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

April 3, 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'esimateur 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 dans l'apprentissage par renforcement.

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

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.

Table des matières

Liste des tableaux

Liste des figures

Notation

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.

θ	$\triangleq$	Learnable parameters of a model
$\mathcal{L}$	$\triangleq$	Loss function
V	$\triangleq$	value function.
$\mathcal{D}_{\text{KL}}$	$\triangleq$	Kullback-Leibler Divergence.
a	$\triangleq$	Actions
r	$\triangleq$	Actions
a	$\triangleq$	Actions
$\tau_{1:T}$	$\triangleq$	$\{\mathbf{s}_i, \mathbf{a}_i\}_{i=1}^{T}$ the state-action pairs.
$\mathcal{O}_t$	$\triangleq$	Optimality variable.
$p(\mathcal{O}_t \mid \mathbf{s}_t, \mathbf{a}_t)$	$\triangleq$	$\exp(r(\mathbf{s}_t, \mathbf{a}_t))$ 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.

Remerciements

test

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 (**???**) and robotics (**?**).

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 (**?**). This analogy later inspired the Temporal Difference algorithm (**?**), 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 (**?**). Reinforcement learning is also strongly linked to fundamental notions in optimal control, such as Bellman equations (**??**), which eventually led to the development of highly successful algorithms like Q-Learning (**?**).

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, 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" (**???**) 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* (**???**), we show how we can use a particle filter method to estimate this distribution. This yield an 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 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 **??**.

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) $\mathbb{P}(a \leq X \leq b) = \int_a^b p(x)dx$

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 (**??**).

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. **?** 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 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

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

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 (**??????**).

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 $\mathcal{D}_{\text{train}} = \{x_i\}_{i=1\ldots N}$ 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

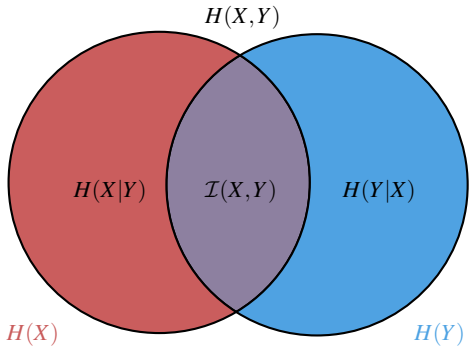


Fig. 1. Mutual information is the information shared between X and Y. From this Venn diagram, we recover the first two definitions of the mutual information in term of the joint and conditional entropy. For instance the red circle represents the entropy associated to X, but as the purple intersection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

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 hundred of billions/ a few trillions of parameters as of late 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 of 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 realted 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 estimate efficiently estimate $\nabla_\theta \mathcal{L}_\theta$ the gradient of $\mathcal{L}_\theta$ using the backpropagation algorithm (**?**). 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** (**?**) 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 (**?**).

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

This algorithm is the central idea behind all popular optimization algorithms used for training deep neural networks, such as Adam (**?**).

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 **?** and **?**. 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 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 $\gamma \in [0,1[$, the *discount factor* is here to ensure the objective remains bounded. Typically the policy π is parametrized by a neural network with weights θ, 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 $\{0, \dots, T\}$ **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
    $\tau_{0:t} \leftarrow \tau_{0:t-1} \cup \{s_t, a_t\}$
    $R_t \leftarrow R_{t-1} + \gamma^t r_t$
end for

Where a trajectory $\tau_{0:T}$ is the sequence of state-action pairs $\tau_{0:T} = \{(s_0,a_0), \dots, (s_T, a_T)\}$ 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 2 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 2, if ran with $T \to \infty$, would eventually sample states and actions according to d^π. If we are interested in the discounted return, we can adapt Algorithm 2 by adding a probability $1-\gamma$ of restarting from the initial distribution μ at every step. 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 though this noisy interactive process. Thus, it is not surprising that many of the algorithms used in reinforcement learning have the following structure

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 for the value it 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

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

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

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 aimed at evaluating.

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

Because of the Markov assumption implicitly, 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

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

In the same manner

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

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

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

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 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

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

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 exit non-parametric methods, often referred to as *memory-based* methods (**?**), 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

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

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

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

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 (**?**). 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*

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

This update rule using this pseudo-gradient is known as the TD(0) algorithm, which stands for **Temporal Difference** learning (**?**) 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 (**??**)

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

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 $Q^\pi(s,a)$, 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.

Alternatively we can use 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^π. Furthermore, for reasons motivated by variance reduction, it is common to subtract the value function at the current state $V_\phi(s)$ from the Q-function estimator. This gives us a stochastic gradient estimate $A_\phi(s,a)\nabla_\theta \log \pi_\theta(a|s)$ where $A_\phi(s,a)$ is called the *advantage function*. $A_\phi(s,a)$ is typically the difference between 1) a quantity whose expectation is the Q function and 2) the value function. The role of subtracting the value function in the advantage will be the subject of our contribution in Chapter 6. A_ϕ is also referred to as a *critic* as it will be positive when choosing a results in a higher value than the expectation of all actions according to the policy. On the other hand, the policy network π_θ chooses which action to perform and as such is called the *actor*. Thus this class of algorithms where we learn on set of parameters θ to act and another set of parameters ϕ to judge the value of the action is called **actor-critic algorithms.**

For instance here is a simple actor-critic algorithm using a 1-step return advantage

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

Algorithm 3 Simple actor critic algorithm with 1-step return

Choose initial θ_0, ϕ_0.
Sample first state $s_0 \sim \mu$
for t in $\{0, \dots, T\}$ **do**
    // Interact with the environment
    $a_t \sim \pi_{\theta_t}(\cdot|s_t)$
    $s_{t+1}, r_t \sim p_{\text{env}}(\cdot, \cdot|s_t, a_t)$
    // Policy evaluation
    $\phi_{t+1} = \phi_t - \eta_\phi \cdot \big(r_t + \gamma V_{\phi_t}(s_{t+1}) - V_{\phi_t}(s)\big)\nabla_{\phi_t} V_{\phi_t}(s)$
    // Policy improvement
    $\theta_{t+1} = \theta_t + \eta_\theta \cdot \big(r_t + \gamma V_{\phi_t}(s_{t+1}) - V_{\phi_t}(s)\big)\nabla_{\theta_t} \log \pi_{\theta_t}(a|s)$
end for

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 (**?**) 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 (**?**) 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 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 starting state and Q network*
$s_0 \sim \mu(\cdot)$
Initialize ϕ_0
for t in $\{0,\dots,\infty\}$ **do**
    // *Sample and execute action*
    $a_t \sim \pi_{\varepsilon-\text{greedy}}(Q_{\phi_t}(s_t,\cdot))$
    $s_{t+1}, r_t \sim p_{\text{env}}(\cdot|s_t,a_t)$
    // *Temporal Difference update on the greedy policy*
    $\phi_{t+1} \leftarrow \phi_t + \alpha\big(r_t + \gamma \max_{a'} Q_{\phi_t}(s_{t+1},a') - Q_{\phi_t}(s_t,a_t)\big)\nabla_{\phi_t} Q_{\phi_t}(s_t,a_t)$
end for

Note here that in the algorithm we sampled actions from $\pi_{\varepsilon-\text{greedy}}$. 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 (**?**), a slightly modified version of this algorithm for deep reinforcement learning was the first algorithm to reach human level performance on the ALE benchmark (**?**).

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 (**?**)

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

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 $d_\gamma^{\pi'}$ 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, **?** was published at RLDM 2017, a subsequent and longer paper, **?** was presented at the Montreal AI Symposium 2017, and finally, a latter and more theoretically sound version, **?** 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 is 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 (**??**). 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 indepedently 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) (**??????**).

The second point, learning disentangled representation for reinforcement learning has been investigated by **?** 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. **?** 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 **?**, where they use a mutual information objective to help exploration and learning of representation (this is however not unsupervised) and **?** 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 **?** 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 (**?**).

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 (**??**), and in the RL literature (**??**). 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 (**????**).

We build on a previously studied (**?**) 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 (**?**). 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 (**?**) , generative adversarial networks (**?**) or non-linear versions of ICA (**??**) 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 $\mathbb{E}_{p(\phi|h)}[\mathcal{S}(h, \phi)]$.

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 (**??**) we have

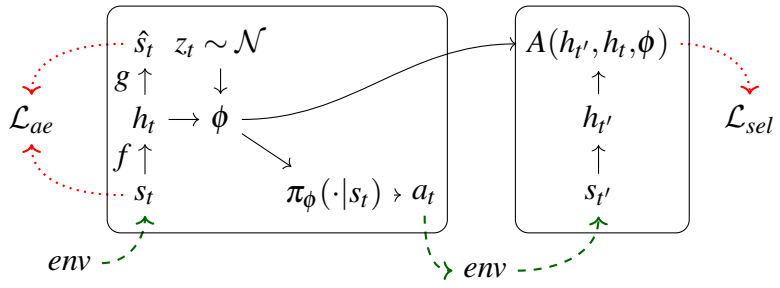


Fig. 1. The computational model of our architecture. s_t is the first state, from its encoding h_t and a noise distribution z, ϕ is generated. ϕ is used to compute the policy π_ϕ, which is used to act in the world. The sequence $h_t, h_{t'}$ is used to update our model through the selectivity loss, as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

$\mathcal{D}_{\mathrm{KL}}(p||q) = \sup_{A\in\mathcal{L}^\infty(q)} \mathbb{E}_p[\log A] - \log \mathbb{E}_q[A]$. Applying this equality to the mutual information $\mathcal{I}_p(\phi, h'|h) = \mathbb{E}_{p(h'|h)}\left[\mathcal{D}_{\mathrm{KL}}\left(p(\phi|h', h)||p(\phi|h)\right)\right]$ gives

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

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 (**?**) or directed (**?**) 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 **??** for details.

3.4. Experiments

We use MazeBase (**?**) 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

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

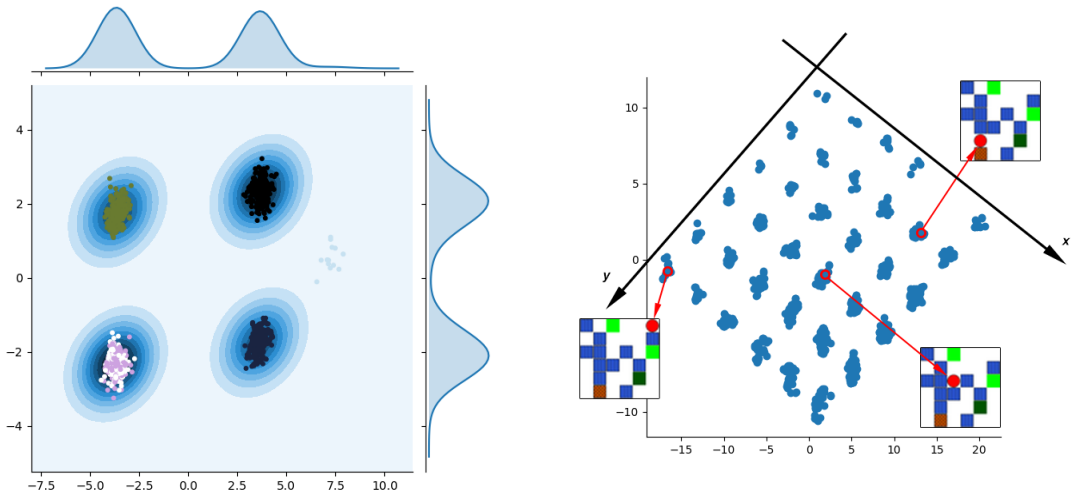


Fig. 2. (a) Sampling of 1000 variations $h' - h$ and its kernel density estimation encountered when sampling random controllable factors ϕ. We observe that our algorithm disentangles these representations on 4 main modes, each corresponding to the action that was actually taken by the agent.[1] (b) The 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.

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 3:

- 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 level. While we currently suffer from mode collapsing of some factors of variations, we

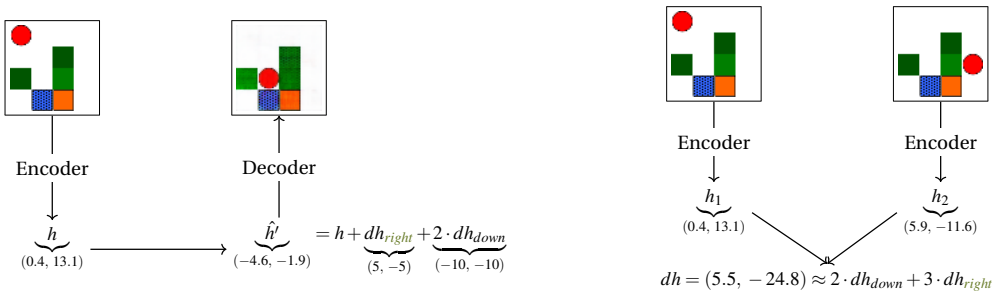


Fig. 3. (left) Predicting the effect of a cause on Mazebase. The leftmost image is the visual input of the environment, where the agent is the round circle, and the switch states are represented by shades of green. After the training, we are able to distinguish one cluster per dh (Figure 2), that is 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.

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.

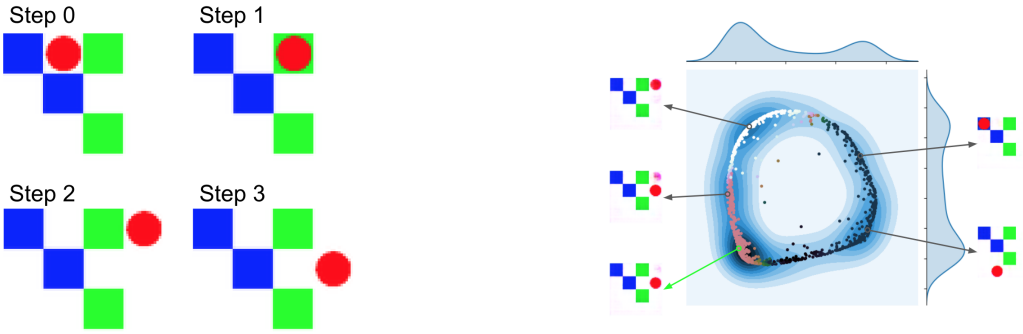


Fig. 4. (a) The actual 3-step trajectory done by the agent. (b) PCA view of the space $\phi(h_0, z), z \sim \mathcal{N}(0,1)$. Each arrow points to the reconstruction of the prediction $T_\theta(h_0, \phi)$ made by different ϕ. The ϕ at the start of the green arrow is the one used by the policy in (a). 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 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 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 (**?**) 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 (**??**). An evalution paper, (**?**), 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 (**?**) 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 plan ning 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. Made the link with the two-filter formula (**?**) and smoothing/forward-backward algorithms
- Determined the expression for the update (maximum entropy advantage) and wrote the proofs in the appendix
- Designed and performed the toy experiment Figure 4b and Figure 4a
- Made **??**, 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 (**?**) relied on Monte Carlo Tree Search (MCTS) (**?**) to achieve super human performances. Cross entropy methods (CEM) (**?**) have enabled robots to perform complex nonprehensile manipulations (**?**) and algorithms to play successfully Tetris (**?**). In addition, iterative linear quadratic regulator (iLQR) (**???**) enabled humanoid robots tasks to get up from an arbitrary seated pose (**?**).

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* (**???????**). 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) (**???**) 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 (**?**).

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 (**???**). First, we introduce an auxiliary binary random variable $\mathcal{O}_t$ denoting the "optimality" of a pair (s_t,a_t) at

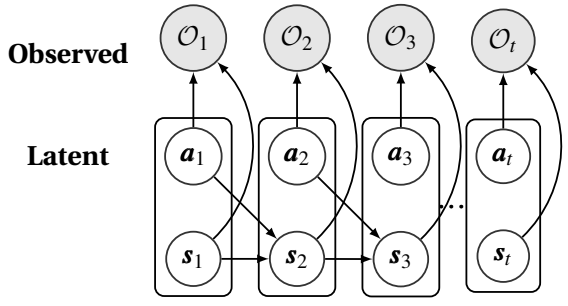


Fig. 1. $\mathcal{O}_t$ is an observed *optimality* variable with probability $p(\mathcal{O}_t|s_t,a_t) = \exp(r(s_t,a_t))$. $\tau_t = (s_t,a_t)$ are the state-action pair variables considered here as latent.

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: $\log p(\mathcal{O}_{1:T}) \geq \mathbb{E}_{\tau_{1:T} \sim q_\theta}[\sum_{t=1}^{T} r(s_t, a_t) - \log \pi_\theta(a_t|s_t)]$ (**??**). 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) (**?**) 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 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*

[1] as in **?**, 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 **?**. We typically consider this term either constant or already included in the reward function. See Appendix **??** for details.

$w(\tau) \propto \frac{p(\tau)}{q(\tau)}$ 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 $p(a_1, s_2, \ldots s_h, a_h|\mathcal{O}_{1:T}, s_1)$, 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* (**??**):

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

This corresponds to a forward-backward messages factorization in a Hidden Markov Model as depicted in Figure **??**. 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.

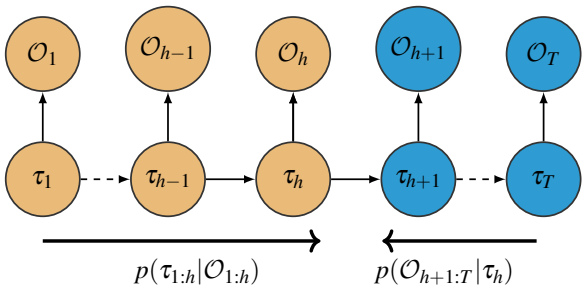


Fig. 2. Factorization of the HMM into forward (orange) and backward (blue) messages. Estimating the forward message is filtering, estimating the value of the latent knowing all the observations is smoothing.

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 (**????**). A more in depth explanation can be found in (**?**). We can show subsequently that:

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

Full details can be found in Appendix **??**. 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 (**?**).

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:

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

Where

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

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 **??**. Our algorithm is similar to the Auxiliary Particle Filter (**?**) 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 (**?**) 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.

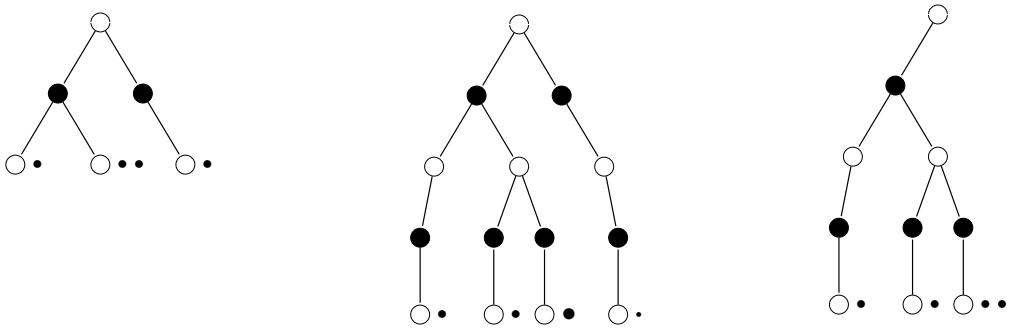
Algorithm 5 SMC Planning using SIR

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 **??** in Appendix **??**.

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



(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.

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, 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 (**?**). 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 (**?**) and the probabilistic version of LQR proposed in **?**.

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 $p_{\text{env}}(s_{t+1}|s_t,a_t,\mathcal{O}_{t:T})$ (**?**). 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

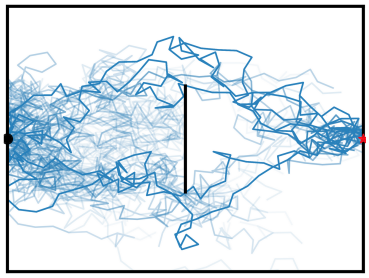
In this section, we show how SMCP can deal with multimodal policies when planning. We believe multimodality is useful for exploring since it allows us to keep a distribution over many promising trajectories and also allows us to adapt to changes in the environment e.g. if a path is suddenly blocked.

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.

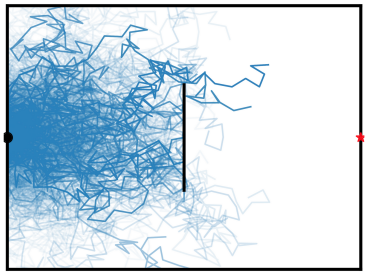
The proposal is an isotropic normal distribution for each planning algorithm, and since the environment's dynamics are known, there is no need for learning: the only difference between the three methods is how they handle planning. We also set the value function to 0 for SIR and SIS as we do not wish 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

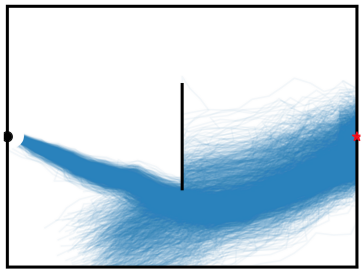
The experiments were conducted on the Open AI Gym Mujoco benchmark suite (**??**). To understand how planning can increase the learning speed of RL agents we focus on



(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.

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 **?**, 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.

We included two popular planning algorithms on Mujoco as baselines: CEM (**?**) and Random Shooting (RS) (**?**). Furthermore, we included SAC (**?**), 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 **?**, 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

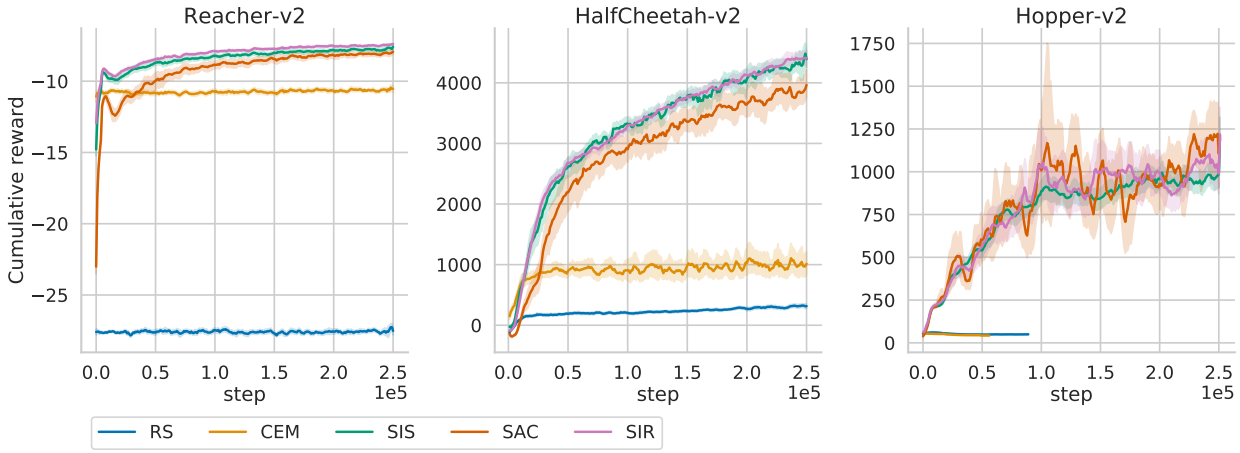


Fig. 5. Training curves on the Mujoco continuous control benchmarks. Sequential Monte Carlo Planning both with resampling (SIR) (pink) and without (SIS) (orange) learns faster than the Soft Actor-Critic model-free baseline (blue) and achieves higher asymptotic performances than the planning methods (Cross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

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 (**?**) or Particle Gibbs with Ancestor Sampling (PGAS) (**?**) have been proposed and using them would certainly improve our results.

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 **?** which validated our results by performing experiments at a larger scale than we did, **?** used our empirical results on the similarity between the empirical and the true Fisher matrix and **?** implemented the Takeuchi Information Criterion and approximations we developed for it as part of their package.

Personal contribution

- Made the link between our original idea and the already existing paper **?** which introduced the estimator first, was only published in Japanese and even reinvented later (**?**).
- Performed 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 (**?**). 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 (**??**) 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 **F** and the Hessian **H**, and the noise matrix, **C**;

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

- We briefly expand the convergence results of **?**, 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 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 **F** and the Hessian **H**, and one associated with noise, the gradients' uncentered covariance **C**. In particular, while **F** and **H** are well understood, **C** is often misinterpreted. For instance, it is often called "empirical Fisher" (**?**) despite bearing no relationship to **F**, the true Fisher. This confusion can have dire consequences and optimizers using **C** as approximation to **F** can have arbitrarily poor performance (**?**).

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 **H**, **F** and **C** are then defined as:

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

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

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

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

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

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

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

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

[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.

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** (**?**). In some recent works (**??**) 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:

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 **C** *does not converge to the actual Fisher matrix nor the Hessian. Hence we shall and will not refer to* **C** *as the "empirical Fisher" matrix.*

In some other settings, however, one can expect a stronger correlation between **C** and **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 **?** to full matrices, expliciting 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 **???**. 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 (**??**). Worthy of note, **??** 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 **?**, 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 (**?????**).

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: **?**) addresses these misunderstandings. It makes use of both the Hessian of the loss with respect to the parameters, $\mathbf{H}$, and the uncentered covariance of the gradients, $\mathbf{C}$. 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 $\mathbf{C}$ 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 **?**, 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 (**?**), 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 (**?**). 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: **?**):

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

where $\hat{\theta}^*$ is a local optimum. Note that $\mathbf{H}$ and $\mathbf{C}$ 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 **?** and similar criteria have been proposed since then (**??**). However, as far as we know, no experimental validation of this criterion has been carried out for deep networks. Indeed, for deep networks, $\mathbf{H}$ is highly degenerate, most of its eigenvalues being close to 0 (**?**). 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 **??**.

Interestingly, the term $\mathrm{Tr}(\mathbf{H}^{-1}\mathbf{C})$ appeared in several works before, whether as an upper bound on the suboptimality (**?**) or as a number of iterates required to reach a certain suboptimality (**?**). 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 (**?**) links the spectrum of the Hessian at a local optimum with the generalization gap. This correlation, observed again by **?**, was already shown to not hold in general (**?**). 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 (**?**) 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 (**?**) 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 **??**.

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

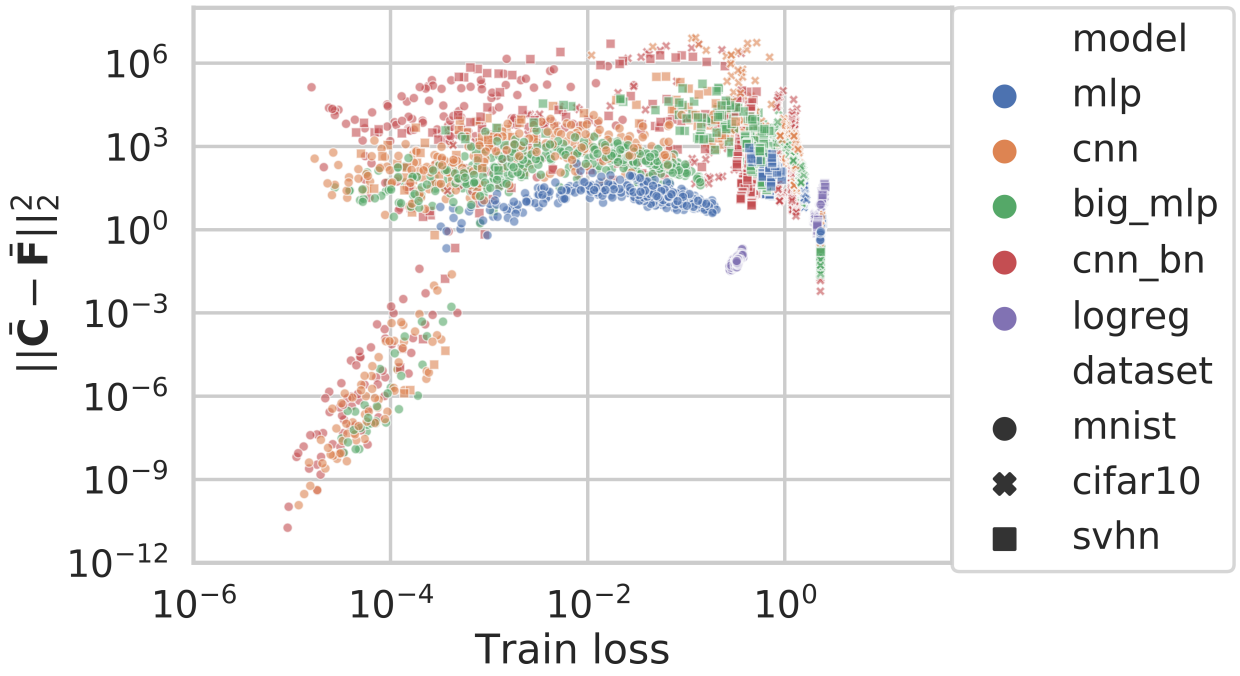


Fig. 1. Squared Frobenius norm between $\bar{\mathbf{F}}$ and $\bar{\mathbf{C}}$ (computed on the training distribution). Even for some low training losses, there can be a significant difference between the two matrices.

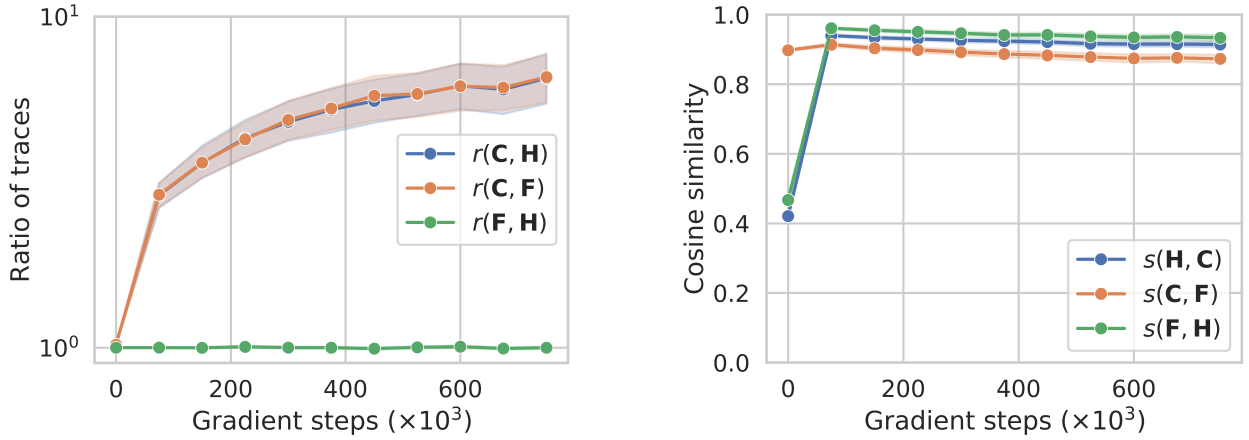


Fig. 2. Scale and angle similarities between information matrices.

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 $\mathbf{C} \propto \mathbf{F} \approx \mathbf{H}$.

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 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 **?**.

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 **?** 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.

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 $\mathbf{S} \propto \mathbf{H}^{\beta}$.

5.5.5. The TIC and the generalization gap

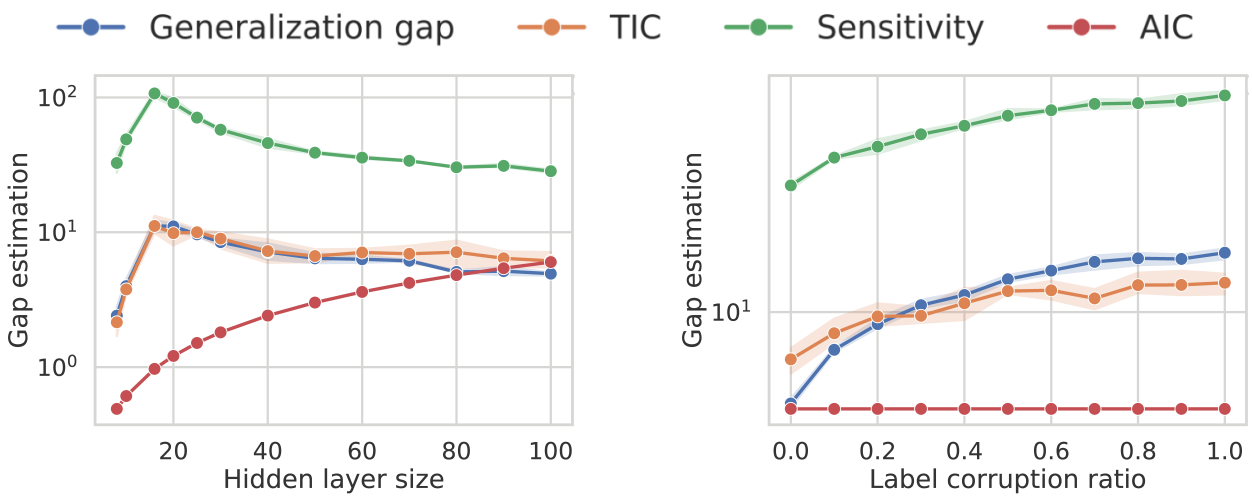
We now empirically test the quality of the TIC as an estimator of the generalization gap in deep networks. Following **?** we assess the behaviour of our generalization gap estimator by varying (1) the number of parameters in a model and (2) the label randomization ratio.

Experiments are performed using a fully connected feedforward network with a single hidden layer trained on a subset of 2k samples of SVHN (**?**). In Figure 3a we vary the number of units in the hidden layer without label randomization while in Figure **??** 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 **??**.

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.

5.5.5.1. Efficient approximations to the TIC

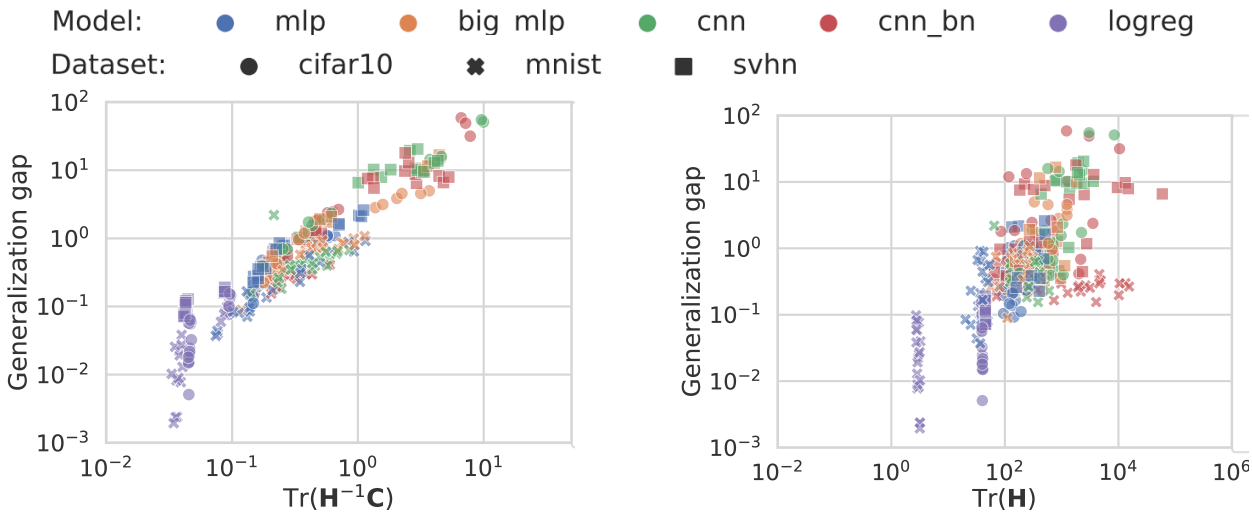
Although the TIC is a good estimate of the generalization gap, it can be expensive to compute on large models. Following our theoretical and empirical analysis of the proximity of $\mathbf{H}$ and $\mathbf{F}$, we propose two approximations to the TIC: $\mathrm{Tr}(\mathbf{F}^{-1}\mathbf{C})$ and $\mathrm{Tr}(\mathbf{C})/\mathrm{Tr}(\mathbf{F})$.



(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.



(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.

They are easier to compute as the **F** is in general easier to compute than **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 **H** with **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.

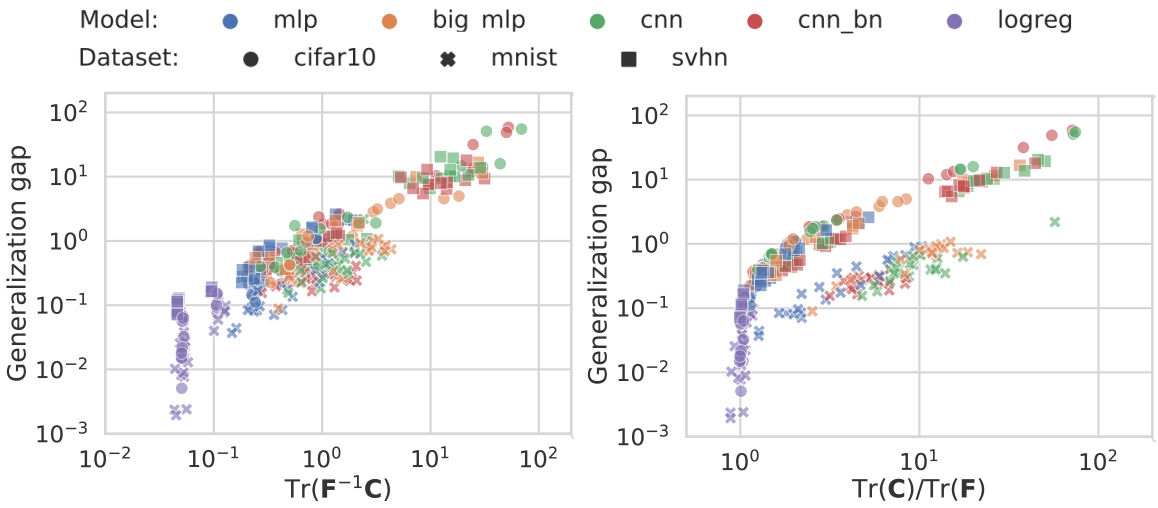


Fig. 5. Generalization gap as a function of two approximations to the Takeuchi Information Criterion: $\mathrm{Tr}(\mathbf{F}^{-1}\mathbf{C})$ (*left*) and $\mathrm{Tr}(\mathbf{C})/\mathrm{Tr}(\mathbf{F})$ (*right*).

Intuition on $\mathrm{Tr}(\mathbf{C})/\mathrm{Tr}(\mathbf{F})$: 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 $\mathrm{Tr}(\mathbf{F}^{-1}\mathbf{C}) = k\alpha$ where k is the dimension of the invertible subspace of $\mathbf{F}$ and $\mathrm{Tr}(\mathbf{C})/\mathrm{Tr}(\mathbf{F}) = d\alpha$ where d is the dimensionality of θ. So, up to a multiplicative constant (or an offset in log scale), we can expect 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 (**?**), 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 (**?**), 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 **?**.

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 V*, Chung W*, Machado M. C., Le Roux N. "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, **?** 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* (**?**) 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
- Collaborated with Wesley on the divergence proofs for the 3 arms setting
- Designed 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 **?**, 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 (**?**). 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 (**??**). 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 **?**.

We analyze the problem of learning to maximize the average reward, J, by gradient ascent:

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

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 J, represented by its curvature. Roughly speaking, the geometry dictates how effective true gradient ascent is at optimizing $J(\theta)$ 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 (**?**):

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

when 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 (**??**), referred to 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

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

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 (**?**) estimator is different and involves the norm of the gradient (**?**). Reducing variance has been the main motivation for many previous works on baselines (e.g., **?????**), 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.

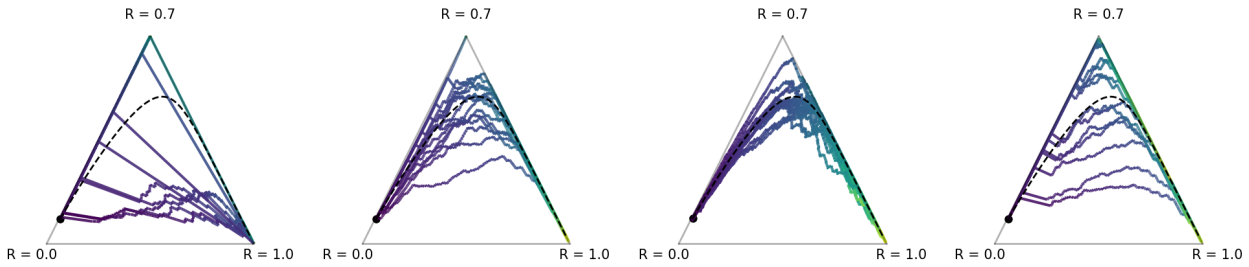
Contributions

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 (**?**).

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., **??**).

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.



(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 (**?**).

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 (**?**).

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 $b^*_\theta = \frac{\mathbb{E}[r(a_i)\|\nabla \log \pi_\theta(a_i)\|_2^2]}{\mathbb{E}[\|\nabla \log \pi_\theta(a_i)\|_2^2]}$ (**??**) (see Appendix **??** 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 **??** for details). Thus, we use these two perturbed baselines to demonstrate that there are phenomena in the optimization process that variance cannot explain.

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 **??**) 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 (**?**, 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 **??**). 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 **??** 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 (**?**), 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., **?**) or, in the stochastic case, assume that the variance of the updates is uniformly bounded for all

parameter values (e.g., **?**). 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 **??** (Prop. **??**).

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

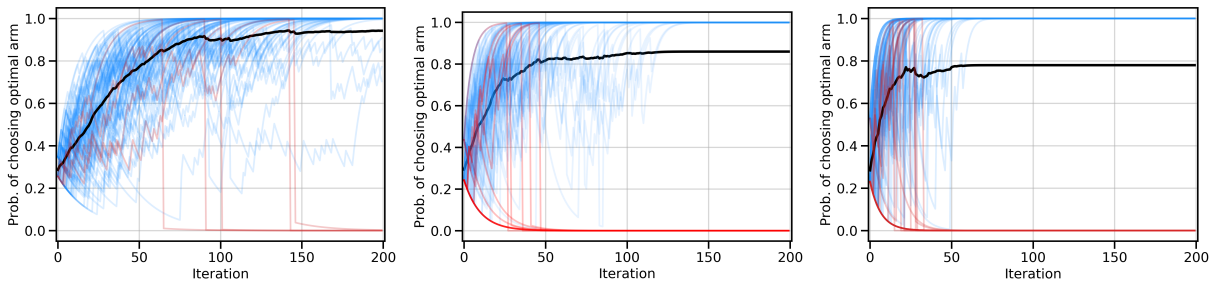
Proposition 6.3.1. *Consider a two-arm bandit with rewards* 1 *and* 0 *for the optimal and suboptimal arms, respectively. Suppose we use natural policy gradient starting from* θ_0*, with a fixed baseline* $b < 0$*, and fixed stepsize* $\alpha > 0$*. If the policy samples the optimal action with probability*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



(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.

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 **??**, 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.

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 (**?**), 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 (**?**). 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 (**??**). 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 $C > 0$,

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

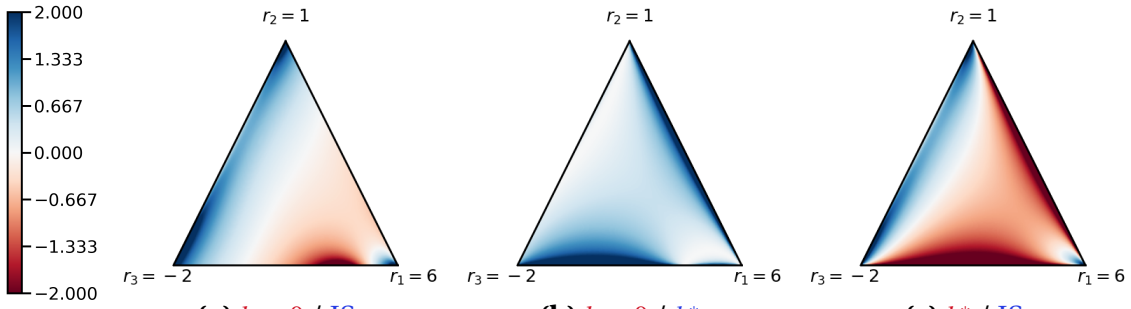
where θ^1 is the parameter associated to the optimal arm. Thus if $\lim_{t\to\infty} t\varepsilon_t^2 = +\infty$, the RHS goes to 1. In a similar manner, we can upper bound $\mathbb{P}\left(\theta_t^i \geq \theta_0^i + \alpha\delta\Delta_i t\right)$ for all suboptimal arms, and applying an union bound,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

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.



(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.

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., **?**), we advocate here for expanding the research on designing appropriate sampling distributions as done by **??** and **?**. This line of work has a long history in statistics (c.f., **?**).

6.4.3. Other mitigating strategies

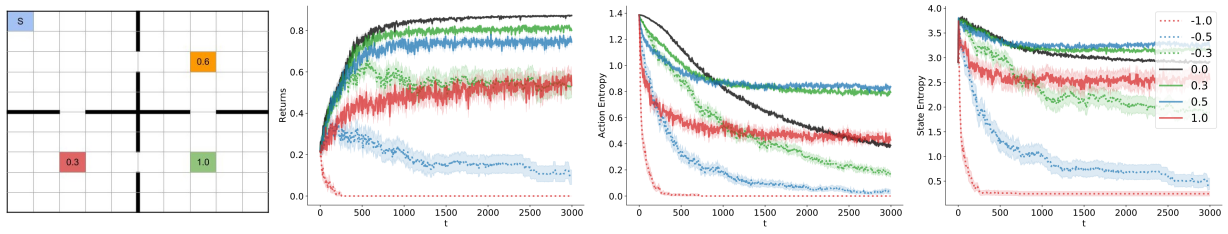
We conclude this section by discussing alternative strategies to mitigate the convergence issues. While they might be effective, and some are indeed used in practice, they are not without pitfalls.

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 **??** in Appendix **??**.

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 (**?**), 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. **?**). For example, by introducing bias in the updates, their variance can be reduced significantly **??**. Finally, other works have also developed provably convergent policy gradient algorithms using different mechanisms, such as exploration bonuses or ensembles of policies (**???**).



(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.

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 (**?**). 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

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

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. **?**). 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 **??** 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 **?**, 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 (**??**). Nevertheless, importance sampling in RL is generally seen as problematic (**?**) 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 **?**.

More broadly, this work suggests that treating bandit and reinforcement learning problems as a black-box optimization of a function $J(\theta)$ 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

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

Our first two contributions focused on reinforcement learning in unknown environments, examining how we can design algorithms that leverage the stochasticity of their policy and the environment to their advantage. In the first contribution, we delved into the unsupervised reinforcement learning setting, demonstrating how an agent can learn which aspects of the environment it can control independently, while simultaneously learning a disentangled latent representation of these factors. The second contribution targeted planning in continuous control tasks. By reframing reinforcement learning as an inference problem, we borrowed tools from the Sequential Monte Carlo literature to develop a theoretically grounded and efficient algorithm for probabilistic planning using a learned model of the world, showing how the agent can leverage SMC methods to imagine diverse solutions.

The latter two contributions analyzed the impact of gradient noise due to sampling in optimization algorithms. In our third contribution, we investigated the role of gradient noise in maximum likelihood estimation with stochastic gradient descent, exploring how the structure of the gradient noise and local curvature affect the generalization and convergence speed of the model. Lastly, our fourth contribution revisited reinforcement learning to understand the impact of sampling noise on the policy gradient algorithm. We found that sampling noise can significantly influence the optimization dynamics and the policies discovered in on-policy reinforcement learning.

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.

Références bibliographiques

Appendix A

Le titre

A.1. Section un de l'Annexe A

...texte...

Appendix B

Les différentes parties et leur ordre d'apparition

J'ajoute ici les différentes parties d'un mémoire ou d'une thèse ainsi que leur ordre d'apparition tel que décrit dans le guide de présentation des mémoires et des thèses de la Faculté des études supérieures. Pour plus d'information, consultez le guide sur le site web de la facutlé (www.fes.umontreal.ca).