Handling Continuous Space Security Games with Neural Networks

Nitin Kamra[1], Fei Fang[2], Debarun Kar[1], Yan Liu[1], Milind Tambe[1]

University of Southern California[1], Carnegie Mellon University[2]
{nkamra, dkar, yanliu.cs, tambe}@usc.edu[1], feifang@cmu.edu[2]

August 16, 2017

Overview

1. Security games
2. Forest protection game
3. Algorithm: OptGradFP
4. Experiments and Results
5. Discussion and Future work

Stackelberg Security Game (SSG)

- A leader-follower game between a defender and opponent.
- Payoff for players (r_O and r_D): decided by their joint actions.
- Defender pure strategy: allocate resources to protect a subset of targets.
- Opponent pure strategy: attack a target.

Adversary

Defender

	Target #1	Target #2
Target #1	5, -3	-1, 1
Target #2	-5, 4	2, -1

SSG: Utilities and Policies

- Mixed strategy (a.k.a. policy): Probability distribution over pure strategies.
- Optimal defender strategy (π_D): Maximizes her expected utility J_D, given that the attacker best responds to it.
- Attacker's best response (π_O): An action or strategy that maximizes his expected utility J_O.
- Zero-sum game: $J_D + J_O = 0$.

$$
\begin{aligned}
AttEU_1 &= 0.556 * (-3) + 0.444 * 4 = 0.11 \\
AttEU_2 &= 0.556 * 1 + 0.444 * (-1) = 0.11 \\
J_o &= \max\{AttEU_1, AttEU_2\} = 0.11
\end{aligned}
$$

Adversary

Defender

		Target #1	Target #2
55.6%	Target #1	5, -3	-1, 1
44.4%	Target #2	-5, 4	2, -1

Challenges

- Previous work considers discrete player actions, even for games with continuous space (through discretization) [1, 4, 5, 12].
- Most approaches solve mixed integer linear programs to obtain Stackelberg Equilibria, which rarely scale to big problems.
- Other solutions rely on exploitable spatio-temporal structures of the problem and cannot be generalized to handle general settings [8, 2, 13].

Our major contributions

- This work is a proof-of-concept showing that deep learning can be used to handle difficult problems in security games.
- This is part of ongoing work and provides encouraging results with a preliminary version of our algorithm.
- We present
    - Continuous space security game model: Infinite action sets over two-dimensional continuous areas with asymmetric target distribution.
    - OptGradFP: General algorithm to optimize parametrized strategies (policies) in continuous adversarial domains.
    - OptGradFP-NN: Application of OptGradFP using CNNs.

Forest protection game

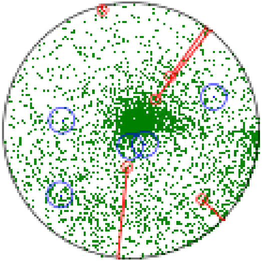


Figure: Forest game with 5 guards and 5 lumberjacks visualized. Trees are green dots, guards are blue dots with enclosing blue circles showing radius R_g and lumberjacks are red dots with enclosing red circles showing R_l.

- **Game model**:
  - Circular forest, prespecified arbitrary tree distribution.
  - n lumberjacks move directly towards center in a straight line, stop, chop wood in radius R_l and return back.
  - m hidden guards attempt to ambush lumberjacks.
  - Forest state: Summarized via 120×120 grayscale image.
- **Defender action**: pick m locations, one for each guard to set ambush.
- **Opponent action**: pick n chopping locations, one for each lumberjack.

Forest protection game

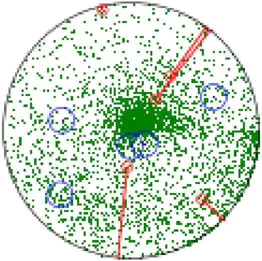


Figure: Forest game with 5 guards and 5 lumberjacks visualized. Trees are green dots, guards are blue dots with enclosing blue circles showing radius R_g and lumberjacks are red dots with enclosing red circles showing R_l.

- **Rewards**:
  - Guard ambushes lumberjacks within R_g radius.
  - Ambushed lumberjack loses all wood and pays penalty r_{pen}.
  - Opponent utility $(r_O) = \#$ trees successfully stolen - total ambush penalty incurred.
  - Defender utility $(r_D) = -r_O$.
- **Game play**: Given a forest:
  - Defender gives m guard locations
  - Opponent gives n chopping locations
  - Game simulator returns (r_D, r_O)

By playing multiple times, defender gets information via rewards and optimizes her strategy.

Definitions

- **Policies (Mixed strategies)**:
  - Policy: Probability distribution over player's actions given state (S).
  - Defender's learnable policy $\pi_D : P(a_D|S; \boldsymbol{w_D})$.
  - Defender's estimate of opponent's policy $\pi_O : P(a_O|S; \boldsymbol{w_O})$.
  - Opponent's real policy: Best response to defender's final policy (not π_O).
- **Utilities**:
  - Defender utility $= J_D(\boldsymbol{w_D}, \boldsymbol{w_O}) = \mathbb{E}_{S,a_D,a_O}[r_D(S, a_D, a_O)]$
  - Opponent's utility: $J_O = -J_D$

Maximizing utilities

- Both players want to maximize their utilities.
- Defender deploys her policy first, without knowing opponent's policy. Defender's optimization:

$$\boldsymbol{w_D^*} = arg \max_{\boldsymbol{w_D}} \min_{\boldsymbol{w_O}} J_D(\boldsymbol{w_D}, \boldsymbol{w_O}) \tag{1}$$

- Opponent observes the defender's policy and reacts with a best response. Opponent's optimization:

$$\boldsymbol{w_O^*} = arg \min_{\boldsymbol{w_O}} J_D(\boldsymbol{w_D^*}, \boldsymbol{w_O}) \tag{2}$$

We approach these problems by taking a gradient optimization based approach.

Policy Gradient Theorem

- Given a function $f(\cdot)$ and a random variable $\boldsymbol{X} \sim p(\boldsymbol{x}|\boldsymbol{\theta})$.
- Gradient of expected value of $f(\cdot)$ with respect to distribution parameters can be computed using Policy Gradient Theorem [10] as:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X})] = \mathbb{E}_{\boldsymbol{X}}[f(\boldsymbol{X}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{X}|\boldsymbol{\theta})] \tag{3}$$

- Useful to compute gradients of players' utilities w.r.t. policy parameters.

Approximating utility function gradients

- Gradient of J_D w.r.t. defender parameters $\boldsymbol{w_D}$ can be found using policy gradient theorem:

$$\boldsymbol{\nabla_{w_D} J_D} = \mathbb{E}_{S,a_D,a_O}[\nabla_{w_D}\pi_D(a_D|S;\boldsymbol{w_D})\ r_D] \tag{4}$$

- Exact computation of above integral is prohibitive, but can be approximated from a batch of B on-policy samples (w.r.t. π_D) using the following unbiased estimator:

$$\boldsymbol{\nabla_{w_D} J_D} \approx \frac{1}{B}\sum_{i=1}^{B}\nabla_{w_D}\pi_D(a_D^i|S^i;\boldsymbol{w_D})\ r_D^i \tag{5}$$

- Gradient of J_O w.r.t. $\boldsymbol{w_O}$ can be similarly approximated.

Best response to average strategy: Fictitious play

- Directly responding to the other player's strategy with a best response is not appropriate since it causes sudden simultaneous changes to players' policies. Both players can diverge because of it.
- Fictitious Play: Respond to the other player's average strategy uptil now [6, 7].
- Converges to Nash Equilibrium under various settings including two-player zero-sum games [3].
- In a zero-sum SSG, Fictitious Play converges to Stackelberg Equilibrium.
- Emulate average play by storing past history of games in replay memories.

OptGradFP: Intuition

- Parametrize players' mixed strategies in continuous space (we use ConvNets).
- Play games with players' policy estimates and keep storing in replay memories.
- Use games from players' current policies and from previous policies (fictitious play) to compute the gradient of utility w.r.t. current policy parameters (policy gradients).
- Update NN policy with gradients to improve against other player's average strategy.

Our algorithm: OptGradFP

Algorithm 1: OptGradFP

Initialize policy parameters $\boldsymbol{w_D}$ and $\boldsymbol{w_O}$ randomly;
Fill replay memories *memD*, *memO* of size E with randomly played games;
for ep *in* $\{0, \ldots, ep_{max}\}$ **do**
- Get game state S;
- Execute $a_D \sim \pi_D(\cdot|S; \boldsymbol{w_D}), a_O \sim \pi_O(\cdot|S; \boldsymbol{w_O})$;
- Get rewards (r_D, r_O) and store $\{S, a_D, a_O, r_D, r_O\}$ in `memD`, `memO`;
- **if** $ep \,\%\, f_D == 0$ **then**
  - Get samples $\{S^i, a_D^i, a_O^i, r_D^i, r_O^i\}_{i \in [E]}$ from `memD`;
  - Replay all games $S^i, \tilde{a}_D^i \sim \pi_D(\cdot|S; \boldsymbol{w_D}), a_O^i$ to obtain $\tilde{r}_D^i, \tilde{r}_O^i$;
  - $\boldsymbol{w_D} := \boldsymbol{w_D} + \frac{\alpha_D}{1+ep\,\beta_D} \frac{1}{E} \sum_{i=1}^{E} \nabla_{\boldsymbol{w_D}} \pi_D(\tilde{a}_D^i | S^i; \boldsymbol{w_D}) \ \tilde{r}_D^i$;
- **if** $ep \,\%\, f_O == 0$ **then**
  - Get samples $\{S^i, a_D^i, a_O^i, r_D^i, r_O^i\}_{i \in [E]}$ from `memO`;
  - Replay all games $S^i, a_D^i, \tilde{a}_O^i \sim \pi_O(\cdot|S; \boldsymbol{w_D})$ to obtain $\tilde{r}_D^i, \tilde{r}_O^i$;
  - $\boldsymbol{w_O} := \boldsymbol{w_O} + \frac{\alpha_O}{1+ep\,\beta_O} \frac{1}{E} \sum_{i=1}^{E} \nabla_{\boldsymbol{w_O}} \pi_O(\tilde{a}_O^i | S^i; \boldsymbol{w_O}) \ \tilde{r}_O^i$;

OptGradFP-NN: Representing policies with ConvNets

- We assume each coordinate (cylindrical: radius, angle) of a_D, a_O to be distributed independently according to logit-normal distribution.
- Our choice of logit-normal distribution meets the requirement of a continuous distribution, differentiable w.r.t. its parameters and having bounded support (since our action spaces are bounded and continuous).
- Two separate ConvNets [9, 14] to learn the required means and standard deviations.

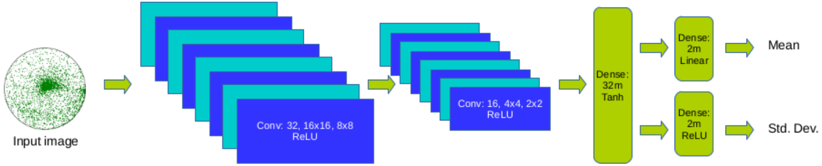


Figure: Defender's policy as a CNN

OptGradFP-NN: Hyperparameter selection

- Our OptGradFP implementation uses a replay memory size of $E = 1000$ samples, maximum episodes $ep_{max} = 10000$, learning rates $\alpha_D = \alpha_O = 10^{-5}$, training rates $f_D = f_O = 50$ and decays $\beta_D = \beta_O = 0.045$.
- The architectures of all neural networks involved and all algorithm hyperparameters were chosen by doing informal grid searches within appropriate intervals. For more information on choosing convolutional neural network architectures, refer [11].

Baselines

Baseline algorithms:

- Cournot Adjustment (CA): Players sequentially best respond to each others' policies.
- StackGrad[1] [1]: Opponent best responds, defender uses a soft policy gradient update (but no fictitious play).
- OptGradFP: Our method.

Other parameters:

- We use $m = 8$ guards and $n = 8$ lumberjacks.
- Ambush penalty $r_{pen} = 10$, guard radius $R_g = 0.1$ and lumberjack radius $R_l = 0.04 < R_g$.

[1] StackGrad uses best response computation for opponent in [1] (approximated by parametrized softmax distribution). Since it is hard to compute the analytic best response to any policy for our domain, we use an approximation to emulate the opponent's best response: we play multiple games with random actions for the opponent while drawing the defender's actions from its current policy. The random action which gets the highest reward against the defender's policy is chosen as the best response action for the opponent.

Training reward curves for defender

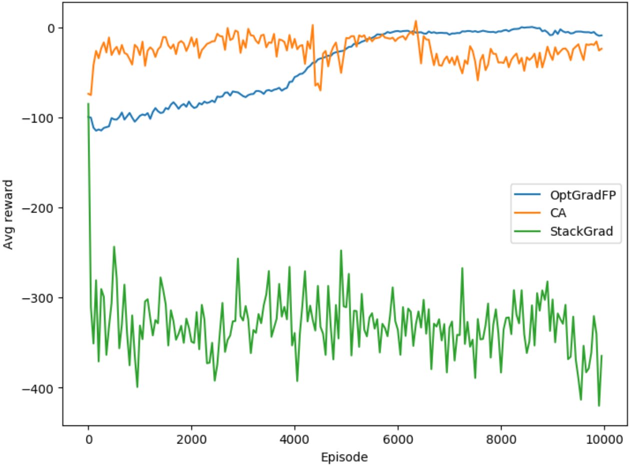


Figure: Average reward for all replayed games before every training iteration. OptGradFP offers the maximum average utility. Note that the reward is averaged on the last E games for OptGradFP, but only on f_D games for CA and StackGrad. Hence, CA seems to approach OptGradFP, but it does not truly respond well to the average response of the opponent.

Opponent's maximum average utility

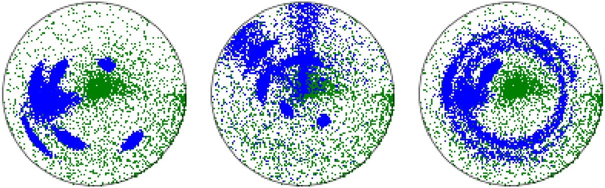
Algorithm	Max Util
CA	567.05
StackGrad	518.34
OptGradFP	499.15

Table: Maximum average utility [2] of the opponent.

- OptGradFP offers the least maximum average utility to the opponent.

[2] Opponent's maximum utility was computed approximately (computing actual values is extremely prohibitive), by sampling 100 random opponent actions and 100 actions from the defender's final policy. 10000 games were played with each combination of the defender's and opponent's actions and the opponent action which led to the maximum reward for the opponent (averaged over all 100 defender actions) was assumed to be the opponent's final action.

Policy visualization



(a) CA (b) StackGrad (c) OptGradFP

Figure: Visualization of defender's policy. Blue dots show sampled positions for the guards. Locations with many blue dots are the regions where the distribution is concentrated.

- OptGradFP's defender policy converges to well-spread concentric rings.
- Other baselines find local regions to guard and leave a lot of space for lumberjacks to go unambushed.
- OptGradFP finds reasonable radii to place the rings (guards).

Effect of fictitious play and replay memory

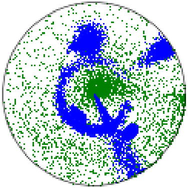


Figure: Policy visualization after removing replay memory

- Disabled fictitious play with small replay memory ($E = f_D = f_O$), containing only games sampled from current player policies.
- Opponent's best response utility: 555.58.
- Defender policy not well spread out: no memory of previous moves.
- **Result**: Trade-off between large memory (smooth convergence, replay bottleneck) vs. small memory (fast, but non-optimal policy).

Discussion

Interpretation for the algorithms' performance:

- Cournot Adjustment:
  - Opponent *runs* from defender; defender keeps *chasing* (oscillatory reward curve).
  - Final defender policy localized due to lack of memory.
- StackGrad:
  - Opponent adapts fast, while defender *chases* around, but never catches up (sudden initial fall in the defender's average reward curve).
  - Final defender policy highly localized due to lack of memory.
- OptGradFP:
  - Soft steps for both players towards best response to each other's *average* strategies (averaged via replay memories).
  - Both players eventually converge to a good average response to each other.
  - Final defender policy well spread out into circular bands around the dense forest center.

Future work

- Generalizing the model and algorithm to handle arbitrary shaped forest regions.
- Training the network to respond to multiple distinct game states.
- Extending to games played over time.

Thank you

Questions?

References I

Kareem Amin, Satinder Singh, and Michael P Wellman.
Gradient methods for stackelberg security games.
In *UAI*, pages 2–11. AUAI Press, 2016.

Fei Fang, Albert Xin Jiang, and Milind Tambe.
Optimal patrol strategy for protecting moving targets with multiple mobile resources.
In *AAMAS*, pages 957–964, 2013.

Drew Fudenberg and David K Levine.
The theory of learning in games, volume 2.
MIT press, 1998.

Jiarui Gan, Bo An, Yevgeniy Vorobeychik, and Brian Gauch.
Security games on a plane.
In *AAAI*, pages 530–536, 2017.

William Haskell, Debarun Kar, Fei Fang, Milind Tambe, Sam Cheung, and Elizabeth Denicola.
Robust protection of fisheries with compass.
In *IAAI*, 2014.

Johannes Heinrich, Marc Lanctot, and David Silver.
Fictitious self-play in extensive-form games.
In *ICML*, pages 805–813, 2015.

References II

Johannes Heinrich and David Silver.
Deep reinforcement learning from self-play in imperfect-information games.
CoRR, abs/1603.01121, 2016.

Matthew P. Johnson, Fei Fang, and Milind Tambe.
Patrol strategies to maximize pristine forest area.
In *AAAI*, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *NIPS*, pages 1097–1105. 2012.

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

Ihsan Ullah and Alfredo Petrosino.
About pyramid structure in convolutional neural networks.
In *IEEE 2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1318–1324, 2016.

Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux.
Adaptive resource allocation for wildlife protection against illegal poachers.
In *AAMAS*, 2014.

References III

Yue Yin, Bo An, and Manish Jain.
Game-theoretic resource allocation for protecting large public events.
In *AAAI*, pages 826–833, 2014.

Matthew D Zeiler and Rob Fergus.
Visualizing and understanding convolutional networks.
In *European conference on computer vision*, pages 818–833. Springer, 2014.