Using Bifurcations for Diversity in Differentiable Games

Jonathan Lorraine [1 2 3] **Jack Parker-Holder** [1 4] **Paul Vicol** [2 3] **Aldo Pacchiano** [5] **Luke Metz** [6] **Tal Kachman** [7] **Jakob Foerster** [1]

Abstract

Ridge Rider (RR) is an algorithm for finding diverse solutions to optimization problems by following eigenvectors of the Hessian ("ridges"). RR is designed for conservative gradient systems (i.e. settings involving a single loss function), where it branches at saddles – the only relevant *bifurcation points*. We generalize this idea to non-conservative, multi-agent gradient systems by identifying new types of bifurcation points and proposing a method to follow eigenvectors with complex eigenvalues. We give theoretical motivation for our method – denoted Game Ridge Rider (GRR) – by leveraging machinery from the field of dynamical systems. Finally, we empirically motivate our method by constructing novel toy problems where we can visualize new phenomena and by finding diverse solutions in the iterated prisoners' dilemma, where existing methods often converge to a single solution mode.

1. Introduction

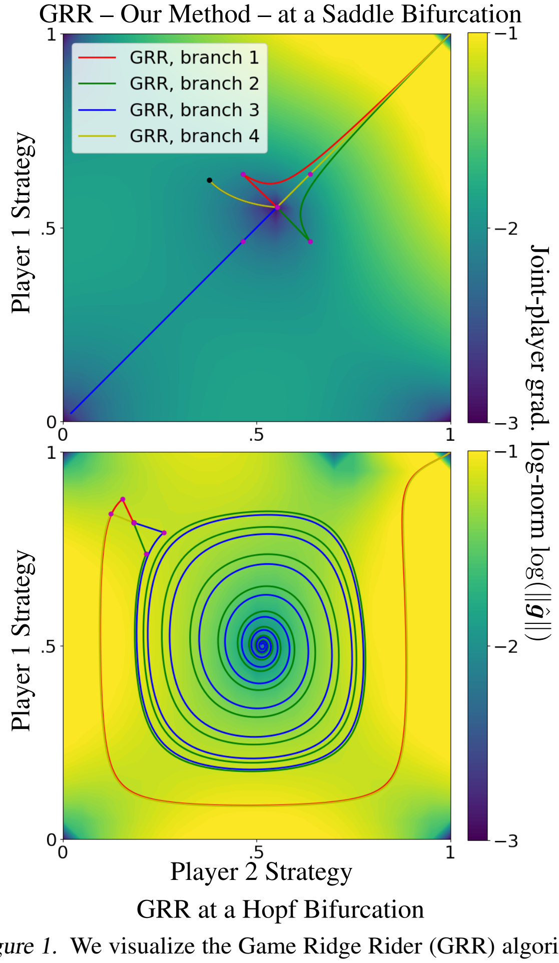
In machine learning, optimizing non-convex losses to local minima is critical in a variety of applications. Often, being able to select a *specific type of minimum* is important such as for policy optimization [1] or to avoid non-transferable features in supervised learning [2, 3].

However, an increasing number of applications require learning in games, generalizing single objective optimization to settings where each agent controls a different subset of parameters to optimize their own objective. Some examples are GANs [4, 5], actor-critic models [5], curriculum learning [6–9], hyperparameter optimization [10–15], adversarial examples [16, 17], learning models [18–20], domain adversarial adaptation [21], neural architecture search [22–26], multi-agent settings [27] and meta-learning [28–30].

There are two major challenges with learning in games: first, the learning dynamics can be *unstable* due to the nonstationarity induced by simultaneously learning agents, e.g. resulting in cycling dynamics rather than convergence. Second, different equilibria can result in vastly different rewards for the different agents. For example, in the iterated prisoners' dilemma (Sec. 4.1), finding solutions that favor cooperation vs selfishness result in higher return for all agents; or in Hanabi, finding solutions that do not arbitrarily break symmetries results in better coordination with humans [31].

Recently, Ridge Rider (RR [32]) provided a general method for finding diverse solutions in single objective optimization, by following *eigenvectors of the Hessian* with negative eigenvalues. We generalize RR to multi-agent settings, each optimizing their own objective. The relevant generalization of the Hessian – the *game Hessian* – is not symmetric and thus, in general, has complex eigenvalues (EVals) and eigenvectors (EVecs). This leads to new types of *bifurcation* points, where small changes in initial parameters lead to very different optimization outcomes.

Our method, Game Ridge Rider (GRR) branches from these novel bifurcation points along the (complex) EVecs to discover diverse solutions in these settings.


GRR – Our Method – at a Saddle Bifurcation

GRR at a Hopf Bifurcation

Figure 1. We visualize the Game Ridge Rider (GRR) algorithm when branching at different types of bifurcations. We have two eigenvectors and can branch in opposite directions, so we have four paths displayed in different colors. Steps with the eigenvector have magenta boundaries. *Top*: For the small Iterated Prisoner's Dilemma (IPD) finding then branching at the max entropy saddle allows us to find defect-defect and tit-for-tat solutions. *Bottom:* For the mixed problem of small IPD and matching pennies, branching at the Hopf bifurcation allows us to find both solutions.

[1] Facebook AI Research [2] University of Toronto [3] Vector Institute [4] University of Oxford [5] UC Berkeley [6] Google Brain [7] Radboud University. Correspondence to: <lorraine@cs.toronto.edu>.

2. Background

Appendix Table 2 summarizes our notation. Consider the minimization problem:

$$\boldsymbol{\theta}^* := \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{1}$$

We denote the gradient of the loss at parameters $\boldsymbol{\theta}^j$ by $\boldsymbol{g}^j := \boldsymbol{g}(\boldsymbol{\theta}^j) := \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^j}$. We can locally minimize the loss $\mathcal{L}$ using (stochastic) gradient descent with step size α.

$$\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j - \alpha \boldsymbol{g}^j \tag{SGD}$$

Due to the potentially non-convex nature of $\mathcal{L}$, multiple solutions can exist and simply following the gradient will not explore the parameter space.

2.1. Ridge Rider

Ridge Rider (RR) [32] finds diverse solutions in gradient systems for single-objective minimization. RR first finds a saddle point, and then branches the optimization procedure following different directions (or "ridges") given by the EVecs of the Hessian $\mathcal{H} = \nabla_{\boldsymbol{\theta}} \boldsymbol{g} = \nabla_{\boldsymbol{\theta}}(\nabla_{\boldsymbol{\theta}} \mathcal{L})$. Full computation of $\mathcal{H}$—and its EVecs and EVals—is often prohibitively expensive; however, we can efficiently access a subset of the eigenspaces in libraries with efficient Hessian-vector products $\mathcal{H}\boldsymbol{v} = \nabla_{\boldsymbol{\theta}}((\nabla_{\boldsymbol{\theta}} \mathcal{L})\boldsymbol{v})$ [33–36]. Algorithm 1 summarizes the RR method.

2.2. Optimization in Games

Instead of simply optimizing a single loss, optimization in games involves multiple agents each with a loss functions that potentially depend on other agents. Examples include multiplayer games (e.g. Go [37, 38] and Hanabi [39]), iterated prisoners' dilemma (IPD) [40, 41] and generative adversarial networks (GANs) [4]. For simplicity, we look at 2-player games with players denoted by A and B. Each player wants to minimize their loss $\mathcal{L}_A, \mathcal{L}_B$ with their parameters $\boldsymbol{\theta}_A \in \mathbb{R}^{d_A}$, $\boldsymbol{\theta}_B \in \mathbb{R}^{d_B}$.

$$\boldsymbol{\theta}_A^* \in \arg\min_{\boldsymbol{\theta}_A} \mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B^*), \boldsymbol{\theta}_B^* \in \arg\min_{\boldsymbol{\theta}_B} \mathcal{L}_B(\boldsymbol{\theta}_A^*, \boldsymbol{\theta}_B) \tag{2}$$

$$\iff \boldsymbol{\theta}_A^* \in \arg\min_{\boldsymbol{\theta}_A} \mathcal{L}_A^*(\boldsymbol{\theta}_A), \boldsymbol{\theta}_B^* \in \arg\min_{\boldsymbol{\theta}_B} \mathcal{L}_B^*(\boldsymbol{\theta}_B) \tag{3}$$

If $\mathcal{L}_B$ and $\mathcal{L}_A$ are differentiable in $\boldsymbol{\theta}_B$ and $\boldsymbol{\theta}_A$ we say the game is differentiable. Unfortunately, for a player to do gradient based optimization of their objective we must compute ${d\mathcal{L}_B^*}/{d\boldsymbol{\theta}_B}$ which often requires ${d\boldsymbol{\theta}_A^*}/{d\boldsymbol{\theta}_B}$, but $\boldsymbol{\theta}_A^*(\boldsymbol{\theta}_B)$ and its Jacobian are typically intractable. There are various algorithms to try to find solutions like Eq. 2, often efficiently approximating ${d\boldsymbol{\theta}_A^*}/{d\boldsymbol{\theta}_B}$ with a method rely on efficient Jacobian-vector products. We overview these in our related work (Appendix A).

One of the most straightforward optimization methods is to find local solutions with simultaneous SGD (SimSGD). This does not take into account ${d\boldsymbol{\theta}_A^*}/{d\boldsymbol{\theta}_B}$ and often fails to converge. Here, $\boldsymbol{g}_A^j := \boldsymbol{g}_A(\boldsymbol{\theta}_A^j, \boldsymbol{\theta}_B^j)$ and $\boldsymbol{g}_B^j := \boldsymbol{g}_B(\boldsymbol{\theta}_A^j, \boldsymbol{\theta}_B^j)$ are estimators for $\nabla_{\boldsymbol{\theta}_A} \mathcal{L}_A|_{\boldsymbol{\theta}_A^j, \boldsymbol{\theta}_B^j}$ and $\nabla_{\boldsymbol{\theta}_B} \mathcal{L}_B|_{\boldsymbol{\theta}_A^j, \boldsymbol{\theta}_B^j}$:

$$\boldsymbol{\theta}_A^{j+1} = \boldsymbol{\theta}_A^j - \alpha \boldsymbol{g}_A^j, \quad \boldsymbol{\theta}_B^{j+1} = \boldsymbol{\theta}_B^j - \alpha \boldsymbol{g}_B^j \tag{SimSGD}$$

We simplify notation by using the concatenation of all players parameters (or joint-parameters) $\boldsymbol{\omega} := [\boldsymbol{\theta}_A, \boldsymbol{\theta}_B] \in \mathbb{R}^d$ and the joint-gradient vector field $\hat{\boldsymbol{g}} : \mathbb{R}^d \to \mathbb{R}^d$, which at the j^{th} iteration is denoted:

$$\hat{\boldsymbol{g}}^j := \hat{\boldsymbol{g}}(\boldsymbol{\omega}^j) := [\boldsymbol{g}_A(\boldsymbol{\omega}^j), \boldsymbol{g}_B(\boldsymbol{\omega}^j)] = [\boldsymbol{g}_A^j, \boldsymbol{g}_B^j] \tag{4}$$

We can write the next iterate in (SimSGD) with a fixed-point operator $\boldsymbol{F}_\alpha$:

$$\boldsymbol{\omega}^{j+1} = \boldsymbol{F}_\alpha(\boldsymbol{\omega}^j) = \boldsymbol{\omega}^j - \alpha \hat{\boldsymbol{g}}^j \tag{5}$$

The Jacobian of the fixed point operator $\boldsymbol{F}_\alpha$ is useful for analysis, including bounding convergence rates near fixed points and finding points where local changes to parameters may cause convergence to qualitatively different solutions. The fixed point operator's Jacobian crucially depends on the Jacobian of our update $\hat{\boldsymbol{g}}$. When our update is the gradient, we call this the *game Hessian* because it generalizes the Hessian:

$$\hat{\mathcal{H}} := \nabla_{\boldsymbol{\omega}} \hat{\boldsymbol{g}} = \begin{bmatrix} \nabla_{\boldsymbol{\theta}_A}^2 \mathcal{L}_A & \nabla_{\boldsymbol{\theta}_A} \nabla_{\boldsymbol{\theta}_B} \mathcal{L}_A \\ \nabla_{\boldsymbol{\theta}_B} \nabla_{\boldsymbol{\theta}_A} \mathcal{L}_B^\top & \nabla_{\boldsymbol{\theta}_B}^2 \mathcal{L}_B \end{bmatrix} \tag{6}$$

$$\nabla_{\boldsymbol{\omega}} \boldsymbol{F}_\alpha(\boldsymbol{\omega}) = \boldsymbol{I} - \alpha \hat{\mathcal{H}} \tag{7}$$

For single-objective optimization $\hat{\mathcal{H}}$ is the Hessian of the loss, which is symmetric and has real EVals yielding parameter updates which form a conservative gradient vector field. However, in games with multiple objectives, $\hat{\mathcal{H}}$ is not symmetric and can have complex Evals, resulting in updates which form a non-conservative vector field.

2.3. Local Bifurcations and Separatrices

The goal of RR was to obtain a method for finding diverse solutions in minimization. There are multiple ways to try to find solutions with RR, but we focus on finding separatrices—i.e., boundaries between phase space regions with different dynamical behavior. Crossing such boundaries leads to different solutions under our updates flow—and branching over these boundaries. Such a behavior of crossing regions and changing behavior is in fact a local bifurcations and a qualitative change in the behavior of the solutions.

When applying RR in conservative gradient systems, saddle points and their EVecs play a key role in the shape of the phase portraits, which are a geometrical representation of the underlying dynamics. The negative EVecs often align with unstable manifolds that are orthogonal to our separatrices [42], thus giving directions in which we can perturb to find different solutions (Lemma 14.3 [43]). However, in non-conservative systems there are a variety of other local bifurcations [44] besides saddle points [45]. For example, by Thm. 11.2 of [43] if all EVals of have negative real part, except a conjugate non-zero pair of purely imaginary EVals, then a *Hopf bifurcation* occurs when changing the parameters causing the pair to cross the imaginary axis. A Hopf bifurcation is a critical point where the stability of a system switches resulting in a periodic solution.

3. Our Method: Game Ridge Rider (GRR)

In this section we introduce Game Ridge Rider (GRR), a generalization of Game Ridge Rider for learning in games. RR is not immediately applicable to complex EVals of the Hessian, because we would need to follow the complex EVecs, resulting in complex weights, and we may need to branch at points besides saddles. When the Hessian has real EVals, GRR is equivalent to RR.

Consider the framework for the RR method in Alg 1. We modify the components of GetRidges and EndRide, along with a proposed starting saddle. We also add a method for running different optimizers after branching with Evecs.

GetRidges finds which EVals and EVecs we should explore from a given branching point. The EVecs of a matrix are not unique in-general, and may have complex entries for complex EVals. We only want real-valued updates when following our EVecs, so we select the choices with largest real part and norm one—this is the default in PyTorch [35]. For a conjugate pair of complex EVals, this selection of EVecs corresponds to spanning the (real-valued) plane that we would rotate in under repeated matrix multiplication of the EVec. This also specifies the order in which we explore the EVals, which we set to be the most negative EVals first. Note that we can explore in opposite directions along each negative EVec.

EndRide is a heuristic that determines how long we follow an EVec. We experiment with stopping after a single step and stopping when the real part of the EVal goes from negative to zero. If we have noisy EVecs estimates, or we are not exactly at a separatrix, we may need to take multiple steps to find different solutions. We can attempt to find Hopf bifurcations by ending and branching when a complex EVals crosses the imaginary axis.

Optimize is a method which runs an optimization procedure. In this work we explore following gradients SimSGD or LOLA [27] with user specified optimization parameters.

Starting location ($\boldsymbol{\omega}^{\textbf{Saddle}}$) We start our method at some some $\boldsymbol{\omega}^{\text{Saddle}} = \arg\min_{\boldsymbol{\omega}} |\hat{\boldsymbol{g}}|$. There are often multiple saddles we can begin at, so for multi-agent Tabular RL – like the IPD – we heuristically begin at the maximum entropy saddle $\arg\min_{\boldsymbol{\omega}} |\hat{\boldsymbol{g}}| - \beta H(\pi_{\boldsymbol{\omega}}(\boldsymbol{a})), \beta > 0$ as in [32].

The other components of ChooseFromArchive and UpdateRidge we did not change, but summarize below. See RR [32] for more details on their implementations.

ChooseFromArchive gives a search order on optimization branches – ex., BFS or DFS – by outputting an index to search and the updated archive of optimization branches.

UpdateRidge updates the currently followed EVec which potentially changed due to the optimization step.

Algorithm 1 Game Ridge Rider (GRR)–red modifications

```
Input: $\boldsymbol{\omega}^{\text{Saddle}}$, $\alpha$, ChooseFromArchive, GetRidges,
        EndRide, Optimize, UpdateRidge
$\mathcal{A}$ = GetRidges($\boldsymbol{\omega}^{\text{Saddle}}$)        # Init. Archive
while Archive $\mathcal{A}$ non-empty do
  $j, \mathcal{A}$ = ChooseFromArchive($\mathcal{A}$)
  $(\boldsymbol{\omega}^j, e_j, \lambda_j) = \mathcal{A}_j$
  while EndRide($\boldsymbol{\omega}^j, e_j, \lambda_j$) not True do
    $\boldsymbol{\omega}^i \leftarrow \boldsymbol{\omega}^j - \alpha e_j$          # Step along the ridge $e_j$
    $e_j, \lambda_j$ = UpdateRidge($\boldsymbol{\omega}^j, e_j, \lambda_j$)
  $\boldsymbol{\omega}^j$ = Optimize($\boldsymbol{\omega}^j$)
  $\mathcal{A} = \mathcal{A} \cup$ GetRidges($\boldsymbol{\omega}^j$)  # Add new ridges
```

4. Experiments

We investigate using Game Ridge Rider on multi-agent learning problems. First, we make a range of two-dimensional games, allowing us to qualitatively study new phenomena that occur in multi-agent settings. Next, we look at a higher dimensional experiment of learning strategies in the iterated prisoners' dilemma (IPD). Our method is able to find a diverse set of solutions improving on baselines.

4.1. Test Problems

Our test problems described in full detail in Appendix Section B.1 and summarized here. We visualize the strategy space for 2-parameter problems in Appendix Fig. 3.

Iterated Prisoners' Dilemma (IPD): This game is an infinite sequence of the Prisoner's Dilemma, where the future payoff is discounted by a factor $\gamma \in [0, 1)$. Each agent is conditioned on the actions in the prior state, so there are 5 parameters for each agent – i.e., the probability of cooperating at start state or given both agents preceding actions. We interested in two Nash equilibria: Defect-Defect (DD) where agents are selfish (giving a poor reward), and *tit-for-tat* (TT) where agents initially cooperate, then copy the opponents action (giving a higher reward).

Small IPD: A is a 2-parameter simplification of IPD, which allows DD and TT Nash equilibria. This game allows us to visualize some of the optimization difficulties for the full-scale IPD, however, the game Hessian has strictly real EVals unlike the full-scale IPD.

Matching Pennies: This is a simplified 2-parameter version of rock-paper scissors. The first player wins if they select the same, while the second player wins if they select different. This game has a Nash equilibrium where each player selects their action with uniform probability. This problem's game Hessian has purely imaginary EVals unlike the small IPD, but only has a single solution and thus is a poor fit for evaluating RR which finds diversity of solutions.

Mixing Small IPD and Matching Pennies: This game interpolates between the Small IPD and matching pennies games with an interpolation factor $\tau \in [0, 1]$. If $\tau = .25$

	Player 1 Loss	Player 1 Strategy Distribution, [min, max]				
Search Strategy	$\mathcal{L}$ [min, max]	C_0	$C\|CC$	$C\|CD$	$C\|DC$	$C\|DD$
Max Entropy Saddle	$[1.000, 2.000]$	$[.001, .999]$	$[.041, .999]$	$[.004, 0.874]$	$[.000, 0.912]$	$[.000, .013]$
20 Random init + grad.	$[1.997, 1.998]$	$[.043, .194]$	$[.142, .480]$	$[.041, .143]$	$[.055, .134]$	$[.001, .001]$
20 Random init + LOLA	$[1.000, 1.396]$	$[.000, 1.00]$	$[.093, 1.00]$	$[.000, .966]$	$[.057, 1.00]$	$[.000, .947]$
1 Random init + branch	$[2.000, 2.000]$	$[.001, .001]$	$[.027, .027]$	$[.003, .003]$	$[.008, .008]$	$[.000, .000]$

Table 1. We compare strategies for finding diverse solutions in the iterated prisoner's dilemma (IPD). The IPD has two solution modes – i.e., solutions where both agents end up defecting with a loss of 2 and where both agents end up cooperating with a loss of 1 (like tit-for-tat). We compare just following gradients with SimSGD and LOLA [27] with random (init)ializations. We look at the impact of starting at a saddle, by branching on the EVecs at a random init. **Takeaway**: Our method finds solutions at both loss modes. Random inits then following the gradient or using LOLA does not find diverse solutions – the gradient often defects, while LOLA often cooperates. If we are not at a stationary point like a saddle, then branching likely does not affect where we converge to.

problem has two solutions – one where both players cooperate and one where both players select actions uniformly, with a Hopf bifurcation separating these.

4.2. Baseline Methods on Toy Problems

Fig. 2 we shows the phase portrait for baseline methods on our *mixed problem* – i.e, the mixture of small IPD and matching pennies with $\tau = .25$.

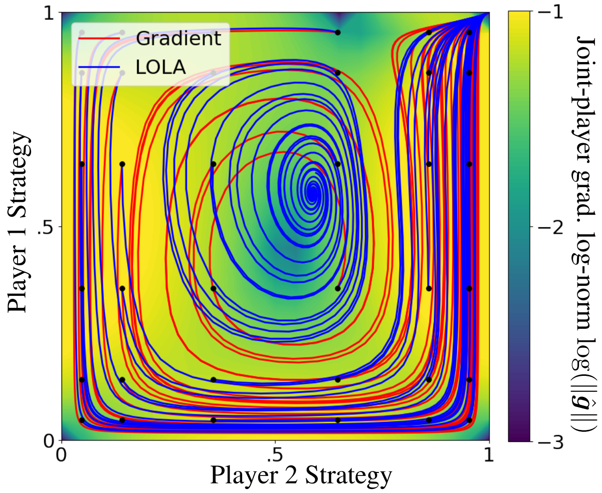


Figure 2. This shows the phase portrait for two standard optimization algorithms on the mixed small IPD and Matching pennies problem. Following the gradient is shown in red, while LOLA – a method for learning in games – is shown in blue. Following the gradient only finds the solution in the top right, because the center solution has imaginary EVals. LOLA [27] can find either solution. **Takeaway**: The mixture game has a range of phenomena, including a imaginary EVal solution, a real EVal solution and a Hopf bifurcation. We may want to use a method for learning in games, so we can robustly converge to different solutions.

4.3. Visualizing the Spectrum on Toy Problems

In Appendix Fig. 5, we visualize the spectrum with the mixed objective to see where stationary points are, which stationary points are solutions, how desirable solutions are for the players, and where the bifurcation occurs.

4.4. Visualizing the Spectrum on the full IPD

Appendix Fig. 4 shows the spectrum on optimization trajectories for the IPD. During training, complex EVals cross the imaginary axis and the final stationary point has positive a& negative real EVals and complex EVals. **Takeaway**: While optimizing the IPD, we have multiple bifurcation candidates and thus multiple potential branching points for GRR.

4.5. Game Ridge Rider (GRR) on Toy Problems

In Figure 1 we use our method to find diverse solutions on toy problems with different types of bifurcations. The small IPD has a saddle bifurcation, while the mixed problem has a Hopf bifurcation. The mixture has a solution with imaginary EVals, which is unstable when following the gradient – see Figure 2 – so we use LOLA after branching. **Takeaway**: By branching our optimization at a bifurcation and using a method for learning in games, we can find all solutions in both toy problems from a single starting point.

4.6. Game Ridge Rider on the IPD

Here, we use our method on the IPD which is a larger scale problem where existing methods have difficulty finding diverse solutions. IPD has two solution modes: ones where both agents end up defecting and cooperating respectively. Table 4 compares our method to following gradients and LOLA each run with random initializations. We also investigate the importance of starting at the max entropy saddle. **Takeaway**: Our method finds solutions at both loss modes by branching at the top few EVecs, where baseline methods failed to find diverse solutions. Starting at an approximate saddle is critical for our branching to find different solutions.

5. Conclusion

In this paper we introduced Game Ridge Rider, an extension of the Ridge Rider algorithm to settings with multiple losses. We showed that in these settings a broader class of bifurcation points needs to be considered and that GRR can indeed discover them in a number of settings. Furthermore, our experimental results showed that GRR obtains a diversity of qualitatively different solutions in multi-agent settings such as iterated prisoner's dilemma. We also provide some theoretical justification for our method by using tools from the dynamical systems literature. Prior work had failed to explore the connection between saddle points in the RR algorithm and the bifurcation points in dynamical systems.

Acknowledgements

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We would also like to thank C. Daniel Freeman for feedback on this work.

References

[1] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[2] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[3] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[5] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48, 2009.

[7] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019.

[8] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pages 434–443. PMLR, 2019.

[9] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[10] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326, 2012.

[11] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning (ICML)*, pages 2113–2122, 2015.

[12] Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

[13] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations (ICLR)*, 2019.

[14] Aniruddh Raghu, Maithra Raghu, Simon Kornblith, David Duvenaud, and Geoffrey Hinton. Teaching with commentaries. *arXiv preprint arXiv:2011.03037*, 2020.

[15] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1552, 2020.

[16] Avishek Joey Bose, Gauthier Gidel, Hugo Berrard, Andre Cianflone, Pascal Vincent, Simon Lacoste-Julien, and William L Hamilton. Adversarial example games. *arXiv preprint arXiv:2007.00720*, 2020.

[17] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30 (9):2805–2824, 2019.

[18] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. *arXiv preprint arXiv:2004.07804*, 2020.

[19] Pierre-Luc Bacon, Florian Schäfer, Clement Gehring, Animashree Anandkumar, and Emma Brunskill. A Lagrangian method for inverse problems in reinforcement learning. *lis.csail.mit.edu/pubs*, 2019.

[20] Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. Control-oriented model-based reinforcement learning with implicit differentiation. *arXiv preprint arXiv:2106.03273*, 2021.

[21] David Acuna, Guojun Zhang, Marc T Law, and Sanja Fidler. f-domain-adversarial learning: Theory and algorithms for unsupervised domain adaptation with neural networks, 2021. URL https://openreview.net/forum?id=WqXAKcwfZtI.

[22] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[23] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

[25] Will Grathwohl, Elliot Creager, Seyed Kamyar Seyed Ghasemipour, and Richard Zemel. Gradient-based optimization of neural network architecture. 2018.

[26] George Adam and Jonathan Lorraine. Understanding neural architecture search techniques. *arXiv preprint arXiv:1904.00438*, 2019.

[27] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130, 2018.

[28] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[29] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. *arXiv preprint arXiv:1909.04630*, 2019.

[30] Mengye Ren, Eleni Triantafillou, Kuan-Chieh Wang, James Lucas, Jake Snell, Xaq Pitkow, Andreas S Tolias, and Richard Zemel. Flexible few-shot learning with contextual similarity. *arXiv preprint arXiv:2012.05895*, 2020.

[31] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[32] Jack Parker-Holder, Luke Metz, Cinjon Resnick, Hengyuan Hu, Adam Lerer, Alistair Letcher, Alexander Peysakhovich, Aldo Pacchiano, and Jakob Foerster. Ridge rider: Finding diverse solutions by following eigenvectors of the hessian. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 753–765. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/08425b881bcde94a383cd258cea331be-Paper.pdf.

[33] Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

[34] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[36] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[37] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[39] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

[40] Robert J Aumann. Acceptable points in games of perfect information. *Pacific Journal of Mathematics*, 10(2):381–417, 1960.

[41] Robert Axelrod and William Donald Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[42] M Tabor. Chaos and integrability in nonlinear dynamics: An introduction, wileyinterscience. *Chaos and Integrability in Nonlinear Dynamics: An Introduction*, 1989.

[43] Jack K Hale and Hüseyin Koçak. *Dynamics and bifurcations*, volume 3. Springer Science & Business Media, 2012.

[44] Ippei Shimada and Tomomasa Nagashima. A numerical approach to ergodic problem of dissipative dynamical systems. *Progress of theoretical physics*, 61(6):1605–1616, 1979.

[45] Davide Bigoni and Oleg Kirillov. *Dynamic Stability and Bifurcation in Nonconservative Mechanics*. Springer, 2019.

[46] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[47] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399 – 1404, 1999. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(99)00073-8. URL http://www.sciencedirect.com/science/article/pii/S0893608099000738.

[48] Samarth Sinha, Homanga Bharadhwaj, Anirudh Goyal, Hugo Larochelle, Animesh Garg, and Florian Shkurti. Diversity inducing information bottleneck in model ensembles. *AAAI*, 2021.

[49] Andrew Slavin Ross, Weiwei Pan, Leo A. Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5527–5536. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6004.

[50] Zelda Mariet and Suvrit Sra. Diversity Networks: Neural Network Compression Using Determinantal Point Processes. In *International Conference on Learning Representations (ICLR)*, May 2016.

[51] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

[52] Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI*. MIT Press, 2008.

[53] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

[54] Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Effective diversity in population-based reinforcement learning. In *Advances in Neural Information Processing Systems 34*. 2020.

[55] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016. ISSN 2296-9144. doi: 10.3389/frobt.2016.00040. URL https://www.frontiersin.org/article/10.3389/frobt.2016.00040.

[56] Galina M Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[57] Waïss Azizian, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. A tight and unified analysis of gradient-based methods for a whole spectrum of differentiable games. In *International Conference on Artificial Intelligence and Statistics*, pages 2863–2873. PMLR, 2020.

[58] Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. *arXiv preprint arXiv:1311.1869*, 2013.

[59] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[60] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1802–1811. PMLR, 2019.

[61] Jonathan Lorraine, David Acuna, Paul Vicol, and David Duvenaud. Complex momentum for learning in games. *arXiv preprint arXiv:2102.08431*, 2021.

[62] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[63] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. *arXiv preprint arXiv:1705.10461*, 2017.

[64] Alistair Letcher, David Balduzzi, Sébastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. Differentiable game mechanics. *The Journal of Machine Learning Research*, 20(1):3032–3071, 2019.

[65] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[66] Florian Schäfer and Anima Anandkumar. Competitive gradient descent. *arXiv preprint arXiv:1905.12103*, 2019.

[67] Marta Garnelo, Wojciech Marian Czarnecki, Siqi Liu, Dhruva Tirumala, Junhyuk Oh, Gauthier Gidel, Hado van Hasselt, and David Balduzzi. Pick your battles: Interaction graphs as population-level objectives for strategic diversity. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1501–1503, 2021.

[68] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[69] Yaodong Yang, Ying Wen, Jun Wang, Liheng Chen, Kun Shao, David Mguni, and Weinan Zhang. Multi-agent determinantal q-learning. In *International Conference on Machine Learning*, pages 10757–10766. PMLR, 2020.

[70] Andrei Lupu, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1593–1595, 2021.

A. Related Work

Diversity in machine learning: Finding diverse solutions is often desirable in machine learning, for example improving performance for model ensembles [46], with canonical approaches directly optimizing for negative correlation amongst model predictions [47]. In recent times these ideas have begun to re-emerge, improving ensemble performance [48–50], robustness [51, 1] and boosting exploration in reinforcement learning [52–55].

Many of these approaches seek to find diverse solutions by following gradients of an altered, typically multi-objective loss function. By contrast, the recent *Ridge Rider* (RR, [32]) algorithm searches for diverse solutions by following EVecs of the Hessian with respect to the original loss function, producing orthogonal (loss reducing) search directions.

Finding solutions in games: There are first-order methods for finding solutions in games including extragradient [56, 57], optimistic gradient [58, 59], negative momentum [60], complex momentum [61], and iterate averaging [62]. There are also higher-order methods like consensus optimization [63], symplectic gradient adjustment (SGA) [64], local symplectic surgery (LSS) [65], competitive gradient descent (CGD) [66].

Diversity in multi-agent RL In recent times a series of works have explore the benefit of diversity in both competitive [8, 67, 68] and cooperative [69, 70] multi-agent RL. However, once again these approaches all consider augmented loss functions. Instead, we take inspiration from RR and extend it to the multi-agent setting.

B. Experiment Details

B.1. Test Problems

Iterated Prisoners' Dilemma (IPD): This game is an infinite sequence of the Prisoner's Dilemma, where the future payoff is discounted by a factor $\gamma \in [0, 1)$. Each agent is conditioned on the actions in the prior state (s). Thus, there are 5 parameters for each agent i: $P^i(C|s)$ the probability of cooperating at start state $s_0 = \emptyset$ or state $s_t = (a^1_{t-1}, a^2_{t-1})$for $t > 0$. There are two Nash equilibria which we interested in: Defect-Defect (DD) where agents are selfish (resulting in poor reward), and *tit-for-tat* (TT) where agents initially cooperate, then copy the opponents action (resulting in higher reward).

Small IPD: This is a 2-parameter simplification of IPD, which allows DD and TT Nash equilibria. We fix the strategy if our opponent defects, to defect with high probability. We also constrain the probability of cooperating to only depend on if the opponent cooperates, and in the initial state we assume our opponent cooperated. This game allows us to visualize some of the optimization difficulties for the full-scale IPD, however, the game hessian has strictly real EVals unlike the full-scale IPD. See Fig 2, top for a visualization of the strategy space.

Matching Pennies: This is a simplified 2-parameter version of rock-paper scissors, where each players selects Cooperate or Defect. This game has a Nash equilibrium where each player selects their action with uniform probability. Notably, this problem's game Hessian has purely imaginary EVals so following the gradient does not converge to solutions and we need a method for learning in games like LOLA. Also, this game only has a single solution thus is a poor fit for evaluating RR which finds diversity of solutions. See Fig 2, bottom for a visualization of the strategy space.

Mixing Small IPD and Matching Pennies: This game interpolates between the Small IPD and matching pennies games with the loss for player i, $\mathcal{L}_{mix,P_i,\tau} = \tau\mathcal{L}_{smallIPD,P_i} + (1 - \tau)\mathcal{L}_{matchingPennies,P_i}$. This problem has two solutions – one where both players cooperate, and one where both players select actions uniformly. The uniform action solution has imaginary EVals, so it is only stable under a method for learning in games, while the both cooperate solution has real EVals. There is a Hopf bifurcation separating these solutions. See Fig 2 for standard methods on this problem and Appendix Fig. 3 to contrast this problem with Small IPD or Matching Pennies. See Appendix Fig 5 to visualize the eigenstructure on this problem.

Table 2. Notation

RR	Ridge Rider [32]
IPD	Iterated Prisoners' Dilemma
GAN	Generative Adversarial Network [4]
LOLA	Learning with opponent learning awareness [27]
EVec, EVal	Shorthand for Eigenvector or Eigenvalue
SGD	Stochastic Gradient Descent
SimSGD	Simultaneous SGD
$:=$	Defined to be equal to
$x, y, z, \dots \in \mathbb{C}$	Scalars
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \dots \in \mathbb{C}^n$	Vectors
$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \dots \in \mathbb{C}^{n \times n}$	Matrices
$\boldsymbol{X}^\top$	The transpose of matrix $\boldsymbol{X}$
$\boldsymbol{I}$	The identity matrix
$\Re(z), \Im(z)$	The real or imaginary component of $z \in \mathbb{C}$
i	The imaginary unit. $z \in \mathbb{C} \implies z = \Re(z) + i\Im(z)$
$\bar{z}$	The complex conjugate of $z \in \mathbb{C}$
$\lvert z \rvert := \sqrt{z\bar{z}}$	The magnitude or modulus of $z \in \mathbb{C}$
$\arg(z)$	The argument or phase of $z \in \mathbb{C} \implies z = \lvert z \rvert \exp(i \arg(z))$
A, B	A symbol for the outer/inner players
$d_A, d_B \in \mathbb{N}$	The number of weights for the outer/inner players
$\boldsymbol{\theta}$	A symbol for the parameters or weights of a player
$\boldsymbol{\theta}_A \in \mathbb{R}^{d_A}, \boldsymbol{\theta}_B \in \mathbb{R}^{d_B}$	The outer/inner parameters or weights
$\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$	A symbol for a loss
$\mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B), \mathcal{L}_B(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$	The outer/inner losses – $\mathbb{R}^{d_A + d_B} \mapsto \mathbb{R}$
$\boldsymbol{g}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B), \boldsymbol{g}_B(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$	Gradient of outer/inner losses w.r.t. their weights in $\mathbb{R}^{d_A / d_B}$
$\boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A) := \arg\min_{\boldsymbol{\theta}_B} \mathcal{L}_B(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$	The best-response of the inner player to the outer player
$\mathcal{L}_A^*(\boldsymbol{\theta}_A) := \mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A))$	The outer loss with a best-responding inner player
$\boldsymbol{\theta}_A^* := \arg\min_{\boldsymbol{\theta}_A} \mathcal{L}_A^*(\boldsymbol{\theta}_A)$	Outer optimal weights with a best-responding inner player
$d := d_A + d_B$	The combined number of weights for both players
$\boldsymbol{\omega} := [\boldsymbol{\theta}_A, \boldsymbol{\theta}_B] \in \mathbb{R}^d$	A concatenation of the outer/inner weights
$\hat{\boldsymbol{g}}(\boldsymbol{\omega}) := [\boldsymbol{g}_A(\boldsymbol{\omega}), \boldsymbol{g}_B(\boldsymbol{\omega})] \in \mathbb{R}^d$	A concatenation of the outer/inner gradients
$\boldsymbol{\omega}^0 = [\boldsymbol{\theta}_A^0, \boldsymbol{\theta}_B^0] \in \mathbb{R}^d$	The initial parameter values
j	An iteration number
$\hat{\boldsymbol{g}}^j := \hat{\boldsymbol{g}}(\boldsymbol{\omega}^j) \in \mathbb{R}^d$	The joint-gradient vector field at weights $\boldsymbol{\omega}^j$
$\nabla_{\boldsymbol{\omega}} \hat{\boldsymbol{g}}^j := \nabla_{\boldsymbol{\omega}} \hat{\boldsymbol{g}} \rvert_{\boldsymbol{\omega}^j} \in \mathbb{R}^{d \times d}$	The Jacobian of the joint-gradient $\hat{\boldsymbol{g}}$ at weights $\boldsymbol{\omega}^j$
$\hat{\mathcal{H}}$	The game Hessian
$\boldsymbol{\omega}^{\text{Saddle}}$	A saddle point
$\alpha \in \mathbb{C}$	The step size or learning rate
$\lambda \in \mathbb{C}, e$	Notation for an arbitrary Eval or Evec
$\text{Sp}(\boldsymbol{M}) \in \mathbb{C}^n$	The spectrum – or set of eigenvalues – of $\boldsymbol{M} \in \mathbb{R}^{n \times n}$
$\rho(\boldsymbol{M}) := \max_{z \in \text{Sp}(\boldsymbol{M})} \lvert z \rvert$	The spectral radius in $\mathbb{R}^+$ of $\boldsymbol{M} \in \mathbb{R}^{n \times n}$
$\boldsymbol{F}_\alpha(\boldsymbol{\omega})$	Fixed point operator for our optimization
$\mathcal{A}$	The archive from our method
γ	Discount Factor
τ	The mixture weighting for the objectives

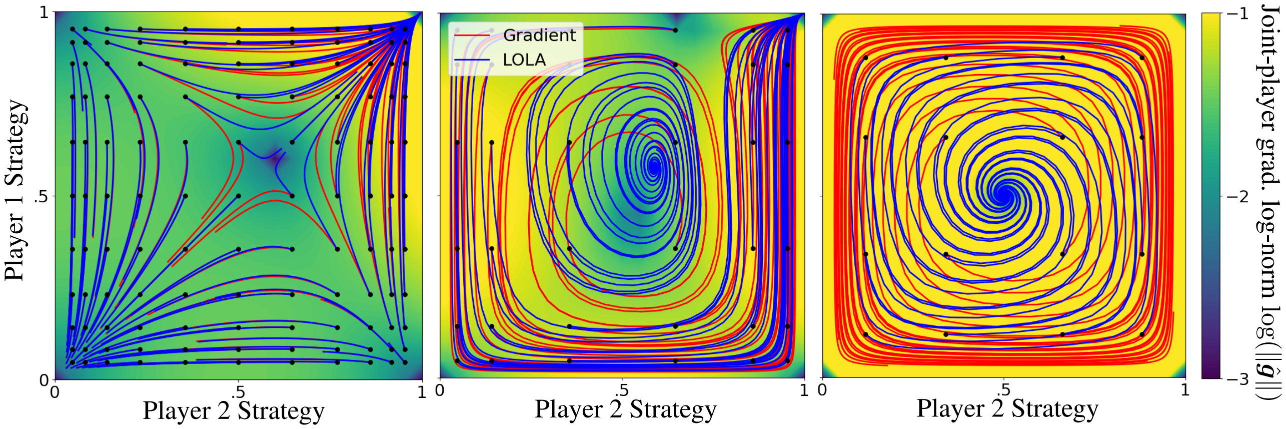


Figure 3. This shows the phase portrait for two standard optimization algorithms on a range of problems. Following the gradient is shown in red, while LOLA – a method for learning in games – is shown in blue. *Left*: The small IPD, which has solutions in the top right and bottom left. *Middle*: Matching pennies, which has a single solution in the middle. Following the gradient does not find this solution because it has imaginary EVals, so we must a method like LOLA. *Right*: A mixture of small IPD and matching pennies. Following the gradient only finds the solution in the top right, because the center solution has imaginary EVals. LOLA can find either solution. **Takeaway**: The mixture game has a range of phenomena, including a imaginary EVal solution, a real EVal solution and a Hopf bifurcation. We may want to use a method for learning in games, so we can robustly converge to different solutions.

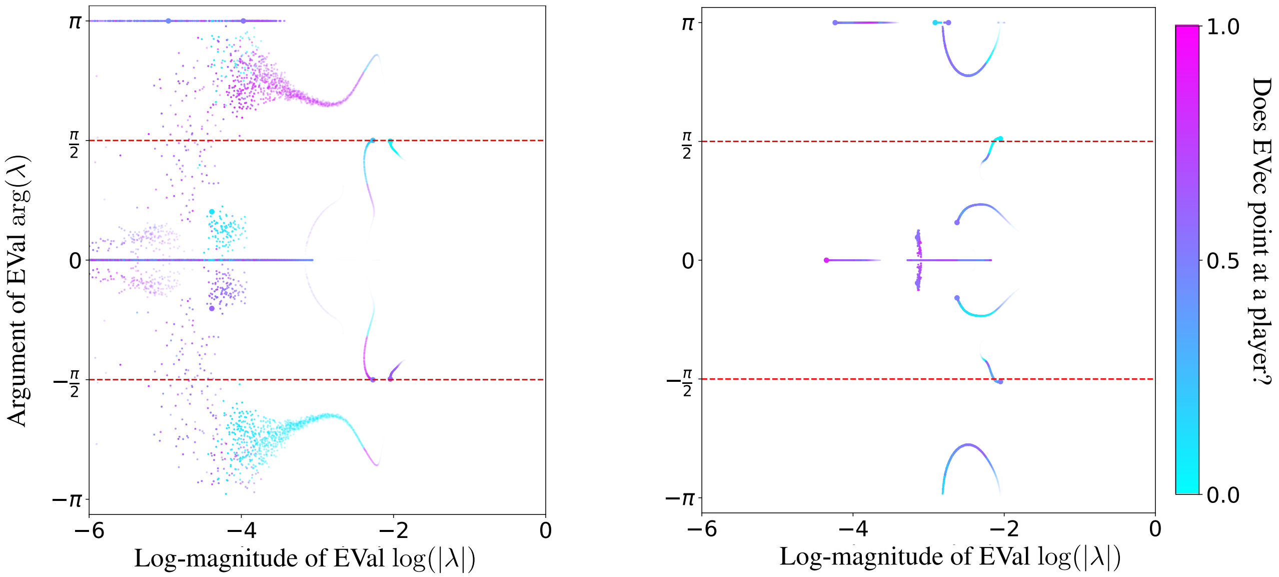


Figure 4. This shows the spectrum of the game Hessian in log-polar coordinates during training. The spectrum at the start of training is in low alpha, while at the end it is in high alpha. We also color each EVec based on how much it points at a player, which we calculate by finding the ratio of the first players component of EVec's norm to the norm of the entire EVec $|e_{1:d_B}|_1/|e|_1$. **Takeaway**: Only some EVals are real and lie entirely in a single players space – these align with search directions for single objective RR. During training, the EVals cross the imaginary axis – i.e., where $\arg(\lambda) = \pm\pi/2$ shown in red– indicating potential Hopf bifurcations. At the end of training we have positive (i.e. $\arg(\lambda) = 0$) and negative (i.e. $\arg(\lambda) = \pm\pi$) real EVals, showing potential bifurcations that are similar to saddles.

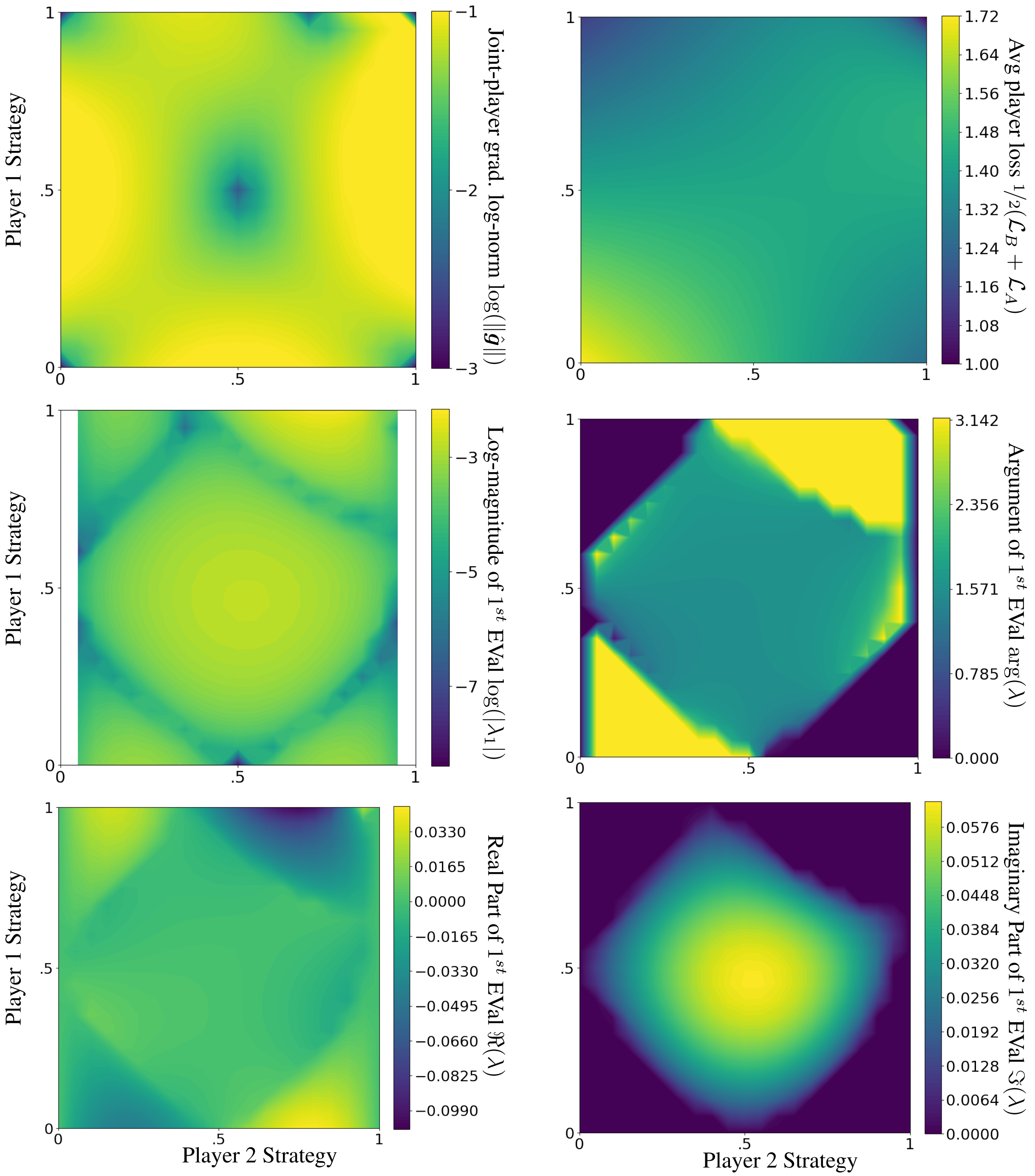


Figure 5. We display various aspects of the players learning dynamics for the small IPD and matching pennies mixture problem. *Top left*: The log-norm of the joint gradient $\hat{\boldsymbol{g}}$. When this is 0 – i.e., the corners of the grid and the center – we are at a stationary point, which is required, but not sufficient for solutions. *Top right*: The loss averaged over both players, allowing us to assess how desirable different solutions are. *Middle left*: The log magnitude of the game Hessian's first Eval λ. *Middle right*: The $\arg$ of λ. *Bottom left*: The real part of λ. *Bottom Right*: The imaginary part of λ. **Takeaway**: This range of visualizations allows us to see where stationary points are, which stationary points are solutions, how desirable solutions are for the players, and where the bifurcation occurs. **Note**: It is difficult to see that the gradient norm goes to 0 near the corners of the grid, but this – in fact – begins to happen if we get close enough (in both the mixed objective and the small IPD).