n=1}^{N} x_n$. This estimator is consistent and converges to μ at rate $\mathcal{O}(\frac{1}{\sqrt{N}})$.*

This problem is an MLE problem if we define $q_\theta(x) = \mathcal{N}(x;\theta,\mathbf{I}_d)$. In this case, we have

$$\mathbf{H}(\theta^{MLE}) = \mathbf{F}(\theta^{MLE}) = \mathbf{I}_d \quad , \quad \mathbf{C}(\theta^{MLE}) = \widehat{\Sigma}_x \,, \tag{5.2.5}$$

where $\widehat{\Sigma}_x$ is the empirical covariance of the x_i's. We see that, even in the limit of infinite data, the covariance $\mathbf{C}$ does not converge to the actual Fisher matrix nor the Hessian. Hence we shall and will not refer to $\mathbf{C}$ as the "empirical Fisher" matrix.

In some other settings, however, one can expect a stronger correlation between $\mathbf{C}$ and $\mathbf{H}$:

Example 3. *(Ordinary Least Squares) Let us assume we have a data distribution (x_n, y_n) so that*

$$y_n = x_n^\top \theta^* + \varepsilon_n, \quad \varepsilon_n \sim \mathcal{N}(0,\Sigma) \tag{5.2.6}$$

with $y_n \in \mathbb{R}^p$ and $x_n, \varepsilon_n \in \mathbb{R}^d$, $\theta \in \mathbb{R}^{p\times d}$. We train a model to minimize the sum of squares residual

$$\min_\theta \frac{1}{2N}\sum_{n=1}^{N} ||y_n - x_n^\top \theta||^2 \,. \tag{5.2.7}$$

In that case, for $\theta = \theta^*$

$$\mathbf{H} = \mathbf{F} = \mathbb{E}_p[xx^\top] \quad , \quad \mathbf{C} = \mathbb{E}_p[x\Sigma x^\top] \tag{5.2.8}$$

First, we observe that the Hessian and the Fisher are equal, for all θ. Second, when the input/output covariance matrix is isotropic $\Sigma = \sigma^2\mathbf{I}$, then we have $\mathbf{C} \propto \mathbf{H} = \mathbf{F}$.

5.3. Information matrices in optimization

Now that the distinction between these matrices has been clarified, we can explain how their interplay affects optimization. In this section, we offer theoretical results, as well as a small empirical study, on the impact of the noise geometry on convergence rates.

5.3.1. Convergence rates

We start here by expanding the the result of Schmidt (2014) to full matrices, explicitiing how the interplay of noise and curvature affects optimization. We then build a toy experiment to validate the results.

5.3.1.1. General setting

Proposition 5.3.1 (Function value)**.** *Let f be a twice-differentiable function. Assume f is μ-strongly convex and there are two matrices* $\mathbf{H}$ *and* $\mathbf{S}$ *such that for all θ, θ':*

$$f(\theta') \leq f(\theta) + \nabla f(\theta)^\top(\theta' - \theta) + \frac{1}{2}(\theta' - \theta)^\top \mathbf{H}(\theta' - \theta)$$
$$\mathbb{E}_p[\nabla \mathcal{L}(\theta, x)\nabla \mathcal{L}(\theta, x)^\top] \preccurlyeq \mathbf{S} + \nabla f(\theta)\nabla f(\theta)^\top .$$

Then stochastic gradient with stepsize α and positive definite preconditioning matrix $\mathbf{M}$ *satisfies*

$$\mathbb{E}[\Delta_k] \leq (1 - 2\alpha\mu_M\mu)^k \Delta_0 + \frac{\alpha}{4\mu_M\mu} \operatorname{Tr}(\mathbf{HMSM}) ,$$

where μ_M is the smallest eigenvalue of $\mathbf{M} - \frac{\alpha}{2}\mathbf{M}^\top\mathbf{HM}$.

$\mathbf{S}$ is the centered covariance matrix of the stochastic gradients and, if f is quadratic, then $\mathbf{H}$ is the Hessian.

5.3.1.2. Centered and uncentered covariance

Proposition 5.3.1, as well as most results on optimization, uses a bound on the uncentered covariance of the gradients. The result is that the noise must be lower far away from the optimum, where the gradients are high. Thus, it seems more natural to define convergence rates as a function of the *centered* gradients' covariance $\mathbf{S}$, although these results are usually weaker as a consequence of the relaxed assumption. For the remainder of this section, focused on the quadratic case, we will use $\mathbf{S}$. Note that the two matrices are equal at any first-order stationary point.

Centered covariance matrices have been used in the past to derive convergence rates, for instance by Bach & Moulines (2013); Flammarion & Bach (2015); Dieuleveut et al. (2016). These works also include a dependence on the geometry of the noise since their constraint is of the form $\mathbf{S} \preccurlyeq \sigma^2\mathbf{H}$. In particular, if $\mathbf{S}$ and $\mathbf{H}$ are not aligned, σ^2 must be larger for the inequality to hold.

5.3.1.3. Quadratic functions

Proposition 5.3.2 (Quadratic case)**.** *Assuming we minimize a quadratic function*

$$f(\theta) = \tfrac{1}{2}(\theta - \hat{\theta}^*)^\top \mathbf{H}(\theta - \hat{\theta}^*)$$

only having access to a noisy estimate of the gradient $g(\theta) \sim \nabla f(\theta) + \varepsilon$ with ε a zero-mean random variable with covariance $\mathbb{E}[\varepsilon\varepsilon^\top] = \mathbf{S}$, then the iterates obtained using stochastic gradient with stepsize α and preconditioning psd matrix $\mathbf{M}$ *satisfy*

$$\mathbb{E}[\theta^k - \hat{\theta}^*] = (1 - \alpha\mathbf{MH})^k(\theta^0 - \hat{\theta}^*) .$$

Further, the covariance of the iterates $\Sigma_k = \mathbb{E}[(\theta^k - \hat{\theta}^*)(\theta^k - \hat{\theta}^*)^\top]$ *satisfies*

$$\Sigma_{k+1} = (\mathbf{I} - \alpha\mathbf{MH})\Sigma_k(\mathbf{I} - \alpha\mathbf{MH})^\top + \alpha^2\mathbf{MSM}^\top \ .$$

In particular, the stationary distribution of the iterates has a covariance Σ_∞ *which verifies the equation*

$$\Sigma_\infty\mathbf{HM} + \mathbf{MH}\Sigma_\infty = \alpha\mathbf{M}\big(\mathbf{S} + \mathbf{H}\Sigma_\infty\mathbf{H}\big)\mathbf{M} \ .$$

Recently, analyzing the stationary distribution of SGD by modelling its dynamics as a stochastic differential equation (SDE) has gained traction in the machine learning community (Chaudhari & Soatto, 2017; Jastrzębski et al., 2017). Worthy of note, Mandt et al. (2017); Zhu et al. (2018) do not make assumptions about the structure of the noise matrix $\mathbf{S}$. Our proposition above extends and corrects some of their results as it does not rely on the continuous-time approximation of the dynamics of SGD. Indeed, as pointed out in Yaida (2018), most of the works using the continuous-time approximation implicitly make the confusion between centered $\mathbf{S}$ and uncentered $\mathbf{C}$ covariance matrices of the gradients.

Proposition 5.3.3 (Limit cycle of SG)**.** *If* f *is a quadratic function and* $\mathbf{H}$, $\mathbf{C}$ *and* $\mathbf{M}$ *are simultaneously diagonalizable, then stochastic gradient with symmetric positive definite preconditioner* $\mathbf{M}$ *and stepsize* α *yields*

$$\mathbb{E}[\Delta_t] = \tfrac{\alpha}{2}\operatorname{Tr}((2\mathbf{I} - \alpha\mathbf{MH})^{-1}\mathbf{MS}) + \mathcal{O}(e^{-t}) \ . \tag{5.3.1}$$

Rather than using a preconditioner, another popular method to reduce the impact of the curvature is Polyak momentum, defined as

$$\begin{aligned} v_0 = 0 \quad , \quad v_t &= \gamma v_{t-1} + \nabla_\theta \mathcal{L}(\theta_t, x_t) \\ \theta_{t+1} &= \theta_t - \alpha v_t \ . \end{aligned}$$

Proposition 5.3.4 (Limit cycle of momentum)**.** *If* f *is a quadratic function and* $\mathbf{H}$ *and* $\mathbf{C}$ *are simultaneously diagonalizable, then Polyak momentum with parameter* γ *and stepsize* α *yields*

$$\mathbb{E}[\Delta_t] = \tfrac{\alpha}{2}\tfrac{(1+\gamma)}{(1-\gamma)}\operatorname{Tr}\big((2(1+\gamma)\mathbf{I} - \alpha\mathbf{H})^{-1}\mathbf{S}\big) + \mathcal{O}(e^{-t}) \ . \tag{5.3.2}$$

5.4. Generalization

So far, we focused on the impact of the interplay between curvature and noise in the optimization setting. However, optimization, i.e. reaching low loss on the training set, is generally not the ultimate goal as one would rather reach a low test loss. The difference between training and test loss is called the generalization gap and estimating it has been

the focus of many authors (Keskar et al., 2017; Neyshabur et al., 2017; Liang et al., 2017; Novak et al., 2018; Rangamani et al., 2019).

We believe there is a fundamental misunderstanding in several of these works, stemming from the confusion between curvature and noise. Rather than proposing a new metric, we empirically show how the Takeuchi information criterion (TIC: Takeuchi, 1976) addresses these misunderstandings. It makes use of both the Hessian of the loss with respect to the parameters, **H**, and the uncentered covariance of the gradients, **C**. While the former represents the curvature of the loss, i.e., the sensitivity of the gradient to a change in parameter space, the latter represents the sensitivity of the gradient to a change in inputs. As the generalization gap is a direct consequence of the discrepancy between training and test sets, the influence of **C** is natural. Thus, our result further reinforces the idea that the Hessian cannot by itself be used to estimate the generalization gap, an observation already made by Dinh et al. (2017), among others.

5.4.1. Takeuchi information criterion

In the simplest case of a well specified least squares regression problem, an unbiased estimator of the generalization gap is the AIC (Akaike, 1974), which is simply the number of degrees of freedom divided by the number of samples: $\hat{\mathcal{G}}(\theta) = \frac{1}{N}d$ where d is the dimensionality of θ. This estimator is valid locally around the maximum likelihood parameters computed on the training data. However, these assumptions do not hold in most cases, leading to the number of parameters being a poor predictor of the generalization gap (Novak et al., 2018). When dealing with maximum likelihood estimation (MLE) in misspecified models, a more general formula for estimating the gap is given by the Takeuchi information criterion (TIC: Takeuchi, 1976):

$$\hat{\mathcal{G}} = \frac{1}{N}\operatorname{Tr}(\mathbf{H}(\hat{\theta}^*)^{-1}\mathbf{C}(\hat{\theta}^*)) \,, \tag{5.4.1}$$

where $\hat{\theta}^*$ is a local optimum. Note that **H** and **C** here are the hessian and covariance of the gradients matrices computed on the *true data distribution*.

This criterion is not new in the domain of machine learning. It was rediscovered by Murata et al. (1994) and similar criteria have been proposed since then (Beirami et al., 2017; Wang et al., 2018). However, as far as we know, no experimental validation of this criterion has been carried out for deep networks. Indeed, for deep networks, **H** is highly degenerate, most of its eigenvalues being close to 0 (Sagun et al., 2016). In this work, the Takeuchi information criterion is computed, in the degenerate case, by only taking into account the eigenvalues of the Hessian of significant magnitude. In practice, we cut all the eigenvalues smaller than a constant times the biggest eigenvalue and perform the inversion on that subspace. Details can be found in appendix C.2.1.

Interestingly, the term $\mathrm{Tr}(\mathbf{H}^{-1}\mathbf{C})$ appeared in several works before, whether as an upper bound on the suboptimality (Flammarion & Bach, 2015) or as a number of iterates required to reach a certain suboptimality (Bottou & Bousquet, 2008). Sadly, it is hard to estimate for large networks but we propose an efficient approximation in Section 5.5.5.1.

5.4.2. Limitations of flatness and sensitivity

We highlight here two commonly used estimators of the generalization gap as they provide good examples of failure modes that can occur when not taking the noise into account. This is not to mean that these estimators cannot be useful for the models that are common nowadays, rather that they are bound to fail in some cases.

Flatness (Hochreiter & Schmidhuber, 1997) links the spectrum of the Hessian at a local optimum with the generalization gap. This correlation, observed again by Keskar et al. (2017), was already shown to not hold in general (Dinh et al., 2017). As we showed in Section 5.2.2, the Hessian does not capture the covariance of the data, which is linked to the generalization gap through the central-limit theorem.

Sensitivity (Novak et al., 2018) links the generalization gap to the derivative of the loss with respect to the input. The underlying idea is that we can expect some discrepancy between train and test data, which will induce changes in the output and a potentially higher test loss. However, penalizing the norm of the Jacobian assumes that changes between train and test data will be isotropic. In practice, we can expect data to vary more along some directions, which is not reflected in the sensitivity. In the extreme case where the test data is exactly the same as the training data, the generalization gap will be 0, which will again not be captured by the sensitivity. In practice, whitening the data makes the sensitivity appropriate, save for a scaling factor, as we will see in the experiments.

5.5. Experiments

We now provide experimental validation of all the results in this paper. We start by analyzing the distance between information matrices, first showing its poor correlation with the training loss, then showing that these matrices appear to be remarkably aligned, albeit with different scales, when training deep networks on standard datasets.

5.5.1. Discrepancies between C, H and F

5.5.1.1. Experimental setup

For comparing the similarities and discrepencies between the information matrices, we tested

- 5 different architectures: logistic regression, a 1-hidden layer and 2-hidden layer fully connected network, and 2 small convolutional neural networks (CNNs, one with batch normalization (Ioffe & Szegedy, 2015) and one without);
- 3 datasets: MNIST, CIFAR-10, SVHN;
- 3 learning rates: 10^{-2}, $5 \cdot 10^{-3}$, and 10^{-3}, using SGD with momentum $\mu = 0.9$;
- 2 batch sizes: 64, 512;
- 5 dataset sizes: 5k, 10k, 20k, 25k, and 50k.

We train for 750k steps and compute the metrics every 75k steps. To be able to compute all the information matrices exactly, we reduced the input dimension by converting all images to greyscale and resizing them to 7×7 pixels. While this makes the classification task more challenging, our neural networks still exhibit the behaviour of larger ones by their ability to fit the training set, even with random labels. Details and additional figures can be found in appendix C.2.2.

5.5.2. Comparing Fisher and empirical Fisher

Figure 1 shows the squared Frobenius norm between $\mathbf{F}$ and $\mathbf{C}$ (on training data) for many architectures, datasets, at various stages of the optimization. We see that, while the two matrices eventually coincide on the training set for some models, the convergence is very weak as even low training errors can lead to a large discrepancy between these two matrices. In practice, $\mathbf{C}$ and $\mathbf{F}$ might be significantly different, even when computed on the training set.

5.5.3. Comparing H, F and C

In this subsection, we analyze the similarities and differences between the information matrices. We will focus on the scale similarity r, defined as the ratio of traces, and the angle similarity s, defined as the cosine between matrices. Note that having both $r(\mathbf{A}, \mathbf{B}) = 1$ and $s(\mathbf{A}, \mathbf{B}) = 1$ implies $\mathbf{A} = \mathbf{B}$.

Figure 2 shows the scale (left) and angle (right) similarities between the three pairs of matrices during the optimization of all models used in figure 4. We can see that $\mathbf{H}$ is not aligned with $\mathbf{C}$ nor $\mathbf{F}$ at the beginning of the optimization but this changes quickly. Then, all three matrices reach a very high cosine similarity, much higher than we would obtain for two random low-rank matrices. For the scaling, $\mathbf{C}$ is "larger" than the other two while $\mathbf{F}$ and $\mathbf{H}$ are very close to each other. Thus, as in the least squares case, we have $\mathbf{C} \propto\!\!\!\sim \mathbf{F} \approx \mathbf{H}$.

5.5.4. Impact of noise on second-order methods

Section 5.3 extended existing results to take the geometry of the noise and the curvature into account. Here, we show how the geometry of the noise, and in particular its

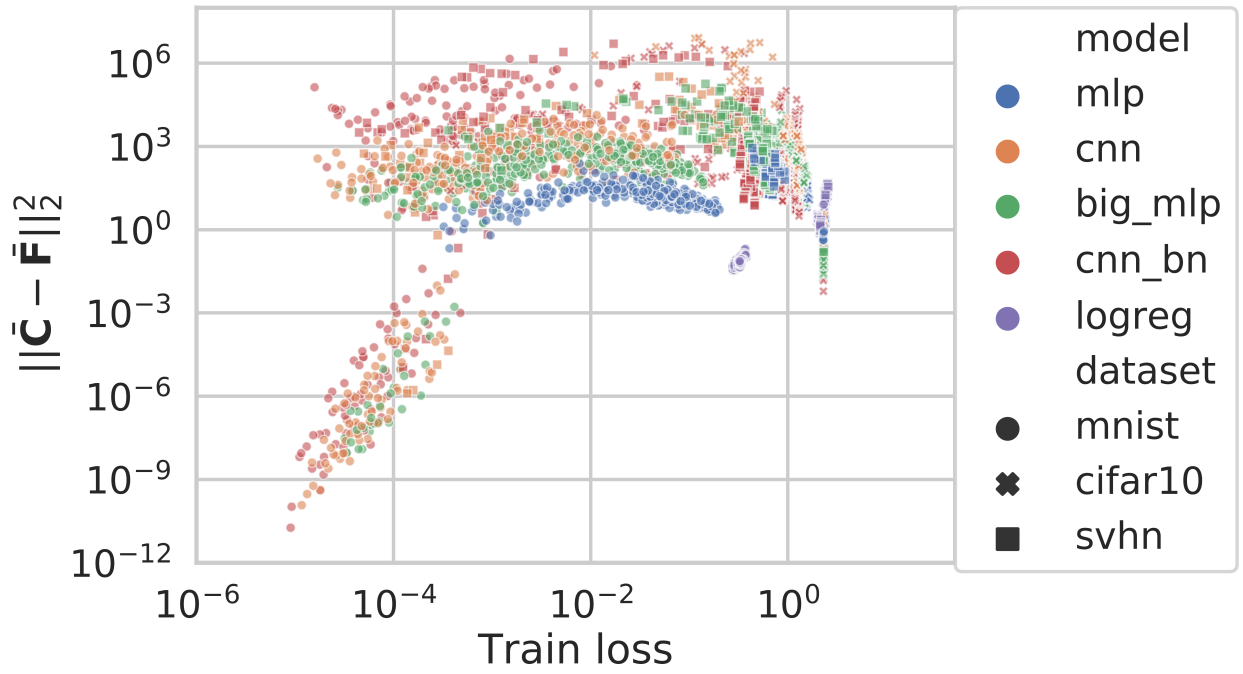


Fig. 1. Squared Frobenius norm between $\bar{\mathbf{F}}$ and $\bar{\mathbf{C}}$ (computed on the training distribution). Even for some low training losses, there can be a significant difference between the two matrices.

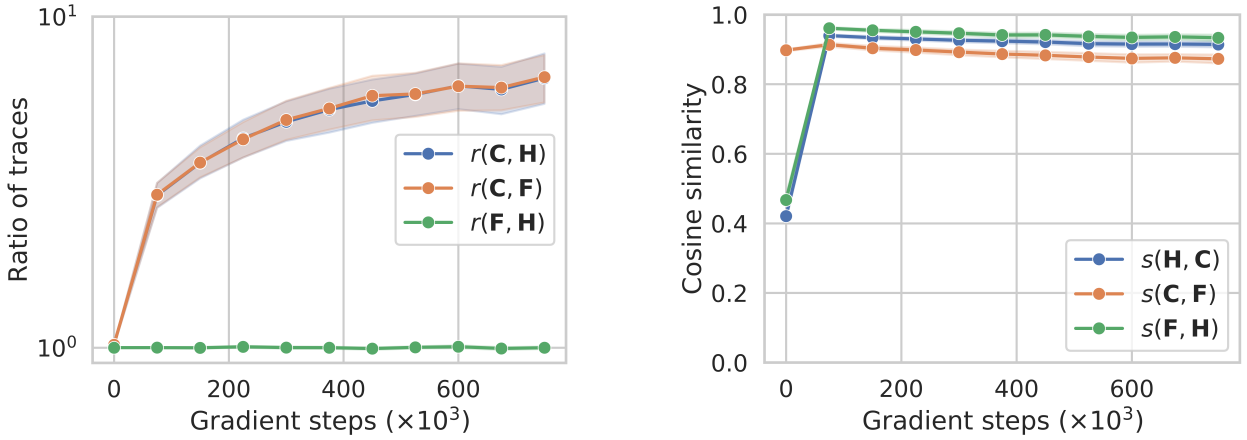


Fig. 2. Scale and angle similarities between information matrices.

relationship to the Hessian, can make or break second-order methods in the stochastic setting. To be clear, we assume here that we have access to the full Hessian and do not address the issue of estimating it from noisy samples.

We assume a quadratic $\mathcal{L}(\theta) = \frac{1}{2}\theta^\top \mathbf{H}\theta$ with $\theta \in \mathbb{R}^{20}$ and $\mathbf{H} \in \mathbb{R}^{20\times 20}$ a diagonal matrix such that $\mathbf{H}_{ii} = i^2$ with a condition number $d^2 = 400$. At each timestep, we have access to an oracle that outputs a noisy gradient, $\mathbf{H}\theta_t + \varepsilon$ with ε drawn from a zero-mean Gaussian

with covariance $\mathbf{S}$. Note here that $\mathbf{S}$ is the *centered* covariance of the gradients. We consider three settings: a) $\mathbf{S} = \alpha_1 \mathbf{H}$; b) $\mathbf{S} = \mathbf{I}$; c) $\mathbf{S} = \alpha_{-1} \mathbf{H}^{-1}$ where the constants α_1 and α_{-1} are chosen such that $\mathrm{Tr}(\mathbf{S}) = d$. Hence, these three settings are indistinguishable from the point of view of the rate of Schmidt (2014).

In this simplified setting, we get an analytic formula for the variance at each timestep and we can compute the exact number of steps t such that $\mathbb{E}[\Delta_t]$ falls below a suboptimality threshold. To vary the impact of the noise, we compute the number of steps for three different thresholds: a) $\varepsilon = 1$; b) $\varepsilon = 0.1$; c) $\varepsilon = 0.01$. For each algorithm and each noise, the stepsize is optimized to minimize the number of steps required to reach the threshold.

The results are in Table 1. We see that, while Stochastic gradient and momentum are insensitive to the geometry of the noise for small ε, Newton method is not and degrades when the noise is large in low curvature directions. For $\varepsilon = 10^{-2}$ and $\mathbf{S} \propto \mathbf{H}^{-1}$, Newton is worse than SG, a phenomenon that is not captured by the bounds of Bottou & Bousquet (2008) since they do not take the structure of the curvature and the noise into account. We also see that the advantage of Polyak momentum over stochastic gradient disappears when the suboptimality is small, i.e. when the noise is large compared to the signal.

Also worthy of notice is the fixed stepsize required to achieve suboptimality ε, as shown in Table 2. While it hardly depends on the geometry of the noise for SG and Polyak, Newton method requires much smaller stepsizes when $\mathbf{S}$ is anticorrelated with $\mathbf{H}$ to avoid amplifying the noise.

5.5.5. The TIC and the generalization gap

We now empirically test the quality of the TIC as an estimator of the generalization gap in deep networks. Following Neyshabur et al. (2017) we assess the behaviour of our generalization gap estimator by varying (1) the number of parameters in a model and (2) the label randomization ratio.

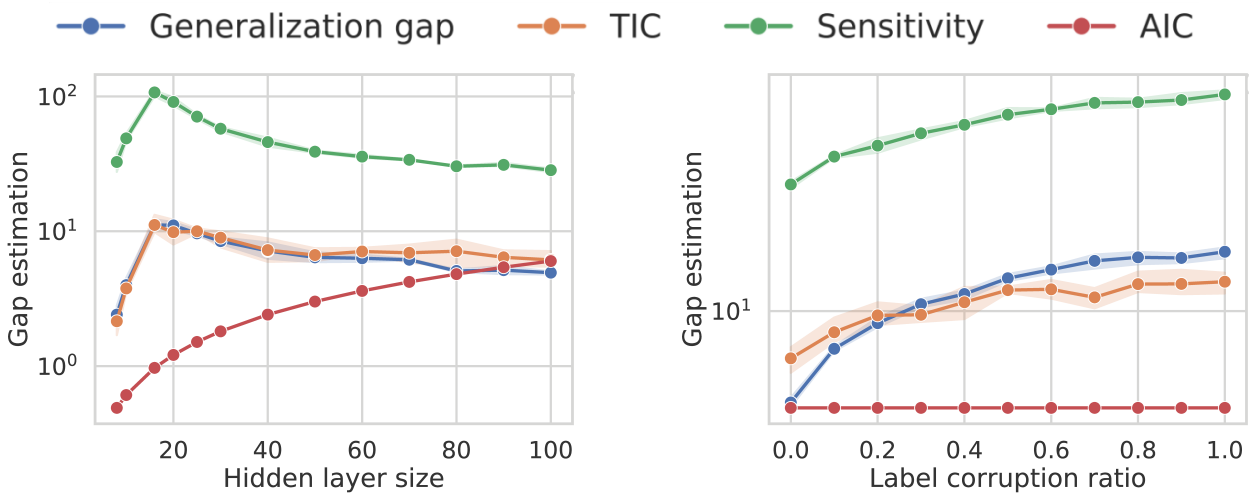
ε	**Method**	$\beta = 1$	$\beta = 0$	$\beta = -1$
10^0	SG	44	43	42
	Newton	3	2	19
	Polyak	36	36	34
10^{-1}	SG	288	253	207
	Newton	3	28	225
	Polyak	119	111	97
10^{-2}	SG	2090	1941	1731
	Newton	29	315	2663
	Polyak	1743	1727	1705

Table 1. Number of updates required to reach suboptimality of ε for various methods and $\mathbf{S} \propto \mathbf{H}^{\beta}$.

ε	**Method**	$\beta = 1$	$\beta = 0$	$\beta = -1$
10^0	SG	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
	Newton	$1 \cdot 10^{0}$	$1 \cdot 10^{0}$	$2 \cdot 10^{-1}$
	Polyak	$5 \cdot 10^{-3}$	$4 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
10^{-1}	SG	$4 \cdot 10^{-3}$	$4 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
	Newton	$1 \cdot 10^{0}$	$2 \cdot 10^{0}$	$3 \cdot 10^{-2}$
	Polyak	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$3 \cdot 10^{-3}$
10^{-2}	SG	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$2 \cdot 10^{-3}$
	Newton	$2 \cdot 10^{-1}$	$2 \cdot 10^{-2}$	$3 \cdot 10^{-3}$
	Polyak	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$

Table 2. Stepsizes achieving suboptimality ε in the fewest updates for various methods and $\mathbf{S} \propto \mathbf{H}^{\beta}$.

Experiments are performed using a fully connected feedforward network with a single hidden layer trained on a subset of 2k samples of SVHN (Netzer et al., 2011). In Figure 3a we vary the number of units in the hidden layer without label randomization while in Figure 3b we vary the label randomization ratio with a fixed architecture. Each point is computed using 3 different random number generator seeds. The neural networks are trained for 750k steps. The confidence intervals are provided using bootstrapping to estimate a 95% confidence interval. The Hessian, covariance matrices and sensitivity are computed on a subset of size 5k of the test data. Details can be found in Appendix C.2.2.



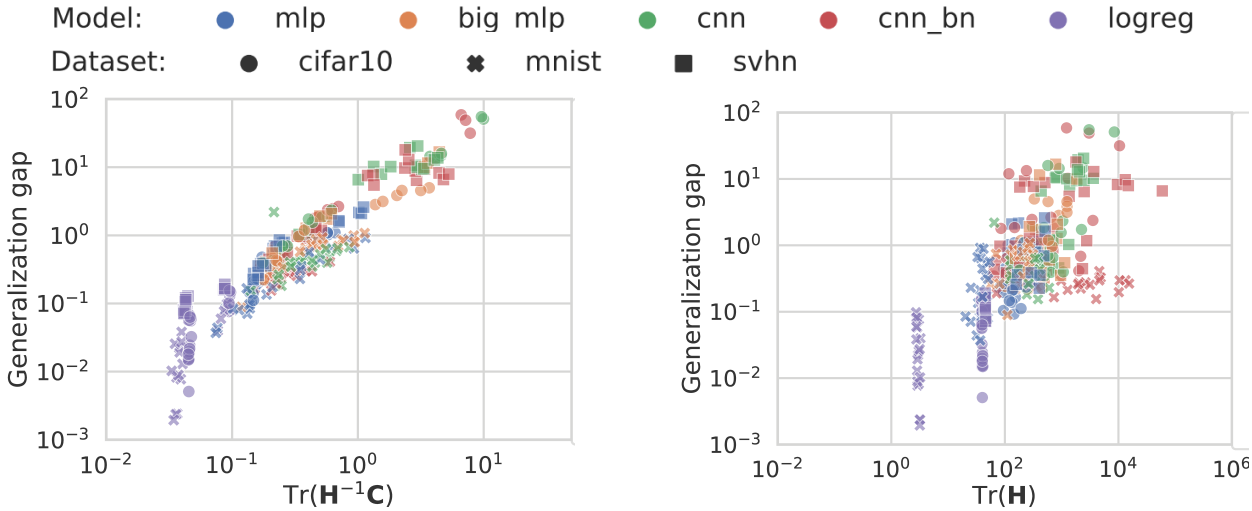
(a) Varying hidden layer size.

(b) Varying the label randomization level.

Fig. 3. Comparing the TIC to other estimators of the generalization gap on SVHN. The TIC matches the generalization gap more closely than both the AIC and the sensitivity.

We now study the ability of the TIC across a wide variety of models, datasets, and hyperparameters. More specifically, we compare the TIC to the generalization gap for: The

experiments of Figure 4 are performed with the experimental setup presented in subsection 5.5.1.1. Figure 4a shows that the TIC using $\mathbf{H}$ and $\mathbf{C}$ computed over the test set is an excellent estimator of the generalization gap. For comparison, we also show in Figure 4b the generalization gap as a function of $\mathbf{H}$ computed over the test set. We see that, even when using the test set, the correlation is much weaker than with the TIC.



(a) Gap vs. TIC.

(b) Gap vs. flatness.

Fig. 4. Generalization gap as a function of the Takeuchi information criterion (*left*) and the trace of the Hessian on the test set (*right*) for many architectures, datasets, and hyperparameters. Correlation is perfect if all points lie on a line. We see that the Hessian cannot by itself capture the generalization gap.

5.5.5.1. Efficient approximations to the TIC

Although the TIC is a good estimate of the generalization gap, it can be expensive to compute on large models. Following our theoretical and empirical analysis of the proximity of $\mathbf{H}$ and $\mathbf{F}$, we propose two approximations to the TIC: $\operatorname{Tr}(\mathbf{F}^{-1}\mathbf{C})$ and $\operatorname{Tr}(\mathbf{C})/\operatorname{Tr}(\mathbf{F})$. They are easier to compute as the $\mathbf{F}$ is in general easier to compute than $\mathbf{H}$ and the second does not require any matrix inversion.

Using the same experimental setting as in 5.5.5, we observe in Figure 5 that the replacing $\mathbf{H}$ with $\mathbf{F}$ leads to almost no loss in predictive performance. On the other hand, the ratio of the traces works best when the generalization gap is high and tends to overestimate it when it is small.

Intuition on $\operatorname{Tr}(\mathbf{C})/\operatorname{Tr}(\mathbf{F})$: it is not clear right away why the ratio of traces might be a interesting quantity. However, as observed in figure 2, $\mathbf{C}$ and $\mathbf{F}$ are remarkably aligned, but there remains a scaling factor. If we had $\mathbf{C} = \alpha\mathbf{F}$, then $\operatorname{Tr}(\mathbf{F}^{-1}\mathbf{C}) = k\alpha$ where k is the dimension of the invertible subspace of $\mathbf{F}$ and $\operatorname{Tr}(\mathbf{C})/\operatorname{Tr}(\mathbf{F}) = d\alpha$ where d is the dimensionality of θ. So, up to a multiplicative constant (or an offset in log scale), we can expect

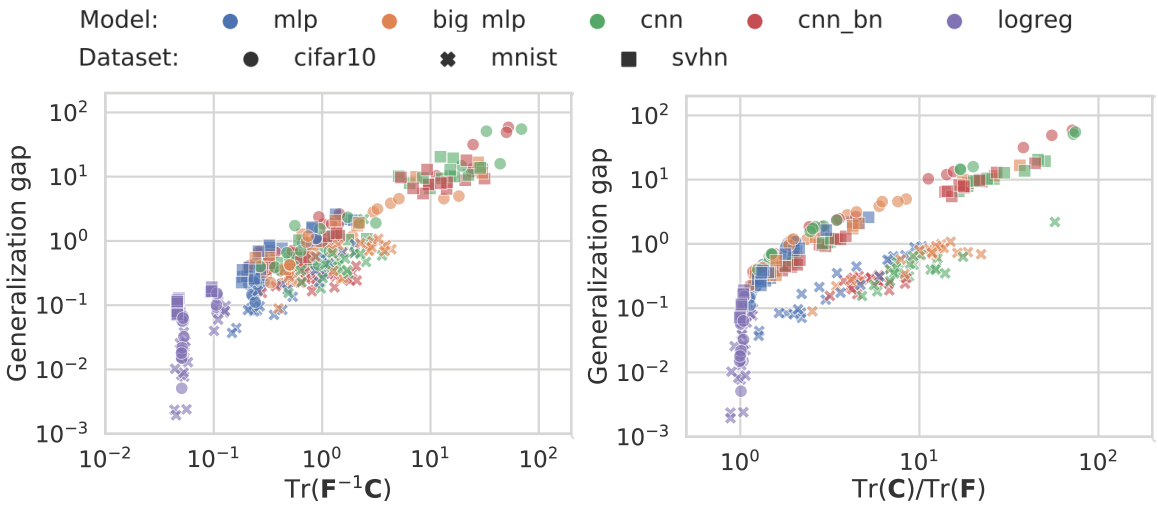


Fig. 5. Generalization gap as a function of two approximations to the Takeuchi Information Criterion: $\mathrm{Tr}(\mathbf{F}^{-1}\mathbf{C})$ (*left*) and $\mathrm{Tr}(\mathbf{C})/\mathrm{Tr}(\mathbf{F})$ (*right*).

these two quantities to exhibit similarities. Notice that on figure 5, this offset does appear and is different for every dataset (MNIST has the smallest one, then SVHN and CIFAR10, just slightly bigger).

5.5.6. The importance of the noise in estimating the generalization gap

For a given model, the generalization gap captures the discrepancy that exists between the training set and the data distribution. Hence, estimating that gap involves the evaluation of the uncertainty around the data distribution. The TIC uses $\mathbf{C}$ to capture that uncertainty but other measures probably exist. However, estimators which do not estimate it are bound to have failure modes. For instance, by using the square norm of the derivative of the loss with respect to the input, the sensitivity implicitly assumes that the uncertainty around the inputs is isotropic and will fail should the data be heavily concentrated in a low-dimensional subspace. It would be interesting to adapt the sensitivity to take the covariance of the inputs into account.

Another aspect worth mentioning is that estimators such as the margin assume that the classifier is fixed but the data is a random variable. Then, the margin quantifies the probability that a new datapoint would fall on the other side of the decision boundary. By contrast, the TIC assumes that the data are fixed but that the classifier is a random variable. It estimates the probability that a classifier trained on slightly different data would classify a training point incorrectly. In that, it echoes the uniform stability theory (Bousquet & Elisseeff, 2002), where a full training with a slightly different training set has been replaced with a local search.

5.6. Conclusion and open questions

We clarified the relationship between information matrices used in optimization. While their differences seem obvious in retrospect, the widespread confusion makes these messages necessary. Indeed, several well-known algorithms, such as Adam (Kingma & Ba, 2014), claiming to use second-order information about the loss to accelerate training seem instead to be using the covariance matrix of the gradients. Equipped with this new understanding of the difference between the curvature and noise information matrices, one might wonder if the success of these methods is not due to variance reduction instead. If so, one should be able to combine variance reduction and geometry adaptation, an idea attempted by Le Roux et al. (2011).

We also showed how, in certain settings, the geometry of the noise could affect the performance of second-order methods. While Polyak momentum is affected by the scale of the noise, its performance is independent of the geometry, similar to stochastic gradient but unlike Newton method. However, empirical results indicate that common loss functions are in the regime favorable to second-order methods.

Finally, we investigated whether the Takeuchi information criterion is relevant for estimating the generalization gap in neural networks. We provided evidence that this complexity measures involving the information matrices is predictive of the generalization performance.

We hope this study will clarify the interplay of the noise and curvature in common machine learning settings, potentially giving rise to new optimization algorithms as well as new methods to estimate the generalization gap.

Acknowledgments

We would like to thank Gauthier Gidel, Reyhane Askari and Giancarlo Kerg for reviewing an earlier version of this paper. We also thank Aristide Baratin for insightful discussions. Valentin Thomas acknowledges funding from the Open Philantropy project.

Chapitre 6

Beyond Variance Reduction: Understanding the True Impact of Baselines on Policy Optimization

Article details

Thomas Valentin*, Chung Wesley*, Machado Marlos C., Le Roux Nicolas "Beyond variance reduction: Understanding the true impact of baselines on policy optimization". In *International Conference on Machine Learning (ICML) 2021. PMLR.*

Foreword

In the summer of 2019, I interned at Google Brain Montréal with Nicolas Le Roux and Marlos C. Machado and my project was initially about using off-policy learning to reduce the variance of the gradients for Policy Gradient methods in Reinforcement Learning. However, when comparing our method to other variance reduction methods, such as the use of baselines, I came to realise that *symmetric* perturbations of the variance-minimizing baseline (e.g $\pm\varepsilon$), while increasing the variance by the same amount, led to *asymmetric* impacts on the regret/average reward. While we initially decided to continue exploring the first project, we came back to this observation which Wesley had begun to work on with Nicolas. By studying this on simple examples such as a two-arm bandits, Nicolas was first able to demonstrate our first divergence result. During the year 2020, we worked together on extending both the divergence and convergence results and deepened our understanding of the problem.

This paper could be described as one pointing out a surprising flaw in a widely used algorithm and as such it inspired future work. The most relevant direct line of work inspired by our article is the one led by Jincheng Mei who worked previously on the convergence of policy gradient methods in the *expected* regime.

Impact since publication

Following our work, Mei et al. (2021) extended our observation to a vast class of algorithms: those which are too greedy, or *committal* and may converge prematurely to a suboptimal solution. Finally, we (Jincheng, Wesley and I) started collaborating together and this culminated in a paper *The Role of Baselines in Policy Optimization* (Mei et al., 2022) published at NeurIPS 2022 which sheds light on how baselines impact the exploration behavior of policy gradient methods and how using the value function as a baseline guarantees policy improvement in expectation while the variance-minimizing baseline may not. As such policy gradient using the true value function as a baseline can converge to the optimal policy fast.

Personal contribution

- The original observation that baselines impact not only the variance but also the convergence of Policy Gradient methods was done by me during my summer internship at Google Brain with Nicolas and Marlos
- Theory (convergence proofs) and experiments for the off-policy setting with the collaboration of Wesley for the extension to $K > 2$ arms
- Collaboration with Wesley on the divergence proofs for the 3 arms setting
- Design of the simplex visualizations (Figure 1 and Figure 3)
- Writing of the paper alongside with all my co-authors

Abstract

Bandit and reinforcement learning (RL) problems can often be framed as optimization problems where the goal is to maximize average performance while having access only to stochastic estimates of the true gradient. Traditionally, stochastic optimization theory predicts that learning dynamics are governed by the curvature of the loss function and the noise of the gradient estimates. In this paper we demonstrate that the standard view is too limited for bandit and RL problems. To allow our analysis to be interpreted in light of multi-step MDPs, we focus on techniques derived from stochastic optimization principles (e.g., natural policy gradient and EXP3) and we show that some standard assumptions from optimization theory are violated in these problems. We present theoretical results showing that, at least for bandit problems, curvature and noise are not sufficient to explain the learning dynamics and that seemingly innocuous choices like the baseline can determine whether an algorithm converges. These theoretical

findings match our empirical evaluation, which we extend to multi-state MDPs.

6.1. Introduction

In the standard multi-arm bandit setting Robbins (1952), an agent needs to choose, at each timestep t, an arm $a_t \in \{1,...,n\}$ to play, receiving a potentially stochastic reward r_t with mean μ_{a_t}. The goal of the agent is usually to maximize the total sum of rewards, $\sum_{i=1}^{T} r_i$, or to maximize the average performance at time T, $\mathbb{E}_{i\sim\pi}\mu_i$ with π being the probability of the agent of drawing each arm (Bubeck & Cesa-Bianchi, 2012). While the former measure is often used in the context of bandits,[1] $\mathbb{E}_{i\sim\pi}\mu_i$ is more common in the context of Markov Decision Processes (MDPs), which have multi-arm bandits as a special case.

In this paper we focus on techniques derived from stochastic optimization principles, such as EXP3 (Auer et al., 2002; Seldin et al., 2013). In particular, we study *policy gradient* methods, a family of algorithms useful in the more general MDP setting which have seen empirical success in recent times Schulman et al. (2017b).

We analyze the problem of learning to maximize the average reward, $\mathcal{J}$, by gradient ascent:

$$\theta^* = \arg\max_\theta \mathcal{J}(\theta) = \arg\max_\theta \sum_a \pi_\theta(a)\mu_a \,, \tag{6.1.1}$$

with μ_a being the average reward of arm a. In this case, we are mainly interested in outputting an effective policy at the end of the optimization process, without explicitly considering the performance of intermediary policies.

Optimization theory predicts that the convergence speed of stochastic gradient methods will be affected by the variance of the gradient estimates and by the geometry of the function $\mathcal{J}$, represented by its curvature. Roughly speaking, the geometry dictates how effective true gradient ascent is at optimizing $\mathcal{J}(\theta)$ while the variance can be viewed as a penalty, capturing how much slower the optimization process is by using noisy versions of this true gradient. More concretely, doing one gradient step with stepsize α, using a stochastic estimate g_t of the gradient, leads to (Bottou et al., 2018):

$$\mathbb{E}[\mathcal{J}(\theta_{t+1})] - \mathcal{J}(\theta_t) \geq (\alpha - \tfrac{L\alpha^2}{2})\|\mathbb{E}[g_t]\|_2^2 - \tfrac{L\alpha^2}{2}\mathrm{Var}[g_t],$$

when $\mathcal{J}$ is L-smooth, i.e. its gradients are L-Lipschitz.

As large variance has been identified as an issue for policy gradient (PG) methods, many works have focused on reducing the noise of the updates. One common technique is the use of control variates (Greensmith et al., 2004; Hofmann et al., 2015), referred to

[1] The objective is usually presented as regret minimization.

as *baselines* in the context of RL. These baselines b are subtracted from the observed returns to obtain shifted returns, $r(a_i) - b$, and do not change the expectation of the gradient. In MDPs, they are typically state-dependent. While the value function is a common choice, previous work showed that the minimum-variance baseline for the REINFORCE (Williams, 1992) estimator is different and involves the norm of the gradient (Peters & Schaal, 2008). Reducing variance has been the main motivation for many previous works on baselines (e.g., Gu et al., 2016; Liu et al., 2017; Grathwohl et al., 2017; Wu et al., 2018; Cheng et al., 2020), but the influence of baselines on other aspects of the optimization process has hardly been studied. We take a deeper look at baselines and their effects on optimization.

Contributions

We show that baselines can impact the optimization process beyond variance reduction and lead to qualitatively different learning curves, even when the variance of the gradients is the same. For instance, given two baselines with the same variance, the more negative baseline promotes *committal* behaviour where a policy quickly tends towards a deterministic one, while the more positive baseline leads to *non-committal* behaviour, where the policy retains higher entropy for a longer period.

Furthermore, we show that **the choice of baseline can even impact the convergence of natural policy gradient** (NPG), something variance cannot explain. In particular, we construct a three-armed bandit where using the baseline minimizing the variance can lead to convergence to a deterministic, sub-optimal policy for any positive stepsize, while another baseline, with larger variance, guarantees convergence to the optimal policy. As such a behaviour is impossible under the standard assumptions in optimization, this result shows how these assumptions may be violated in practice. It also provides a counterexample to the convergence of NPG algorithms in general, a popular variant with much faster convergence rates than vanilla PG when using the true gradient in tabular MDPs (Agarwal et al., 2019).

Further, **we identify on-policy sampling as a key factor to these convergence issues** as it induces a vicious cycle where making bad updates can lead to worse policies, in turn leading to worse updates. A natural solution is to break the dependency between the sampling distribution and the updates through off-policy sampling. We show that ensuring all actions are sampled with sufficiently large probability at each step is enough to guarantee convergence in probability. Note that this form of convergence is stronger than convergence of the expected iterates, a more common type of result (e.g., Mei et al., 2020b; Agarwal et al., 2019).

We also perform an empirical evaluation on multi-step MDPs, showing that baselines have a similar impact in that setting. We observe **a significant impact on the empirical**

performance of agents when using two different sets of baselines yielding the same variance, once again suggesting that learning dynamics in MDPs are governed by more than the curvature of the loss and the variance of the gradients.

6.2. Baselines, learning dynamics & exploration

The problem defined in Eq. 6.1.1 can be solved by gradient ascent. Given access only to samples, the true gradient cannot generally be computed and the true update is replaced with a stochastic one, resulting in the following update:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N} \sum_i r(a_i) \nabla_\theta \log \pi_\theta(a_i) \,, \tag{6.2.1}$$

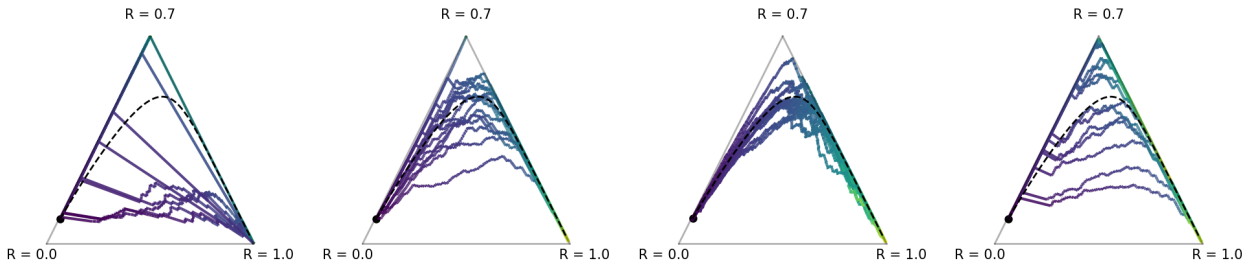
where a_i are actions drawn according to the agent's current policy π_θ, α is the stepsize, and N, which can be 1, is the number of samples used to compute the update. To reduce the variance of this estimate without introducing bias, we can introduce a baseline b, resulting in the gradient estimate $(r(a_i) - b)\nabla_\theta \log \pi_\theta(a_i)$.

While the choice of baseline is known to affect the variance, we show that baselines can also lead to qualitatively different behaviour of the optimization process, even when the variance is the same. This difference cannot be explained by the expectation or variance, quantities which govern the usual bounds for convergence rates (Bottou et al., 2018).

6.2.1. Committal and non-committal behaviours

To provide a complete picture of the optimization process, we analyze the evolution of the policy during optimization. We start in a simple setting, a deterministic three-armed bandit, where it is easier to produce informative visualizations.

To eliminate variance as a potential confounding factor, we consider different baselines with the same variance. We start by computing the baseline leading to the minimum-variance of the gradients for the algorithm we use. For vanilla policy gradient, we have $b^*_\theta = \frac{\mathbb{E}[r(a_i)\|\nabla \log \pi_\theta(a_i)\|_2^2]}{\mathbb{E}[\|\nabla \log \pi_\theta(a_i)\|_2^2]}$ (Peters & Schaal, 2008; Greensmith et al., 2004) (see Appendix D.4.1 for details and the NPG version). Note that this baseline depends on the current policy and changes throughout the optimization. As the variance is a quadratic function of the baseline, the two baselines $b^+_\theta = b^*_\theta + \varepsilon$ and $b^-_\theta = b^*_\theta - \varepsilon$ result in gradients with the same variance (see Appendix D.4.4 for details). Thus, we use these two perturbed baselines to demonstrate that there are phenomena in the optimization process that variance cannot explain.



(a) $b_\theta^- = b_\theta^* - 1/2$ **(b)** $b_\theta = b_\theta^*$ **(c)** $b_\theta^+ = b_\theta^* + 1/2$ **(d)** $b_\theta = V^{\pi_\theta}$

Fig. 1. We plot 15 different trajectories of natural policy gradient with softmax parameterization, when using various baselines, on a 3-arm bandit problem with rewards (1,0.7,0) and stepsize $\alpha = 0.025$ and $\theta_0 = (0,3,5)$. The black dot is the initial policy and colors represent time, from purple to yellow. The dashed black line is the trajectory when following the true gradient (which is unaffected by the baseline). Different values of ε denote different perturbations to the minimum-variance baseline. We see some cases of convergence to a suboptimal policy for both $\varepsilon = -1/2$ and $\varepsilon = 0$. This does not happen for the larger baseline $\varepsilon = 1/2$ or the value function as baseline. Figure made with Ternary (Harper & Weinstein, 2015).

Fig. 1 presents fifteen learning curves on the probability simplex representing the space of possible policies for the three-arm bandit, when using NPG and a softmax parameterization. We choose $\varepsilon = 1/2$ to obtain two baselines with the same variance: $b_\theta^+ = b_\theta^* + 1/2$ and $b_\theta^- = b_\theta^* - 1/2$.

Inspecting the plots, the learning curves for $\varepsilon = -1/2$ and $\varepsilon = 1/2$ are qualitatively different, even though the gradient estimates have the same variance. For $\varepsilon = -1/2$, the policies quickly reach a deterministic policy (i.e., a neighborhood of a corner of the probability simplex), which can be suboptimal, as indicated by the curves ending up at the policy choosing action 2. On the other hand, for $\varepsilon = 1/2$, every learning curve ends up at the optimal policy, although the convergence might be slower. The learning curves also do not deviate much from the curve for the true gradient. Again, these differences cannot be explained by the variance since the baselines result in identical variances.

Additionally, for $b_\theta = b_\theta^*$, the learning curves spread out further. Compared to $\varepsilon = 1/2$, some get closer to the top corner of the simplex, leading to convergence to a suboptimal solution, suggesting that the minimum-variance baseline may be worse than other, larger baselines. In the next section, we theoretically substantiate this and show that, for NPG, it is possible to converge to a suboptimal policy with the minimum-variance baseline; but there are larger baselines that guarantee convergence to an optimal policy.

We look at the update rules to explain these different behaviours. When using a baseline b with NPG, sampling a_i results in the update

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha[r(a_i) - b]F_\theta^{-1}\nabla_\theta \log \pi_\theta(a_i) \\ &= \theta_t + \alpha\frac{r(a_i) - b}{\pi_\theta(a_i)}\mathbf{1}_{a_i} + \alpha\lambda e\end{aligned}$$

where $F_\theta^{-1} = \mathbb{E}_{a\sim\pi}[\nabla \log \pi_\theta(a)\nabla \log \pi_\theta(a)^\top]$, $\mathbf{1}_{a_i}$ is a one-hot vector with 1 at index i, and λe is a vector containing λ in each entry. The second line follows for the softmax policy (see Appendix D.4.2) and λ is arbitrary since shifting θ by a constant does not change the policy.

Thus, supposing we sample action a_i, if $r(a_i) - b$ is positive, which happens more often when the baseline b is small (more negative), the update rule will increase the probability $\pi_\theta(a_i)$. This leads to an increase in the probability of taking the actions the agent took before, regardless of their quality (see Fig.1a for $\varepsilon = -1/2$). Because the agent is likely to choose the same actions again, we call this *committal* behaviour.

While a smaller baseline leads to committal behaviour, a larger (more positive) baseline makes the agent second-guess itself. If $r(a_i) - b$ is negative, which happens more often when b is large, the parameter update decreases the probability $\pi_\theta(a_i)$ of the sampled action a_i, reducing the probability the agent will re-take the actions it just took, while increasing the probability of other actions. This might slow down convergence but it also makes it harder for the agent to get stuck. This is reflected in the $\varepsilon = 1/2$ case (Fig.1c), as all the learning curves end up at the optimal policy. We call this *non-committal* behaviour.

While the previous experiments used perturbed variants of the minimum-variance baseline to control for the variance, this baseline would usually be infeasible to compute in more complex MDPs. Instead, a more typical choice of baseline would be the value function (Sutton & Barto, 2018, Ch. 13), which we evaluate in Fig. 1d. Choosing the value function as a baseline generated trajectories converging to the optimal policy, even though their convergence may be slow, despite it not being the minimum variance baseline. The reason becomes clearer when we write the value function as $V^\pi = b^*_\theta - \frac{\mathrm{Cov}(r,\|\nabla \log \pi\|^2)}{\mathbb{E}[\|\nabla \log \pi\|^2]}$ (see Appendix D.4.3). The term $\mathrm{Cov}(r, \|\nabla \log \pi\|^2)$ typically becomes negative as the gradient becomes smaller on actions with high rewards during the optimization process, leading to the value function being a noncommittal baseline, justifying a choice often made by practitioners.

Additional empirical results can be found in Appendix D.1.1 for natural policy gradient and vanilla policy gradient for the softmax parameterization. Furthermore, we explore the use of different parameterizations: First, we test projected stochastic gradient ascent and directly optimizing the policy probabilities $\pi_\theta(a)$. Next, we try the escort

transform (Mei et al., 2020a), which was designed to improve the curvature of the objective. We find qualitatively similar results in all cases; baselines can induce *committal* and *non-committal* behaviour.

6.3. Convergence to suboptimal policies with natural policy gradient (NPG)

We empirically showed that PG algorithms can reach suboptimal policies and that the choice of baseline can affect the likelihood of this occurring. In this section, we provide theoretical results proving that it is indeed possible to converge to a suboptimal policy when using NPG. We discuss how this finding fits with existing convergence results and why standard assumptions are not satisfied in this setting.

6.3.1. A simple example

Standard convergence results assume access to the true gradient (e.g., Agarwal et al., 2019) or, in the stochastic case, assume that the variance of the updates is uniformly bounded for all parameter values (e.g., Bottou et al., 2018). These assumptions are in fact quite strong and are violated in a simple two-arm bandit problem with fixed rewards. Pulling the optimal arm gives a reward of $r_1 = +1$, while pulling the suboptimal arm leads to a reward of $r_0 = 0$. We use the sigmoid parameterization and call $p_t = \sigma(\theta_t)$ the probability of sampling the optimal arm at time t.

Our stochastic estimator of the natural gradient is

$$g_t = \begin{cases} \frac{1-b}{p_t}, \text{with probability } p_t \\ \frac{b}{1-p_t}, \text{with probability } 1-p_t, \end{cases}$$

where b is a baseline that does not depend on the action sampled at time t but may depend on θ_t. By computing the variance of the updates, $\text{Var}[g_t] = \frac{(1-p_t-b)^2}{p_t(1-p_t)}$, we notice it is unbounded when the policy becomes deterministic, i.e. $p_t \to 0$ or $p_t \to 1$, violating the assumption of uniformly bounded variance, unless $b = 1-p_t$, which is the optimal baseline. Note that using vanilla (non-natural) PG would, on the contrary, yield a bounded variance. In fact, we prove a convergence result in its favour in Appendix D.2 (Prop. D.2.2).

For NPG, the proposition below establishes potential convergence to a suboptimal arm and we demonstrate this empirically in Fig. 2.

Proposition 6.3.1. *Consider a two-arm bandit with rewards* 1 *and* 0 *for the optimal and suboptimal arms, respectively. Suppose we use natural policy gradient starting from* θ_0*, with a fixed baseline* $b < 0$*, and fixed stepsize* $\alpha > 0$*. If the policy samples the optimal action with probability* $\sigma(\theta)$*, then the probability of picking the suboptimal action forever and*

having θ_t go to $-\infty$ is strictly positive. Additionally, if $\theta_0 \leq 0$, we have

$$P(\text{suboptimal action forever}) \geq (1-e^{\theta_0})(1-e^{\theta_0+\alpha b})^{-\frac{1}{\alpha b}}.$$

PROOF. All the proofs may be found in the appendix. □

The updates provide some intuition as to why there is convergence to suboptimal policies. The issue is the *committal* nature of the baseline. Choosing an action leads to an increase of that action's probability, even if it is a poor choice. Choosing the suboptimal arm leads to a decrease in θ by $\frac{\alpha b}{1-p_t}$, thus increasing the probability the same arm is drawn again and further decreasing θ. By checking the probability of this occurring forever, $P(\text{suboptimal arm forever}) = \prod_{t=1}^{\infty}(1-p_t)$, we show that $1-p_t$ converges quickly enough to 1 that the infinite product is nonzero, showing it is possible to get trapped choosing the wrong arm forever (Prop. 6.3.1), and $\theta_t \to -\infty$ as t grows.

This issue could be solved by picking a baseline with lower variance. For instance, the minimum-variance baseline $b = 1-p_t$ leads to 0 variance and both possible updates are equal to $+\alpha$, guaranteeing that $\theta \to +\infty$, thus convergence. In fact, any baseline $b \in (0,1)$ suffices since both updates are positive and greater than $\alpha \min(b, 1-b)$. However, this is not always the case, as we show in the next section.

To decouple the impact of the variance with that of the committal nature of the baseline, Prop. 6.3.2 analyzes the learning dynamics in the two-arm bandit case for perturbations of the optimal baseline, i.e. we study baselines of the form $b = b^* + \varepsilon$ and show how ε, and particularly its sign, affects learning. Note that, because the variance is a quadratic function with its minimum in b^*, both $+\varepsilon$ and $-\varepsilon$ have the same variance. Our findings can be summarized as follows:

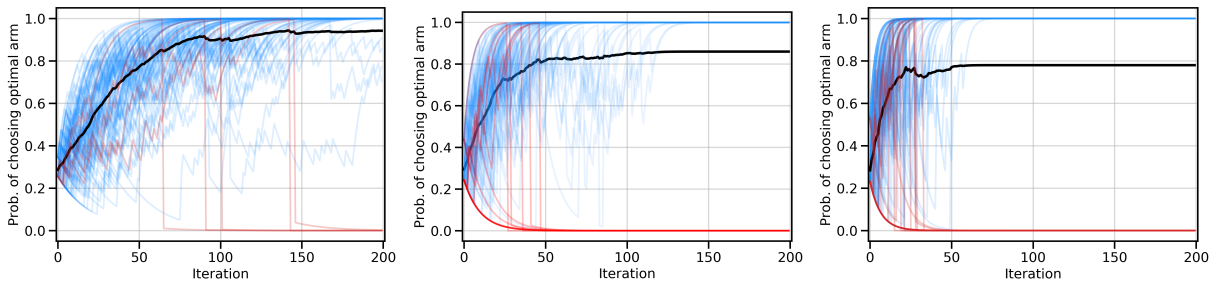
Proposition 6.3.2. *For the two-armed bandit defined in Prop. 6.3.1, when using a perturbed min-variance baseline $b = b^* + \varepsilon$, the value of ε determines the learning dynamics as follows:*

- *For $\varepsilon < -1$, there is a positive probability of converging to the suboptimal arm.*
- *For $\varepsilon \in (-1,1)$, we have convergence in probability to the optimal policy.*
- *For $\varepsilon \geq 1$, the supremum of the iterates goes to $+\infty$ in probability.*

While the proofs can be found in Appendix D.2.2, we provide here some intuition behind these results.

For $\varepsilon < -1$, we reuse the same argument as for $b < 0$ in Prop. 6.3.1. The probability of drawing the correct arm can decrease quickly enough to lead to convergence to the suboptimal arm.

For $\varepsilon \in (-1,1)$, the probability of drawing the correct arm cannot decrease too fast. Hence, although the updates, as well as the variance of the gradient estimate, are potentially unbounded, we still have convergence to the optimal solution in probability.



(a) $\alpha = 0.05$ **(b)** $\alpha = 0.1$ **(c)** $\alpha = 0.15$

Fig. 2. Learning curves for 100 runs of 200 steps, on the two-arm bandit, with baseline $b = -1$ for three different stepsizes α. *Blue:* Curves converging to the optimal policy. *Red:* Curves converging to a suboptimal policy. *Black:* Avg. performance. The number of runs that converged to the suboptimal solution are 5%, 14% and 22% for the three α's. Larger α's are more prone to getting stuck at a suboptimal solution but settle on a deterministic policy more quickly.

Finally, for $\varepsilon \geq 1$, we can reuse an intermediate argument from the $\varepsilon \in (0,1)$ case to argue that for any threshold C, the parameter will eventually exceed that threshold. For $\varepsilon \in (0,1)$, once a certain threshold is crossed, the policy is guaranteed to improve at each step. However, with a large positive perturbation, updates are larger and we lose this additional guarantee, leading to the weaker result.

We want to emphasize that not only we get provably different dynamics for $\varepsilon < -1$ and $\varepsilon \geq 1$, showing the importance of the sign of the perturbation, but that there also is a sharp transition around $|\varepsilon| = 1$, which cannot be captured solely by the variance.

The above analysis was specific to these updates. To predict committal vs. non-committal behaviour more generally, it may be possible to utilize higher order moments or other distributional properties, even when the mean and variance is the same. Unfortunately, it is difficult to utilize higher-moment information in theoretical bounds in a general manner as Markov-type inequalities do not take into account the sign of the higher moment, which we think is where the committal vs. non-committal distinction would appear.

6.3.2. Reducing variance with baselines can be detrimental

As we saw with the two-armed bandit, the direction of the updates is important in assessing convergence. More specifically, problems can arise when the choice of baseline induces committal behaviour. We now show a different bandit setting where committal behaviour happens even when using the minimum-variance baseline, thus leading to convergence to a suboptimal policy. Furthermore, we design a better baseline which ensures all updates move the parameters towards the optimal policy. This cements the idea that the quality of parameter updates must not be analyzed in terms of variance but

rather in terms of the probability of going in a bad direction, since a baseline that induces higher variance leads to convergence while the minimum-variance baseline does not. The following theorem summarizes this.

Theorem 1. There exists a three-arm bandit where using the stochastic natural gradient on a softmax-parameterized policy with the minimum-variance baseline can lead to convergence to a suboptimal policy with probability $\rho > 0$, and there is a different baseline (with larger variance) which results in convergence to the optimal policy with probability 1.

The bandit used in this theorem is the one we used for the experiments depicted in Fig. 1. The key is that the minimum-variance baseline can be lower than the second best reward; so pulling the second arm will increase its probability and induce committal behaviour. This can cause the agent to prematurely commit to the second arm and converge to the wrong policy. On the other hand, using any baseline whose value is between the optimal reward and the second best reward, which we term a *gap* baseline, will always increase the probability of the optimal action at every step, no matter which arm is drawn. Since the updates are sufficiently large at every step, this is enough to ensure convergence with probability 1, despite the higher variance compared to the minimum variance baseline. The key is that whether a baseline underestimates or overestimates the second best reward can affect the algorithm convergence and this is more critical than the resulting variance of the gradient estimates.

As such, more than lower variance, good baselines are those that can assign positive effective returns to the good trajectories and negative effective returns to the others. These results cast doubt on whether finding baselines which minimize variance is a meaningful goal to pursue. The baseline can affect optimization in subtle ways, beyond variance, and further study is needed to identify the true causes of some improved empirical results observed in previous works. This importance of the sign of the returns, rather than their exact value, echoes with the cross-entropy method (De Boer et al., 2005), which maximizes the probability of the trajectories with the largest returns, regardless of their actual value.

6.4. Off-policy sampling

So far, we have seen that *committal* behaviour can be problematic as it can cause convergence to a suboptimal policy. This can be especially problematic when the agent follows a near-deterministic policy as it is unlikely to receive different samples which would move the policy away from the closest deterministic one, regardless of the quality of that policy.

Up to this point, we assumed that actions were sampled according to the current policy, a setting known as *on-policy*. This setting couples the updates and the policy and is a root cause of the *committal* behaviour: the update at the current step changes the policy, which affects the distribution of rewards obtained and hence the next updates. However, we know from the optimization literature that bounding the variance of the updates will lead to convergence (Bottou et al., 2018). As the variance becomes unbounded when the probability of drawing some actions goes to 0, a natural solution to avoid these issues is to sample actions from a behaviour policy that selects every action with sufficiently high probability. Such a policy would make it impossible to choose the same, suboptimal action forever.

6.4.1. Convergence guarantees with IS

Because the behaviour policy changed, we introduce importance sampling (IS) corrections to preserve the unbiased updates (Kahn & Harris, 1951; Precup, 2000a). These changes are sufficient to guarantee convergence for any baseline:

Proposition 6.4.1. *Consider a n-armed bandit with stochastic rewards with bounded support and a unique optimal action. The behaviour policy μ_t selects action i with probability $\mu_t(i)$ and let $\varepsilon_t = \min_i \mu_t(i)$. When using NPG with importance sampling and a bounded baseline b, if $\lim_{t\to\infty} t\ \varepsilon_t^2 = +\infty$, then the target policy π_t converges to the optimal policy in probability.*

Proof. *(Sketch)* Using Azuma-Hoeffding's inequality, we can show that for well chosen constants Δ_i, δ and $C > 0$,

$$\mathbb{P}\left(\theta_t^1 \geq \theta_0^1 + \alpha\delta\Delta_1 t\right) \quad \geq \quad 1 - \exp\left(-\frac{\delta^2\Delta_1^2}{2C^2} t\varepsilon_t^2\right)$$

where θ^1 is the parameter associated to the optimal arm. Thus if $\lim_{t\to\infty} t\varepsilon_t^2 = +\infty$, the RHS goes to 1. In a similar manner, we can upper bound $\mathbb{P}\left(\theta_t^i \geq \theta_0^i + \alpha\delta\Delta_i t\right)$ for all suboptimal arms, and applying an union bound, we get the desired result. □

The condition on μ_t imposes a cap on how fast the behaviour policy can become deterministic: no faster than $t^{-1/2}$. Intuitively, this ensures each action is sampled sufficiently often and prevents premature convergence to a suboptimal policy. The condition is satisfied for any sequence of behaviour policies which assign at least ε_t probability to each action at each step, such as ε-greedy policies. It also holds if ε_t decreases over time at a sufficiently slow rate. By choosing as behaviour policy μ a linear interpolation between π and the uniform policy, $\mu(a) = (1-\gamma)\pi(a) + \frac{\gamma}{K}, \gamma \in (0,1]$, where K is the number of arms, we recover the classic EXP3 algorithm (Auer et al., 2002; Seldin et al., 2012).

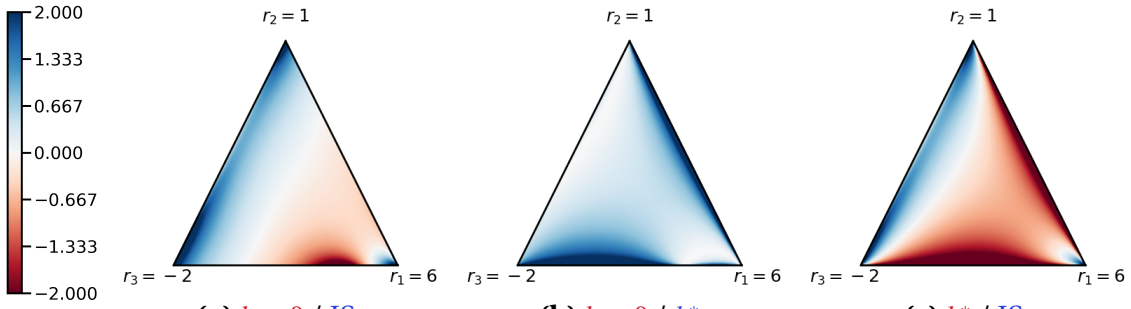
We can also confirm that this condition is not satisfied for the simple example we presented when discussing convergence to suboptimal policies. There, p_t could decrease exponentially fast since the tails of the sigmoid function decay exponentially and the parameters move by at least a constant at every step. In this case, $\varepsilon_t = \Omega(e^{-t})$, resulting in $\lim_{t\to\infty} te^{-2t} = 0$, so Proposition 6.4.1 does not apply.

6.4.2. Importance sampling, baselines & variance

As we have seen, using a separate behaviour policy that samples all actions sufficiently often may lead to stronger convergence guarantees, even if it increases the variance of the gradient estimates in most of the space, as what matters is what happens in the high variance regions, which are usually close to the boundaries. Fig. 3 shows the ratios of gradient variances between on-policy PG without baseline, on-policy PG with the minimum variance baseline, and off-policy PG using importance sampling (IS) where the sampling distribution is $\mu(a) = \frac{1}{2}\pi(a) + \frac{1}{6}$, i.e. a mixture of the current policy π and the uniform distribution. While using the minimum variance baseline decreases the variance on the entire space compared to not using a baseline, IS actually *increases* the variance when the current policy is close to uniform. However, IS does a much better job at reducing the variance close to the boundaries of the simplex, where it actually matters to guarantee convergence.

This suggests that convergence of PG methods is not so much governed by the variance of the gradient estimates in general, but by the variance in the worst regions, usually near the boundary. While baselines can reduce the variance, they generally cannot prevent the variance in those regions from exploding, leading to the policy getting stuck. Thus, good baselines are not the ones reducing the variance across the space but rather those that can prevent the learning from reaching these regions altogether. Large values of b, such that $r(a_i) - b$ is negative for most actions, achieve precisely that. On the other hand, due to the increased flexibility of sampling distributions, IS can limit the nefariousness of these critical regions, offering better convergence guarantees despite not reducing variance everywhere.

Importantly, although IS is usually used in RL to correct for the distribution of past samples (e.g., Munos et al., 2016), we advocate here for expanding the research on designing appropriate sampling distributions as done by Hanna et al. (2017, 2018) and Parmas & Sugiyama (2019). This line of work has a long history in statistics (c.f., Liu, 2008).



(a) $b = 0$ / IS. **(b)** $b = 0$ / b^*. **(c)** b^* / IS.

Fig. 3. Comparison between the variance of different methods on a 3-arm bandit. Each plot depicts the log of the ratio between the variance of two approaches. For example, Fig. (a) depicts $\log \frac{\mathrm{Var}[g_{b=0}]}{\mathrm{Var}[g_{\mathrm{IS}}]}$, the log of the ratio between the variance of the gradients of PG without a baseline and PG with IS. The triangle represents the probability simplex with each corner representing a deterministic policy on a specific arm. The method written in blue (resp. red) in each figure has lower variance in blue (resp. red) regions of the simplex. The sampling policy μ, used in the PG method with IS, is a linear interpolation between π and the uniform distribution, $\mu(a) = \frac{1}{2}\pi(a) + \frac{1}{6}$. Note that this is not the min. variance sampling distribution and it leads to higher variance than PG without a baseline in some parts of the simplex.

6.4.3. Other mitigating strategies

We conclude this section by discussing alternative strategies to mitigate the convergence issues. While they might be effective, and some are indeed used in practice, they are not without pitfalls.

First, one could consider reducing the stepsizes, with the hope that the policy would not converge as quickly towards a suboptimal deterministic policy and would eventually leave that bad region. Indeed, if we are to use vanilla PG in the two-arm bandit example, instead of NPG, this effectively reduces the stepsize by a factor of $\sigma(\theta)(1-\sigma(\theta))$ (the Fisher information). In this case, we are able to show convergence in probability to the optimal policy. See Proposition D.2.2 in Appendix D.2.

Empirically, we find that, when using vanilla PG, the policy may still remain stuck near a suboptimal policy when using a negative baseline, similar to Fig. 2. While the previous proposition guarantees convergence eventually, the rate may be very slow, which remains problematic in practice. There is theoretical evidence that following even the true vanilla PG may result in slow convergence (Schaul et al., 2019), suggesting that the problem is not necessarily due to noise.

An alternative solution would be to add entropy regularization to the objective. By doing so, the policy would be prevented from getting too close to deterministic policies.

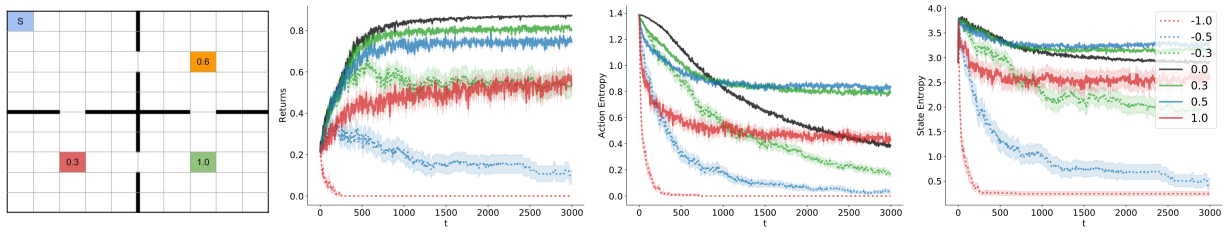
While this might prevent convergence to a suboptimal policy, it would also exclude the possibility of fully converging to the optimal policy, though the policy may remain near it.

In bandits, EXP3 has been found not to enjoy high-probability guarantees on its regret so variants have been developed to address this deficiency (c.f. Lattimore & Szepesvári, 2020). For example, by introducing bias in the updates, their variance can be reduced significantly Auer et al. (2002); Neu (2015). Finally, other works have also developed provably convergent policy gradient algorithms using different mechanisms, such as exploration bonuses or ensembles of policies (Cai et al., 2019; Efroni et al., 2020; Agarwal et al., 2020).

6.5. Extension to multi-step MDPs

We focused our theoretical analyses on multi-arm bandits so far. However, we are also interested in more general environments where gradient-based methods are commonplace. We now turn our attention to the Markov Decision Process (MDP) framework (Puterman, 2014). An MDP is a set $\{\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho\}$ where $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, P is the environment transition function, r is the reward function, $\gamma \in [0,1)$ the discount factor, and ρ is the initial state distribution. The goal of RL algorithms is to find a policy π_θ, parameterized by θ, which maximizes the (discounted) expected return; i.e. Eq. 6.1.1 becomes

$$\arg\max_\theta \mathcal{J}(\theta) = \arg\max_\theta \sum_s d_\gamma^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) r(s,a),$$



(a) MDP **(b)** Returns **(c)** Entropy (A) **(d)** Entropy (S)

Fig. 4. We plot the discounted returns, the entropy of the policy over the states visited in each trajectory, and the entropy of the state visitation distribution, averaged over 50 runs, for multiple baselines. The baselines are of the form $b(s) = b^*(s) + \varepsilon$, perturbations of the minimum-variance baseline, with ε indicated in the legend. The shaded regions denote one standard error. Note that the policy entropy of lower baselines tends to decay faster than for larger baselines. Also, smaller baselines tend to get stuck on suboptimal policies, as indicated by the returns plot. See text for additional details.

where there is now a discounted distribution over states induced by π_θ. Although that distribution depends on π_θ in a potentially complex way, the parameter updates are similar to Eq. 6.2.1:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N} \sum_i [Q(s_i, a_i) - b(s_i)] \nabla_\theta \log \pi_\theta(a_i|s_i) \ ,$$

where (a_i, s_i) pairs are drawn according to the discounted state-visitation distribution induced by π_θ and Q is the state-action value function induced by π_θ (c.f. Sutton & Barto, 2018). To match the bandit setting and common practice, we made the baseline state dependent.

Although our theoretical analyses do not easily extend to multi-step MDPs, we empirically investigated if the similarity between these formulations leads to similar differences in learning dynamics when changing the baseline. We consider a 10x10 gridworld consisting of 4 rooms as depicted on Fig. 4a. We use a discount factor $\gamma = 0.99$. The agent starts in the upper left room and two adjacent rooms contain a goal state of value 0.6 or 0.3. The best goal (even discounted), with a value of 1, lies in the furthest room, so that the agent must learn to cross the sub-optimal rooms and reach the furthest one.

Similar to the bandit setting, for a state s, we can derive the minimum-variance baseline $b^*(s)$ assuming access to state-action values $Q(s,a)$ for π_θ and consider perturbations to it. Again, we use baselines $b(s) = b^*(s) + \varepsilon$ and $b(s) = b^*(s) - \varepsilon$, since they result in identical variances (this would not be the case if we used standard REINFORCE). We use a natural policy gradient estimate, which substitutes $\nabla \log \pi(a_i|s_i)$ by $F_{s_i}^{-1} \nabla \log \pi(a_i|s_i)$ in the update rule, where F_{s_i} is the Fisher information matrix for state s_i and solve for the exact $Q(s,a)$ values using dynamic programming for all updates (see Appendix D.4.6 for details).

In order to identify the committal vs. non-committal behaviour of the agent depending on the baseline, we monitor the entropy of the policy and the entropy of the stationary state distribution over time. Fig.4b shows the average returns over time and Fig.4c and 4d show the entropy of the policy in two ways. The first is the average entropy of the action distribution along the states visited in each trajectory, and the second is the entropy of the distribution of the number of times each state is visited up to that point in training.

The action entropy for smaller baselines tends to decay faster compared to larger ones, indicating convergence to a deterministic policy. This quick convergence is premature in some cases since the returns are not as high for the lower baselines. In fact for $\varepsilon = -1$, we see that the agent gets stuck on a policy that is unable to reach any goal within the time limit, as indicated by the returns of 0. On the other hand, the larger baselines tend to achieve larger returns with larger entropy policies, but do not fully converge to the optimal policy as evidenced by the gap in the returns plot.

Since committal and non-committal behaviour can be directly inferred from the PG and the sign of the effective rewards $R(\tau) - b$, we posit that these effects extend to all MDPs. In particular, in complex MDPs, the first trajectories explored are likely to be suboptimal and a low baseline will increase their probability of being sampled again, requiring the use of techniques such as entropy regularization to prevent the policy from getting stuck too quickly. In some preliminary experiments with a deep RL policy gradient algorithm, PPO Schulman et al. (2017b), where we perturb the baseline by a fixed constant, seem to indicate that negative perturbations perform slightly worse than positive perturbations. The results are not conclusive though and there are many confounding factors in this setting which could affect the outcome, including clipping due to PPO, neural network generalization, and adaptive optimizers. It is likely that a more careful strategy to perturb the baseline is needed to gain benefits, similar to using exploration bonuses.

6.6. Conclusion

We presented results that dispute common beliefs about baselines, variance, and policy gradient methods in general. As opposed to the common belief that baselines only provide benefits through variance reduction, we showed that they can significantly affect the optimization process in ways that cannot be explained by the variance and that lower variance can even sometimes be detrimental.

Different baselines can give rise to very different learning dynamics, even when they reduce the variance of the gradients equally. They do that by either making a policy quickly tend towards a deterministic one (*committal* behaviour) or by maintaining high-entropy for a longer period of time (*non-committal* behaviour). We showed that *committal* behaviour can be problematic and lead to convergence to a suboptimal policy. Specifically, we showed that stochastic natural policy gradient does not always converge to the optimal solution due to the unusual situation in which the iterates converge to the optimal policy in expectation but not almost surely. Moreover, we showed that baselines that lead to lower-variance can sometimes be detrimental to optimization, highlighting the limitations of using variance to analyze the convergence properties of these methods. We also showed that standard convergence guarantees for PG methods do not apply to some settings because the assumption of bounded variance of the updates is violated.

The aforementioned convergence issues are also caused by the problematic coupling between the algorithm's updates and its sampling distribution since one directly impacts the other. As a potential solution, we showed that off-policy sampling can sidestep these

difficulties by ensuring we use a sampling distribution that is different than the one induced by the agent's current policy. This supports the hypothesis that on-policy learning can be problematic, as observed in previous work (Schaul et al., 2019; Hennes et al., 2020). Nevertheless, importance sampling in RL is generally seen as problematic (van Hasselt et al., 2018) due to instabilities it introduces to the learning process. Moving from an imposed policy, using past trajectories, to a chosen sampling policy reduces the variance of the gradients for near-deterministic policies and can lead to much better behaviour. In general, other variance-reduction strategies may also be more effective Xu et al. (2019).

More broadly, this work suggests that treating bandit and reinforcement learning problems as a black-box optimization of a function $\mathcal{J}(\theta)$ may be insufficient to perform well. As we have seen, the current parameter value can affect all future parameter values by influencing the data collection process and thus the updates performed. Theoretically, relying on immediately available quantities such as the gradient variance and ignoring the sequential nature of the optimization problem is not enough to discriminate between certain optimization algorithms. In essence, to design highly-effective policy optimization algorithms, it may be necessary to develop a better understanding of how the optimization process evolves over many steps.

Acknowledgements

We would like to thank Kris de Asis, Alan Chan, Ofir Nachum, Doina Precup, Dale Schuurmans, and Ahmed Touati for helpful discussions. We also thank Courtney Paquette, Vincent Liu, Scott Fujimoto and Csaba Szepesvári for reviewing an earlier version of this paper. Marlos C. Machado and Nicolas Le Roux are supported by a Canada CIFAR AI Chair.

Conclusion and future works

Conclusion

In this thesis, we have explored the roles and challenges presented by stochasticity in reinforcement learning and optimization algorithms. We have investigated the impact of noise in the learning process and presented four significant contributions that address various aspects of this issue. The main axes of our research were the following:

Not sure about the name of this first point

- **Learning diverse behaviors for planning:** Our first two articles focused on learning stochastic policies able to explore the environment and discover diverse behaviors. In our first article, *Independently Controllable Factors*, we learn policies, or options, indexed on a latent factor representation z. We were able to show that we both learned disentangled representations of the world as well as policies that were able to modify these factors. Finally, we were able to demonstrate how this representation could be used for planning. In our second article, *Sequential Monte Carlo Planning*, we designed a general purpose planning algorithm that can be used in continuous control tasks. Our SMCP algorithm can be viewed as a maximum entropy planning algorithm and as such can discover stochastic policies discovering diverse solutions.
- **Understanding the role of noise:** The next two contributions were more focused on understanding the role of noise during the optimization process. In our third article, we analyse the role of the interplay between the gradient noise and the local curvature of the loss function. Specifically, we show it can impact the optimization speed and generalization properties of models trained via maximum likelihood. Finally, in our fourth article, we investigate the role of *baselines* in policy gradient methods. Baselines are often presented as a way to reduce the variance of the gradient estimator without affecting the bias. We show that baselines have an effect beyond variance reduction directly impact the exploration/exploitation trade-off and as such can impact which policies are discovered.

The work presented in this thesis provides valuable insights into the role of stochasticity in reinforcement learning and optimization algorithms, offering a solid foundation for future research in this area.

By further examining the interplay between noise and learning, we can continue to develop more robust, adaptive, and efficient algorithms that can better handle the challenges of real-world environments. Ultimately, understanding and harnessing the power of stochasticity will bring us closer to achieving the long-term goal of creating intelligent machines capable of interacting with the world and learning from experience, just as humans and animals do.

Future works

For future works, we would like to continue to explore the role of stochasticity in reinforcement learning and optimization algorithms. In particular, we would like to investigate the following directions:

- Stochasticity and credit assignement? How does the the structure of the environment can effect learning of the value function?
- Concentration in high dimensions and the behavior of some objects become deterministic (Thomas, 2022).

End goal: being able to understand the properties that affects the behaviors of large scale models for reinfrocement learning. Application to hyperparameter tuning or scaling laws?

Références bibliographiques

Abdolmaleki, Abbas, Springenberg, Jost Tobias, Tassa, Yuval, Munos, Remi, Heess, Nicolas, and Riedmiller, Martin. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Achiam, Joshua, Edwards, Harrison, Amodei, Dario, and Abbeel, Pieter. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Agarwal, Alekh, Kakade, Sham M, Lee, Jason D, and Mahajan, Gaurav. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

Agarwal, Alekh, Henaff, Mikael, Kakade, Sham, and Sun, Wen. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020.

Akaike, Hirotugu. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

Anand, Ankesh, Racah, Evan, Ozair, Sherjil, Bengio, Yoshua, Côté, Marc-Alexandre, and Hjelm, R Devon. Unsupervised state representation learning in atari. *arXiv preprint arXiv:1906.08226*, 2019.

Asadi, Kavosh and Littman, Michael L. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pp. 243–252. PMLR, 2017.

Atkeson, Christopher G, Moore, Andrew W, and Schaal, Stefan. Locally weighted learning. *Lazy learning*, pp. 11–73, 1997.

Attias, Hagai. Planning by probabilistic inference. In *AISTATS*. Citeseer, 2003.

Auer, Peter, Cesa-Bianchi, Nicolo, Freund, Yoav, and Schapire, Robert E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Bach, Francis and Moulines, Eric. Non-strongly-convex smooth stochastic approximation with convergence rate o (1/n). In *Advances in Neural Information Processing Systems*, pp. 773–781, 2013.

Baird, Leemon. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Barto, Andrew G, Singh, Satinder, and Chentanez, Nuttapong. Intrinsically motivated learning of hierarchical collections of skills.

Beirami, Ahmad, Razaviyayn, Meisam, Shahrampour, Shahin, and Tarokh, Vahid. On optimal generalizability in parametric learning. In *Advances in Neural Information Processing Systems*, pp. 3455–3465, 2017.

Bellemare, Marc G, Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellman, Richard and Kalaba, Robert. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

Bellman, Richard et al. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

Belousov, Boris, Abdulsamad, Hany, Klink, Pascal, Parisi, Simone, and Peters, Jan. *Reinforcement learning algorithms: analysis and applications*. Springer, 2021.

Bengio, Emmanuel, Thomas, Valentin, Pineau, Joelle, Precup, Doina, and Bengio, Yoshua. Independently controllable features. *arXiv preprint arXiv:1703.07718*, 2017.

Bengio, Yoshua. *Learning deep architectures for AI*. Now Publishers, 2009.

Berlyne, Daniel E. Curiosity and exploration. *Science*, 153(3731):25–33, 1966.

Billingsley, Patrick. *Probability and measure*. John Wiley & Sons, 2008.

Bottou, Léon and Bousquet, Olivier. The tradeoffs of large scale learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL `https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf`.

Bottou, Léon and Bousquet, Olivier. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Bottou, Léon, Curtis, Frank E, and Nocedal, Jorge. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Bousquet, Olivier and Elisseeff, André. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

Bresler, Yoram. Two-filter formulae for discrete-time non-linear bayesian smoothing. *International Journal of Control*, 43(2):629–641, 1986.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bubeck, Sébastien and Cesa-Bianchi, Nicolo. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

Byravan, Arunkumar, Hasenclever, Leonard, Trochim, Piotr, Mirza, Mehdi, Ialongo, Alessandro Davide, Tassa, Yuval, Springenberg, Jost Tobias, Abdolmaleki, Abbas,

Heess, Nicolas, Merel, Josh, and Riedmiller, Martin A. Evaluating model-based planning and planner amortization for continuous control. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=SS8F6tFX3-.

Cai, Qi, Yang, Zhuoran, Jin, Chi, and Wang, Zhaoran. Provably efficient exploration in policy optimization. *arXiv preprint arXiv:1912.05830*, 2019.

Chaudhari, Pratik and Soatto, Stefano. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *arXiv preprint arXiv:1710.11029*, 2017.

Cheng, Ching-An, Yan, Xinyan, and Boots, Byron. Trajectory-wise control variates for variance reduction in policy gradient methods. In *Conference on Robot Learning*, pp. 1379–1394, 2020.

Chowdhery, Aakanksha, Narang, Sharan, Devlin, Jacob, Bosma, Maarten, Mishra, Gaurav, Roberts, Adam, Barham, Paul, Chung, Hyung Won, Sutton, Charles, Gehrmann, Sebastian, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chua, Kurtland, Calandra, Roberto, McAllister, Rowan, and Levine, Sergey. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

Colas, Cédric, Sigaud, Olivier, and Oudeyer, Pierre-Yves. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

Dayan, Peter. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Dayan, Peter and Hinton, Geoffrey E. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

De Boer, Pieter-Tjerk, Kroese, Dirk P, Mannor, Shie, and Rubinstein, Reuven Y. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Dieuleveut, Aymeric, Bach, Francis, et al. Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, 44(4):1363–1399, 2016.

Dinh, Laurent, Krueger, David, and Bengio, Yoshua. NICE: Non-linear Independent Components Estimation. arXiv:1410.8516, ICLR 2015 workshop, 2014.

Dinh, Laurent, Pascanu, Razvan, Bengio, Samy, and Bengio, Yoshua. Sharp minima can generalize for deep nets. *International Conference on Machine Learning (ICML)*, 2017.

Donsker, Monroe D and Varadhan, SR Srinivasa. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

Efroni, Yonathan, Shani, Lior, Rosenberg, Aviv, and Mannor, Shie. Optimistic policy optimization with bandit feedback. *arXiv preprint arXiv:2002.08243*, 2020.

Eslami, SM Ali, Rezende, Danilo Jimenez, Besse, Frederic, Viola, Fabio, Morcos, Ari S, Garnelo, Marta, Ruderman, Avraham, Rusu, Andrei A, Danihelka, Ivo, Gregor, Karol, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Eysenbach, Benjamin, Gupta, Abhishek, Ibarz, Julian, and Levine, Sergey. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Fedus, William, Zoph, Barret, and Shazeer, Noam. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL `http://jmlr.org/papers/v23/21-0998.html`.

Finn, Chelsea and Levine, Sergey. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2786–2793. IEEE, 2017.

Flammarion, Nicolas and Bach, Francis. From averaging to acceleration, there is only a step-size. In *Conference on Learning Theory*, pp. 658–695, 2015.

Florensa, Carlos, Duan, Yan, and Abbeel, Pieter. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

George, Thomas, Laurent, César, Bouthillier, Xavier, Ballas, Nicolas, and Vincent, Pascal. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, pp. 9550–9560, 2018.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Networks. In *NIPS'2014*, 2014.

Gopnik, Alison, Meltzoff, Andrew N, and Kuhl, Patricia K. *The scientist in the crib: Minds, brains, and how children learn*. William Morrow & Co, 1999.

Gordon, Neil J, Salmond, David J, and Smith, Adrian FM. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, volume 140, pp. 107–113. IET, 1993.

Grathwohl, Will, Choi, Dami, Wu, Yuhuai, Roeder, Geoff, and Duvenaud, David. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Greensmith, Evan, Bartlett, Peter L, and Baxter, Jonathan. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Gregor, K., Jimenez Rezende, D., and Wierstra, D. Variational Intrinsic Control. *InProceedings of the Interna-tional Conference on Learning Representations (ICLR)*, November 2017.

Gregor, Karol, Rezende, Danilo Jimenez, and Wierstra, Daan. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E, and Levine, Sergey. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hanna, Josiah P, Thomas, Philip S, Stone, Peter, and Niekum, Scott. Data-efficient policy evaluation through behavior policy search. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1394–1403. JMLR. org, 2017.

Hanna, Josiah P, Niekum, Scott, and Stone, Peter. Importance sampling policy evaluation with an estimated behavior policy. *arXiv preprint arXiv:1806.01347*, 2018.

Harper, Marc and Weinstein, Bryan. python-ternary: Ternary plots in python. *Zenodo 10.5281/zenodo.594435*, 2015. doi: 10.5281/zenodo.594435. URL `https://github.com/marcharper/python-ternary`.

Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Hennes, Daniel, Morrill, Dustin, Omidshafiei, Shayegan, Munos, Rémi, Perolat, Julien, Lanctot, Marc, Gruslys, Audrunas, Lespiau, Jean-Baptiste, Parmas, Paavo, Duéñez-Guzmán, Edgar, et al. Neural replicator dynamics: Multiagent learning via hedging policy gradients. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 492–501, 2020.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Hofmann, Thomas, Lucchi, Aurelien, Lacoste-Julien, Simon, and McWilliams, Brian. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pp. 2305–2313, 2015.

Hyvarinen, Aapo and Morioka, Hiroshi. Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA. In *NIPS*, 2016.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine*

Learning, pp. 448–456, 2015.

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z, Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Jastrzębski, Stanisław, Kenton, Zachary, Arpit, Devansh, Ballas, Nicolas, Fischer, Asja, Bengio, Yoshua, and Storkey, Amos. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Kahn, Herman and Harris, Theodore E. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

Kakade, Sham and Langford, John. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kalman, Rudolf Emil. When is a linear control system optimal? *Journal of Basic Engineering*, 86(1):51–60, 1964.

Kalman, Rudolf Emil et al. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

Kearns, Michael, Mansour, Yishay, and Ng, Andrew Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2-3): 193–208, 2002.

Keskar, Nitish Shirish, Mudigere, Dheevatsa, Nocedal, Jorge, Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations(ICLR)*, 2017.

Kim, Hyoungseok, Kim, Jaekyeom, Jeong, Yeonwoo, Levine, Sergey, and Song, Hyun Oh. Emi: Exploration with mutual information. In *International Conference on Machine Learning*, pp. 3360–3369, 2019.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Durk P. and Welling, Max. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Kitagawa, Genshiro. The two-filter formula for smoothing and an implementation of the gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623, 1994.

Kitagawa, Genshiro. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

Kolmogorov, Andreĭ Nikolaevich and Bharucha-Reid, Albert T. *Foundations of the theory of probability: Second English Edition*. Courier Dover Publications, 2018.

Kullback, Solomon. *Information theory and statistics*. Courier Corporation, 1997.

Kullback, Solomon and Leibler, Richard A. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Kunstner, Frederik, Balles, Lukas, and Hennig, Philipp. Limitations of the empirical fisher approximation. *arXiv preprint arXiv:1905.12558*, 2019.

Lattimore, Tor and Szepesvári, Csaba. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

Le Roux, Nicolas, Bengio, Yoshua, and Fitzgibbon, Andrew. Improving first and second-order methods by modeling uncertainty. *Optimization for Machine Learning*, pp. 403, 2011.

LeCun, Yann. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Levine, Sergey. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Levine, Sergey and Koltun, Vladlen. Variational policy search via trajectory optimization. In *Advances in Neural Information Processing Systems*, pp. 207–215, 2013.

Levine, Sergey, Pastor, Peter, Krizhevsky, Alex, Ibarz, Julian, and Quillen, Deirdre. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

Li, Yingzhen and Mandt, Stephan. A deep generative model for disentangled representations of sequential data. *CoRR*, abs/1803.02991, 2018. URL `http://arxiv.org/abs/1803.02991`.

Liang, Tengyuan, Poggio, Tomaso, Rakhlin, Alexander, and Stokes, James. Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Lindsten, Fredrik, Schön, Thomas B, et al. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013.

Lindsten, Fredrik, Jordan, Michael I, and Schön, Thomas B. Particle gibbs with ancestor sampling. *The Journal of Machine Learning Research*, 15(1):2145–2184, 2014.

Lioutas, Vasileios, Lavington, Jonathan Wilder, Sefas, Justice, Niedoba, Matthew, Liu, Yunpeng, Zwartsenberg, Berend, Dabiri, Setareh, Wood, Frank, and Scibior, Adam. Critic sequential monte carlo. *arXiv preprint arXiv:2205.15460*, 2022.

Liu, Hao, Feng, Yihao, Mao, Yi, Zhou, Dengyong, Peng, Jian, and Liu, Qiang. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

Liu, Jun S. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

Maddison, Chris J, Lawson, Dieterich, Tucker, George, Heess, Nicolas, Doucet, Arnaud, Mnih, Andriy, and Teh, Yee Whye. Particle value functions. *arXiv preprint arXiv:1703.05820*, 2017.

Mandt, Stephan, Hoffman, Matthew D, and Blei, David M. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1): 4873–4907, 2017.

Martens, James. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Massey, James. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, pp. 303–305, 1990.

Mei, Jincheng, Xiao, Chenjun, Dai, Bo, Li, Lihong, Szepesvári, Csaba, and Schuurmans, Dale. Escaping the gravitational pull of softmax. *Advances in Neural Information Processing Systems*, 33, 2020a.

Mei, Jincheng, Xiao, Chenjun, Szepesvari, Csaba, and Schuurmans, Dale. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pp. 6820–6829. PMLR, 2020b.

Mei, Jincheng, Dai, Bo, Xiao, Chenjun, Szepesvari, Csaba, and Schuurmans, Dale. Understanding the effect of stochasticity in policy optimization. *Advances in Neural Information Processing Systems*, 34:19339–19351, 2021.

Mei, Jincheng, Chung, Wesley, Thomas, Valentin, Dai, Bo, Szepesvari, Csaba, and Schuurmans, Dale. The role of baselines in policy gradient optimization. *Advances in Neural Information Processing Systems*, 35:17818–17830, 2022.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mohamed, Shakir and Rezende, Danilo Jimenez. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Munos, Rémi, Stepleton, Tom, Harutyunyan, Anna, and Bellemare, Marc G. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1046–1054, 2016.

Murata, Noboru, Yoshizawa, Shuji, and Amari, Shun-ichi. Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6):865–872, 1994.

Nachum, Ofir, Norouzi, Mohammad, Xu, Kelvin, and Schuurmans, Dale. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Nagabandi, Anusha, Kahn, Gregory, Fearing, Ronald S, and Levine, Sergey. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

Naganuma, Hiroki, Suzuki, Taiji, Yokota, Rio, Nomura, Masahiro, Ishikawa, Kohta, and Sato, Ikuro. Takeuchi's information criteria as generalization measures for DNNs close to NTK regime, 2022. URL https://openreview.net/forum?id=FH_mZOKFX-b.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems (NeurIPS)*, 2011.

Neu, Gergely. Explore no more: Improved high-probability regret bounds for non-stochastic bandits. *arXiv preprint arXiv:1506.03271*, 2015.

Neyshabur, Behnam, Bhojanapalli, Srinadh, McAllester, David, and Srebro, Nati. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Nota, Chris and Thomas, P. Is the policy gradient a gradient? *Adaptive Agents And Multi-agent Systems*, 2019.

Novak, Roman, Bahri, Yasaman, Abolafia, Daniel A, Pennington, Jeffrey, and Sohl-Dickstein, Jascha. Sensitivity and generalization in neural networks: an empirical study. *International Conference on Learning Representations (ICLR)*, 2018.

O'Donoghue, Brendan, Munos, Remi, Kavukcuoglu, Koray, and Mnih, Volodymyr. Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

Oudeyer, Pierre-Yves and Kaplan, Frederic. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Parmas, Paavo and Sugiyama, Masashi. A unified view of likelihood ratio and reparameterization gradients and an optimal importance sampling scheme. *arXiv preprint arXiv:1910.06419*, 2019.

Peters, Jan and Schaal, Stefan. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Pitt, Michael K and Shephard, Neil. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

Pong, Vitchyr. rlkit. https://github.com/vitchyr/rlkit/, 2018.

Precup, Doina. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000a.

Precup, Doina. Temporal abstraction in reinforcement learning. 2000b.

Puterman, Martin L. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Rame, Alexandre, Dancette, Corentin, and Cord, Matthieu. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*, pp. 18347–18377. PMLR, 2022.

Rangamani, Akshay, Nguyen, Nam H, Kumar, Abhishek, Phan, Dzung, Chin, Sang H, and Tran, Trac D. A scale invariant flatness measure for deep network minima. *arXiv*

preprint arXiv:1902.02434, 2019.

Rawlik, Konrad, Toussaint, Marc, and Vijayakumar, Sethu. An approximate inference approach to temporal optimization in optimal control. In *Advances in neural information processing systems*, pp. 2011–2019, 2010.

Rawlik, Konrad, Toussaint, Marc, and Vijayakumar, Sethu. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: science and systems*, volume 13, pp. 3052–3056, 2012.

Robbins, Herbert. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Rubinstein, RY and Kroese, DP. A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning. *Springer-Verlag New York, LLC*, 2004.

Ruderman, Avraham, Reid, Mark, García-García, Darío, and Petterson, James. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv preprint arXiv:1206.4664*, 2012.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 1986. doi: 10.1038/323533a0. URL `https://doi.org/10.1038/323533a0`.

Sagun, Levent, Bottou, Léon, and LeCun, Yann. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Salge, Christoph, Glackin, Cornelius, and Polani, Daniel. Empowerment - an introduction. *CoRR*, abs/1310.1863, 2013. URL `http://arxiv.org/abs/1310.1863`.

Schaul, Tom, Borsa, Diana, Modayil, Joseph, and Pascanu, Razvan. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.

Schmidt, Mark. Convergence rate of stochastic gradient with constant step size. Technical report, UBC, 2014.

Schneider, Frank, Dangel, Felix, and Hennig, Philipp. Cockpit: A practical debugging tool for the training of deep neural networks. *Advances in Neural Information Processing Systems*, 34:20825–20837, 2021.

Schulman, John, Chen, Xi, and Abbeel, Pieter. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Schultz, Wolfram, Dayan, Peter, and Montague, P Read. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

Seldin, Yevgeny, Szepesvári, Csaba, Auer, Peter, and Abbasi-Yadkori, Yasin. Evaluation and analysis of the performance of the exp3 algorithm in stochastic environments. In *EWRL*, pp. 103–116, 2012.

Seldin, Yevgeny, Szepesvári, Csaba, Auer, Peter, and Abbasi-Yadkori, Yasin. Evaluation and analysis of the performance of the exp3 algorithm in stochastic environments. In *European Workshop on Reinforcement Learning*, pp. 103–116. PMLR, 2013.

Shannon, Claude Elwood. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Stewart, Leland and McCarty, Perry. Use of bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment. In *Signal Processing, Sensor Fusion, and Target Recognition*, volume 1699, pp. 177–186. International Society for Optics and Photonics, 1992.

Still, Susanne and Precup, Doina. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Strouse, DJ, Baumli, Kate, Warde-Farley, David, Mnih, Vlad, and Hansen, Steven. Learning more skills through optimistic exploration. *arXiv preprint arXiv:2107.14226*, 2021.

Sukhbaatar, Sainbayar, Szlam, Arthur, Synnaeve, Gabriel, Chintala, Soumith, and Fergus, Rob. MazeBase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*, 2015.

Sutton, Richard S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, Richard S and Barto, Andrew G. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review*, 88(2):135, 1981.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Sutton, Richard S, McAllester, David A, Singh, Satinder P, Mansour, Yishay, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pp. 1057–1063, 1999.

Szita, István and Lörincz, András. Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941, 2006.

Takeuchi, Kei. The distribution of information statistics and the criterion of goodness of fit of models. *Mathematical Science*, 153:12–18, 1976.

Tassa, Yuval, Erez, Tom, and Todorov, Emanuel. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4906–4913. IEEE, 2012.

Tesauro, Gerald. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

Thomas, Valentin. On the role of overparameterization in off-policy temporal difference learning with linear function approximation. *Advances in Neural Information Processing Systems*, 35:37228–37240, 2022.

Thomas, Valentin, Pondard, Jules, Bengio, Emmanuel, Sarfati, Marc, Beaudoin, Philippe, Meurs, Marie-Jean, Pineau, Joelle, Precup, Doina, and Bengio, Yoshua. Independently controllable features. *arXiv preprint arXiv:1708.01289*, 2017.

Thomas, Valentin, Bengio, Emmanuel, Fedus, William, Pondard, Jules, Beaudoin, Philippe, Larochelle, Hugo, Pineau, Joelle, Precup, Doina, and Bengio, Yoshua. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018.

Todorov, Emanuel and Li, Weiwei. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pp. 300–306. IEEE, 2005.

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Toussaint, Marc. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056. ACM, 2009.

Toussaint, Marc and Storkey, Amos. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pp. 945–952. ACM, 2006.

van Hasselt, Hado, Doron, Yotam, Strub, Florian, Hessel, Matteo, Sonnerat, Nicolas, and Modayil, Joseph. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018.

Wang, Shuaiwen, Zhou, Wenda, Lu, Haihao, Maleki, Arian, and Mirrokni, Vahab. Approximate leave-one-out for fast parameter tuning in high dimensions. *arXiv preprint arXiv:1807.02694*, 2018.

Wang, Yunbo, Liu, Bo, Wu, Jiajun, Zhu, Yuke, Du, Simon S, Fei-Fei, Li, and Tenenbaum, Joshua B. Dual sequential monte carlo: Tunneling filtering and planning in continuous pomdps. *arXiv preprint arXiv:1909.13003*, 2019.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wu, Cathy, Rajeswaran, Aravind, Duan, Yan, Kumar, Vikash, Bayen, Alexandre M, Kakade, Sham, Mordatch, Igor, and Abbeel, Pieter. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

Xu, Pan, Gao, Felicia, and Gu, Quanquan. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*, 2019.

Yaida, Sho. Fluctuation-dissipation relations for stochastic gradient descent. *arXiv preprint arXiv:1810.00004*, 2018.

Zhu, Zhanxing, Wu, Jingfeng, Yu, Bing, Wu, Lei, and Ma, Jinwen. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018.

Ziebart, Brian D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, CMU, 2010.

Appendix A

Appendix A

A.1. Additional details

A.1.1. Architecture

Our architecture is as follows: the encoder, mapping the raw pixel state to a latent representation, is a 4-layer convolutional neural network with batch normalization (Ioffe & Szegedy, 2015) and leaky ReLU activations. The decoder uses the transposed architecture with ReLU activations. The noise z is sampled from a 2-dimensional gaussian distribution and both the generator $\Phi(h,z)$ and the policy $\pi(h,\phi)$ are neural networks consisting of 2 fully-connected layers. In practice, a minibatch of $n = 256$ or 1024 vectors $\phi_1,\ldots,\phi_n$ is sampled at each step. The agent randomly choses one $\phi = \phi_{behavior}$ and samples actions from its policy $a \sim \pi(h,\phi_{behavior})$. Our model parameters are then updated using policy gradient with the REINFORCE estimator and a state-dependent baseline and importance sampling. For each selectivity reward, the term $\mathbb{E}_{\phi'}[A(h',h,\phi')]$ is estimated as $\frac{1}{n}\sum_{i=1}^{n} A(h',h,\phi_i)$.

In practice, we don't use concatenation of vectors when feeding two vectors as input for a network (like (h,z) for the factor generator or (h,ϕ) for the policy). For vectors $a,b \in \mathbb{R}^{n_a \times n_b}$. We use a bilinear operation $bil(a,b) = (a_i * b_j)_{i\in[[n_a]],j\in[[n_b]]}$ as in Florensa et al. (2017). We observe the bilinear integrated input to more strongly enforce dependence on both vectors; in contrast, our models often ignored one input when using a simple concatenation.

Through our research, we experiment with different outputs for our generator $\Phi(h,z)$. We explored embedding the ϕ-vectors into a hypercube, a hypersphere, a simplex and also a simplex multiplied by the output of a $tanh(\cdot)$ operation on a scalar.

A.1.2. First experiment

In the first experiment, figure 2, we used a gaussian similarity kernel i.e $A(h', h, \phi) = \exp(-\frac{||h'-(h+\phi)||^2}{2\sigma^2})$ with $\sigma = \sqrt{dim(h)}$. In this experiment only, for clarity of the figure, we only allowed permissible actions in the environment (no no-op action).

A.2. Additional Figures

A.2.1. Discrete simple case

Here we consider the case where we learn a latent space H of size K, with K factors corresponding to the coordinates of h (h_i, $i \in [k]$), and learn K separately parameterized policies $\pi_i(a|h)$, $i \in [k]$. We train our model with the selectivity objective, but no autoencoder loss, and find that we correctly recover independently controllable features on a simple environment. Albeit slower than when jointly training an autoencoder, this shows that the objective we propose is strong enough to provide a learning signal for discovering a disentangled latent representation.

We train such a model on a gridworld MNIST environment, where there are two MNIST digits . The two digits can be moved on the grid via 4 directional actions (so there are 8 actions total), the first digit is always odd and the second digit always even, so they are distiguishable. In Figure 1 we plot each latent feature h_k as a curve, as a function of each ground truth. For example we see that the black feature recovers $+x_1$, the horizontal position of the first digit, or that the purple feature recovers $-y_2$, the vertical position of the second digit.

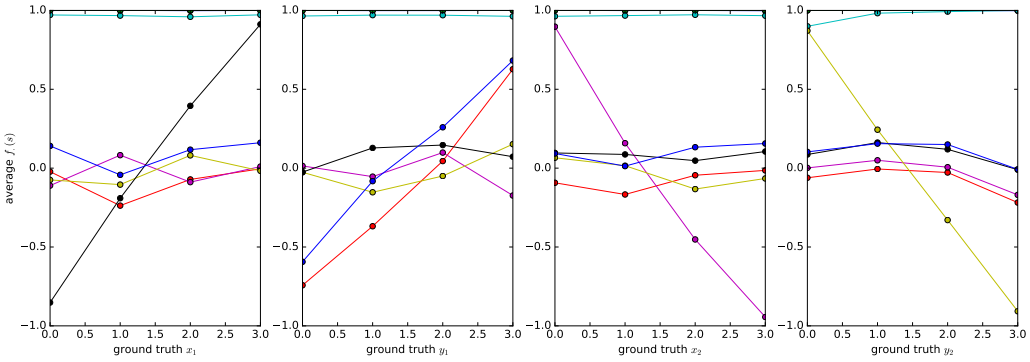


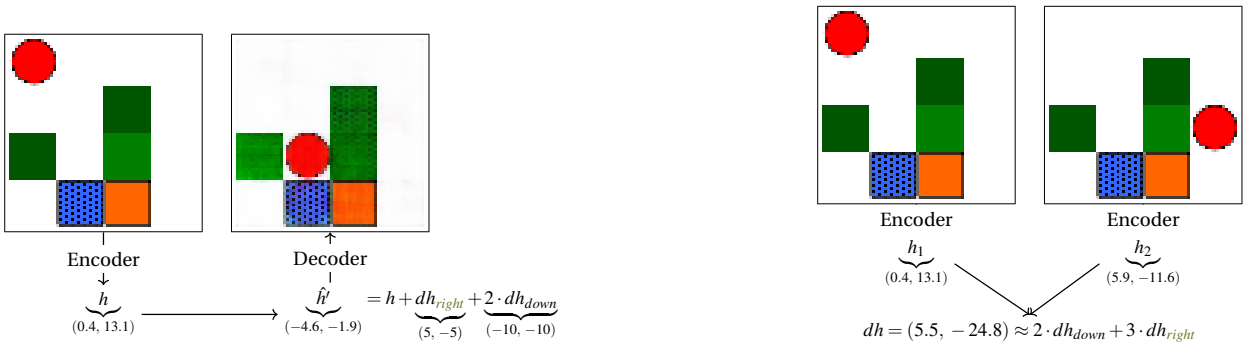
Fig. 1. In a gridworld environment with 2 objects (in this case 2 MNIST digits), we know there are 4 underlying features, the (x_i, y_i) position of each digit i. Here each of the four plots represents the evolution of the f_k's as a function of their underlying feature, from left to right x_1, y_1, x_2, y_2. We see that for each of them, at least one f_k recovers it almost linearly, from the raw pixels only.

A.2.2. Planning and policy inference example in 1-step

This disentangled structure could be used to address many challenging issues in reinforcement learning. We give two examples in figure 2:

- Model-based predictions: Given an initial state, s_0, and an action sequence $a_{\{0:T-1\}}$, we want to predict the resulting state s_T.
- A simplified deterministic policy inference problem: Given an initial state s_{start} and a terminal state s_{goal}, we aim to find a suitable action sequence $a_{\{0:T-1\}}$ such that s_{goal} can be reached from s_{start} by following it.

Because of the *tanh* activation on the last layer of $\Phi(h,z)$, the different factors of variation $dh = h' - h$ are placed on the vertices of a hypercube of dimension K, and we can think of the the policy inference problem as finding a path in that simpler space, where the starting point is h_{start} and the goal is h_{goal}. We believe this could prove to be a much easier problem to solve.



(a) **(b)**

Fig. 2. (a) Predicting the effect of a cause on Mazebase. The leftmost image is the visual input of the environment, where the agent is the round circle, and the switch states are represented by shades of green. After the training, we are able to distinguish one cluster per dh (Figure 2), that is to say per variation obtained after performing an action, independently from the position h. Therefore, we are able to move the agent just by adding the corresponding dh to our latent representation h. The second image is just the reconstruction obtained by feeding the resulting h' into the decoder. (b) Given a starting state and a goal state, we are able to decompose the difference of the two representations dh into a (non-directed) sequence of movements.

A.2.3. Multistep Example

We demonstrate an instance of ICF operating in a 4×4 Mazebase enviroment over five time steps in Figure 3. We consistently witness a failure of mode collapse in our generator Φ and therefore the generator only produces a subset of all possible ϕ-variations. In

Figure 3, we observe the ϕ governing the agent's policy π_ϕ appears to correspond to moving two positions down and then to repeatedly toggle the switch. A random action due to ε-greedy led to the agent moving up and off the switch at time step-4. This perturbation is corrected by the policy π_ϕ by moving down in order to return to toggling the relevant switch.

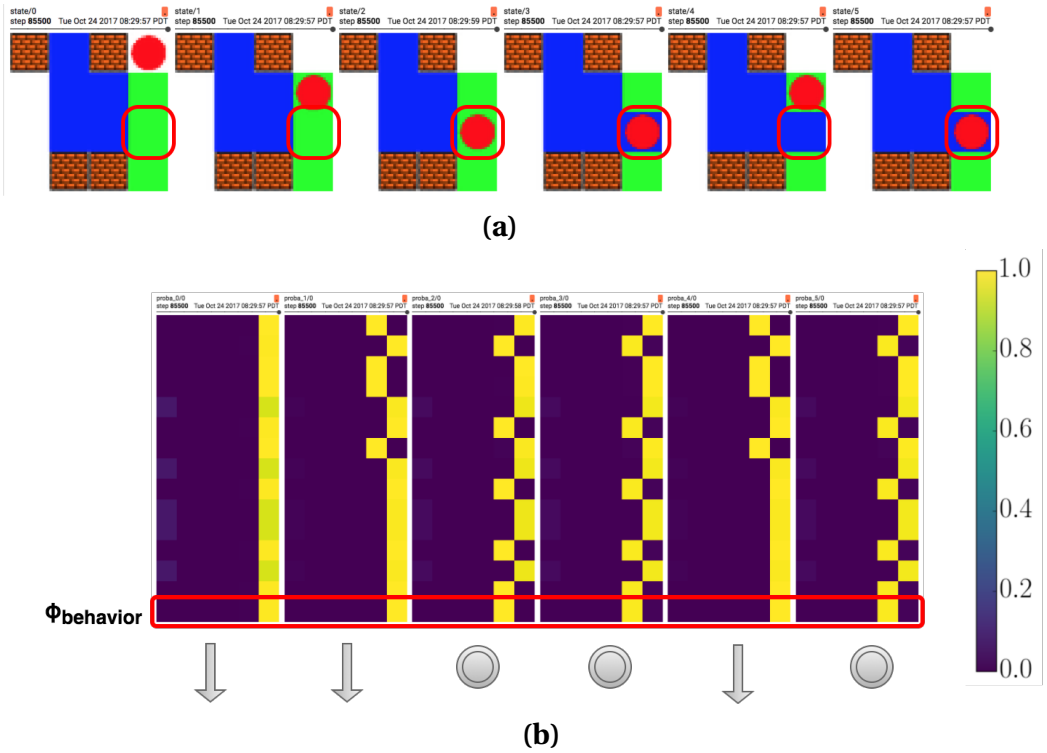


Fig. 3. (a) Mazebase environment over five time-steps. Here the red dot denotes the position of the agent. The $\phi_{behavior}$ governing the agent's policy appears to control toggling the switch indicated by the red rounded box. (b) Visualization of the policies instantiated by different ϕs. Each box represents the probability distribution of the policies at that time step. Each row is generated by a different ϕ and each column corresponds to an action (up, left, pass, right, toggle, down) in order. The boxed column shows the $\phi_{behavior}$. The symbols below each box represent the most-probable action for the behavioral policy, where the grey circle indicates toggling the switch.

A.3. Variational bound and the selectivity

Let us call $p(h_{t+1}|\phi_{t+1},h_t) = \mathcal{P}^{\phi}_{h',h}$ the probability distribution over final hidden states starting from h and using the policy parametrized by the embedding ϕ.

$p(h_{t+1}|\phi_{t+1},h_t) = \Pi_{k=1}^{K} \pi_{\phi_{t+1}}(a_{t+\frac{k-1}{K}}|h_{t+\frac{k-1}{K}}) p_{env}(s_{t+\frac{k}{K}}|a_{t+\frac{k-1}{K}}, s_{t+\frac{k-1}{K}})$. where p_{env} is the transition probability of the environment.

For simplicity, let's refer to h_t as h, h_{t+1} as h' and ϕ_{t+1} as ϕ.

A.3.1. Lower bound on the mutual information

The bound

$$\mathcal{I}_p(\phi, h'|h) \geq \sup_{\theta} \mathbb{E}_{p(\phi|h)}\big[\mathcal{S}(h,\phi)\big]$$

can be proven by using Donsker-Varadhan variational representation of the KL divergence (Donsker & Varadhan, 1975; Ruderman et al., 2012):

$$\mathcal{D}_{\mathrm{KL}}\big(p||q\big) = \sup_{T \in \mathcal{L}^{\infty}(q)} \mathbb{E}_p\big[T\big] - \log \mathbb{E}_q\big[e^T\big]$$

For $A = e^T$ and using the identity $\mathcal{I}_p(X,Y) = \mathbb{E}_{p(y)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(x|y)||p(x)\big)\big]$ with $X = \phi|h$ and $Y = h'|h$, we have:

$$\begin{aligned}
\mathcal{I}_p(\phi, h'|h) &= \mathbb{E}_{h'|h} \sup_{A} \mathbb{E}_{\phi|h,h'}\big[\log A(h',h,\phi)\big] - \log \mathbb{E}_{\varphi|h}\big[A(h',h,\varphi)\big] \\
&= \mathbb{E}_{h'|h} \sup_{A} \mathbb{E}_{\phi|h,h'}\big[\log \frac{A(h',h,\phi)}{\mathbb{E}_{\varphi|h}\big[A(h',h,\varphi)\big]}\big] \\
&\geq \sup_{A} \mathbb{E}_{\phi|h} \mathbb{E}_{h'|\phi,h}\big[\log \frac{A(h',h,\phi)}{\mathbb{E}_{\varphi|h}\big[A(h',h,\varphi)\big]}\big] \\
&\geq \sup_{\theta} \mathbb{E}_{\phi|h} \mathbb{E}_{h'|\phi,h}\big[\log \frac{A(h',h,\phi;\theta)}{\mathbb{E}_{\varphi|h}\big[A(h',h,\varphi;\theta)\big]}\big]
\end{aligned}$$

for parametric A functions.

As we sample the factors ϕ uniformly, our total objective is then a lower bound on $\sum_t \mathcal{I}(\phi_t, h_t|h_{t-1})$ which corresponds here to the *directed information* (Massey, 1990) Ziebart (2010) as ϕ_t is sampled independently from $\phi_{1:t-1}$.

A.4. Additional information on the training

In our experiments, we use the selectivity objective, an autoencoding loss and an entropy regularization loss $\mathcal{H}(\pi_\phi)$ for each of the policies π_ϕ. Furthermore, in experiment 4.2 we added the model-based cost $||h' - T(h,\phi)||^2$ with T a learned two layer fully connected neural network.

The selectivity is used to update the parameters of the encoder, factor generator and policy networks. We use the following equation for computing the gradients

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\big[f_\theta\big] = \mathbb{E}_{\pi_\theta}\big[\nabla_\theta f_\theta + f_\theta \nabla_\theta \log \pi_\theta\big]$$

We also use a state dependent baseline V as a control variate to reduce the variance of the REINFORCE estimator.

Furthermore, to be able to train the factor generator efficiently, we train all ϕ sampled in a mini-batch (of size 1024) by importance sampling on the probability ratio of the trajectory under each ϕ

Appendix B

SMCP appendix

B.1. Appendix

B.1.1. Abbreviation and Notation

Table 1. Abbreviation

SMCP:	Sequential Monte Carlo Planning.
SAC:	Soft Actor Critic.
CEM:	Cross Entropy Method.
RS:	Random Shooting.
MCTS:	Monte Carlo Tree Search.
SMC:	Sequential Monte Carlo.
SIR:	Sequential Importance Resampling.
SIS:	Sequential Importance Sampling.
IS:	Importance Sampling.
MPC:	Model Predictive Control

Table 2. Notation

$\tau_{1:T}$	$\triangleq$	$\{\mathbf{s}_i, \mathbf{a}_i\}_{i=1}^T$ the state-action pairs.
V	$\triangleq$	value function.
$\mathcal{O}_t$	$\triangleq$	Optimality variable.
$p(\mathcal{O}_t \vert \mathbf{s}_t, \mathbf{a}_t)$	$\triangleq$	$\exp(r(\mathbf{s}_t, \mathbf{a}_t))$ Probability of a pair state action of being optimal.
p_{env}	$\triangleq$	Transition probability of the environment. Takes state and action $(\mathbf{s}_t, \mathbf{a}_t)$ as argument and return next state and reward $(\mathbf{s}_{t+1}, r_t)$.
p_{model}	$\triangleq$	Model of the environment. Takes state and action $(\mathbf{s}_t, \mathbf{a}_t)$ as argument and return next state and reward $(\mathbf{s}_{t+1}, r_t)$.
w_t	$\triangleq$	Importance sampling weight.
$p(\tau)$	$\triangleq$	Density of interest.
$q(\tau)$	$\triangleq$	Approximation of the density of interest.
$t \in \{1, \dots T\}$	$\triangleq$	time steps.
$n \in \{1, \dots N\}$	$\triangleq$	particle number.
h	$\triangleq$	horizon length.

B.1.2. The action prior

The true joint distribution 4.2.1 in section 4.2.1 should actually be written:

$$\begin{aligned} p(\tau_{1:T}, \mathcal{O}_{1:T}) &= \mu(\mathbf{s}_1) \prod_{t=1}^{T-1} p_{\text{env}}(\mathbf{s}_{t+1} \vert \mathbf{a}_t, \mathbf{s}_t) \prod_{t=1}^{T} p(\mathbf{a}_t) \exp\big(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\big) \\ &= \mu(\mathbf{s}_1) \prod_{t=1}^{T-1} p_{\text{env}}(\mathbf{s}_{t+1} \vert \mathbf{a}_t, \mathbf{s}_t) \exp\big(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) + \log p(\mathbf{a}_t)\big) \end{aligned}$$

In Mujoco environments, the reward is typically written as

$$r(\mathbf{s}_t, \mathbf{a}_t) = f(\mathbf{s}_t) - \alpha ||\mathbf{a}_t||_2^2$$

where f is a function of the state (velocity for HalfCheetah on Mujoco for example). The part $\alpha ||\mathbf{a}_t||_2^2$ can be seen as the contribution from the action prior (here a gaussian prior). One can also consider the prior to be constant (and potentially improper) so that is does not change the posterior $p(\tau_{1:T} \vert \mathcal{O}_{1:T})$.

B.1.3. Value function: backward message

$$
\begin{aligned}
p(\mathcal{O}_{t+1:T}|\tau_t) &= \int_{\tau_{t+1}} p(\mathcal{O}_{t+1:T}, \tau_{t+1}|\tau_t) d\tau_{t+1} \\
&= \int_{\tau_{t+1}} p(\tau_{t+1}|\tau_t, \mathcal{O}_{t+1:T}) p(\mathcal{O}_{t+1:T}|\tau_{t+1}) d\tau_{t+1} \\
&= \int_{\mathbf{s}_{t+1}} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \left[\int_{\mathbf{a}_{t+1}} p(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}, \mathcal{O}_{t+1:T}) \exp Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) d\mathbf{a}_{t+1} \right] d\mathbf{s}_{t+1} \\
&= \int_{\mathbf{s}_{t+1}} p_{\text{env}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \exp\left(V(\mathbf{s}_{t+1})\right) d\mathbf{s}_{t+1} \\
&= \mathbb{E}_{\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t}[\exp\left(V(\mathbf{s}_{t+1})\right)] \qquad \text{(B.1.1)}
\end{aligned}
$$

By definition of the optimal value function in (Levine, 2018).

B.1.4. Recursive weights update

$$
\begin{aligned}
w_t &= \frac{p(\tau_{1:t}|\mathcal{O}_{1:T})}{q(\tau_{1:t})} \\
&= \frac{p(\tau_{1:t-1}|\mathcal{O}_{1:T})}{q(\tau_{1:t-1})} \frac{p(\tau_t|\tau_{1:t-1}, \mathcal{O}_{1:T})}{q(\tau_t|\tau_{1:t-1})} \\
&= w_{t-1} \cdot \frac{p(\tau_t|\tau_{1:t-1}, \mathcal{O}_{1:T})}{q(\tau_t|\tau_{1:t-1})} \\
&= w_{t-1} \frac{1}{q(\tau_t|\tau_{1:t-1})} \frac{p(\tau_{1:t}|\mathcal{O}_{1:T})}{p(\tau_{1:t-1}|\mathcal{O}_{1:T})}
\end{aligned}
$$

We use there the forward-backward equation 4.3.1 for the numerator and the denominator

$$
\begin{aligned}
&\propto w_{t-1} \frac{1}{q(\tau_t|\tau_{1:t-1})} \frac{p(\tau_{1:t}|\mathcal{O}_{1:t})}{p(\tau_{1:t-1}|\mathcal{O}_{1:t-1})} \frac{p(\mathcal{O}_{t+1:T}|\tau_t)}{p(\mathcal{O}_{t:T}|\tau_{t-1})} \\
&= w_{t-1} \frac{p(\tau_t|\tau_{1:t-1})}{q(\tau_t|\tau_{1:t-1})} p(\mathcal{O}_t|\tau_t) \frac{p(\mathcal{O}_{t+1:T}|\tau_t)}{p(\mathcal{O}_{t:T}|\tau_{t-1})} \\
&= w_{t-1} \frac{p_{\text{env}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})}{p_{\text{model}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})} \frac{\exp(r_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \frac{\mathbb{E}_{\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t}[\exp\left(V(\mathbf{s}_{t+1})\right)]}{\mathbb{E}_{\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}}[\exp\left(V(\mathbf{s}_t)\right)]} \\
&= w_{t-1} \frac{p_{\text{env}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})}{p_{\text{model}}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})} \mathbb{E}_{\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t}[\exp\left(r_t - \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + V(\mathbf{s}_{t+1}) - \log \mathbb{E}_{\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}}[\exp\left(V(\mathbf{s}_t)\right)]\right)] \qquad \text{(B.1.2)}
\end{aligned}
$$

B.1.5. Experiment Details

Random samples: 1000 transitions are initially collected by a random policy to pretrain the model and the proposal distribution. After which the agents start following their respective policy.
Data preprocessing: We normalize the observations to have zero mean and standard deviation 1.
Model Predictive Control: The model is used to predict the planning distribution for the horizon h of N particles. We then sample a trajectory according to its weight and return the first action of this trajectory. In our experiments, we fix the maximum number of particles for every method to 2500. For SMCP, the temperature and horizon length are described in Table 3.
Soft Actor Critic: We used a custom implementation with a Gaussian policy for both the SAC baseline and the proposal distribution used for both versions of SMCP. We used Adam (Kingma & Ba, 2014) with a learning rate of 0.001. The reward scaling suggested by Haarnoja et al. (2018) for all experiments and used an implementation inspired by Pong (2018). We used a two hidden layers with 256 hidden units for the three networks: the value function, the policy and the soft Q functions.
Model: We train the model p_{model} to minimize the negative log likelihood of $p(s_{t+1}|s_t + \Delta_t(s_t,a_t), \sigma_t(s_t,a_t))$. The exact architectures are detailed in Table 3. We train the model to predict the distribution of the change in states and learn a deterministic reward function from the current state and predict the change in state. Additionally, we manually add a penalty on the action magnitude in the reward function to simplify the learning. At the end of each episode we train the model for 10 epochs. Since the training is fairly short, we stored every transitions into the buffer. The model is defined as:

$$\Delta s_t \sim p(\cdot|s_t,a_t) \tag{B.1.3}$$

$$r_t = g(s_t,\Delta s_t) - \alpha\|a\|^2 \tag{B.1.4}$$

where α was taken from the Mujoco gym environments. We used Adam (Kingma & Ba, 2014) with a learning rate of 0.001 and leaky ReLU activation function.

Environment	Temperature	Horizon length	Number of Dense Layers	Layer Dimension
Hopper-v2	1	10	3	256
Walker2d-v2	10	20	3	256
HalfCheetah-v2	10	20	3	256

Table 3. Hyperparameters for the experiments.

B.1.6. Sequential Importance Sampling Planning

Algorithm 6 SMC Planning using SIS

for t in $\{1,\dots,T\}$ **do**
  $\{\mathbf{s}_t^{(n)} = \mathbf{s}_t\}_{n=1}^N$
  $\{w_t^{(n)} = 1\}_{n=1}^N$
  for i in $\{t,\dots,t+h\}$ **do**
    // Update
    $\{\mathbf{a}_i^{(n)} \sim \pi(\mathbf{a}_i^{(n)}|\mathbf{s}_i^{(n)})\}_{n=1}^N$
    $\{\mathbf{s}_{i+1}^{(n)}, r_i^{(n)} \sim p_{\text{model}}(\cdot|\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)})\}_{n=1}^N$
    $\{w_i^{(n)} \propto w_{i-1}^{(n)} \cdot \exp\left(A(\mathbf{s}_i^{(n)}, \mathbf{a}_i^{(n)}, \mathbf{s}_{i+1}^{(n)})\right)\}_{n=1}^N$
  end for
  Sample $n \sim \text{Categorical}(w_{t+h}^{(1)}, \dots, w_{t+h}^{(N)})$.
  // Model Predictive Control
  Select $\mathbf{a}_t$, first action of $\tau_{t:t+h}^{(n)}$
  $\mathbf{s}_{t+1}, r_t \sim p_{\text{env}}(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
  Add $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to buffer $\mathcal{B}$
  Update π, V and p_{model} with $\mathcal{B}$
end for

B.1.7. Significance of the results

The significance of our results is done following guidelines from Colas et al. (2018). We test the hypothesis that the mean return of our method is superior to the one of SAC. We use 20 random seeds (from 0 to 19pro) for each method on each environment.

For this we look at the average return from steps 150k to 250k for SIR-SAC and SAC, and conduct a Welch's t-test with unknown variance. We report the p-value for each environment tested on Mujoco. A $p_{\text{val}} < 0.05$ usually indicates that there is strong evidence to suggest that our method outperforms SAC.

- `HalfCheetah-v2`: $p_{\text{val}} = 0.003$. There is very compelling evidence suggesting we outperform SAC.
- `Hopper-v2`: $p_{\text{val}} = 0.09$. There is no significant evidence suggesting we outperform SAC.
- `Walker2d-v2`: $p_{\text{val}} = 0.03$. There is compelling evidence suggesting we outperform SAC.

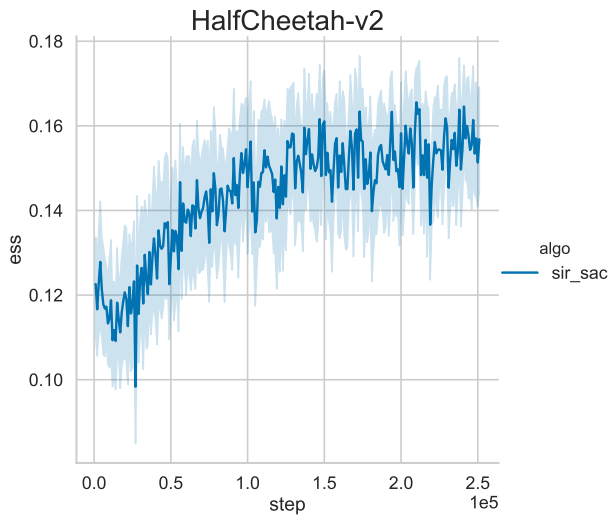


Fig. 1. Effective sample size for HalfCheetah. The shaded area represents the standard deviation over 20 seeds.

B.1.8. Additional experimental results

B.1.8.1. Effective Sample Size

The values reported on Figure 1 are the harmonic mean of the ratio of the effective sample size by the actual number of particles.

More precisely the values are

$$y_t = \left(\prod_{i=1}^{h} \mathrm{ESS}_i(t)/N\right)^{1/h}$$

where i is the depth of the planning, N is the number of particles and

$$\mathrm{ESS}_i(t) = \frac{(\sum_{n=1}^{N} w_{t+i}^{(n)})^2}{\sum_{n=1}^{N} (w_{t+i}^{(n)})^2}$$

We can see that as the proposal distribution improves the ESS also increases. The ESS on HalfCheetah is representative of the one obtained on the other environments. While these values are not high, we are still around 15% thus we do not suffer heavily from weight degeneracy.

B.1.8.2. Model loss

We also report the negative log likelihood loss of the environment's model during the training on Figure 2.

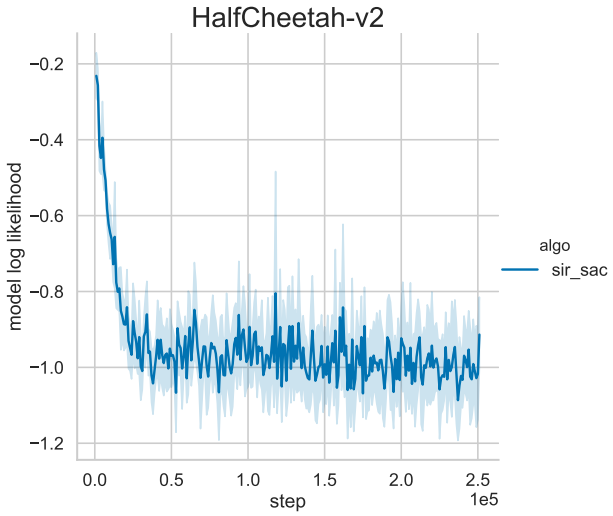


Fig. 2. Negative log likelihood for the model on HalfCheetah. The shaded area represents the standard deviation over 20 seeds.

Appendix C

Appendix HFC

C.1. Proofs

C.1.1. Bounds between H, F and C

C.1.1.1. Bounds with backward χ^2 divergence

$$
\begin{aligned}
|\mathbf{F}_{ij} - \mathbf{H}_{ij}|^2 &= |\int q_\theta(x,y)\big(\nabla^2_\theta \ell(x,y)\big)_{ij} d(x,y) - \int p(x,y)\big(\nabla^2_\theta \ell(x,y)\big)_{ij} d(x,y)|^2 \\
&= |\int \big(q_\theta(x,y) - p(x,y)\big)\big(\nabla^2_\theta \ell(x,y)\big)_{ij} d(x,y)|^2 \\
&= |\int \frac{\big(q_\theta(x,y) - p(x,y)\big)}{\sqrt{p(x,y)}}\big(\sqrt{p(x,y)}\nabla^2_\theta \ell(x,y)\big)_{ij} d(x,y)|^2 \\
&\leq \int \frac{\big(q_\theta(x,y) - p(x,y)\big)^2}{p(x,y)} d(x,y) \int p(x,y)\big(\nabla^2_\theta \ell(x,y)\big)^2_{ij} d(x,y) \\
&= \mathcal{D}_{\chi^2}(q_\theta||p)\, \mathbb{E}_p[\big(\nabla^2_\theta \ell(x,y)\big)^2_{ij}]
\end{aligned}
$$

Where we used Cauchy-Schwarz inequality and $\mathcal{D}_{\chi^2}$ denotes the χ^2 divergence.

$$
||\mathbf{F} - \mathbf{H}||^2 \leq \mathcal{D}_{\chi^2}(q_\theta||p)\, \mathbb{E}_p[||\mathbf{H}(x,y)||_2^2]
$$

Where $\mathbf{H}(x,y) \triangleq \nabla^2_\theta \ell(x,y)$ is the empirical hessian for one sample and the $||\cdot||_2$ is the Frobenius norm.

In the same way

$$
\begin{aligned}
|\mathbf{F}_{ij} - \mathbf{C}_{ij}|^2 &= |\int q_\theta(x,y)\big(\nabla_\theta \ell(x,y)\nabla_\theta \ell(x,y)^\top\big)_{ij} d(x,y) - \int p(x,y)\big(\nabla_\theta \ell(x,y)\nabla_\theta \ell(x,y)^\top\big)_{ij} d(x,y)|^2 \\
&\leq \mathcal{D}_{\chi^2}(q_\theta||p)\, \mathbb{E}_p[\big(\nabla_\theta \ell(x,y)\nabla_\theta \ell(x,y)^\top\big)^2_{ij}]
\end{aligned}
$$

For $\mathbf{C}(x,y) \triangleq \nabla_\theta \ell(x,y) \nabla_\theta \ell(x,y)^\top$ we have

$$||\mathbf{F}-\mathbf{C}||^2 \;\leq\; \mathcal{D}_{\chi^2}(q_\theta||p)\, \mathbb{E}_p[||\mathbf{C}(x,y)||^2]$$

Hence

$$||\mathbf{C}-\mathbf{H}||^2 \;\leq\; \mathcal{D}_{\chi^2}(q_\theta||p)\, \mathbb{E}_p[||\mathbf{C}(x,y)||^2 + ||\mathbf{H}(x,y)||^2]$$

C.1.1.2. Bounds with forward χ^2 divergence

Note that in the above proof, breaking the integral in two with Cauchy-Schwarz inequality could have been done using

$$\begin{aligned}
|\mathbf{F}_{ij}-\mathbf{H}_{ij}|^2 &= |\int \frac{\big(q_\theta(x,y)-p(x,y)\big)}{\sqrt{q_\theta(x,y)}} \big(\sqrt{q_\theta(x,y)}\nabla^2_\theta \ell(x,y)\big)_{ij} d(x,y)|^2 \\
&\leq \int \frac{\big(q_\theta(x,y)-p(x,y)\big)^2}{q_\theta(x,y)} d(x,y) \int q_\theta(x,y) \big(\nabla^2_\theta \ell(x,y)\big)^2_{ij} d(x,y) \\
&= \mathcal{D}_{\chi^2}(p||q_\theta)\, \mathbb{E}_{q_\theta}[\big(\nabla^2_\theta \ell(x,y)\big)^2_{ij}]
\end{aligned}$$

Similarly

$$|\mathbf{F}_{ij}-\mathbf{C}_{ij}|^2 \;\leq\; \mathcal{D}_{\chi^2}(p||q_\theta)\, \mathbb{E}_{q_\theta}[\big(\nabla_\theta \ell(x,y) \nabla_\theta \ell(x,y)^\top\big)^2_{ij}]$$

Thus

$$||\mathbf{C}-\mathbf{H}||^2 \;\leq\; \mathcal{D}_{\chi^2}(p||q_\theta)\, \mathbb{E}_{q_\theta}[||\mathbf{C}(x,y)||^2 + ||\mathbf{H}(x,y)||^2]$$

C.1.1.3. Proof of Proposition 5.3.1

From the upper bound assumption we have

$$\begin{aligned}
f(\theta^{k+1}) &\leq f(\theta^k) + \nabla f(\theta^k)^\top(\theta^{k+1}-\theta^k) + \tfrac{1}{2}(\theta^{k+1}-\theta^k)^\top \mathbf{H}(\theta^{k+1}-\theta^k) \\
&= f(\theta^k) - \alpha \nabla f(\theta^k)^\top \mathbf{M} \nabla \ell(\theta^k,x) + \tfrac{\alpha^2}{2} \nabla \ell(\theta^k,x)^\top \mathbf{M}^\top \mathbf{H} \mathbf{M} \nabla \ell(\theta^k,x)\,.
\end{aligned}$$

Subtracting $f(\theta^*)$ from both sides and taking conditional expectation we have

$$\begin{aligned}
\mathbb{E}[f(\theta^{k+1}) - f(\hat{\theta}^*)] &\leq f(\theta^k) - f(\hat{\theta}^*) - \alpha \nabla f(\theta^k)^\top \mathbf{M} \mathbb{E}\big[\nabla \ell(\theta^k,x)\big] + \tfrac{\alpha^2}{2} \mathbb{E}\big[\operatorname{Tr}\big(\mathbf{M}^\top \mathbf{H} \mathbf{M} \nabla \ell(\theta^k,x) \nabla \ell(\theta^k,x)^\top\big)\big] \\
&\leq f(\theta^k) - f(\hat{\theta}^*) - \alpha \nabla f(\theta^k)^\top \mathbf{M} \nabla f(\theta^k) + \tfrac{\alpha^2}{2} \operatorname{Tr}\big(\mathbf{M}^\top \mathbf{H} \mathbf{M} (\mathbf{C} + \nabla f(\theta^k) \nabla f(\theta^k))\big) \\
&= f(\theta^k) - f(\hat{\theta}^*) - \alpha \nabla f(\theta^k)^\top (\mathbf{M} - \frac{\alpha}{2} \mathbf{M}^\top \mathbf{H} \mathbf{M}) \nabla f(\theta^k) + \tfrac{\alpha^2}{2} \operatorname{Tr}\big(\mathbf{M}^\top \mathbf{H} \mathbf{M} \mathbf{C})\big)\,,
\end{aligned}$$

where in the second inequality we have used the covariance bound.

For $\mu_M \mathbf{I} \preccurlyeq \mathbf{M} - \frac{\alpha}{2}\mathbf{M}^\top \mathbf{H}\mathbf{M}$ and using the strong convexity bound $\frac{1}{2\mu}\|\nabla f(\theta)\|^2 \geq f(\theta) - f(\hat{\theta}^*)$, we can simplify to

$$\begin{aligned}
\mathbb{E}[f(\theta^{k+1}) - f(\hat{\theta}^*)] &\leq f(\theta^k) - f(\hat{\theta}^*) - \alpha\mu_M \nabla f(\theta^k)^\top \nabla f(\theta^k) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right) \\
&\leq f(\theta^k) - f(\hat{\theta}^*) - 2\alpha\mu_M\mu\left(f(\theta^k) - f(\hat{\theta}^*)\right) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right) \\
&= \left(1 - 2\alpha\mu_M\mu\right)\left(f(\theta^k) - f(\hat{\theta}^*)\right) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right)
\end{aligned}$$

Assuming $\alpha\mu_M\mu \leq \frac{1}{2}$, we have $\sum_{i=0}^{k}\left(1 - 2\alpha\mu_M\mu\right)^i \leq \sum_{i=0}^{\infty}\left(1 - 2\alpha\mu_M\mu\right)^i = \frac{1}{2\alpha\mu_M\mu}$. Therefore

$$\begin{aligned}
\mathbb{E}[f(\theta^{k+1}) - f(\hat{\theta}^*)] &\leq f(\theta^k) - f(\hat{\theta}^*) - \alpha\mu_M \nabla f(\theta^k)^\top \nabla f(\theta^k) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right) \\
&\leq f(\theta^k) - f(\hat{\theta}^*) - 2\alpha\mu_M\mu\left(f(\theta^k) - f(\hat{\theta}^*)\right) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right) \\
&= \left(1 - 2\alpha\mu_M\mu\right)\left(f(\theta^k) - f(\hat{\theta}^*)\right) + \tfrac{\alpha^2}{2}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right)
\end{aligned}$$

Assuming $\alpha\mu_M\mu \leq \frac{1}{2}$, we have $\sum_{i=0}^{k}\left(1 - 2\alpha\mu_M\mu\right)^i \leq \sum_{i=0}^{\infty}\left(1 - 2\alpha\mu_M\mu\right)^i = \frac{1}{2\alpha\mu_M\mu}$. Taking full expectations and chaining inequalities we then have

$$\mathbb{E}[f(\theta^k) - f(\hat{\theta}^*)] \leq \left(1 - 2\alpha\mu_M\mu\right)^k\left(f(\theta^0) - f(\hat{\theta}^*)\right) + \tfrac{\alpha}{4\mu_M\mu}\operatorname{Tr}\left(\mathbf{M}^\top \mathbf{H}\mathbf{M}\mathbf{C}\right).$$

This concludes the proof.

C.1.1.4. Convergence to limit cycles in the quadratic case

For SGD with constant stepsize α and preconditioner $\mathbf{M}$, the update equation on the parameters is

$$\theta_{t+1} = \theta_t - \alpha\mathbf{M}(\nabla f(\theta_t) + \varepsilon_t)$$

In our quadratic case, $\nabla f(\theta_t) = \mathbf{H}(\theta_t - \theta^*)$ with $\mathbb{E}[\varepsilon_t] = 0$ and $\mathbb{E}[\varepsilon_t \varepsilon_t^\top] = \mathbf{S}$. By defining $\delta_t = \mathbb{E}[\theta_t - \theta^*]$, we have

$$\begin{aligned}
\delta_{t+1} &= (\mathbf{I} - \alpha\mathbf{M}\mathbf{H})\delta_t \\
&= (\mathbf{I} - \alpha\mathbf{M}\mathbf{H})^{t+1}\delta_0
\end{aligned}$$

This concludes the first result of proposition on the quadratic case.

By defining, $\Sigma_t = \mathbb{E}[(\theta_t - \theta^*)(\theta_t - \theta^*)^\top]$, we get

$$
\begin{aligned}
\Sigma_{t+1} &= \Sigma_t - \mathbb{E}\big[\alpha\mathbf{M}\big(\mathbf{H}(\theta_t-\theta^*)+\varepsilon_t\big)(\theta_t-\theta^*)^\top\big] && \text{(C.1.1)}\\
&- \alpha\mathbb{E}\big[(\theta_t-\theta^*)\big(\theta_t-\theta^*+\varepsilon_t\big)^\top\mathbf{H}\mathbf{M}^\top\big] && \text{(C.1.2)}\\
&+ \alpha^2\mathbb{E}\big[\mathbf{M}\mathbf{H}(\theta_t-\theta^*)(\theta_t-\theta^*)^\top\mathbf{H}\mathbf{M}^\top\big] && \text{(C.1.3)}\\
&+ \alpha^2\mathbb{E}\big[\mathbf{M}\varepsilon_t\varepsilon_t^\top\mathbf{M}^\top\big] && \text{(C.1.4)}\\
&= \Sigma_t - \alpha\mathbf{M}\mathbf{H}\Sigma_t - \alpha\Sigma_t\mathbf{H}\mathbf{M}^\top + \alpha^2\mathbf{M}\mathbf{H}\Sigma_t\mathbf{H}\mathbf{M}^\top + \alpha^2\mathbf{M}\mathbf{S}\mathbf{M}^\top && \text{(C.1.5)}\\
&= (\mathbf{I}-\alpha\mathbf{M}\mathbf{H})\Sigma_t(\mathbf{I}-\alpha\mathbf{M}\mathbf{H})^\top + \alpha^2\mathbf{M}\mathbf{S}\mathbf{M}^\top && \text{(C.1.6)}\\
& && \text{(C.1.7)}
\end{aligned}
$$

C.1.2. Expected suboptimality for SG and Polyak momentum on quadratic functions

We detail here the computation of the expected suboptimality at each timestep when optimizing a quadratic function with a diagonal Hessian when the noise is also diagonal. Note that all these results apply if $\mathbf{H}$ and $\mathbf{S}$ are simultaneously diagonalizable by a change of basis.

We assume that f is a quadratic with Hessian $\mathbf{H}$ and that, at each time step, we receive a gradient perturbed by a random variable ε with $\mathbb{E}[\varepsilon]=0$, $\mathbb{E}[\varepsilon\varepsilon^\top]=\mathbf{S}$. Further, we shall assume that $\mathbf{H}$ and $\mathbf{S}$ are both diagonal. With these assumptions, the optimization occurs in each dimension independently and we can thus focus on a single dimension. We will denote by h and c the hessian and noise variance along that direction.

C.1.2.1. Proof of proposition 5.3.3

We can compare this result to the same setting where we use stochastic gradient with a diagonal preconditioning matrix $\mathbf{M}$. Then we get

$$
\begin{aligned}
s_i &= (1-\alpha\mathbf{M}_{ii}\mathbf{H}_{ii})^2 s_i + \alpha^2\mathbf{M}_{ii}^2\mathbf{S}_{ii}\\
s_i &= \frac{\alpha\mathbf{M}_{ii}\mathbf{S}_{ii}}{2\mathbf{H}_{ii}-\alpha\mathbf{M}_{ii}\mathbf{H}_{ii}^2}\,,
\end{aligned}
$$

and

$$
\mathbb{E}[f(\theta_t)-f(\hat{\theta}^*)] = \frac{1}{2}\sum_i \frac{\alpha\mathbf{M}_{ii}\mathbf{S}_{ii}}{2-\alpha\mathbf{M}_{ii}\mathbf{H}_{ii}} + \mathcal{O}(e^{-t})\,.
$$

Generalizing to simultaneously diagonalizable matrices, we get

$$
\mathbb{E}[f(\theta_t)-f(\hat{\theta}^*)] = \frac{\alpha}{2}\operatorname{Tr}((2\mathbf{I}-\alpha\mathbf{M}\mathbf{H})^{-1}\mathbf{M}\mathbf{S}) + \mathcal{O}(e^{-t})\,.
$$

C.1.2.2. Proof of proposition 5.3.4

Polyak momentum update equations are:

$$v_t = \gamma v_{t-1} + \nabla f(\theta_t) + \varepsilon \tag{C.1.8}$$

$$\theta_{t+1} = \theta_t - \alpha v_t \ . \tag{C.1.9}$$

Using the quadratic assumption, we can rewrite

$$\begin{aligned} v_{t+1} &= \gamma v_t + \nabla f(\theta_{t+1}) + \varepsilon \\ &= \gamma v_t + h\theta_{t+1} + \varepsilon \\ &= \gamma v_t + h\theta_t - \alpha h v_t + \varepsilon \ , \end{aligned}$$

and the full update can be written in matrix form

$$\begin{bmatrix} \theta_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & -\alpha \\ h & \gamma - \alpha h \end{bmatrix} \begin{bmatrix} \theta_{t-1} \\ v_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \varepsilon \end{bmatrix} \tag{C.1.10}$$

Denoting $P = \begin{bmatrix} 1 & -\alpha \\ h & \gamma - \alpha h \end{bmatrix}$ and $S_t = [\begin{bmatrix} \theta_t \\ v_t \end{bmatrix} \begin{bmatrix} \theta_t \\ v_t \end{bmatrix}^T$, we have

$$\mathbb{E}[S_t | S_{t-1}] = P S_{t-1} P^T + \begin{bmatrix} 0 & 0 \\ 0 & c \end{bmatrix} \ . \tag{C.1.11}$$

If there is a limit cycle for $\begin{bmatrix} \theta_t \\ v_t \end{bmatrix}$, it will satisfy

$$S = PSP^T + \begin{bmatrix} 0 & 0 \\ 0 & c \end{bmatrix} \ . \tag{C.1.12}$$

Writing $S = \begin{bmatrix} s_\theta & s_{v\theta} \\ s_{v\theta} & s_v \end{bmatrix}$, we have

$$\begin{aligned} s_\theta &= s_\theta - 2\alpha s_{v\theta} + \alpha^2 s_v \\ s_v &= h^2 s_\theta + 2h(\gamma - \alpha h) s_{v\theta} + (\gamma - \alpha h)^2 s_v + c \\ s_{v\theta} &= h s_\theta + (\gamma - 2\alpha h) s_{v\theta} - \alpha(\gamma - \alpha h) s_v \ . \end{aligned}$$

The first equation gives $s_{v\theta} = \frac{\alpha}{2} s_v$ and the last one becomes

$$\begin{aligned} \frac{\alpha}{2} s_v &= h s_\theta + (\gamma - 2\alpha h)\frac{\alpha}{2} s_v - \alpha(\gamma - \alpha h) s_v \\ s_\theta &= \frac{\alpha(1+\gamma)}{2h} s_v \ . \end{aligned}$$

Finally, the second equation gives

$$s_v = \left(h^2 \frac{\alpha(1+\gamma)}{2h} + 2h(\gamma - \alpha h)\frac{\alpha}{2} + (\gamma - \alpha h)^2\right) s_v + c$$
$$s_v = \frac{c}{(1-\gamma)\left(1+\gamma - \frac{\alpha h}{2}\right)}$$

and

$$s_\theta = \frac{\alpha(1+\gamma)c}{h(1-\gamma)(2+2\gamma-\alpha h)} .$$

Adding all dimensions together and multiplying by the Hessian to get the value function, we get

$$\mathbb{E}[f(\theta_t) - f(\hat{\theta}^*)] = \frac{1}{2}\sum_i \frac{\alpha(1+\gamma)\mathbf{S}_{ii}}{(1-\gamma)(2+2\gamma-\alpha\mathbf{H}_{ii})} + \mathcal{O}(e^{-t}) .$$

Generalizing to simultaneously diagonalizable matrices, we get

$$\mathbb{E}[f(\theta_t) - f(\hat{\theta}^*)] = \frac{\alpha}{2}\frac{(1+\gamma)}{(1-\gamma)} \operatorname{Tr}\left((2(1+\gamma)\mathbf{I} - \alpha\mathbf{H})^{-1}\mathbf{S}\right) + \mathcal{O}(e^{-t}) . \tag{C.1.13}$$

C.1.2.3. Comparison between stochastic gradient and Polyak momentum in the large noise regime

When the desired suboptimality is small, it requires a small α and the two suboptimality can be approximated by

$$f(\theta_t) - f(\hat{\theta}^*) \approx \frac{1}{4}\sum_i \frac{\alpha\mathbf{S}_{ii}}{(1-\gamma)} + o(1) \qquad \text{(Momentum)}$$

$$f(\theta_t) - f(\hat{\theta}^*) \approx \frac{1}{4}\sum_i \alpha\mathbf{S}_{ii} + o(1) , \qquad \text{(Stochastic gradient)}$$

and we see that momentum needs a stepsize α that is $(1-\gamma)$ times that of stochastic gradient to achieve the same suboptimality, countering any gain. This is what we see in Table 2.

C.2. Experimental details

C.2.1. Details on the Hessian inverse

As $\mathbf{H}$ is highly degenerate in neural networks, we compute an inverse of $\mathbf{H}$ by cutting all the eigenvalues smaller than $10^{-3} \times \lambda_{max}$ where λ_{max} is the biggest eigenvalue of $\mathbf{H}$. We observed that 10^{-3} and 10^{-3} were reasonable constants for selecting the eigenvalues of significant magnitude. Using smaller constant sometimes lead to very noisy estimates of the TIC while using a bigger constant would lead to severe underestimation of the criterion.

C.2.2. Details on the large scale experiments

These details apply for the experiments conducted in subsection 5.5.5, figure 4 and all figures in subsection 5.5.1.

We remind the reader the setup.

- 5 different architectures: logistic regression, a 1-hidden layer and 2-hidden layer fully connected network, and 2 small convolutional neural networks (CNNs, one with batch normalization (Ioffe & Szegedy, 2015) and one without);
- 3 datasets: MNIST, CIFAR-10, SVHN;
- 3 learning rates: 10^{-2}, $5 \cdot 10^{-3}$, 10^{-3} using vanilla SGD with momentum $\mu = 0.9$;
- 2 batch sizes: 64, 512;
- 5 dataset sizes: 5k, 10k, 20k, 25k, 50k.

We train for 750k steps and compute our metrics every 75k steps.

Data preprocessing: We choose to greyscale, resize to 7×7 pixels and normalize all the images in the 3 datasets used (CIFAR-10, MNIST and SVHN). This way, we can design architectures with a relatively low number of parameters.

Architectures:

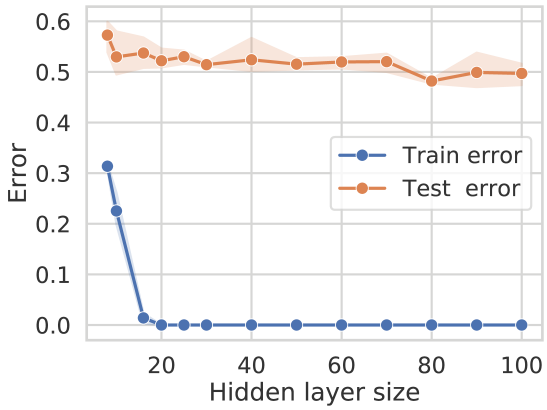
- `mlp`: This one is a one hidden layer MLP. Input size is $7 \times 7 = 49$ and output size is 10. The default number of hidden units is 70. We use ReLU activations.
- `big_mlp`: The architecture is the same as above but with one additional hidden layer.
- `logreg`: This is simple a 49×10 linear classifier.
- `cnn`: It is a small CNN with 3 layers. A first conv layer with kernel 3×3, 0 padding and 15 channels. The next layer has 20 channels and same parameters. The last layer has 10 channels and directly outputs the class scores.
- `cnn_bn`: Same architecture as above, except for a spatial batch-norm after the second layer.

C.2.3. Details on experiments of subsection 5.5.5

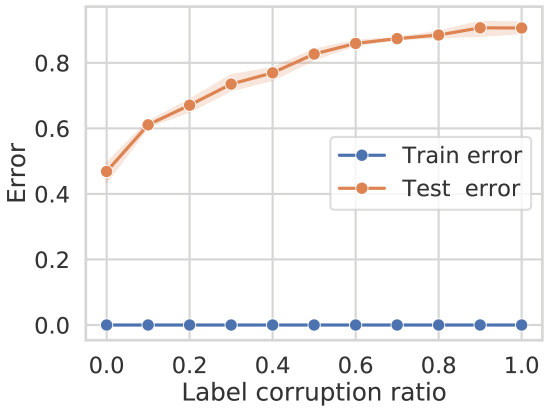
For these experiments we train one hidden layer MLPs on SVHN. Each points is computed by training three times with three different random seed until convergence. In figure 3a, the labels are kept without corruption and we vary the hidden size layer by using $\{8, 10, 16, 20, 25, 30, 40, 50, 60, 70, 80, 100\}$ hidden units in the hidden layer.

In figure 3b, we fix the number of hidden units to 70 but we vary the labels corruption percentage from 0% to 100% (included) by increments of 10%.

The networks are trained for $150k$ gradients steps with a learning rate of $5e{-}3$ and a batch size of 256. We used a subset of 2000 samples of SVHN to remain in the highly overparametrized regime, our networks were able to fit random data.



(a) Varying hidden layer size.



(b) Varying label randomization level.

Fig. 1. The train and test errors associated with the experiments 3a and 3b. We see that while we use small networks, they are still able to fit the data completely provided we use more than 20 hidden units. This behavior mirrors the one of bigger networks.

Appendix D

Beyond variance reduction

Organization of the appendix

We organize the appendix into several thematic sections.

The first one, section D.1 contains additional experiments and figures on bandits and MDPs. We have further investigations into committal and non-committal behaviour with baselines. More precisely subsection D.1.1 contains additional experiments for the 3 arm bandits for vanilla policy gradient, natural policy gradient and policy gradient with direct parameterization and a discussion on the effect the hyperparameters have on the results. In all cases, we find evidence for committal and non-committal behaviours. In the rest of the section, we investigate this in MDPs, starting with a smaller MDP with 2 different goals in subsection D.1.2 and constant baselines. We also provide additional experiments on the 4 rooms environment in subsection D.1.3, including the vanilla policy gradient and constant baselines with REINFORCE.

Then, section D.2 contains theory for the two-armed bandit case, namely proofs of convergence to a suboptimal policy (Proposition 6.3.1 in Appendix D.2.1) and an analysis of perturbed minimum-variance baselines (Proposition 6.3.2 in Appendix D.2.2). For the latter, depending on the perturbation, we may have possible convergence to a suboptimal policy, convergence to the optimal policy in probability, or a weaker form of convergence to the optimal policy. Finally, we also show vanilla policy gradient converges to the optimal policy in probability regardless of the baseline in Appendix D.2.3.

Section D.3 contains the theory for multi-armed bandit, including the proof of theorem 1. This theorem presents a counterexample to the idea that reducing variance always improves optimization. We show that there is baseline leading to reduced variance which may converge to a suboptimal policy with positive probability (see Appendix D.3.1) while there is another baseline with larger variance that converges to the optimal policy with probability 1 (see Appendix D.3.2). We identify on-policy sampling as being a potential source of these convergence issues. We provide proofs of proposition 6.4.1

in Appendix D.3.3, which shows convergence to the optimal policy in probability when using off-policy sampling with importance sampling.

Finally, in section D.4, we provide derivations of miscellaneous, smaller results such as the calculation of the minimum-variance baseline (Appendix D.4.1), the natural policy gradient update for the softmax parameterization (Appendix D.4.2) and the connection between the value function and the minimum-variance baseline (Appendix D.4.3).

D.1. Other experiments

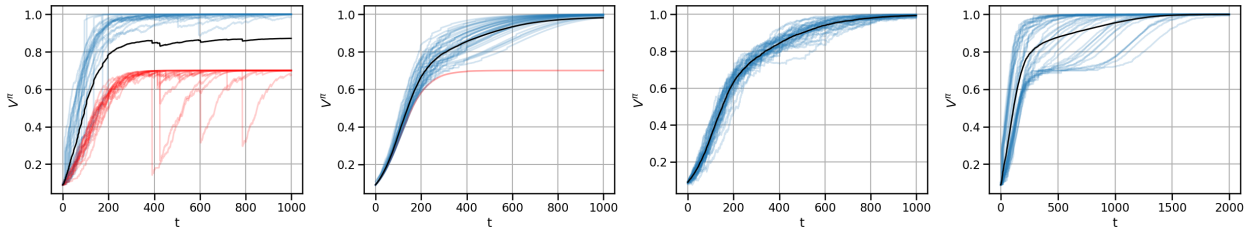
D.1.1. Three-armed bandit

In this subsection, we provide additional experiments on the three-armed bandit with natural and vanilla policy gradients for the softmax parameterization, varying the initializations. Additionally, we present results for the direct parameterization and utilizing projected stochastic gradient ascent.

The main takeaway is that the effect of the baselines appears more strongly when the initialization is unfavorable (for instance with a high probability of selecting a suboptimal action at first). The effect also are diminished when using small learning rates as in that case the effect of the noise on the optimization process lessens.

While the simplex visualization is very appealing, we mainly show here learning curves as we can showcase more seeds that way and show the effects are noticeable across many runs.

Natural policy gradient

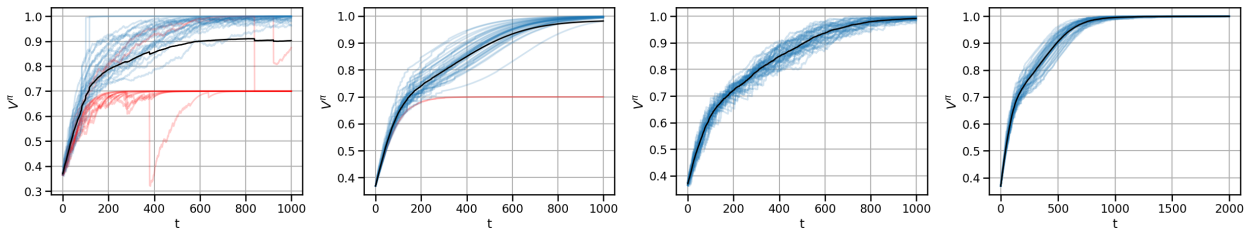


(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 1. We plot 40 different learning curves (in blue and red) of natural policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.025$ and $\theta_0 = (0, 3, 5)$. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training. Note that the value function baseline convergence was slow and thus was trained for twice the number of time steps.

Figure 1 uses the same setting as Figure 1 with 40 trajectories instead of 15. We do once again observe many cases of convergence to the wrong arm for the negative baseline and

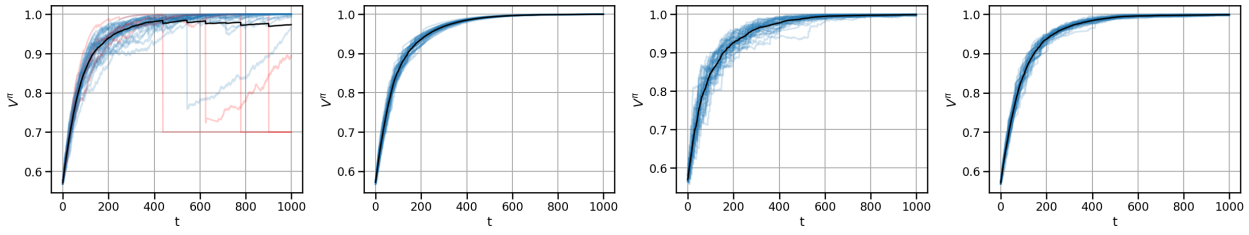
some cases for the minimum variance baseline, while the positive baseline converges reliably. In this case the value function also converges to the optimal solution but is much slower.



(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 2. We plot 40 different learning curves (in blue and red) of natural policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.025$ and $\theta_0 = (0, 3, 3)$. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training.

Figure 2 shows a similar setting to Figure 1 but where the initialization parameter is not as extreme. We observe the same type of behavior, but not as pronounced as before; fewer seeds converge to the wrong arm.



(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

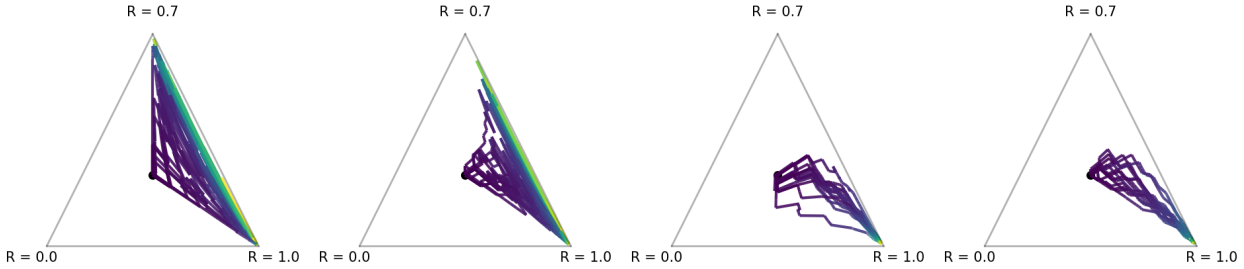
Fig. 3. We plot 40 different learning curves (in blue and red) of natural policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.025$ and $\theta_0 = (0, 0, 0)$ i.e the initial policy is uniform. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training.

In Figure 3 whose initial policy is the uniform, we observe that the minimum variance baseline and the value function as baseline perform very well. On the other hand the committal baseline still has seeds that do not converge to the right arm. Interestingly, while all seeds for the non-committal baseline identify the optimal arm, the variance of the return is higher than for the optimal baseline, suggesting a case similar to the result presented in Proposition 6 where a positive baseline ensured we get close to the optimal arm but may not remain arbitrary close to it.

Vanilla policy gradient

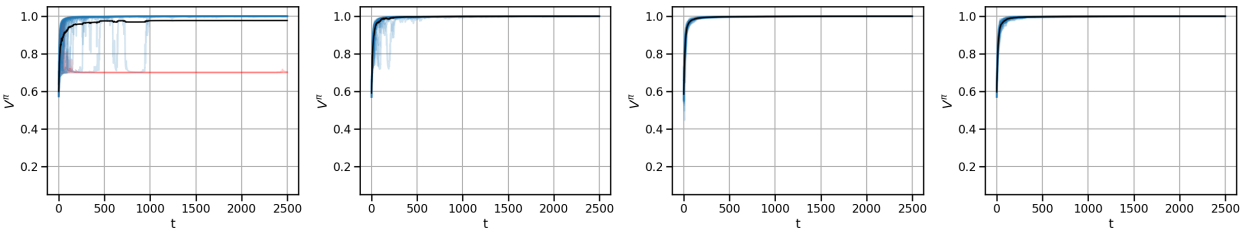
While we have no theory indicating that we may converge to a suboptimal arm with vanilla policy gradient, we can still observe some effect in terms of learning speed in practice (see Figures 4 to 7).

On Figures 4 and 5 we plot the simplex view and the learning curves for vanilla policy gradient initialized at the uniform policy. We do observe that some trajectories did not converge to the optimal arm in the imparted time for the committal baseline, while they converged in all other settings. The mininum variance baseline is slower to converge than the non-committal and the value function in this setting as can be seem both in the simplex plot and learning curves.



(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 4. Simplex plot of 15 different learning curves for vanilla policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.5$ and $\theta_0 = (0,0,0)$. Colors, from purple to yellow represent training steps.

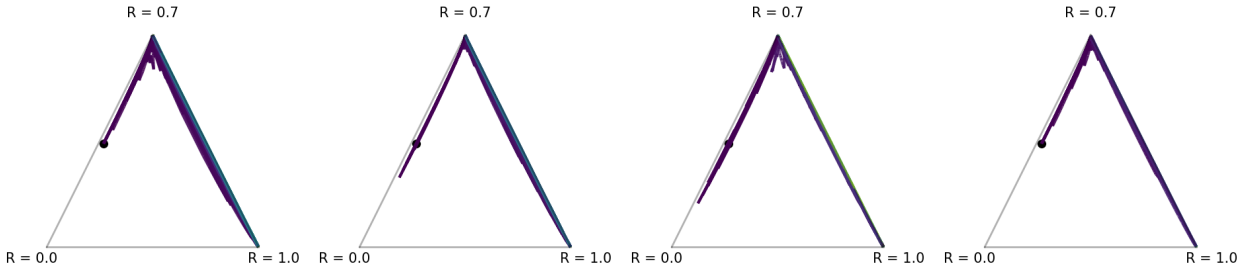


(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 5. We plot 40 different learning curves (in blue and red) of vanilla policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.5$ and $\theta_0 = (0,0,0)$. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training.

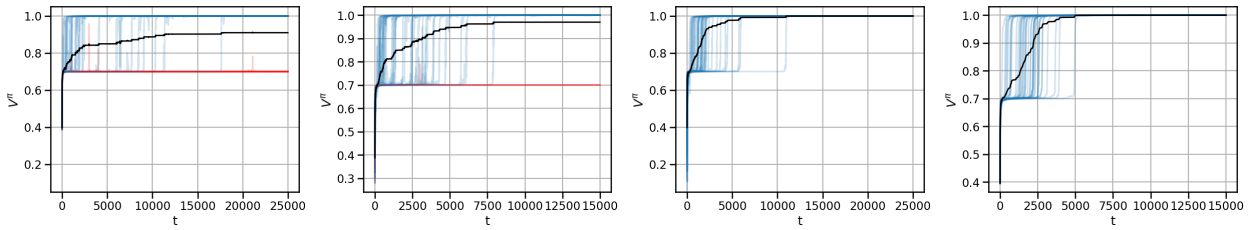
On Figures 6 and 7 we plot the simplex view and the learning curves for vanilla policy gradient initialized at a policy yielding a very high probability of sampling the suboptimal actions, 48.7% for each. We do observe a similar behavior than for the previous plots with vanilla PG, but in this setting the minimum variance baseline is even slower to converge

and a few seeds did not identify the optimal arm. As the gradient flow leads the solutions closer to the simplex edges, the simplex plot is not as helpful in this setting to understand the behavior of each baseline option.



(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 6. Simplex plot of 15 different learning curves for vanilla policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.5$ and $\theta_0 = (0, 3, 3)$. Colors, from purple to yellow represent training steps.



(a) $b = b^* - 1/2$ **(b)** $b = b^*$ **(c)** $b = b^* + 1/2$ **(d)** $b = V^\pi$

Fig. 7. We plot 40 different learning curves (in blue and red) of vanilla policy gradient, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.5$ and $\theta_0 = (0, 3, 3)$. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training.

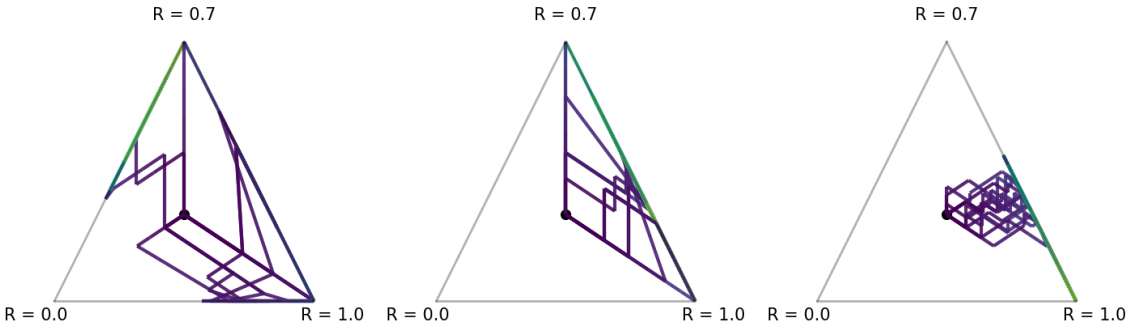
Policy gradient with direct parameterization

Here we present results with the direct parameterization, i.e where θ contains directly the probability of drawing each arm. In that case the gradient update is

$$\theta_{t+1} = \text{Proj}_{\Delta_3}\left[\theta_t + \alpha \frac{r(a_i) - b}{\theta(a_i)} \mathbf{1}_{a_i}\right]$$

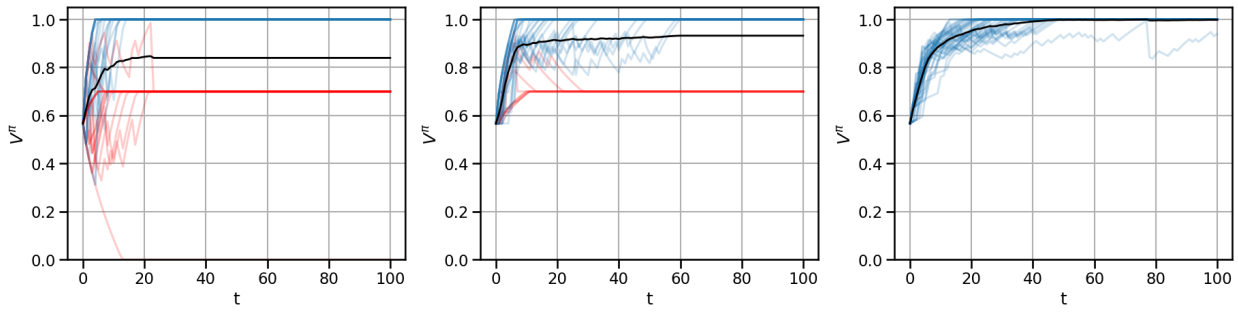
where Δ_3 is the three dimensional simplex $\Delta_3 = \{u, v, w \geq 0, u + v + w = 1\}$. In this case, however, because the projection step is non trivial and doesn't have an easy explicit closed form solution (but we can express it as the output of an algorithm), we cannot explicitly write down the optimal baseline. Again, because of the projection step, baselines of this form are not guaranteed to preserve unbiasedness of the gradient estimate. For

this reason, we only show experiments with fixed baselines, but keep in mind that these results are not as meaningful as the ones presented above. We present the results in Figures 8 and 9.



(a) $b = -1/2$ **(b)** $b = 0$ **(c)** $b = 1/2$

Fig. 8. We plot 15 different learning curves of vanilla policy gradient with direct parameterization, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.1$ and $\theta_0 = (1/3, 1/3, 1/3)$, the uniform policy on the simplex.



(a) $b = -1/2$ **(b)** $b = 0$ **(c)** $b = 1/2$

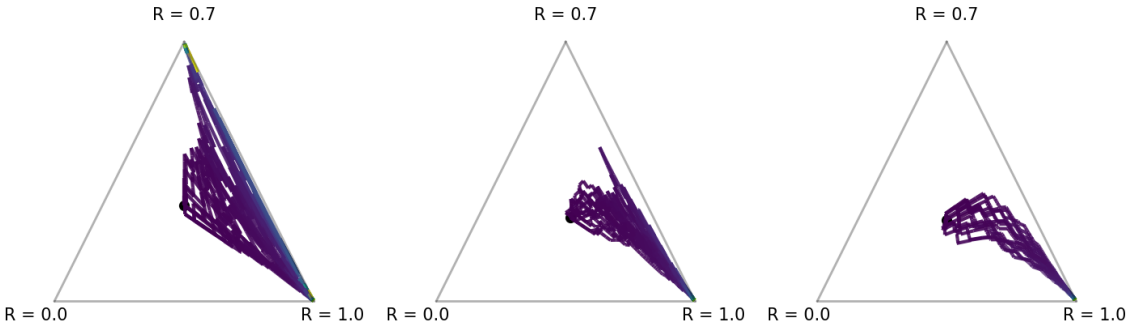
Fig. 9. We plot 40 different learning curves (in blue and red) of vanilla policy gradient with direct parameterization, when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.1$ and $\theta_0 = (1/3, 1/3, 1/3)$, the uniform policy. The black line is the average value over the 40 seeds for each setting. The red curves denote the seeds that did not reach a value of at least 0.9 at the end of training.

Once again in this setting we can see that negative baselines tend to encourage convergence to a suboptimal arm while positive baselines help converge to the optimal arm.

Policy gradient with escort transform parameterization

We try the escort transform Mei et al. (2020a) which was found to lead to better curvature properties of the objective than the softmax parameterization. We use the escort transform parameter $p = 2$ as in the experiments for the original paper and find results

similar to the softmax parameterization. In fact, since this parameterization has larger updates near deterministic policies, it may be more prone to getting stuck at suboptimal policies when a choosing a committal baseline.

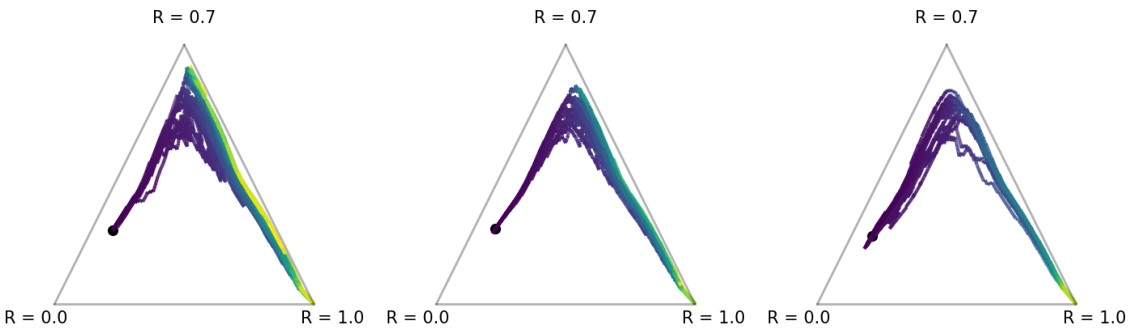


(a) $b = -1/2$ **(b)** $b = 0$ **(c)** $b = 1/2$

Fig. 10. We plot 15 different learning curves of vanilla policy gradient with the escort transform with parameter $p = 2$ (Mei et al., 2020a), when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.25$ and $\theta_0 = (1,1,1)$, the uniform policy on the simplex.

Policy gradient with mellowmax parameterization

As an alternate parameterization, we try the mellowmax function Asadi & Littman (2017). Unfortunately, it is not trivial to utilize it with policy gradient methods. The mellowmax algorithm was designed for SARSA as it requires Q-function estimates and the temperature parameter β has to be computed using a black-box optimizer to find a maximum-entropy policy, thus cannot be differentiated through easily. However, using a naive version (treating β as a constant in the policy gradient, setting $\omega = 1$ and using the parameters directly in place of Q), we observe that the committal vs. non-committal behaviors are greatly mitigated and all paths conserve a higher entropy and converge to the optimal arm. This strategy could be viewed as adding an entropy-regularizer with biased updates. Note that the baseline we used as the "minimum-variance" is not the true minimizer due to this bias too. Furthermore, even though the divergence is mitigated, the complexity per iteration rises significantly due to solving a black-box optimization problem at every step.



(a) $b = -1/2$ **(b)** $b = 0$ **(c)** $b = 1/2$

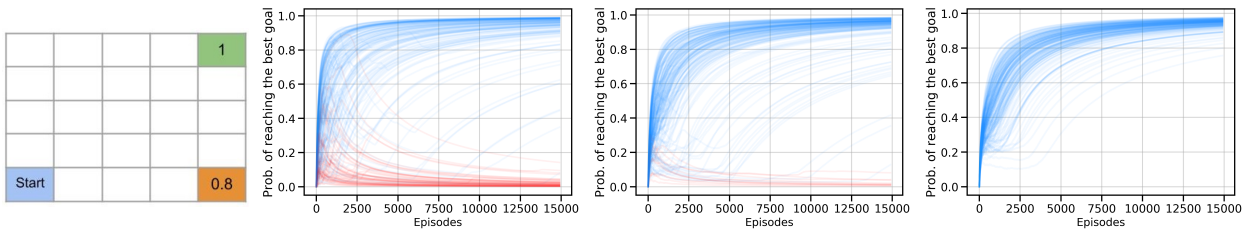
Fig. 11. We plot 15 different learning curves of a policy gradient with the mellowmax transform Asadi & Littman (2017), when using various baselines, on a 3-arm bandit problem with rewards $(1, 0.7, 0)$, $\alpha = 0.25$ and $\theta_0 = (0,3,5)$.

D.1.2. Simple gridworld

As a simple MDP with more than one state, we experiment using a 5x5 gridworld with two goal states, the closer one giving a reward of 0.8 and the further one a reward of 1. We ran the vanilla policy gradient with a fixed stepsize and discount factor of 0.99 multiple times for several baselines. Fig. 12 displays individual learning curves with the index of the episode on the x-axis, and the fraction of episodes where the agent reached the reward of 1 up to that point on the y-axis. To match the experiments for the four rooms domain in the main text, Fig. 13 shows returns and the entropy of the actions and state visitation distributions for multiple settings of the baseline. Once again, we see a difference between the smaller and larger baselines. In fact, the difference is more striking in this example since some learning curves get stuck at suboptimal policies. Overall, we see two main trends in this experiment: a) The larger the baseline, the more likely the agent converges to the optimal policy, and b) Agents with negative baselines converge faster, albeit sometimes to a suboptimal behaviour. We emphasize that a) is not universally true and large enough baselines will lead to an increase in variance and a decrease in performance.

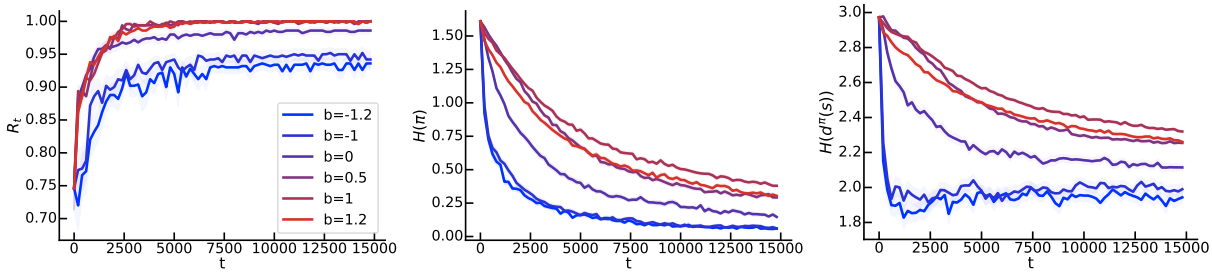
D.1.3. Additional results on the 4 rooms environment

For the four-rooms gridworld discussed in the main text, we extend the experiments and provide additional details. The environment is a 10x10 gridworld consisting of 4 rooms as depicted on Fig. 4a with a discount factor $\gamma = 0.99$. The agent starts in the upper left room and two adjacent rooms contain a goal state of value 0.6 (discounted, ≈ 0.54) or 0.3 (discounted, ≈ 0.27). However, the best goal, with a value of 1 (discounted, ≈ 0.87),



(a) MDP used **(b)** $b = -1$ **(c)** $b = 0$ **(d)** $b = 1$

Fig. 12. Learning curves for a 5x5 gridworld with two goal states where the further goal is optimal. Trajectories in red do not converge to an optimal policy.



(a) Returns **(b)** Action entropy **(c)** State visitation entropy

Fig. 13. We plot the returns, the entropy of the policy over the states visited in each trajectory, and the entropy of the state visitation distribution averaged over 100 runs for multiple baselines for the 5x5 gridworld. The shaded regions denote one standard error and are close to the mean curve. Similar to the four rooms, the policy entropy of lower baselines tends to decay faster than for larger baselines, and smaller baselines tend to get stuck on suboptimal policies, as indicated by the returns plot.

lies in the furthest room, so that the agent must learn to cross the sub-optimal rooms and reach the furthest one.

For the NPG algorithm used in the main text, we required solving for $Q_\pi(s,a)$ for the current policy π. This was done using dynamic programming on the true MDP, stopping when the change between successive approximations of the value function didn't differ more than 0.001. Additionally, a more thorough derivation of the NPG estimate we use can be found in Appendix D.4.6.

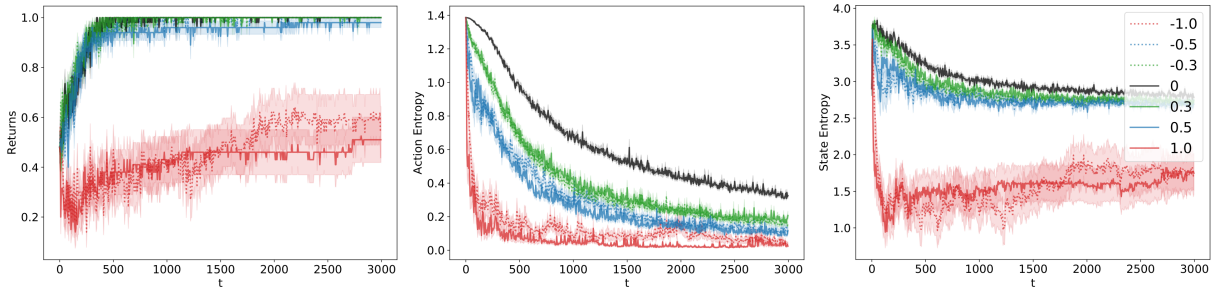
We also experiment with using the vanilla policy gradient with the tabular softmax parameterization in the four-rooms environment. We use a similar estimator of the policy gradient which makes updates of the form:

$$\theta \leftarrow \theta + \alpha(Q_{\pi_\theta}(s_i,a_i) - b)\nabla \log \pi_\theta(a_i|s_i)$$

for all observed s_i, a_i in the sampled trajectory. As with the NPG estimator, we can find the minimum-variance baseline b^*_θ in closed-form and thus can choose baselines of the form $b^+ = b^*_\theta + \varepsilon$ and $b^-_\theta = b^*_\theta - \varepsilon$ to ensure equal variance as before. Fig. 15 plots the

results. In this case, we find that there is not a large difference between the results for $+\varepsilon$ and $-\varepsilon$, unlike the results for NPG and those for vanilla PG in the bandit setting.

The reason for this discrepancy may be due to the magnitudes of the perturbations ε relative to the size of the unperturbed update $Q_\pi(s_i,a_i) - b^*_\theta$. The magnitude of $Q_\pi(s_i,a_i) - b^*$ varies largely from the order of 0.001 to 0.1, even within an episode. To investigate this further, we try another experiment using perturbations $\varepsilon = c(\max_a Q_\pi(s_i,a) - b^*_\theta$ for various choices of $c > 0$. This would ensure that the magnitude of the perturbation is similar to the magnitude of $Q_\pi(s_i,a_i) - b^*$, while still controlling for the variance of the gradient estimates. In Fig. 14, we see that there is a difference between the $+\varepsilon$ and $-\varepsilon$ settings. As expected, the $+\varepsilon$ baseline leads to larger action and state entropy although, in this case, this results in a reduction of performance. Overall, the differences between vanilla PG and natural PG are not fully understood and there may be many factors playing a role, possibly including the size of the updates, step sizes and the properties of the MDP.

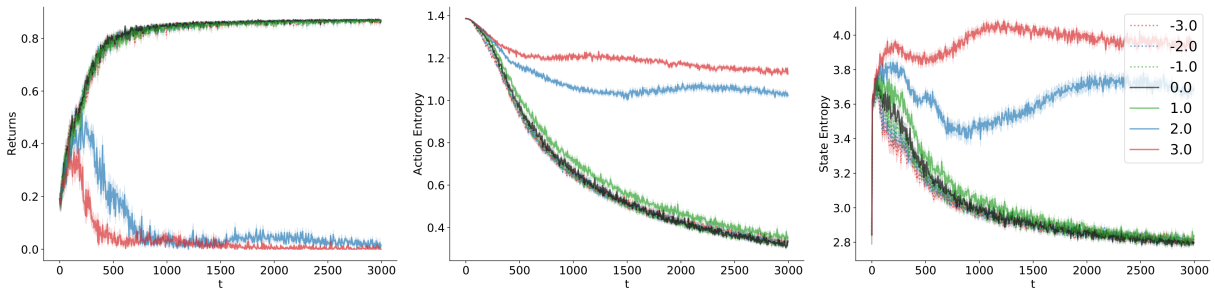


(a) Returns **(b)** Action entropy **(c)** State visitation entropy

Fig. 14. We plot results for vanilla policy gradient with perturbed minimum-variance baselines of the form $b^*_\theta + \varepsilon$, with ε denoted in the legend. The step size is 0.5 and 20 runs are done. We see smaller differences between positive and negative ε values.

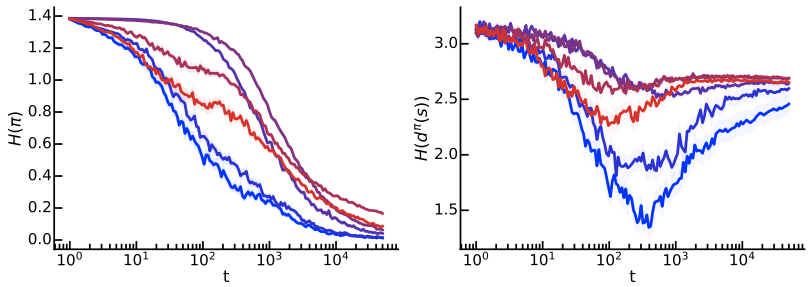
Finally, we also experiment with the vanilla REINFORCE estimator with softmax parameterization where the estimated gradient for a trajectory is $(R(\tau_i) - b)\nabla \log \pi(\tau_i)$ for τ_i being a trajectory of state, actions and rewards for an episode. For the REINFORCE estimator, it is difficult to compute the minimum-variance baseline so, instead, we utilize constant baselines. Although we cannot ensure that the variance of the various baselines are the same, we could still expect to observe committal and non-committal behaviour depending on the sign of $R(\tau_i) - b$. We use a step size of 0.1.

We consider an alternative visualization for the experiment of vanilla policy gradient with constant baselines: Figures 17a, 17b and 17c. Each point in the simplex is a policy, and the position is an estimate, computed with $1{,}000$ Monte-Carlo samples, of the probability of the agent reaching each of the 3 goals. We observe that the starting point of the curve is equidistant to the 2 sub-optimal goals but further from the best goal, which is coherent with the geometry of the MDP. Because we have a discount factor of $\gamma = 0.99$,



(a) Returns **(b)** Action entropy **(c)** State visitation entropy

Fig. 15. We plot results for vanilla policy gradient with perturbed minimum-variance baselines of the form $b^*_\theta + \varepsilon$, where $\varepsilon = c(\max_a Q_\pi(s_i,a) - b^*_\theta$ and c is denoted in the legend. For a fixed c, we can observe a difference between the learning curves for the $+c$ and $-c$ settings. The step size is 0.5 and 50 runs are done. As expected, the action and state entropy for the positive settings of c are larger than for the negative settings. In this case, this increased entropy does not translate to larger returns though and is a detriment to performance,

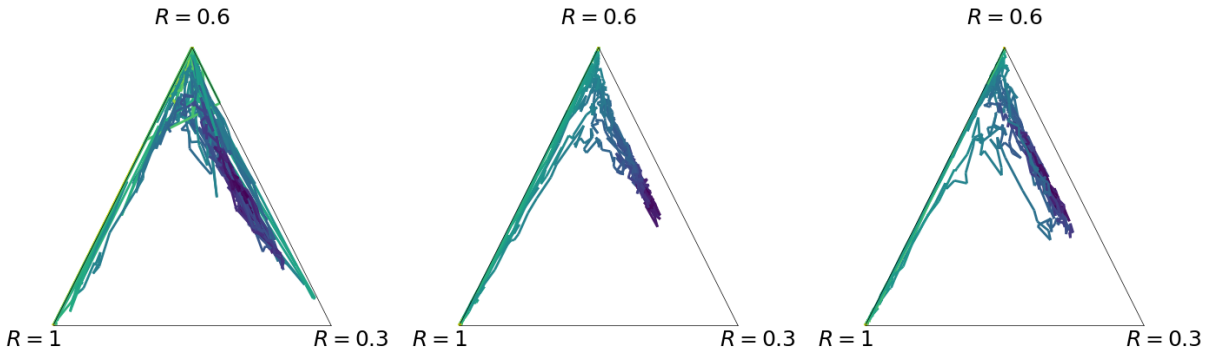


(a) Returns **(b)** Action entropy **(c)** State visitation entropy

Fig. 16. We plot the results for using REINFORCE with constant baselines. Once again, the policy entropy of lower baselines tends to decay faster than for larger baselines, and smaller baselines tend to get stuck on suboptimal policies, as indicated by the returns plot.

the agent first learns to reach the best goal in an adjacent room to the starting one, and only then it learns to reach the globally optimal goal fast enough for its reward to be the best one.

In these plots, we can see differences between $b = -1$ and $b = 1$. For the lower baseline, we see that trajectories are much more noisy, with some curves going closer to the bottom-right corner, corresponding to the worst goal. This may suggest that the policies exhibit committal behaviour by moving further towards bad policies. On the other hand, for $b = 1$, every trajectory seems to reliably move towards the top corner before converging to the bottom-left, an optimal policy.



(a) $b = -1$ **(b)** $b = 0$ **(c)** $b = 1$

Fig. 17. We plot 10 different trajectories of vanilla policy gradient (REINFORCE) using different constant on a 4 rooms MDP with goal rewards $(1, 0.6, 0.3)$. The color of each trajectory represents time and each point of the simplex represents the probability that a policy reaches one of the 3 goals.

D.2. Two-armed bandit theory

In this section, we expand on the results for the two-armed bandit. First, we show that there is some probability of converging to the wrong policy when using natural policy gradient with a constant baseline. Next, we consider all cases of the perturbed minimum-variance baseline ($b = b^* + \varepsilon$) and show that some cases lead to convergence to the optimal policy with probability 1 while others do not. In particular there is a difference between $\varepsilon < -1$ and $\varepsilon > 1$, even though these settings can result in the same variance of the gradient estimates. Finally, we prove that the vanilla policy gradient results in convergence in probability to the optimal policy regardless of the baseline, in contrast to the natural policy gradient.

Notations:

- Our objective is $J(\theta) = \mathbb{E}_{\pi_\theta}[R(\tau)]$, the expected reward for current parameter θ.
- $p_t = \sigma(\theta_t)$ is the probability of sampling the optimal arm (arm 1).
- P_1 is the distribution over rewards than can be obtained from pulling arm 1. Its expected value is $\mu_1 = \mathbb{E}_{r_1 \sim P_1}[r_1]$. Respectively P_0, μ_0 for the suboptimal arm.
- g_t is a stochastic unbiased estimate of $\nabla_\theta J(\theta_t)$. It will take different forms depending on whether we use vanilla or natural policy gradient and whether we use importance sampling or not.
- For $\{\alpha_t\}_t$ the sequence of stepsizes, the current parameter θ_t is a random variable equal to $\theta_t = \sum_{i=1}^{t} \alpha_i g_i + \theta_0$ where θ_0 is the initial parameter value.

For many convergence proofs, we will use the fact that the sequence $\theta_t - \mathbb{E}[\theta_t]$ forms a martingale. In other words, the noise around the expected value is a martingale, which we define below.

Definition 5 (Martingale). *A discrete-time martingale is a stochastic process* $\{X_t\}_{t\in\mathbb{N}}$ *such that*

- $\mathbb{E}[|X_t|] < +\infty$
- $\mathbb{E}[X_{t+1}|X_t, \dots X_0] = X_t$

Example 4. *For* g_t *a stochastic estimate of* $\nabla J(\theta_t)$ *we have* $X_t = \mathbb{E}[\theta_t] - \theta_t$ *is a martingale. As* $\theta_t = \theta_0 + \sum_i \alpha_i g_i$, X_t *can also be rewritten as* $X_t = \mathbb{E}[\theta_t - \theta_0] - (\theta_t - \theta_0) = \sum_{i=0}^{t} \alpha_i \big(\mathbb{E}[g_i|\theta_0] - g_i\big)$.

We will also be making use of Azuma-Hoeffding's inequality to show that the iterates stay within a certain region with high-probability, leading to convergence to the optimal policy.

Lemma 1 (Azuma-Hoeffding's inequality). *For* $\{X_t\}$ *a martingale, if* $|X_t - X_{t-1}| \leq c_t$ *almost surely, then we have* $\forall t, \varepsilon \geq 0$

$$\mathbb{P}(X_t - X_0 \geq \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sum_{i=1}^{t} c_i^2}\right)$$

D.2.1. Convergence to a suboptimal policy with a constant baseline

For the proofs in this subsection, we assume that the step size is constant i.e. $\alpha_t = \alpha$ for all t and that the rewards are deterministic.

Proposition D.2.1. *Consider a two-arm bandit with rewards* 1 *and* 0 *for the optimal and suboptimal arms, respectively. Suppose we use natural policy gradient starting from* θ_0, *with a fixed baseline* $b < 0$, *and fixed stepsize* $\alpha > 0$. *If the policy samples the optimal action with probability* $\sigma(\theta)$, *then the probability of picking the suboptimal action forever and having* θ_t *go to* $-\infty$ *is strictly positive. Additionally, if* $\theta_0 \leq 0$, *we have*

$$P(\text{suboptimal action forever}) \geq (1 - e^{\theta_0})(1 - e^{\theta_0 + \alpha b})^{-\frac{1}{\alpha b}}.$$

Proof. First, we deal with the case where $\theta_0 < 0$.

$$1 - \sigma(\theta_0 - \alpha b t) \geq 1 - \exp(\theta_0 - \alpha b t)$$

Next, we use the bound $1-x \geq \exp(\frac{-x}{1-x})$. This bound can be derived as follows:

$$\begin{aligned}
1-u &\leq e^{-u} \\
1-e^{-u} &\leq u \\
1-\frac{1}{y} &\leq \log y, \quad \text{substitute } u = \log y \text{ for } y > 0 \\
\frac{-x}{1-x} &\leq \log(1-x), \quad \text{substitute } y = 1-x \text{ for } x \in [0,1) \\
\exp\left(\frac{-x}{1-x}\right) &\leq 1-x.
\end{aligned}$$

Continuing with $x = \exp(\theta_0 - \alpha bt)$, the bound holds when $x \in [0,1)$, which is satisfied assuming $\theta_0 \leq 0$.

$$1-\sigma(\theta_0-\alpha bt) \geq \exp\left(\frac{-1}{e^{-\theta_0+\alpha bt}-1}\right)$$

For now we ignore $t=0$ and we will just multiply it back in at the end.

$$\begin{aligned}
\prod_{t=1}^{\infty}[1-\sigma(\theta_0-\alpha bt)] &\geq \prod_{t=1}^{\infty}\exp\left(\frac{-1}{e^{-\theta_0+\alpha bt}-1}\right) \\
&= \exp\sum_{t=1}^{\infty}\left(\frac{-1}{e^{-\theta_0+\alpha bt}-1}\right) \\
&\geq \exp\left(-\int_{t=1}^{\infty}\frac{1}{e^{-\theta_0+\alpha bt}-1}dt\right)
\end{aligned}$$

The last line follows by considering the integrand as the right endpoints of rectangles approximating the area above the curve.

Solving this integral by substituting $y = -\theta_0 + \alpha bt$, multiplying the numerator and denominator by e^y and substituting $u = e^y$, we get:

$$\begin{aligned}
&= \exp\left(\frac{1}{\alpha b}\log(1-e^{\theta_0-\alpha b})\right) \\
&= \left(1-e^{\theta_0-\alpha b}\right)^{\frac{1}{\alpha b}}
\end{aligned}$$

Finally we have:

$$P(\text{left forever}) \geq (1-e^{\theta_0})(1-e^{\theta_0-\alpha b})^{\frac{1}{\alpha b}}$$

If $\theta_0 > 0$, then there is a positive probability of reaching $\theta < 0$ in a finite number of steps since choosing action 2 makes a step of size αb in the left direction and we will reach $\theta_t < 0$ after $m = \frac{\theta_0 - 0}{\alpha b}$ steps leftwards. The probability of making m left steps in a row

is positive. So, we can simply lower bound the probability of picking left forever by the product of that probability and the derived bound for $\theta_0 \leq 0$. □

Corollary 1.1. *The regret for the previously described two-armed bandit is linear.*

PROOF. Letting R_t be the reward collected at time t,

$$\begin{aligned} Regret(T) &= \mathbb{E}\left[\sum_{t=1}^{T}(1-b-R_t)\right] \\ &\geq \sum_{t=1}^{T} 1 \times Pr(\text{left } T \text{ times}) \\ &\geq \sum_{t=1}^{T} P(\text{left forever}) \\ &= T \times P(\text{left forever}). \end{aligned}$$

The second line follows since choosing the left action at each step incurs a regret of 1 and this is one term in the entire expectation. The third line follows since choosing left T times is a subset of the event of choosing left forever. The last line implies linear regret since we know $Pr(\text{left forever}) > 0$ by the previous theorem. □

D.2.2. Analysis of perturbed minimum-variance baseline

In this section, we look at perturbations of the minimum-variance baseline in the two-armed bandit, i.e. baselines of the form $b = 1 - p_t + \varepsilon$. In summary:

- For $\varepsilon < -1$, convergence to a suboptimal policy is possible with positive probability.
- For $\varepsilon \in (-1,1)$, we have convergence almost surely to the optimal policy.
- For $\varepsilon \geq 1$, the supremum of the iterates goes to ∞ (but we do not have convergence to an optimal policy)

It is interesting to note that there is a subtle difference between the case of $\varepsilon \in (-1,0)$ and $\varepsilon \in (0,1)$, even though both lead to convergence. The main difference is that when θ_t is large, positive ε leads to both updates being positive and hence improvement is guaranteed at every step. But, when ε is negative, then only one of the actions leads to improvement, the other gives a large negative update. So, in some sense, for $\varepsilon \in (-1,0)$, convergence is less stable because a single bad update could be catastrophic.

Also, the case of $\varepsilon = -1$ proved to be difficult. Empirically, we found that the agent would incur linear regret and it seemed like some learning curves also got stuck near $p = 0$, but we were unable to theoretically show convergence to a suboptimal policy.

Lemma 2. *For the two-armed bandit with sigmoid parameterization, natural policy gradient and a perturbed minimum-variance baseline $b = 1 - p_t + \varepsilon$, with $\varepsilon < -1$, there is a positive probability of choosing the suboptimal arm forever and diverging.*

PROOF. We can reuse the result for the two-armed bandit with constant baseline $b < 0$. Recall that for the proof to work, we only need θ to move by at least a constant step $\delta > 0$ in the negative direction at every iteration.

In detail, the update after picking the worst arm is $\theta_{t+1} = \theta_t + \alpha(1 + \frac{\varepsilon}{1-p_t})$. So, if we choose $\varepsilon < -1 - \delta$ for some $\delta > 0$, we get the update step magnitude is $\frac{\delta+p}{1-p} > \delta$ and hence the previous result applies (replace αb by δ). □

Lemma 3. *For the two-armed bandit with sigmoid parameterization, natural policy gradient and a perturbed minimum-variance baseline $b = 1 - p_t + \varepsilon$, with $\varepsilon \in (-1,0)$, the policy converges to the optimal policy in probability.*

PROOF. Recall that the possible updates when the parameter is θ_t are:

- $\theta_{t+1} = \theta_t + \alpha(1 - \frac{\varepsilon}{\sigma(\theta_t)})$ if we choose action 1, with probability $\sigma(\theta_t)$
- $\theta_{t+1} = \theta_t + \alpha(1 + \frac{\varepsilon}{1-\sigma(\theta_t)})$ if we choose action 2, with probability $1 - \sigma(\theta_t)$.

First, we will partition the real line into three regions (A, B, and C with $a < b < c$ for $a \in A, b \in B, c \in C$), depending on the values of the updates. Then, each region will be analyzed separately.

We give an overview of the argument first. For region A (θ very negative), both updates are positive so θ_t is guaranteed to increase until it reaches region B.

For region C (θ very positive), sampling action 2 leads to the update $\alpha(1 + \frac{\varepsilon}{1-\sigma(\theta_t)})$, which has large magnitude and results in θ_{t+1} being back in region A. So, once θ_t is in C, the agent needs to sample action 1 forever to stay there and converge to the optimal policy. This will have positive probability (using the same argument as the divergence proof for the two-armed bandit with constant baseline).

For region B, the middle region, updates to θ_t can make it either increase or decrease and stay in B. For this region, we will show that θ_t will eventually leave B with probability 1 in a finite number of steps, with some lower-bounded probability of reaching A.

Once we've established the behaviours in the three regions, we can argue that for any initial θ_0 there is a positive probability that θ_t will eventually reach region C and take action 1 forever to converge. In the event that does not occur, then θ_t will be sent back to A and the agent gets another try at converging. Since we are looking at the behaviour when $t \to \infty$, the agent effectively gets infinite tries at converging. Since each attempt has some positive probability of succeeding, convergence will eventually happen.

We now give additional details for each region.

To define region A, we check when both updates will be positive. The update from action 1 is always positive so we are only concerned with the second update.

$$\begin{aligned} 1+\frac{\varepsilon}{1-p} &> 0 \\ 1-p+\varepsilon &> 0 \\ 1+\varepsilon &> p \\ \sigma^{-1}(1+\varepsilon) &> \theta \end{aligned}$$

Hence, we set $A = (-\infty, \sigma^{-1}(1+\varepsilon))$. Since every update in this region increases θ_t by at least a constant at every iteration, θ_t will leave A in a finite number of steps.

For region C, we want to define it so that an update in the negative direction from any $\theta \in C$ will land back in A. So $C = [c, \infty)$ for some $c \geq \sigma^{-1}(1+\varepsilon)$. By looking at the update from action 2, $\alpha(1+\frac{\varepsilon}{1-\sigma(\theta)}) = \alpha(1+\varepsilon(1+e^{\theta}))$, we see that it is equal to 0 at $\theta = \sigma^{-1}(1+\varepsilon)$ but it is a decreasing function of θ and it decreases at an exponential rate. So, eventually for θ_t sufficiently large, adding this update will make $\theta_{t+1} \in A$.

So let $c = \inf\{\theta : \theta + \alpha\left(1-\frac{\varepsilon}{1-\sigma(\theta)}\right), \theta \geq \sigma^{-1}(1+\varepsilon)\}$. Note that it is possible that $c = \sigma^{-1}(1+\varepsilon)$. If this is the case, then region B does not exist.

When $\theta_t \in C$, we know that there is a positive probability of choosing action 1 forever and thus converging (using the same proof as the two-armed bandit with constant baseline).

Finally, for the middle region $B = [a, c)$ ($a = \sigma^{-1}(1+\varepsilon)$), we know that the updates for any $\theta \in B$ are uniformly bounded in magnitude by a constant u.

We define a stopping time $\tau = \inf\{t; \theta_t \leq a \text{ or } \theta_t \geq c\}$. This gives the first time θ_t exits the region B. Let "$\wedge$" denote the min operator.

Since the updates are bounded, we can apply Azuma's inequality to the stopped martingale $\theta_{t\wedge\tau} - \alpha(t \wedge \tau)$, for $\lambda \in \mathbb{R}$.

$$\begin{aligned} P(\theta_{t\wedge\tau} - \alpha(t\wedge\tau) < \lambda) &\leq \exp\left(\frac{-\lambda^2}{2tu}\right) \\ P(\theta_{t\wedge\tau} - \alpha(t-(t\wedge\tau)) \leq c) &< \exp\left(-\frac{(c+\alpha t)^2}{2tu}\right) \end{aligned}$$

The second line follows from substituting $\lambda = -\alpha t + c$. Note that the RHS goes to 0 as t goes to ∞.

Next, we continue from the LHS. Let $\theta_t^* = \sup_{0 \le n \le t} \theta_n$

$$
\begin{aligned}
& P(\theta_{t\wedge\tau} - \alpha(t - (t \wedge \tau)) < c) \\
& \geq P(\theta_{t\wedge\tau} - \alpha(t - (t \wedge \tau)) < c, t \leq \tau) \\
& \quad + P(\theta_{t\wedge\tau} - \alpha(t - (t \wedge \tau)) < c, t > \tau), \quad \text{splitting over events} \\
& \geq P(\theta_{t\wedge\tau} < c, t < \tau), \quad \text{dropping the second term} \\
& \geq P(\theta_t < c, \sup \theta_t < c, \inf \theta_t < a), \quad \text{definition of } \tau \\
& = P(\sup \theta_t < c, \inf \theta_t < a), \quad \text{this event is a subset of the other} \\
& = P(\tau > t)
\end{aligned}
$$

Hence the probability the stopping time exceeds t goes to 0 and it is guaranteed to be finite almost surely.

Now, if θ_t exits B, there is some positive probability that it reached C. We see this by considering that taking action 1 increases θ by at least a constant, so the sequence of only taking action 1 until θ_t reaches C has positive probability. This is a lower bound on the probability of eventually reaching C given that θ_t is in B.

Finally, we combine the results for all three regions to show that convergence happens with probability 1. Without loss of generality, suppose $\theta_0 \in A$. If that is not the case, then keep running the process until either θ_t is in A or convergence occurs.

Let E_i be the event that θ_t returns to A after leaving it for the i-th time. Then $E_i^\complement$ is the event that $\theta_t \to \infty$ (convergence occurs). This is the case because, when $\theta_t \in C$, those are the only two options and, when $\theta_t \in B$ we had shown that the process must exit B with probability 1, either landing in A or C.

Next, we note that $P(E_i^\complement) > 0$ since, when θ_t is in B, the process has positive probability of reaching C. Finally, when $\theta_t \in C$, the process has positive probability of converging. Hence, $P(E_i^\complement) > 0$.

To complete the argument, whenever E_i occurs, then θ_t is back in A and will eventually leave it almost surely. Since the process is Markov and memoryless, E_{i+1} is independent of E_i. Thus, by considering a geometric distribution with a success being E_i^C occurring, E_i^C will eventually occur with probability 1. In other words, θ_t goes to $+\infty$.

□

Lemma 4. *For the two-armed bandit with sigmoid parameterization, natural policy gradient and a perturbed minimum-variance baseline $b = 1 - p_t + \varepsilon$, with $\varepsilon = 0$, the policy converges to the optimal policy with probability 1.*

Proof. By directly writing the updates, we find that both updates are always equal to the expected natural policy gradient, so that $\theta_{t+1} = \theta_t + \alpha$ for any θ_t. Hence $\theta_t \to \infty$ as $t \to \infty$ with probability 1. □

Lemma 5. *For the two-armed bandit with sigmoid parameterization, natural policy gradient and a perturbed minimum-variance baseline $b = 1 - p_t + \varepsilon$, with $\varepsilon \in (0,1)$, the policy converges to the optimal policy in probability.*

PROOF. The overall idea is to ensure that the updates are always positive for some region $A = \{\theta : \theta > \theta_A\}$ then show that we reach this region with probability 1.

Recall that the possible updates when the parameter is θ_t are:

- $\theta_{t+1} = \theta_t + \alpha(1 - \frac{\varepsilon}{\sigma(\theta_t)})$ if we choose action 1, with probability $\sigma(\theta_t)$
- $\theta_{t+1} = \theta_t + \alpha(1 + \frac{\varepsilon}{1-\sigma(\theta_t)})$ if we choose action 2, with probability $1 - \sigma(\theta_t)$.

First, we observe that the update for action 2 is always positive. As for action 1, it is positive whenever $p \geq \varepsilon$, equivalently $\theta \geq \theta_A$, where $\theta_A = \sigma^{-1}(\varepsilon)$. Call this region $A = \{\theta : \theta > \theta_A (= \sigma^{-1}(\varepsilon))\}$.

If $\theta_t \in A$, then we can find a $\delta > 0$ such that the update is always greater than δ in the positive direction, no matter which action is sampled. So, using the same argument as for the $\varepsilon = 0$ case with steps of $+\delta$, we get convergence to the optimal policy (with only constant regret).

In the next part, we show that the iterates will enter the good region A with probability 1 to complete the proof. We may assume that $\theta_0 < \theta_A$ since if that is not the case, we are already done. The overall idea is to create a transformed process which stops once it reaches A and then show that the stopping time is finite with probability 1. This is done using the fact that the expected step is positive ($+\alpha$) along with Markov's inequality to bound the probability of going too far in the negative direction.

We start by considering a process equal to θ_t except it stops when it lands in A. Defining the stopping time $\tau = \inf\{t : \theta_t > \theta_A\}$ and "$\wedge$" by $a \wedge b = \min(a,b)$ for $a, b \in \mathbb{R}$, the process $\theta_{t\wedge\tau}$ has the desired property.

Due to the stopping condition, $\theta_{t\wedge\tau}$ will be bounded above and hence we can shift it in the negative direction to ensure that the values are all nonpositive. So we define $\tilde{\theta}_t = \theta_{t\wedge\tau} - C$ for all t, for some C to be determined.

Since we only stop the process $\{\theta_{t\wedge\tau}\}$ *after* reaching A, then we need to compute the largest value $\theta_{t\wedge\tau}$ can take after making an update which brings us inside the good region. In other words, we need to compute $\sup_\theta\{\theta + \alpha(1 + \frac{\varepsilon}{1-\sigma(\theta)}) : \theta \in A^\complement\}$. Fortunately, since the function to maximize is an increasing function of θ, the supremum is easily obtained by choosing the largest possible θ, that is $\theta = \sigma^{-1}(\varepsilon)$. This gives us that $C = \theta_A + U_A$, where $U_A = \alpha(1 + \frac{\varepsilon}{1-\varepsilon})$.

All together, we have $\tilde{\theta}_t = \theta_{t\wedge\tau} - \theta_A - U_A$. By construction, $\tilde{\theta}_t \leq 0$ for all t (note that by assumption, $\theta_0 < \theta_A$ which is equivalent to $\tilde{\theta}_0 < -U_A$ so the process starts at a negative value).

Next, we separate the expected update from the process. We form the nonpositive process $Y_t = \tilde{\theta}_t - \alpha(t \wedge \tau) = \theta_{t\wedge\tau} - U_A - \theta_A - \alpha(t \wedge \tau)$. This is a martingale as it is a stopped version of the martingale $\{\theta_t - U_A - \theta_A - \alpha t\}$.

Applying Markov's inequality, for $\lambda > 0$ we have:

$$P(Y_t \leq -\lambda) \leq -\frac{\mathbb{E}[Y_t]}{\lambda}$$

$$P(Y_t \leq -\lambda) \leq -\frac{Y_0}{\lambda}, \quad \text{since } \{Y_t\} \text{ is a martingale}$$

$$P(\theta_{\tau\wedge t} - \alpha(\tau \wedge t) - \theta_A - U_A \leq -\lambda) \leq \frac{\theta_A + U_A - \theta_0}{\lambda}$$

$$P(\theta_{\tau\wedge t} \leq \alpha(\tau \wedge t - t) + \theta_A) \leq \frac{\theta_A + U_A - \theta_0}{\alpha t + U_A}, \quad \text{choosing } \lambda = \alpha t + U_A$$

Note that the RHS goes to 0 as $t \to \infty$. We then manipulate the LHS to eventually get an upper bound on $P(t \leq \tau)$.

$$\begin{aligned}
&P(\theta_{\tau\wedge t} \leq \alpha(\tau \wedge t - t) + \theta_A) \\
&= P(\theta_{\tau\wedge t} \leq \alpha(\tau \wedge t - t) + \theta_A, t \leq \tau) + P(\theta_{\tau\wedge t} \leq \alpha(\tau \wedge t - t) + \theta_A, t > \tau), \quad \text{splitting over disjoint events} \\
&\geq P(\theta_{\tau\wedge t} \leq \alpha(\tau \wedge t - t), t \leq \tau), \quad \text{second term is nonnegative} \\
&= P(\theta_t \leq \theta_A, t \leq \tau), \quad \text{since } t \leq \tau \text{ in this event} \\
&= P(\theta_t \leq \theta_A, \sup_{0\leq n\leq t} \theta_n \leq \theta_A), \quad \text{by definition of } \tau \\
&\geq P(\sup_{0\leq n\leq t} \theta_n \leq \theta_A), \quad \text{this event is a subset of the other} \\
&= P(t \leq \tau)
\end{aligned}$$

Since the first line goes to 0, the last line goes to 0 and hence we have that θ_t will enter the good region with probability 1.

□

Note that there is no contradiction with the nonconvergence result for $\varepsilon < -1$ as we cannot use Markov's inequality to show that the probability that $\theta_t < c$ ($c > 0$) goes to 0. The argument for the $\varepsilon \in (0,1)$ case relies on being able to shift the iterates θ_t sufficiently left to construct a nonpositive process $\tilde{\theta}_t$. In the case of $\varepsilon < 0$, for $\theta < c$ ($c \in \mathbb{R}$), the right update $(1 - \frac{\varepsilon}{\sigma(\theta)})$ is unbounded hence we cannot guarantee the process will be nonpositive. As a sidenote, if we were to additionally clip the right update so that it is $\max(B, 1 - \frac{\varepsilon}{\sigma(\theta)})$ for some $B > 0$ to avoid this problem, this would still not allow this approach to be used because then we would no longer have a submartingale. The expected update would be negative for θ sufficiently negative.

Lemma 6. *For the two-armed bandit with sigmoid parameterization, natural policy gradient and a perturbed minimum-variance baseline $b = 1 - p_t + \varepsilon$, with $\varepsilon \geq 1$, we have that $P(\sup_{0\leq n\leq t} \theta_n > C) \to 1$ as $t \to \infty$ for any $C \in \mathbb{R}$.*

PROOF. We follow the same argument as in the $\varepsilon \in (0,1)$ case with a stopping time defined as $\tau = \inf\{t : \theta_t > c\}$ and using $\theta_A = c$, to show that

$$P\left(\sup_{0\leq n\leq t} \theta_t \leq c\right) \to 0$$

□

D.2.3. Convergence with vanilla policy gradient

In this section, we show that using vanilla PG on the two-armed bandit converges to the optimal policy in probability. This is shown for on-policy and off-policy sampling with importance sampling corrections. The idea to show optimality of policy gradient will be to use Azuma's inequality to prove that θ_t will concentrate around their mean $\mathbb{E}[\theta_t]$, which itself converges to the right arm.

We now proceed to prove the necessary requirements.

Lemma 7 (Bounded increments for vanilla PG)**.** *Assuming bounded rewards and a bounded baseline, the martingale $\{X_t\}$ associated with vanilla policy gradient has bounded increments*

$$|X_t - X_{t-1}| \leq C\alpha_t$$

PROOF. Then, the stochastic gradient estimate is

$$g_t = \begin{cases} (r_1 - b)(1 - p_t), \text{with probability } p_t, r_1 \sim P_1 \\ -(r_0 - b)p_t, \text{with probability } (1 - p_t), r_0 \sim P_0 \end{cases}$$

Furthermore, $\mathbb{E}[g_t|\theta_0] = \mathbb{E}[\mathbb{E}[g_t|\theta_t]|\theta_0] = \mathbb{E}[\Delta p_t(1 - p_t)|\theta_0]$. As the rewards are bounded, for $i = 0,1$, $\exists R_i > 0$ so that $|r_i| \leq R_i$

$$\begin{aligned} |X_t - X_{t-1}| &= |\sum_{i=1}^{t} \alpha_i(g_i - \mathbb{E}[g_i]) - \sum_{i=1}^{t-1} \alpha_i(g_i - \mathbb{E}[g_i])| \\ &= \alpha_t|g_t - \mathbb{E}[\Delta p_t(1 - p_t)]| \\ &\leq \alpha_t\big(|g_t| + |\mathbb{E}[\Delta p_t(1 - p_t)]|\big) \\ &\leq \alpha_t\big(\max(|r_1 - b|, |r_0 - b|) + |\mathbb{E}[\Delta p_t(1 - p_t)]|\big), \quad r_1 \sim P_1, r_0 \sim P_0 \\ &\leq \alpha_t\big(\max(|R_1| + |b|, |R_0| + |b|) + \frac{\Delta}{4}\big) \end{aligned}$$

Thus $|X_t - X_{t-1}| \leq C\alpha_t$

□

Lemma 8 (Bounded increments with IS)**.** *Assuming bounded rewards and a bounded baseline, the martingale* $\{X_t\}$ *associated with policy gradient with importance sampling distribution* q *such that* $\min\{q, 1-q\} \geq \varepsilon > 0$ *has bounded increments*

$$|X_t - X_{t-1}| \leq C\alpha_t$$

PROOF. Let us also call $\varepsilon > 0$ the lowest probability of sampling an arm under q.

Then, the stochastic gradient estimate is

$$g_t = \begin{cases} \frac{(r_1-b)p_t(1-p_t)}{q_t}, \text{with probability } q_t, r_1 \sim P_1 \\ -\frac{(r_0-b)p_t(1-p_t)}{1-q_t}, \text{with probability } (1-q_t), r_0 \sim P_0 \end{cases}$$

As the rewards are bounded, $\exists R_i > 0$ such that $|r_i| \leq R_i$ for all i

$$\begin{aligned} |X_t - X_{t-1}| &= |\sum_{i=1}^{t} \alpha_i(g_i - \mathbb{E}[g_i]) - \sum_{i=1}^{t-1} \alpha_i(g_i - \mathbb{E}[g_i])| \\ &= \alpha_t |g_t - \mathbb{E}[\Delta p_t(1-p_t)]| \\ &\leq \frac{\alpha_t\big(\max(|R_1|+|b|, |R_0|+|b|) + \Delta\big)}{4\varepsilon} \quad \text{as } q_t, 1-q_t \geq \varepsilon \end{aligned}$$

Thus $|X_t - X_{t-1}| \leq C\alpha_t$

□

We call non-singular importance sampling any importance sampling distribution so that the probability of each action is bounded below by a strictly positive constant.

Lemma 9. *For vanilla policy gradient and policy gradient with nonsingular importance sampling, the expected parameter* θ_t *has infinite limit. i.e. if* $\mu_1 \neq \mu_0$*,*

$$\lim_{t \to +\infty} \mathbb{E}[\theta_t - \theta_0] = +\infty$$

In other words, the expected parameter value converges to the optimal arm.

PROOF. We reason by contradiction. The contradiction stems from the fact that on one hand we know θ_t will become arbitrarily large with t with high probability as this setting satisfies the convergence conditions of stochastic optimization. On the other hand, because of Azuma's inequality, if the average θ_t were finite, we can show that θ_t cannot deviate arbitrarily far from its mean with probability 1. The contradiction will stem from the fact that the expected θ_t cannot have a finite limit.

We have $\theta_t - \theta_0 = \sum_{i=0}^{t} \alpha_i g_i$. Thus

$$\begin{aligned}
\mathbb{E}[\theta_t - \theta_0] &= \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i | \theta_0] \\
&= \sum_{i=0}^{t} \alpha_i \mathbb{E}[g_i|\theta_0] \\
&= \sum_{i=0}^{t} \alpha_i \mathbb{E}[\mathbb{E}[g_i|\theta_i]|\theta_0] \quad \text{using the law of total expectations} \\
&= \sum_{i=0}^{t} \alpha_i \mathbb{E}[\Delta p_i (1-p_i)|\theta_0]
\end{aligned}$$

where $\Delta = \mu_1 - \mu_0 > 0$ the optimality gap between the value of the arms. As it is a sum of positive terms, its limit is either positive and finite or $+\infty$.

(1) **Let us assume that** $\lim_{t\to+\infty} \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] = \beta > 0$.
As $\sum_{i=0}^{\infty} \alpha_i^2 = \gamma$, using Azuma-Hoeffing's inequality

$$\begin{aligned}
\mathbb{P}(\theta_t \geq M) &= \mathbb{P}(\theta_t - \theta_0 - \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] \geq M - \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] - \theta_0) \\
&\leq \exp\big(-\frac{(M - \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] - \theta_0)^2}{2\sum_{i=1}^{t} c_i^2}\big)
\end{aligned}$$

where $c_i = \alpha_i C$ like in the proposition above. And for $M > |\theta_0| + \beta + 2C\sqrt{\gamma \log 2}$ we have

$$\begin{aligned}
\lim_{t\to+\infty} M - \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] - \theta_0 &\geq |\theta_0| + \beta + 2C\sqrt{\gamma \log 2} - \beta - \theta_0 \\
&\geq 2C\sqrt{\gamma \log 2}
\end{aligned}$$

As $\sum_{i=0}^{\infty} c_i = \gamma C^2$, we have

$$\lim_{t\to+\infty} \frac{(M - \mathbb{E}[\sum_{i=0}^{t} \alpha_i g_i] - \theta_0)^2}{2\sum_{i=1}^{t} c_i^2} = \frac{4C^2\gamma \log 2}{2\gamma C^2} \geq 2\log 2 = \log 4$$

Therefore

$$\lim_{t\to+\infty} \mathbb{P}(\theta_t \geq M) \leq \frac{1}{4}$$

By a similar reasoning, we can show that

$$\lim_{t\to+\infty} \mathbb{P}(\theta_t \leq -M) \leq \frac{1}{4}$$

Thus

$$\lim_{t\to+\infty} \mathbb{P}(|\theta_t| \leq M) \geq \frac{1}{2}$$

i.e for any M large enough, the probability that $\{\theta_t\}$ is bounded by M is bigger than a strictly positive constant.

(2) Because policy gradient with diminishing stepsizes satisfies the convergence conditions defined by Bottou et al. (2018), we have that

$$\forall \varepsilon > 0, \mathbb{P}(\|\nabla J(\theta_t)\| \geq \varepsilon) \leq \frac{\mathbb{E}[\|\nabla J(\theta_t)\|^2]}{\varepsilon^2} \xrightarrow[t\to\infty]{} 0$$

(see proof of Corollary 4.11 by Bottou et al. (2018)). We also have $\|\nabla J(\theta_t)\| = \|\Delta\sigma(\theta_t)(1-\sigma(\theta_t))\| = \Delta\sigma(\theta_t)(1-\sigma(\theta_t))$ for $\Delta = \mu_1 - \mu_0 > 0$ for μ_1 (resp. μ_0) the expected value of the optimal (res. suboptimal arm). Furthermore, $f : \theta_t \mapsto \Delta\sigma(\theta_t)(1-\sigma(\theta_t))$ is symmetric, monotonically decreasing on $\mathbb{R}^+$ and takes values in $[0,\Delta/4]$. Let's call f^{-1} its inverse on $\mathbb{R}^+$.
We have that

$$\forall \varepsilon \in [0,\Delta/4],\ \Delta\sigma(\theta)(1-\sigma(\theta)) \geq \varepsilon \iff |\theta| \leq f^{-1}(\varepsilon)$$

Thus $\forall M > 0$,

$$\begin{aligned}
\mathbb{P}(|\theta_t| \leq M) &= \mathbb{P}(\|\nabla J(\theta_t)\| \geq f(M)) \\
&\leq \frac{\mathbb{E}[\|\nabla J(\theta_t)\|^2]}{(\Delta\sigma(M)(1-\sigma(M)))^2} \\
&\xrightarrow[t\to\infty]{} 0
\end{aligned}$$

Here we show that θ_t cannot be bounded by any constant with non-zero probability at $t \to \infty$. This contradicts the previous conclusion.

Therefore $\lim_{t\to+\infty} \mathbb{E}[\theta_t - \theta_0] = +\infty$

□

Proposition D.2.2 (Optimality of stochastic policy gradient on the 2-arm bandit)**.** *Policy gradient with stepsizes satisfying the Robbins-Monro conditions ($\sum_t \alpha_t = \infty, \sum_t \alpha_t^2 < \infty$) converges to the optimal arm.*

Note that this convergence result addresses the stochastic version of policy gradient, which is not covered by standard results for stochastic gradient algorithms due to the nonconvexity of the objective.

PROOF. We prove the statement using Azuma's inequality again. We can choose $\varepsilon = (1-\beta)\mathbb{E}[\sum_{i=0}^t \alpha_i g_i] \geq 0$ for $\beta \in]0,1[$.

$$\begin{aligned}
\mathbb{P}\left(\theta_t > \theta_0 + \beta\mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i]\right) &= \mathbb{P}\left(\theta_t - \mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i] - \theta_0 > \beta\mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i] - \mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i]\right) \\
&= 1 - \mathbb{P}\left(\theta_t - \theta_0 - \mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i] \leq -\varepsilon\right) \\
&= 1 - \mathbb{P}\Bigg(\underbrace{\theta_0 + \mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i] - \theta_t}_{\text{Martingale } X_t} \geq \varepsilon\Bigg) \\
&\geq 1 - \exp\left(-\frac{(1-\beta)^2\,\mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i]^2}{2\sum_{i=1}^{t}\alpha_i^2 C^2}\right)
\end{aligned}$$

Thus $\lim_{t\to\infty}\mathbb{P}\left(\theta_t > \theta_0 + \beta\mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i]\right) = 1$, as $\lim_{t\to\infty}\mathbb{E}[\sum_{i=0}^{t}\alpha_i g_i] = +\infty$ and $\sum_{t=0}^{\infty}\alpha_t^2 < +\infty$. Therefore $\lim_{t\to\infty}\theta_t = +\infty$ almost surely. □

D.3. Multi-armed bandit theory

Theorem 1. There exists a three-arm bandit where using the stochastic natural gradient on a softmax-parameterized policy with the minimum-variance baseline can lead to convergence to a suboptimal policy with probability $\rho > 0$, and there is a different baseline (with larger variance) which results in convergence to the optimal policy with probability 1.

Proof. The example of convergence to a suboptimal policy for the minimum-variance baseline and convergence to the optimal policy for a gap baseline are outlined in the next two subsections. □

D.3.1. Convergence issues with the minimum-variance baseline

Proposition D.3.1. *Consider a three-armed bandit with rewards of 1, 0.7 and 0. Let the policy be parameterized by a softmax ($\pi_i \propto e^{\theta_i}$) and optimized using natural policy gradient paired with the mininum-variance baseline. If the policy is initialized to be uniform random, there is a nonzero probability of choosing a suboptimal action forever and converging to a suboptimal policy.*

Proof. The policy probabilities are given by $\pi_i = \frac{e_i^\theta}{\sum_j e_j^\theta}$ for $i = 1,2,3$. Note that this parameterization is invariant to shifting all θ_i by a constant.

The natural policy gradient estimate for

The gradient for sampling arm i is given by $g_i = e_i - \pi$, where e_i is the vector of zeros except for a 1 in entry i. The Fisher information matrix can be computed to be $F =$

$diag(\pi) - \pi\pi^T$.
Since F is not invertible, then we can instead find the solutions to $Fx = g_i$ to obtain our updates. Solving this system gives us $x = \lambda e + \frac{1}{\pi_i} e_i$, where e is a vector of ones and $\lambda \in \mathbb{R}$ is a free parameter.

Next, we compute the minimum-variance baseline. Here, we have two main options. We can find the baseline that minimizes the variance of the sampled gradients g_i, the "standard" choice, or we can instead minimize the variance of the sampled *natural* gradients, $F^{-1}g_i$. We analyze both cases separately.

The minimum-variance baseline for gradients is given by $b^* = \frac{\mathbb{E}[R(\tau)||\nabla \log \pi(\tau)||^2]}{\mathbb{E}[||\nabla \log \pi(\tau)||^2]}$. In this case, $\nabla \log \pi_i = e_i - \pi$, where e_i is the i-th standard basis vector and π is a vector of policy probabilities. Then, $||\nabla \log \pi_i|| = (1-\pi_i)^2 + \pi_j^2 + \pi_k^2$, where π_j and π_k are the probabilities for the other two arms. This gives us

$$b^* = \frac{\sum_{i=1}^3 r_i w_i}{\sum_{i=1}^3 w_i}$$

where $w_i = ((1-\pi_i)^2 + \pi_j^2 + \pi_k^2)\pi_i$.

The proof idea is similar to that of the two-armed bandit. Recall that the rewards for the three actions are 1, 0.7 and 0. We will show that this it is possible to choose action 2 (which is suboptimal) forever.

To do so, it is enough to show that we make updates that increase θ_2 by at least δ at every step (and leave θ_1 and θ_3 the same). In this way, the probability of choosing action 2 increases sufficiently fast, that we can use the proof for the two-armed bandit to show that the probability of choosing action 2 forever is nonzero.

In more detail, suppose that we have established that, at each step, θ_2 increases by at least δ. The policy starts as the uniform distribution so we can choose any initial θ as long as three components are the same ($\theta_1 = \theta_2 = \theta_3$). Choosing the initialization $\theta_i = -\log(1/2)$ for all i, we see that $\pi_2 = \frac{e^{\theta_2}}{\sum_{i=1}^3 \theta_i} = \frac{e^{\theta_2}}{1+e^{\theta_2}} = \sigma(\theta_2)$ where $\sigma(.)$ is the sigmoid function. Since at the n-th step, $\theta_2 > \theta_0 + n\delta$, we can reuse the proof for the two-armed bandit to show $Pr(\text{action 2 forever}) > 0$.

To complete the proof, we need to show that the updates are indeed lower bounded by a constant. Every time we sample action 2, the update is $\theta \leftarrow \theta + \alpha(r_2 - b^*)(\lambda e + \frac{1}{\pi_2} e_2)$. We can choose any value of λ since they produce the same policy after an update due to the policy's invariance to a constant shift of all the parameters. We thus choose $\lambda = 0$ for simplicity. In summary, an update does $\theta_2 \leftarrow \theta_2 + \alpha(r_2 - b^*)\frac{1}{\pi_2}$ and leaves the other parameters unchanged.

In the next part, we use induction to show the updates are lower bounded at every step. For the base case, we need $r_2 - b^* > \delta$ for some $\delta > 0$. Since we initialize the policy

to be uniform, we can directly compute the value of $b^* \approx 0.57$, so the condition is satisfied for, say, $\delta = 0.1$.

For the inductive case, we assume that $r_2 - b^* > \delta$ for $\delta > 0$ and we will show that $r_2 - b^*_+ > \delta$ also, where b^*_+ is the baseline after an update. It suffices to show that $b^*_+ \leq b^*$.

To do so, we examine the ratio $\frac{w_2}{w_1}$ in b^* and show that this decreases. Let $\left(\frac{w_2}{w_1}\right)_+$ be the ratio after an update and let $c = r_2 - b^*$.

$$\begin{aligned}\left(\frac{w_2}{w_1}\right) &= \frac{2(\pi_1^2 + \pi_3^2 + \pi_1\pi_3)\pi_2}{2(\pi_2^2 + \pi_3^2 + \pi_2\pi_3)\pi_1} \\ &= \frac{(e^{2\theta_1} + e^{2\theta_3} + e^{\theta_1+\theta_3})e^{\theta_2}}{(e^{2\theta_2} + e^{2\theta_3} + e^{\theta_2+\theta_3})e^{\theta_1}} \\ \left(\frac{w_2}{w_1}\right)_+ &= \frac{(e^{2\theta_1} + e^{2\theta_3} + e^{\theta_1+\theta_3})e^{\theta_2+\frac{c}{\pi_2}}}{(e^{2\theta_2+2\frac{c}{\pi_2}} + e^{2\theta_3} + e^{\theta_2+\theta_3+\frac{c}{\pi_2}})e^{\theta_1}}\end{aligned}$$

We compare the ratio of these:

$$\begin{aligned}\frac{\left(\frac{w_2}{w_1}\right)_+}{\left(\frac{w_2}{w_1}\right)} &= \frac{e^{\theta_2+\frac{c}{\pi_2}}}{e^{\theta_2}} \frac{e^{2\theta_2} + e^{2\theta_3} + e^{\theta_2+\theta_3}}{e^{2\theta_2+2\frac{c}{\pi_2}} + e^{2\theta_3} + e^{\theta_2+\theta_3+\frac{c}{\pi_2}}} \\ &= \frac{e^{2\theta_2} + e^{2\theta_3} + e^{\theta_2+\theta_3}}{e^{2\theta_2+\frac{c}{\pi_2}} + e^{2\theta_3-\frac{c}{\pi_2}} + e^{\theta_2+\theta_3}} \\ &< \frac{e^{2\theta_2} + e^{2\theta_3} + e^{\theta_2+\theta_3}}{e^{2\theta_2+\delta} + e^{2\theta_3-\delta} + e^{\theta_2+\theta_3}}\end{aligned}$$

The last line follows by considering the function $f(z) = e^{x-z} + e^{y-z}$ for a fixed $x \leq y$. $f'(z) = -e^{x-z} + e^{y+z} > 0$ for all z, so $f(z)$ is an increasing function. By taking $x = 2\theta_2$ and $y = 2\theta_3$ ($\theta_2 \geq \theta_3$), along with the fact that $\frac{c}{\pi_2} > \delta$ (considering these as z values), then we we see that the denominator has increased in the last line and the inequality holds.

By the same argument, recalling that $\delta > 0$, we have that the last ratio is less than 1. Hence, $\left(\frac{w_2}{w_1}\right)_+ < \left(\frac{w_2}{w_1}\right)$.

Returning to the baseline, $b^* = \frac{w_1 r_1 + w_2 r_2 + w3 r_3}{w_1 + w_2 + w_3}$. We see that this is a convex combination of the rewards. Focusing on the (normalized) weight of r_2:

$$\begin{aligned}\frac{w_2}{w_1 + w_2 + w_3} &= \frac{w_2}{2w_1 + w_2} \\ &= \frac{w_2/w_1}{2 + w_2/w_1}\end{aligned}$$

The first line follows since $w_1 = w_3$ and the second by dividing the numerator and denominator by w_1. This is an increasing function of w_2/w_1 so decreasing the ratio will decrease

the normalized weight given to r_2. This, in turn, increases the weight on the other two rewards equally. As such, since the value of the baseline is under $r_2 = 0.7$ (recall it started at $b^* \approx 0.57$) and the average of r_1 and r_3 is 0.5, the baseline must decrease towards 0.5.

Thus, we have shown that the gap between r_2 and b^* remains at least δ and this completes the proof for the minimum-variance baseline of the gradients.

Next, we tackle the minimum-variance baseline for the updates. Recall that the natural gradient updates are of the form $x_i = \lambda e + \frac{1}{\pi_i} e_i$ for action i where e is a vector of ones and e_i is the i-th standard basis vector.

The minimum-variance baseline for updates is given by

$$b^* = \frac{\mathbb{E}[R_i ||x_i||^2]}{\mathbb{E}[||x_i||^2]}$$

We have that $||x_i||^2 = 2\lambda^2 = (\lambda + \frac{1}{\pi_i})^2$. At this point, we have to choose which value of λ to use since it will affect the baseline. The minimum-norm solution is a common choice (corresponding to use of the Moore-Penrose pseudoinverse of the Fisher information instead of the inverse). We also take a look at fixed values of λ, but we find that this requires an additional assumption $3\lambda^2 < 1/\pi_1^2$.

First, we consider the minimum-norm solution. We find that the minimum-norm solution gives $\frac{2}{3\pi_i^2}$ for $\lambda = \frac{-1}{3\pi_i^2}$.

We will reuse exactly the same argument as for the minimum-variance baseline for the gradients. The only difference is the formula for the baseline, so all we need to check is the that the ratio of the weights of the rewards decreases after one update, which implies that the baseline decreases after an update.

The baseline can be written as:

$$\begin{aligned} b^* &= \frac{\sum_{i=1}^3 r_i \frac{2}{3\pi_i^2} \pi_i}{\sum_{i=1}^3 \frac{2}{3\pi_i^2}} \\ &= \frac{\sum_{i=1}^3 r_i \frac{1}{\pi_i}}{\sum_{i=1}^3 \frac{1}{\pi_i}} \end{aligned}$$

So we have the weights $w_i = \frac{1}{\pi_i}$ and the ratio is

$$\begin{aligned} \left(\frac{w_2}{w_1}\right) &= \frac{\pi_1}{\pi_2} \\ &= \frac{e^{\theta_1}}{e^{\theta_2}} \\ &= e^{\theta_1 - \theta_2} \end{aligned}$$

So, after an update, we get

$$\left(\frac{w_2}{w_1}\right)_+ = e^{\theta_1-\theta_2-\frac{c}{\pi_2}}$$

for $c = \alpha(r_2 - b^*)$, which is less than the initial ratio. This completes the case where we use the minimum-norm update.

Finally, we deal with the case where $\lambda \in \mathbb{R}$ is a fixed constant. We don't expect this case to be very important as the minimum-norm solution is almost always chosen (the previous case). Again, we only need to check the ratio of the weights.

The weights are given by $w_i = (2\lambda^2 + (\lambda + \frac{1}{\pi_i})^2)\pi_i$

$$\begin{aligned}\left(\frac{w_2}{w_1}\right) &= \frac{(2\lambda^2 + (\lambda + \frac{1}{\pi_2})^2)\pi_2}{(2\lambda^2 + (\lambda + \frac{1}{\pi_1})^2)\pi_1} \\ &= \frac{2\lambda^2\pi_2 + (\lambda + \frac{1}{\pi_2})^2\pi_2}{2\lambda^2\pi_1 + (\lambda + \frac{1}{\pi_1})^2\pi_1}\end{aligned}$$

We know that after an update π_2 will increase and π_1 will decrease. So, we check the partial derivative of the ratio to assess its behaviour after an update.

$$\frac{d}{d\pi_1}\left(\frac{w_2}{w_1}\right) = -\frac{2\lambda^2\pi_2 + (\lambda + \frac{1}{\pi_2})^2\pi_2}{(2\lambda^2\pi_1 + (\lambda + \frac{1}{\pi_1})^2\pi_1}(3\lambda^2 - 1/\pi_1^2)$$

We need this to be an increasing function in π_1 so that a decrease in π_1 implies a decrease in the ratio. This is true when $3\lambda^2 < 1/\pi_1^2$. So, to ensure the ratio decreases after a step, we need an additional assumption on λ and π_1, which is that $3\lambda^2 < 1/\pi_1^2$. This is notably always satisfied for $\lambda = 0$.

□

D.3.2. Convergence with gap baselines

Proposition D.3.2. *For a three-arm bandit with deterministic rewards, choosing the baseline b so that $r_1 > b > r_2$ where r_1 (resp. r_2) is the value of the optimal (resp. second best) arm, natural policy gradient converges to the best arm almost surely.*

PROOF. Let us define $\Delta_i = r_i - b$ which is striclty positive for $i = 1$, stricly negative otherwise. Then the gradient on the parameter θ^i of arm i

$$g_t^i = \mathbf{1}_{\{A_t=i\}}\frac{\Delta_i}{\pi_t(i)},\ i \sim \pi_t(\cdot)$$

Its expectation is therefore

$$\mathbb{E}[\theta_t^i] \quad = \quad \alpha t \Delta_i + \theta_0^i$$

Also note that there is a nonzero probability of sampling each arm at $t = 0$: $\theta_0 \in \mathbb{R}^3$, $\pi_0(i) > 0$. Furthermore, $\pi_t(1) \geq \pi_0(1)$ as θ_1 is increasing and $\theta_i, i > 1$ decreasing because of the choice of our baseline. Indeed, the updates for arm 1 are always positive and negative for other arms.

For the martingale $X_t = \alpha \Delta_1 t + \theta_0^1 - \theta_t^1$, we have

$$|X_t - X_{t-1}| \leq \alpha \frac{\Delta_1}{\pi_0(1)}$$

thus satisfying the *bounded increments* assumption of Azuma's inequality. We can therefore show

$$\begin{aligned}
\mathbb{P}\big(\theta_t^1 > \frac{\alpha\Delta_1}{2} t + \theta_0^1\big) &= \mathbb{P}\big(\theta_t^1 - \alpha\Delta_1 t - \theta_0^1 > -\frac{\alpha\Delta_1}{2} t\big) \\
&= \mathbb{P}\big(X_t < \frac{\alpha\Delta_1}{2} t\big) \\
&= 1 - \mathbb{P}\big(X_t \geq \frac{\alpha\Delta_1}{2} t\big) \\
&\geq 1 - \exp\big(-\frac{(\frac{\alpha\Delta_1}{2} t)^2 \pi_0(1)^2}{2t\alpha^2\Delta_1^2}\big) \\
&\geq 1 - \exp\big(-\frac{\pi_0(1)^2}{8} t\big)
\end{aligned}$$

This shows that θ_t^1 converges to $+\infty$ almost surely while the $\theta_t^i, i > 1$ remain bounded by θ_0^i, hence we converge to the optimal policy almost surely.

□

D.3.3. Convergence with off-policy sampling

We show that using importance sampling with a separate behaviour policy can guarantee convergence to the optimal policy for a three-armed bandit.

Suppose we have an n-armed bandit where the rewards for choosing action i are distributed according to P_i, which has finite support and expectation r_i. Assume at the t-th round the behaviour policy selects each action i with probability $\mu_t(i)$. Then, if we draw

action i, the stochastic estimator for the natural policy gradient with importance sampling is equal to

$$g_t = \frac{R_i - b}{\mu_t(i)} \mathbf{1}_{\{A_t = i\}}$$

with probability $\mu_t(i)$ and R_i drawn from P_i.

We have that $\mathbb{E}[g_t] = r - be$, where r is a vector containing elements r_i and e is a vector of ones. We let $\mathbb{E}[g_t] = \Delta$ for notational convenience.

By subtracting the expected updates, we define the multivariate martingale $X_t = \theta_t - \theta_0 - \alpha\Delta t$. Note that the i-th dimension X_t^i is a martingale for all i.

Lemma 10 (Bounded increments)**.** *Suppose we have bounded rewards and a bounded baseline and a behaviour policy selecting all actions with probability at least ε_t at round t. Then, the martingale $\{X_t\}$ associated with natural policy gradient with importance sampling has bounded increments*

$$|X_t^i - X_{t-1}^i| \leq \frac{C\alpha}{\varepsilon_t}$$

for all dimensions i and some fixed constant C.

Proof. The updates and X_t are defined as above.

Furthermore $\mathbb{E}[g_t|\theta_0] = \mathbb{E}[\mathbb{E}[g_t|\theta_t]|\theta_0] = \Delta$. As the rewards are bounded, $\exists R_{max} > 0$ such that, for all actions i, $|R_i| \leq R_{max}$ with probability 1.

For the i-th dimension,

$$\begin{aligned}
|X_t^i - X_{t-1}^i| &= \alpha|g_t^i - |\Delta_i|| \\
&\leq \alpha\big(|g_t^i| + |\Delta_i|\big) \\
&\leq \alpha\big(\frac{|R_{max} - b|}{\varepsilon_t} + |\Delta_i|\big) \\
&\leq \alpha\frac{R_{max} + |b| + |\Delta_i|}{\varepsilon_t} \quad \text{as } \varepsilon_t \leq 1
\end{aligned}$$

Thus $|X_t^i - X_{t-1}^i| \leq \frac{C\alpha}{\varepsilon_t}$ for all i. □

Proposition D.3.3. *Consider a n-armed bandit with stochastic rewards with bounded support and a unique optimal action. The behaviour policy μ_t selects action i with probability $\mu_t(i)$ and let $\varepsilon_t = \min_i \mu_t(i)$. When using NPG with importance sampling and a bounded baseline b, if $\lim_{t\to\infty} t\,\varepsilon_t^2 = +\infty$, then the target policy π_t converges to the optimal policy in probability.*

Proof. Let $r_i = \mathbb{E}[R_i]$, the expected reward for choosing action i. Without loss of generality, we order the arms such that $r_1 > r_2 > \ldots > r_n$. Also, let $\Delta_i = r_i - b$, the expected natural gradient for arm i.

Next, we choose $\delta \in (0,1)$ such that $(1-\delta)\Delta_1 > (1+\delta)\Delta_j$. We apply Azuma's inequality to X_t^1, the martingale associated to the optimal action, with $\varepsilon = \alpha\delta\Delta_i t$.

$$\begin{aligned}
\mathbb{P}(\theta_t^1 \leq \theta_0^1 + \alpha(1-\delta)\Delta_1 t) &= \mathbb{P}(\theta_t^1 - \theta_0^1 - \alpha\Delta_1 t \leq -\alpha\delta\Delta_1 t) \\
&\leq \exp\left(-\frac{(\alpha\delta\Delta_1 t)^2 \varepsilon_t^2}{2t\alpha^2 C^2}\right) \\
&= \exp\left(-\frac{\delta^2\Delta_1^2}{2C^2} t\varepsilon_t^2\right)
\end{aligned}$$

Similarly, we can apply Azuma's inequality to actions $i \neq 1$ and obtain

$$\begin{aligned}
\mathbb{P}(\theta_t^i \geq \theta_0^i + \alpha(1+\delta)\Delta_i t) &= \mathbb{P}(\theta_t^i - \theta_0^i - \alpha\Delta_i t \geq \alpha\delta\Delta_i t) \\
&\leq \exp\left(-\frac{\delta^2\Delta_i^2}{2C^2} t\varepsilon_t^2\right)
\end{aligned}$$

Letting A be the event $\theta_t^1 \leq \theta_0^1 + \alpha(1-\delta)\Delta_1 t$ and B_i be the event that $\theta_t^i - \theta_0^i \geq \alpha(1+\delta)\Delta_i t$ for $i \neq 1$, we can apply the union bound to get

$$\mathbb{P}(A \cup B_1 \cup \ldots \cup B_n) \leq \sum_{i=1}^{n} \exp\left(-\frac{\delta^2\Delta_i^2}{2C^2} t\varepsilon_t^2\right)$$

The RHS goes to 0 when $\sum_{t\geq 0} t\varepsilon_t^2 = \infty$.

Notice that $A^\complement$ is the event $\theta_t^1 > \theta_0^1 + \alpha(1-\delta)\Delta_1 t$ and $B^\complement$ is the event $\theta_t^i < \theta_0^i + \alpha(1+\delta)\Delta_i t$. Then, inspecting the difference between θ_t^1 and θ_t^i, we have

$$\begin{aligned}
\theta_t^1 - \theta_t^i &> \theta_0^1 + \alpha(1-\delta)\Delta_1 t - (\theta_0^i + \alpha(1+\delta)\Delta_i t) \\
&= \theta_0^1 - \theta_0^i + \alpha((1-\delta)\Delta_1 - (1+\delta)\Delta_i)t
\end{aligned}$$

By our assumption on δ, the term within the parenthesis is positive and hence the difference grows to infinity as $t \to \infty$. Taken together with the above probability bound, we have convergence to the optimal policy in probability.

□

D.4. Other results

D.4.1. Minimum-variance baselines

For completeness, we include a derivation of the minimum-variance baseline for the trajectory policy gradient estimate (REINFORCE) and the state-action policy gradient estimator (with the true state-action values).

Trajectory estimator (REINFORCE)

We have that $\nabla J(\theta) = \mathbb{E}_{\tau\sim\pi}[R(\tau)\nabla\log\pi(\tau)] = \mathbb{E}_{\tau\sim\pi}[(R(\tau)-b)\nabla\log\pi(\tau)]$ and our estimator is $g = (R(\tau)-b)\nabla\log\pi(\tau)$ for a sampled τ for any fixed b. Then we would like to minimize the variance:

$$\begin{aligned} Var(g) &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[g]\|_2^2 \\ &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[(R(\tau)-b)\nabla\log\pi(\tau)]\|_2^2 \\ &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[R(\tau)\nabla\log\pi(\tau)]\|_2^2 \end{aligned}$$

The second equality follows since the baseline doesn't affect the bias of the estimator. Thus, since the second term does not contain b, we only need to optimize the first term.

Taking the derivative with respect to b, we have:

$$\begin{aligned} \frac{\partial}{\partial b}\mathbb{E}[\|g\|_2^2] &= \frac{\partial}{\partial b}\mathbb{E}[\|R(\tau)\nabla\log\pi(\tau)\|^2 - 2\cdot R(\tau)b\|\nabla\log\pi(\tau)]\|^2 + b^2\|\nabla\log\pi(\tau)]\|^2] \\ &= 2\left(b\cdot\mathbb{E}[\|\nabla\log\pi(\tau)]\|^2] - \mathbb{E}[R(\tau)\|\nabla\log\pi(\tau)]\|^2]\right) \end{aligned}$$

The minimum of the variance can then be obtained by finding the baseline b^* for which the gradient is 0, i.e

$$b^* = \frac{\mathbb{E}[R(\tau)\|\nabla\log\pi(\tau)]\|^2]}{\mathbb{E}[\|\nabla\log\pi(\tau)]\|^2]}$$

State-action estimator (actor-critic)

In this setting we assume access to the Q-value for each state-action pair $Q^\pi(s,a)$, in that case the update rule is $\nabla J(\theta) = \mathbb{E}_{s,a\sim d^\pi}[Q^\pi(s,a)\nabla\log\pi(a|s)] = \mathbb{E}_{s,a\sim d^\pi}[(Q^\pi(s,a) - b(s))\nabla\log\pi(a|s)]$ and our estimator is $g = (Q^\pi(s,a)-b(s))\nabla\log\pi(a|s)$ for a sampled s,a. We will now derive the best baseline for a given state s in the same manner as above

$$\begin{aligned} Var(g|s) &= \mathbb{E}_{a\sim\pi}[\|g\|^2] - \|\mathbb{E}_{a\sim\pi}[g]\|^2 \\ &= \mathbb{E}_{a\sim\pi}[\|g\|^2] - \|\mathbb{E}_{a\sim\pi}[Q^\pi(s,a)\nabla\log\pi(a|s)]\|^2 \end{aligned}$$

So that we only need to take into account the first term.

$$\begin{aligned} \frac{\partial}{\partial b}\mathbb{E}_{a\sim\pi}[\|g\|^2] &= \frac{\partial}{\partial b}\mathbb{E}_{a\sim\pi}[\|Q^\pi(s,a)\nabla\log\pi(a|s))\|^2 - 2\cdot Q^\pi(s,a)b(s)\|\nabla\log\pi(a|s)]\|^2 + b(s)^2\|\nabla\log\pi(a|s)]\|^2] \\ &= 2\left(b(s)\cdot\mathbb{E}[\|\nabla\log\pi(a|s)]\|^2] - \mathbb{E}[Q^\pi(s,a)\|\nabla\log\pi(a|s)]\|^2]\right) \end{aligned}$$

Therefore the baseline that minimizes the variance for each state is

$$b^*(s) = \frac{\mathbb{E}[Q^\pi(s,a)\|\nabla \log \pi(a|s)]\|^2]}{\mathbb{E}[\|\nabla \log \pi(a|s)]\|^2])}$$

Note that for the natural policy gradient, the exact same derivation holds and we obtain that

$$b^*(s) = \frac{\mathbb{E}[Q^\pi(s,a)\|F_s^{-1}\nabla \log \pi(a|s)]\|^2]}{\mathbb{E}[\|F_s^{-1}\nabla \log \pi(a|s)]\|^2])}$$

where $F_s^{-1} = \mathbb{E}_{a\sim\pi(\cdot,s)}[\nabla \log \pi(a|s)\nabla \log \pi(a|s)^\top]$

D.4.2. Natural policy gradient for softmax policy in bandits

We derive the natural policy gradient estimator for the multi-armed bandit with softmax parameterization.

The gradient for sampling arm i is given by $g_i = e_i - \pi$, where e_i is the vector of zeros except for a 1 in entry i. The Fisher information matrix can be computed to be $F = diag(\pi) - \pi\pi^T$, where $diag(\pi)$ is a diagonal matrix containing π_i as the i-th diagonal entry. Since F is not invertible, then we can instead find the solutions to $Fx = g_i$ to obtain our updates. Solving this system gives us $x = \lambda e + \frac{1}{\pi_i}e_i$, where e is a vector of ones and $\lambda \in \mathbb{R}$ is a free parameter. Since the softmax policy is invariant to the addition of a constant to all the parameters, we can choose any value for λ.

D.4.3. Link between minimum variance baseline and value function

We show here a simple link between the minimum variance baseline and the value function. While we prove this for the REINFORCE estimator, a similar relation holds for the state-action value estimator.

$$\begin{aligned}
b^* &= \frac{\mathbb{E}[R(\tau)\|\nabla \log \pi(\tau)]\|^2]}{\mathbb{E}[\|\nabla \log \pi(\tau)]\|^2]} \\
&= \frac{\mathbb{E}[R(\tau)\|\nabla \log \pi(\tau)]\|^2]}{\mathbb{E}[\|\nabla \log \pi(\tau)]\|^2]} - V^\pi + V^\pi \\
&= \frac{\mathbb{E}[R(\tau)\|\nabla \log \pi(\tau)]\|^2] - \mathbb{E}[R(\tau)]\mathbb{E}[\|\nabla \log \pi(\tau)]\|^2}{\mathbb{E}[\|\nabla \log \pi(\tau)]\|^2]} + V^\pi \\
&= \frac{\mathrm{Cov}\big(R(\tau), \|\nabla \log \pi(\tau)]\|^2\big)}{\mathbb{E}[\|\nabla \log \pi(\tau)]\|^2]} + V^\pi
\end{aligned}$$

D.4.4. Variance of perturbed minimum-variance baselines

Here, we show that the variance of the policy gradient estimator is equal for baselines $b_+ = b^* + \varepsilon$ and $b_- = b^* - \varepsilon$, where $\varepsilon > 0$ and b^* is the minimum-variance baseline. We will use the trajectory estimator here but the same argument applies for the state-action estimator.

We have $g = R(\tau) - b)\nabla \log \pi(\tau)$ and the variance is given by

$$\begin{aligned} Var(g) &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[g]\|_2^2 \\ &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[(R(\tau) - b)\nabla \log \pi(\tau)]\|_2^2 \\ &= \mathbb{E}[\|g\|_2^2] - \|\mathbb{E}[R(\tau)\nabla \log \pi(\tau)]\|_2^2 \end{aligned}$$

where the third line follows since the baseline does not affect the bias of the policy gradient.

Focusing on the first term:

$$\begin{aligned} \mathbb{E}[||g||_2^2||] &= \mathbb{E}[R(\tau) - b)\nabla \log \pi(\tau)] \\ &= \mathbb{E}[(R(\tau) - b)^2||\nabla \log \pi(\tau)||_2^2] \\ &= \sum_\tau (R(\tau) - b)^2||\nabla \log \pi(\tau)||_2^2 \pi(\tau) \end{aligned}$$

Since $(R(\tau) - b)^2$ is a convex quadratic in b and $||\nabla \log \pi(\tau)||_2^2 \pi(\tau)$ is a positive constant for a fixed τ, the sum of these terms is also a convex quadratic in b. Hence, it can be rewritten in vertex form $\mathbb{E}[||g||_2^2||] = a(b - b_0)^2 + k$ for some $a > 0$, $b_0, k \in \mathbb{R}$.

We see that the minimum is achieved at $b^* = b_0$ (in fact, b_0 is equal to the previously-derived expression for the minimum-variance baseline). Thus, choosing baselines $b_+ = b^* + \varepsilon$ or $b_- = b^* - \varepsilon$ result in identical expressions $\mathbb{E}[||g||_2^2||] = a\varepsilon^2 + k$ and therefore yield identical variance.

Note this derivation also applies for the natural policy gradient. The only change would be the substitution of $\nabla \log \pi(\tau)$ by $F^{-1}\nabla \log \pi(\tau)$ where $F = \mathbb{E}_{s_t \sim d_\pi, a_t \sim \pi}[\nabla \log \pi(a_t|s_t)\nabla \log \pi(a_t|s_t)^\top]$

D.4.5. Baseline for natural policy gradient and softmax policies

We show that introducing a baseline does not affect the bias of the stochastic estimate of the natural policy gradient. The estimator is given by $g = (R_i - b)F^{-1}\nabla \log \pi(a_i)$, where $F^{-1} = \mathbb{E}_{a \sim \pi}[\nabla \log \pi(a)\nabla \log \pi(a)^\top]$.

For a softmax policy, this is: $g = (R_i - b)(\frac{1}{\pi_\theta(i)} e_i + \lambda e)$, where e_i is a vector containing a 1 at position i and 0 otherwise, e is a vector of all one and λ is an arbitrary constant.

Checking the expectation, we see that

$$\begin{aligned}\mathbb{E}[g] &= \mathbb{E}[(R_i - b)\left(\frac{1}{\pi_\theta(a_i)}e_i + \lambda e\right)] \\ &= \mathbb{E}[R_i\left(\frac{1}{\pi_\theta(a_i)}e_i + \lambda e\right)] - b\mathbb{E}[\left(\frac{1}{\pi_\theta(a_i)}e_i + \lambda e\right)] \\ &= \mathbb{E}[R_i\left(\frac{1}{\pi_\theta(a_i)}e_i + \lambda e\right)] - b(e + \lambda e)\end{aligned}$$

So the baseline only causes a constant shift in all the parameters. But for the softmax parameterization, adding a constant to all the parameters does not affect the policy, so the updates remained unbiased. In other words, we can always add a constant vector to the update to ensure the expected update to θ does not change, without changing the policy obtained after an update.

D.4.6. Natural policy gradient estimator for MDPs

In this section, we provide a detailed derivation of the natural policy gradient with Q-values estimate used in the MDP experiments.

Suppose we have a policy π_θ. Then, the (true) natural policy gradient is given by $u = F^{-1}(\theta)\nabla J(\theta)$ where $F(\theta) = \mathbb{E}_{s\sim d_{\pi_\theta}}[F_s(\theta)]$ and $F_s(\theta) = \mathbb{E}_{a\sim\pi}[\nabla \log \pi(a|s) \nabla \log \pi(a|s)^\top]$. We want to approximate these quantities with trajectories gathered with the current policy. Assuming that we have a tabular representation for the policy (one parameter for every state-action pair), our estimators for a single trajectory of experience $(s_0, a_0, r_0, ..., s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ are as follows: $\hat{F} = \frac{1}{T}\sum_{i=0}^{T-1} F(s_i)$ and $\widehat{\nabla J} = \frac{1}{T}\sum_{i=0}^{T-1}(Q_\pi(s_i,a_i) - b(s))\nabla \log \pi(a_i|s_i)$.

Together, our estimate of the policy gradient is

$$\begin{aligned}\hat{F}^{-1}\widehat{\nabla J} &= \left(\frac{1}{T}\sum_{i=0}^{T-1} F(s_i)\right)^{-1}\left(\frac{1}{T}\sum_{i=0}^{T-1}(Q_\pi(s_i,a_i) - b(s))\nabla \log \pi(a_i|s_i)\right) \\ &= \left(\sum_{i=0}^{T-1} F(s_i)\right)^{-1}\left(\sum_{i=0}^{T-1}(Q_\pi(s_i,a_i) - b(s))\nabla \log \pi(a_i|s_i)\right)\end{aligned}$$

Since we have a tabular representation, $F(s_i)$ is a block diagonal matrix where each block corresponds to one state and $F(s_i)$ contains nonzero entries only for the block corresponding to state s_i. Hence, the sum is a block diagonal matrix with nonzero entries corresponding to the blocks of states $s_0, ..., s_{T-1}$ and we can invert the sum by inverting the

blocks. It follows that the inverse of the sum is the sum of the inverses.

$$= \left(\sum_{i=0}^{T-1} F(s_i)^{-1} \right) \left(\sum_{i=0}^{T-1} (Q_\pi(s_i,a_i) - b(s)) \nabla \log \pi(a_i|s_i) \right)$$
$$= \sum_{i=0}^{T-1} (Q_\pi(s_i,a_i) - b(s)) \left(\sum_{j=0}^{T-1} F(s_j)^{-1} \right) \nabla \log \pi(a_i|s_i)$$

Finally, we notice that $\nabla \log \pi(a_i|s_i)$ is a vector of zeros except for the entries corresponding to state s_i. So, $F(s_j)^{-1} \nabla \log \pi(a_i|s_i)$ is nonzero only if $i = j$ giving us our final estimator

$$\hat{u} = \sum_{i=0}^{T-1} (Q_\pi(s_i,a_i) - b(s)) F(s_i)^{-1} \nabla \log \pi(a_i|s_i).$$

Note that this is the same as applying the natural gradient update for bandits at each sampled state s, where the rewards for each action is given by $Q_\pi(s,a)$.

D.4.7. Connection between optimistic initialization and positive baseline perturbations

Using a positive perturbation to the baseline seems reminiscent of optimistic initialization for value-based methods like Q-learning, but there are some key differences. For optimistic initialization, the expected Q-learning/TD-based update (averaged over all states and actions) is actually modified since we change the value estimates. But for policy gradient methods, the baseline has no effect on the expected update. Furthermore, for baselines, improved exploration is only seen after multiple updates. Meanwhile, optimistic initialization directly impacts the action selection to promote exploration. Although they are different, there may be deeper links between baselines and optimistic methods.