Universal parameter optimisation in games: SPSA and related algorithms

Csaba Szepesvári

Computer and Automation Research Institute of the
Hungarian Academy of Sciences
Kende u. 13-17, Budapest 1111, Hungary
E-mail: **szcsaba@sztaki.hu**

Games Group Presentation
U. Alberta, 2005

Joint work with Levente Kocsis and Mark H.M. Winands

Universal parameter optimisation in games: SPSA and related algorithms

Csaba Szepesvári

Computer and Automation Research Institute of the
Hungarian Academy of Sciences
Kende u. 13-17, Budapest 1111, Hungary
E-mail: **szcsaba@sztaki.hu**

Games Group Presentation
U. Alberta, 2005

Joint work with Levente Kocsis and Mark H.M. Winands

Outline

Outline

Outline

Outline

Outline

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Motivation

- Game programs have many parameters
- Hand tuning of parameters is difficult
- Automatic tuning – stochastic search
  - No gradient: Global search (GA, Simulated Annealing)
  - Gradient: Stochastic Gradient Ascent (SGA)
    - Gradient estimation – how?
- Example domains:
  - Omaha Hi-Lo Poker
  - Lines of Actions, MIA++

Formalism

- Optimization problem:

$$f(\theta) = \mathbb{E}[R(\theta)] \rightarrow \max, \quad \operatorname{argmax}_\theta f(\theta) = ?$$

- Simulation Optimization[1]
- 1950s–present (Metropolis, Rosenbluth, Teller, Robbins, Monro, ..)

[1] J.R. Swisher et. al: "A survey of simulation optimization techniques and procedures", Proc. of Winter-2000

Formalism

- Optimization problem:

$$f(\theta) = \mathbb{E}[R(\theta)] \rightarrow \max, \quad \operatorname{argmax}_\theta f(\theta) = ?$$

- Simulation Optimization[1]
- 1950s–present (Metropolis, Rosenbluth, Teller, Robbins, Monro, ..)

[1] J.R. Swisher et. al: "A survey of simulation optimization techniques and procedures", Proc. of Winter-2000

Formalism

- Optimization problem:

$$f(\theta) = \mathbb{E}[R(\theta)] \rightarrow \max, \quad \operatorname{argmax}_\theta f(\theta) = ?$$

- Simulation Optimization[1]
- 1950s–present (Metropolis, Rosenbluth, Teller, Robbins, Monro, ..)

[1] J.R. Swisher et. al: "A survey of simulation optimization techniques and procedures", Proc. of Winter-2000

Simulation Optimization

"Simulation optimization can be defined as the process of finding the best input variable values from among all possibilities without explicitly evaluating each possibility. The objective of simulation optimization is to minimize the resources spent while maximizing the information obtained in a simulation experiment."[a]

[a] Yolanda Carson & Anu Maria: "Simulation Optimization: Methods and Applications", Winter-1997

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Methods

Methods:

- Gradient-based Method:
  - Infinitesimal Perturbation Analysis (IPA)
  - Likelihood Ratio Method (LR)[2]
  - IPA/LR combined
  - Sample path optimization
- Response Surface Methodology
- Heuristic Methods: GA, ES, SA, Tabu-Search, ...

[2] Also: Score function method

Connection to Reinforcement Learning (RL)

- Policy-gradient
  - LR-methods
- PEGASUS[3]
  - Sample-path optimization
- TD-methods:
  - No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Connection to Reinforcement Learning (RL)

- Policy-gradient
  - LR-methods
- PEGASUS[3]
  - Sample-path optimization
- TD-methods:
  - No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Connection to Reinforcement Learning (RL)

- Policy-gradient
  - LR-methods
- PEGASUS[3]
  - Sample-path optimization
- TD-methods:
  - No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Connection to Reinforcement Learning (RL)

- Policy-gradient

  LR-methods
- PEGASUS[3]

  Sample-path optimization
- TD-methods:

  No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Connection to Reinforcement Learning (RL)

- Policy-gradient
  LR-methods
- PEGASUS[3]
  Sample-path optimization
- TD-methods:
  No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Connection to Reinforcement Learning (RL)

- Policy-gradient

  LR-methods
- PEGASUS[3]

  Sample-path optimization
- TD-methods:

  No counterpart

[3] A. Ng & M. Jordan: "PEGASUS: A policy search method for large MDPs and POMDPs", *UAI-2000*.

Outline

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

---

[a] Kiefer-Wolfowitz (1952), Blum (1964)

[b] Spall (1992)

[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

---

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

[a] Kiefer-Wolfowitz (1952), Blum (1964)
[b] Spall (1992)
[c] Spall (1997)

Gradient methods

Analytic expression for the gradient available?

YES

- IPA
- LR
- Sample-path optimization

NO

Finite Difference Methods

- FDSA[a]
- SPSA[b]
- Higher order methods
- One-measurement SPSA[c]

---

[a] Kiefer-Wolfowitz (1952), Blum (1964)

[b] Spall (1992)

[c] Spall (1997)

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable
  - e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Why gradient-free?

PRO

- Too expensive to maintain analytic gradient (change-mgmt)
- Analytic gradient computationally intractable

  e.g. features computed using Monte-Carlo
- Easy to implement
- Efficient???

CONTRA

- Inefficient compared to analytic-methods?

Outline

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}[\hat{g}_t(\theta_t) \mid \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \,|\, \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}\,[\hat{g}_t(\theta_t) \,|\, \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Stochastic Gradient Ascent (SGA)

SGA

Objective:

$$f(\theta) = \mathbb{E}[f(\theta; Y)] \to \max$$

Procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \hat{g}_t(\theta_t)$$

Assumptions:

- Step-size: $\sum_t \alpha_t = +\infty$, $\sum_t \alpha_t^2 < +\infty$
- Gradient-estimate: $\mathbb{E}[\hat{g}_t(\theta_t) \,|\, \mathcal{F}_t] = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$.
- Conditional variance: $\mathrm{Var}\,[\hat{g}_t(\theta_t) \,|\, \mathcal{F}_t] < K^2(1 + \|\theta_t\|^2)$.

Robbins-Monro/IPA

Assumption:

$$Y \sim p(\cdot),$$

p is not dependent on θ

Gradient estimate:

$$\hat{g}_t(\theta_t) = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$$

"Infinitesimal Perturbation Analysis" $\equiv$ IPA

Robbins-Monro/IPA

Assumption:

$$Y \sim p(\cdot),$$

p is not dependent on θ
Gradient estimate:

$$\hat{g}_t(\theta_t) = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$$

"Infinitesimal Perturbation Analysis" $\equiv$ IPA

Robbins-Monro/IPA

Assumption:

$$Y \sim p(\cdot),$$

p is not dependent on θ
Gradient estimate:

$$\hat{g}_t(\theta_t) = \frac{\partial}{\partial \theta} f(\theta_t; Y_t)$$

"Infinitesimal Perturbation Analysis" $\equiv$ IPA

Likelihood-Ratio Method – I.

Assumption

$$Y \sim p_\theta(\cdot),$$

p is dependent on θ, but

$$f(\theta; Y) = f(Y),$$

i.e., f is independent of θ.

Likelihood-ratio Method – II.

- Gradient estimate:

$$\begin{aligned} f'(\theta) &= \frac{d}{d\theta}\int f(y)p_\theta(y)dy \\ &= \int f(y)\frac{d}{d\theta}p_\theta(y)dy \\ &= \int f(y)\frac{\frac{d}{d\theta}p_\theta(y)}{p_\theta(y)}p_\theta(y)dy \\ &= \int f(y)\frac{d}{d\theta}\ln p_\theta(y)p_\theta(y)dy, \end{aligned}$$

- .. hence

$$\begin{aligned} Y_t &\sim p_{\theta_t}(\cdot), \\ \hat{g}_t(\theta_t) &= f(Y_t)\frac{d}{d\theta}\ln p_{\theta_t}(Y_t) \end{aligned}$$

Likelihood-ratio Method – II.

- Gradient estimate:

$$\begin{aligned} f'(\theta) &= \frac{d}{d\theta}\int f(y)p_\theta(y)dy \\ &= \int f(y)\frac{d}{d\theta}p_\theta(y)dy \\ &= \int f(y)\frac{\frac{d}{d\theta}p_\theta(y)}{p_\theta(y)}p_\theta(y)dy \\ &= \int f(y)\frac{d}{d\theta}\ln p_\theta(y)p_\theta(y)dy, \end{aligned}$$

- .. hence

$$\begin{aligned} Y_t &\sim p_{\theta_t}(\cdot), \\ \hat{g}_t(\theta_t) &= f(Y_t)\frac{d}{d\theta}\ln p_{\theta_t}(Y_t) \end{aligned}$$

Variance of LR estimate

Assumption: $Y = (Y_0, Y_1, \ldots, Y_T)$ is a long trajectory.

$$\begin{aligned}\hat{g}(\theta) &= f(Y)\frac{d}{d\theta}\ln p_\theta(Y) \\ &= f(Y)\sum_{t=0}^{T}\frac{d}{d\theta}\ln p_\theta(Y_t|Y_{0:t-1})\end{aligned}$$

$\Rightarrow$ Variance can be high if T is big

Variance of LR estimate

Assumption: $Y = (Y_0, Y_1, \ldots, Y_T)$ is a long trajectory.

$$\begin{aligned} \hat{g}(\theta) &= f(Y) \frac{d}{d\theta} \ln p_\theta(Y) \\ &= f(Y) \sum_{t=0}^{T} \frac{d}{d\theta} \ln p_\theta(Y_t | Y_{0:t-1}) \end{aligned}$$

$\Rightarrow$ Variance can be high if T is big

Variance of LR estimate

Assumption: $Y = (Y_0, Y_1, \ldots, Y_T)$ is a long trajectory.

$$\begin{aligned} \hat{g}(\theta) &= f(Y) \frac{d}{d\theta} \ln p_\theta(Y) \\ &= f(Y) \sum_{t=0}^{T} \frac{d}{d\theta} \ln p_\theta(Y_t | Y_{0:t-1}) \end{aligned}$$

$\Rightarrow$ Variance can be high if T is big

FDSA

FDSA Gradient Estimate

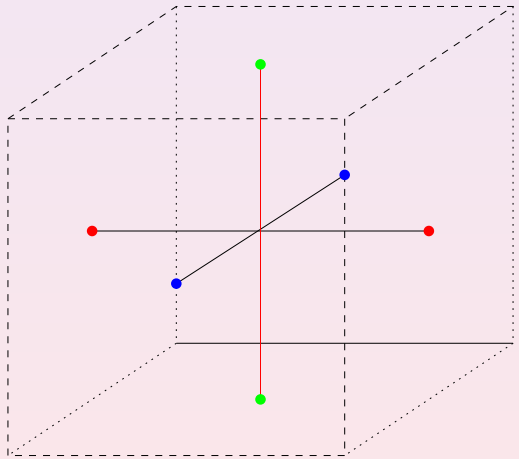
$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t e_i; Y_{ti}^+) - f(\theta_t - c_t e_i; Y_{ti}^-)}{2c_t}$$

2d evaluations of f

FDSA

FDSA Gradient Estimate

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t e_i; Y_{ti}^+) - f(\theta_t - c_t e_i; Y_{ti}^-)}{2c_t}$$

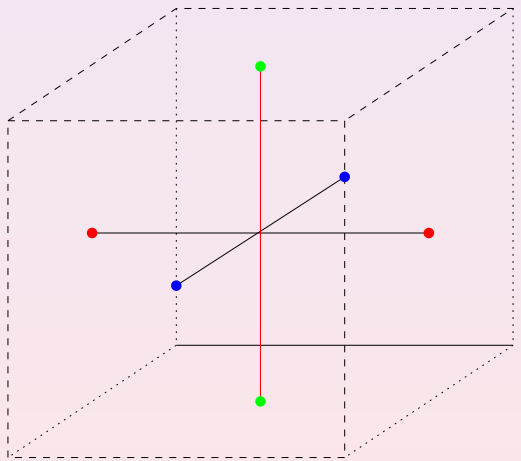


2d evaluations of f

FDSA

FDSA Gradient Estimate

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t e_i; Y_{ti}^+) - f(\theta_t - c_t e_i; Y_{ti}^-)}{2c_t}$$

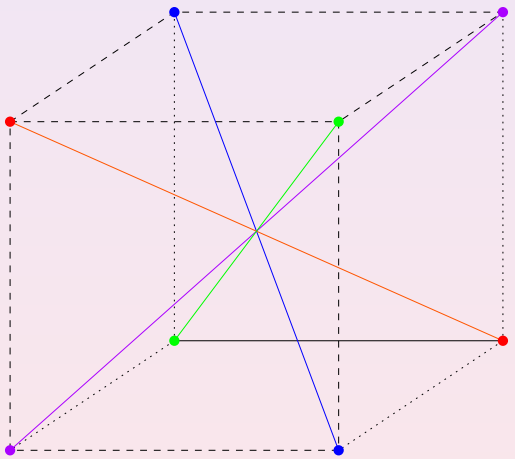


2d evaluations of f

SPSA

SPSA Gradient Estimate

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}$$



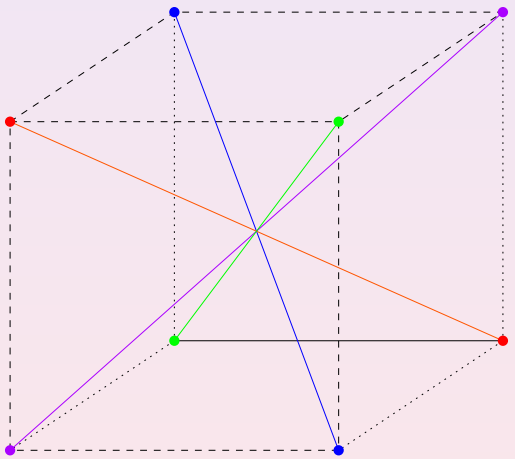
Only 2 evaluations of f

Variants: Discrete-parameter spaces, Non-smooth optimization via noise injection, higher-order methods

SPSA

SPSA Gradient Estimate

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}$$



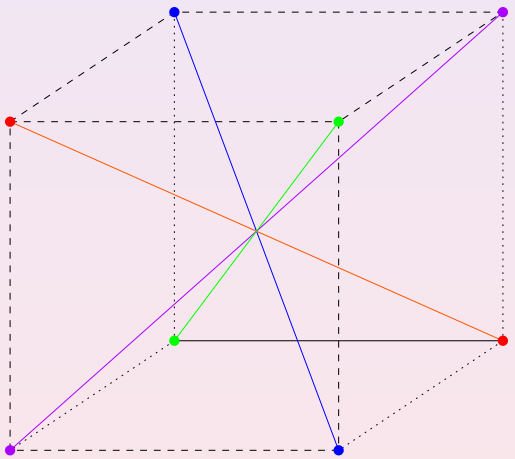
Only 2 evaluations of f

Variants: Discrete-parameter spaces, Non-smooth optimization via noise injection, higher-order methods

SPSA

SPSA Gradient Estimate

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}$$



Only 2 evaluations of f

Variants: Discrete-parameter spaces, Non-smooth optimization via noise injection, higher-order methods

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

SPSA Convergence

(A1) Δ_t are independent of the past
(A2) $\{\Delta_{ti}\}_i$ is i.i.d.
(A3) Distribution of Δ_{ti} is symmetric around zero
(A4) $|\Delta_{ti}|$ is bounded w.p.1
(A5) $\mathbb{E}[\Delta_{ti}^{-1}]$ is finite

KW step-size conditions:

- $\alpha_t, c_t > 0$, $\lim_{t\to\infty} c_t = 0$
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 / c_t^2 < \infty$

Theorem (Spall, 1992)

If f is sufficiently smooth, the step-sizes satisfy the KW conditions then θ_t converges w.p.1 to a local optimum of f. The bias of the estimate of the gradient is of order $O(c_t^2)$.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$

  Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]

  $p = 3 \Rightarrow$ optimal rate

---

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$
- Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]
- $p = 3 \Rightarrow$ optimal rate

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$

  Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]

  $p = 3 \Rightarrow$ optimal rate

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$

  Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]

  $p = 3 \Rightarrow$ optimal rate

---

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$

  Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]

  $p = 3 \Rightarrow$ optimal rate

---

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Efficiency

- Asymptotic Performance: SPSA is d-times more efficient than FDSA[4]
- Transient Performance: FDSA is more efficient than SPSA[5]
- Robbins-Monro: $O(t^{-1/2})$
- SPSA and FDSA: $O(t^{-1/3})$

  Optimal rate for randomized KW: $O(t^{-(p-1)/(2p)})$[6]

  $p = 3 \Rightarrow$ optimal rate

---

[4] J. C. Spall: "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation", IEEE TAC, 37:332–341, 1992

[5] J. Dippon: "Accelerated randomized stochastic optimization", *Annals of Stat.*, 41:12600–1281, 2003

[6] H. Chen: "Lower rate convergence for locating a maximum of a function", Ann.Stat., 16:1330–1334, 1988.

Outline

Common Random Numbers – I.

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}.$$

$$\Delta F_i = F_i^+ - F_i^-$$

$$\mathrm{Var}\left[F_i^+ - F_i^-\right] = \mathrm{Var}\left[F_i^+\right] + \mathrm{Var}\left[F_i^-\right] - 2\mathrm{Cov}\left(F_i^+, F_i^-\right),$$

- Y_t^-, Y_t^+ independent: $\mathrm{Cov}\left(F_i^+, F_i^-\right) = 0$
- $Y_t^- = Y_t^+ = Y_t$: $\mathrm{Cov}\left(F_i^+, F_i^-\right) > 0$.

Common Random Numbers – I.

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}.$$

$$\Delta F_i = F_i^+ - F_i^-$$

$$\mathrm{Var}\left[F_i^+ - F_i^-\right] = \mathrm{Var}\left[F_i^+\right] + \mathrm{Var}\left[F_i^-\right] - 2\mathrm{Cov}\left(F_i^+, F_i^-\right),$$

- Y_t^-, Y_t^+ independent: $\mathrm{Cov}\left(F_i^+, F_i^-\right) = 0$
- $Y_t^- = Y_t^+ = Y_t$: $\mathrm{Cov}\left(F_i^+, F_i^-\right) > 0$.

Common Random Numbers – I.

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}.$$

$$\Delta F_i = F_i^+ - F_i^-$$

$$\mathrm{Var}\left[F_i^+ - F_i^-\right] = \mathrm{Var}\left[F_i^+\right] + \mathrm{Var}\left[F_i^-\right] - 2\mathrm{Cov}\left(F_i^+, F_i^-\right),$$

- Y_t^-, Y_t^+ independent: $\mathrm{Cov}\left(F_i^+, F_i^-\right) = 0$
- $Y_t^- = Y_t^+ = Y_t$: $\mathrm{Cov}\left(F_i^+, F_i^-\right) > 0$.

Common Random Numbers – I.

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}.$$

$$\Delta F_i = F_i^+ - F_i^-$$

$$\mathrm{Var}\left[F_i^+ - F_i^-\right] = \mathrm{Var}\left[F_i^+\right] + \mathrm{Var}\left[F_i^-\right] - 2\mathrm{Cov}\left(F_i^+, F_i^-\right),$$

- Y_t^-, Y_t^+ independent: $\mathrm{Cov}\left(F_i^+, F_i^-\right) = 0$
- $Y_t^- = Y_t^+ = Y_t$: $\mathrm{Cov}\left(F_i^+, F_i^-\right) > 0$.

Common Random Numbers – I.

$$\hat{g}_{ti}(\theta_t) = \frac{f(\theta_t + c_t\Delta_t; Y_t^+) - f(\theta_t - c_t\Delta_t; Y_t^-)}{2c_t\Delta_{ti}}.$$

$$\Delta F_i = F_i^+ - F_i^-$$

$$\mathrm{Var}\left[F_i^+ - F_i^-\right] = \mathrm{Var}\left[F_i^+\right] + \mathrm{Var}\left[F_i^-\right] - 2\mathrm{Cov}\left(F_i^+, F_i^-\right),$$

- Y_t^-, Y_t^+ independent: $\mathrm{Cov}\left(F_i^+, F_i^-\right) = 0$
- $Y_t^- = Y_t^+ = Y_t$: $\mathrm{Cov}\left(F_i^+, F_i^-\right) > 0$.

Common Random Numbers – Example

$$f(\theta; Y) = \theta Y$$

$$\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta} = Y$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta}|\Delta\right] = \mathrm{Var}\left[Y\right] \left(\stackrel{\mathrm{def}}{=} V\right)$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y^+)-f(\theta-c\Delta;Y^-)}{2c\Delta}|\Delta\right] = \left(\frac{\theta^2}{2c^2\Delta^2} + 1/2\right)V$$

Common Random Numbers – Example

$$f(\theta; Y) = \theta Y$$

$$\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta} = Y$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta}|\Delta\right] = \mathrm{Var}\left[Y\right] \left(\stackrel{\mathrm{def}}{=} V\right)$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y^+)-f(\theta-c\Delta;Y^-)}{2c\Delta}|\Delta\right] = \left(\frac{\theta^2}{2c^2\Delta^2} + 1/2\right)V$$

Common Random Numbers – Example

$$f(\theta; Y) = \theta Y$$

$$\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta} = Y$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta}|\Delta\right] = \mathrm{Var}\left[Y\right] \left(\stackrel{\mathrm{def}}{=} V\right)$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y^+)-f(\theta-c\Delta;Y^-)}{2c\Delta}|\Delta\right] = \left(\frac{\theta^2}{2c^2\Delta^2} + 1/2\right)V$$

Common Random Numbers – Example

$$f(\theta; Y) = \theta Y$$

$$\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta} = Y$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y)-f(\theta-c\Delta;Y)}{2c\Delta}|\Delta\right] = \mathrm{Var}\left[Y\right] (\stackrel{\mathrm{def}}{=} V)$$

$$\mathrm{Var}\left[\frac{f(\theta+c\Delta;Y^+)-f(\theta-c\Delta;Y^-)}{2c\Delta}|\Delta\right] = (\frac{\theta^2}{2c^2\Delta^2} + 1/2)V$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}[0,1]$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta}\left(\frac{Y}{\theta} - 1\right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

LR or SPSA? – Example[7]

- LR-estimate
  - $Y = Y_\theta \sim \exp(- \cdot /\theta)/\theta$
  - Objective: $f(\theta) = E[Y_\theta]$
  - $f(\theta) = \int y p_\theta(y) dy$.
  - LR-estimate:

$$Y \frac{\partial}{\partial \theta} \ln p_\theta(Y) = \frac{Y}{\theta} \left(\frac{Y}{\theta} - 1 \right).$$

- SPSA-estimate
  - $U \sim \mathcal{U}_{[0,1]}$
  - $Y = \theta \log U \sim p_\theta(\cdot)$
  - Define: $f(\theta; U) = -\theta \log U$.
  - CRN-based SPSA-gradient:

$$\frac{(\theta + c\Delta)(-\log U) - (\theta - c\Delta)(-\log U)}{2c\Delta} = -\log U.$$

Application to Games

State-space view:

$$\begin{aligned} X_{t+1} &= f(X_t, A_t, U_t) \\ A_t &= \pi(X_t, U'_t) \\ R_t &= R(X_t, A_t) \\ & \mathbb{E}[\sum_{t=0}^{\infty} R_t] \to \max \end{aligned}$$

Application to Games

- Use the same sequence of random variables U_t, U'_t for both evaluations:

$$Y = (U_1, \ldots, U_T, U'_1, \ldots, U'_T)$$

(fix an infinite random sample in advance)

- Easy impl: Manipulate the seed of pseudorandom-generators
- $f = f(\theta; Y)$ – can be complicated!
- ..but the distribution of Y is independent of θ
- $f(\theta; Y)$ might be non-differentiable w.r.t. θ, even when $f(\theta) = \mathbb{E}[f(\theta; Y)]$ is differentiable!

Application to Games

- Use the same sequence of random variables U_t, U'_t for both evaluations:

$$Y = (U_1, \ldots, U_T, U'_1, \ldots, U'_T)$$

(fix an infinite random sample in advance)

- Easy impl: Manipulate the seed of pseudorandom-generators
- $f = f(\theta; Y)$ – can be complicated!
- ..but the distribution of Y is independent of θ
- $f(\theta; Y)$ might be non-differentiable w.r.t. θ, even when $f(\theta) = \mathbb{E}[f(\theta; Y)]$ is differentiable!

Application to Games

- Use the same sequence of random variables U_t, U'_t for both evaluations:

$$Y = (U_1, \ldots, U_T, U'_1, \ldots, U'_T)$$

(fix an infinite random sample in advance)

- Easy impl: Manipulate the seed of pseudorandom-generators
- $f = f(\theta; Y)$ – can be complicated!
- ..but the distribution of Y is independent of θ
- $f(\theta; Y)$ might be non-differentiable w.r.t. θ, even when $f(\theta) = \mathbb{E}[f(\theta; Y)]$ is differentiable!

Application to Games

- Use the same sequence of random variables U_t, U'_t for both evaluations:

$$Y = (U_1, \ldots, U_T, U'_1, \ldots, U'_T)$$

(fix an infinite random sample in advance)

- Easy impl: Manipulate the seed of pseudorandom-generators
- $f = f(\theta; Y)$ – can be complicated!
- ..but the distribution of Y is independent of θ
- $f(\theta; Y)$ might be non-differentiable w.r.t. θ, even when $f(\theta) = \mathbb{E}[f(\theta; Y)]$ is differentiable!

Application to Games

- Use the same sequence of random variables U_t, U'_t for both evaluations:

$$Y = (U_1, \ldots, U_T, U'_1, \ldots, U'_T)$$

(fix an infinite random sample in advance)

- Easy impl: Manipulate the seed of pseudorandom-generators
- $f = f(\theta; Y)$ – can be complicated!
- ..but the distribution of Y is independent of θ
- $f(\theta; Y)$ might be non-differentiable w.r.t. θ, even when $f(\theta) = \mathbb{E}[f(\theta; Y)]$ is differentiable!

Outline

Batch-size

$$\hat{g}_{q,i}(\theta) = \frac{1}{q}\sum_{j=1}^{q}\frac{f(\theta + c\Delta; Y_j) - f(\theta - c\Delta; Y_j)}{2c\,\Delta_i}.$$

$$\hat{g}_{q,r,i} = \frac{1}{r}\sum_{k=1}^{r}\hat{g}_{q,i}^{(k)}(\theta)$$

How to select q, r s.t. $q \times r \equiv p$?

Use as many independent samples as possible:

$$q = 1, \quad r = p$$

Batch-size

$$\hat{g}_{q,i}(\theta) = \frac{1}{q}\sum_{j=1}^{q}\frac{f(\theta + c\Delta; Y_j) - f(\theta - c\Delta; Y_j)}{2c\,\Delta_i}.$$

$$\hat{g}_{q,r,i} = \frac{1}{r}\sum_{k=1}^{r}\hat{g}_{q,i}^{(k)}(\theta)$$

How to select q, r s.t. $q \times r \equiv p$?

Use as many independent samples as possible:

$$q = 1, \quad r = p$$

Batch-size

$$\hat{g}_{q,i}(\theta) = \frac{1}{q}\sum_{j=1}^{q}\frac{f(\theta + c\Delta; Y_j) - f(\theta - c\Delta; Y_j)}{2c\,\Delta_i}.$$

$$\hat{g}_{q,r,i} = \frac{1}{r}\sum_{k=1}^{r}\hat{g}_{q,i}^{(k)}(\theta)$$

How to select q, r s.t. $q \times r \equiv p$?

Use as many independent samples as possible:

$$q = 1, \quad r = p$$

Batch-size

$$\hat{g}_{q,i}(\theta) = \frac{1}{q}\sum_{j=1}^{q}\frac{f(\theta + c\Delta; Y_j) - f(\theta - c\Delta; Y_j)}{2c\,\Delta_i}.$$

$$\hat{g}_{q,r,i} = \frac{1}{r}\sum_{k=1}^{r}\hat{g}_{q,i}^{(k)}(\theta)$$

How to select q, r s.t. $q \times r \equiv p$?

Use as many independent samples as possible:

$$q = 1, \quad r = p$$

Outline

RPROP – I.

Problem

How to select α_t, c_t?

- Idea: Use different values for each of the d components of θ
- SGA step-size methods
- RPROP – one of the best performing first-order batch neural network method
- Combine with RPROP

RPROP – I.

Problem

How to select α_t, c_t?

- Idea: Use different values for each of the d components of θ
- SGA step-size methods
- RPROP – one of the best performing first-order batch neural network method
- Combine with RPROP

RPROP – I.

Problem

How to select α_t, c_t?

- Idea: Use different values for each of the d components of θ
- SGA step-size methods
- RPROP – one of the best performing first-order batch neural network method
- Combine with RPROP

RPROP – I.

Problem

How to select α_t, c_t?

- Idea: Use different values for each of the d components of θ
- SGA step-size methods
- RPROP – one of the best performing first-order batch neural network method
- Combine with RPROP

RPROP – I.

Problem

How to select α_t, c_t?

- Idea: Use different values for each of the d components of θ
- SGA step-size methods
- RPROP – one of the best performing first-order batch neural network method
- Combine with RPROP

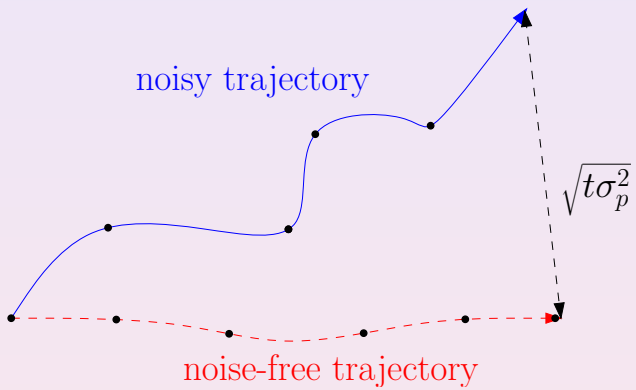
Resiliant backPROPagation (RPROP) – II.

- Individual learning rates δ_{ti}
- Sign-based updates
- $\theta_{t+1,i} = \theta_{t,i} + \delta_{ti}\text{sign}(g_{ti})$
- δ_{ti} increased if $g_{t-1,i}$ and g_{ti} have the same sign, decreased otherwise

RPROP – Update Equations

$$
\begin{aligned}
p_{t,i} &= g_{t-1,i} f_i'(\theta_t), \\
g_{ti} &= \mathbb{I}(p_{t,i} \geq 0) f_i'(\theta_t), \\
\eta_{ti} &= \mathbb{I}(p_{t,i} > 0)\eta^+ + \mathbb{I}(p_{t,i} < 0)\eta^- + \mathbb{I}(p_{t,i} = 0), \\
\delta_{ti} &= P_{[\delta^-,\delta^+]}\left(\eta_{ti}\delta_{t-1,i}\right)
\end{aligned}
$$

Noise suppression

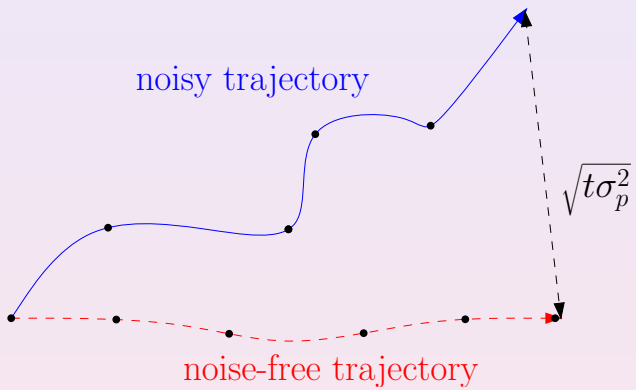


$$\delta_p \sim \exp(-pG^2/(2\sigma_1^2))$$

$$\sigma_p^2 = \frac{1}{p}\sigma_1^2$$

Note: RPROP's equilibrium behaviour is not "ideal"

Noise suppression

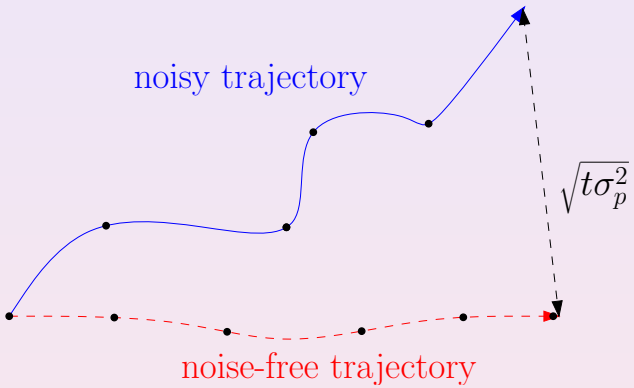


$$\sigma_p^2 = \frac{1}{p}\sigma_1^2$$

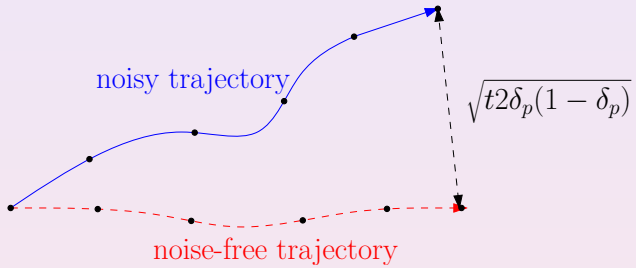
$$\delta_p \sim \exp(-pG^2/(2\sigma_1^2))$$

Note: RPROP's equilibrium behaviour is not "ideal"

Noise suppression



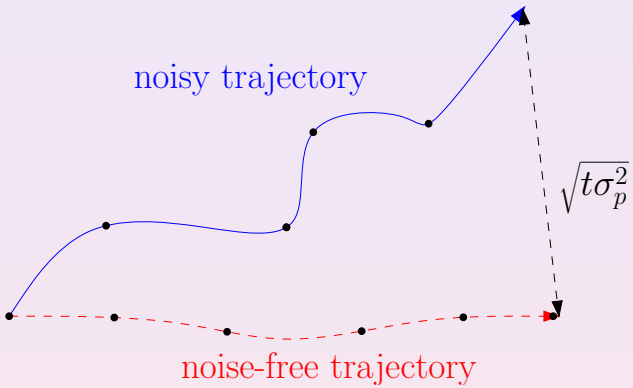
$$\sigma_p^2 = \frac{1}{p}\sigma_1^2$$



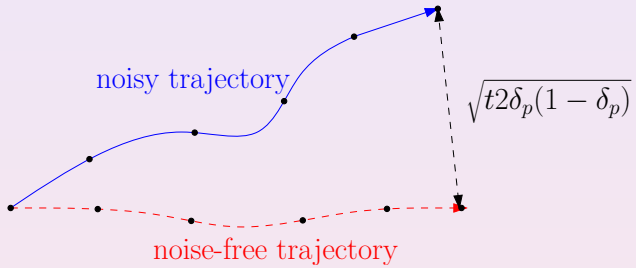
$$\delta_p \sim \exp(-pG^2/(2\sigma_1^2))$$

Note: RPROP's equilibrium behaviour is not "ideal"

Noise suppression



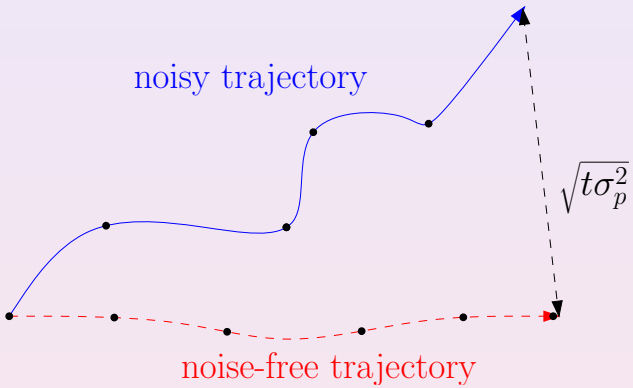
$$\sigma_p^2 = \frac{1}{p}\sigma_1^2$$



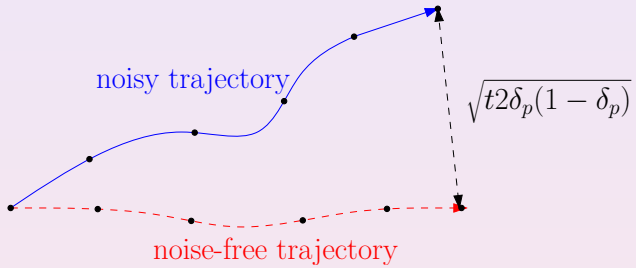
$$\delta_p \sim \exp(-pG^2/(2\sigma_1^2))$$

Note: RPROP's equilibrium behaviour is not "ideal"

Noise suppression



$$\sigma_p^2 = \frac{1}{p}\sigma_1^2$$



$$\delta_p \sim \exp(-pG^2/(2\sigma_1^2))$$

Note: RPROP's equilibrium behaviour is not "ideal"

The RSPSA algorithm

Idea: Combine SPSA and RPROP – couple step-sizes

for (*iteration*)
  $d\theta = 0$
  for (*batch*)
    for (*i*) $p_i = \rho \cdot \delta_i \cdot randb(\pm 1)$
    generate(Y_t)
    $f^+ = \text{eval}(\theta + p, Y_t)$
    $f^- = \text{eval}(\theta - p, Y_t)$
    for (*i*) $d\theta_i$ += $(f^+ - f^-)/(2p_i)$
  RPROPupdate(θ, $d\theta$, δ)

The RSPSA algorithm

Idea: Combine SPSA and RPROP – couple step-sizes

for (*iteration*)
$d\theta = 0$
for (*batch*)
for (*i*) $p_i = \rho \cdot \delta_i \cdot randb(\pm 1)$
generate(Y_t)
$f^+ = \text{eval}(\theta + p, Y_t)$
$f^- = \text{eval}(\theta - p, Y_t)$
for (*i*) $d\theta_i$ += $(f^+ - f^-)/(2p_i)$
RPROPupdate(θ, $d\theta$, δ)

Outline

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:

$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$

- Variance is $(1/n)\,\mathrm{Var}\,[R]$
- Assume $n = 2k$, consider

$$I_n^A = \frac{1}{k}\sum_{i=1}^{k} \frac{R_i^+ + R_i^-}{2},$$

- Variance of a pair:

$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+, R_i^-\right)}{4},$$

- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+, R_i^-\right) \leq 0$

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:

$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$

- Variance is $(1/n)\,\mathrm{Var}[R]$
- Assume $n = 2k$, consider

$$I_n^A = \frac{1}{k}\sum_{i=1}^{k} \frac{R_i^+ + R_i^-}{2},$$

- Variance of a pair:

$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+, R_i^-\right)}{4},$$

- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+, R_i^-\right) \leq 0$

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:

$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$

- Variance is $(1/n)\,\mathrm{Var}[R]$
- Assume $n = 2k$, consider

$$I_n^A = \frac{1}{k}\sum_{i=1}^{k} \frac{R_i^+ + R_i^-}{2},$$

- Variance of a pair:

$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+, R_i^-\right)}{4},$$

- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+, R_i^-\right) \leq 0$

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:

$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$

- Variance is $(1/n)\,\mathrm{Var}[R]$
- Assume $n = 2k$, consider

$$I_n^A = \frac{1}{k}\sum_{i=1}^{k}\frac{R_i^+ + R_i^-}{2},$$

- Variance of a pair:

$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+, R_i^-\right)}{4},$$

- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+, R_i^-\right) \leq 0$

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:
$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$
- Variance is $(1/n)\,\mathrm{Var}\,[R]$
- Assume $n = 2k$, consider
$$I_n^A = \frac{1}{k}\sum_{i=1}^{k}\frac{R_i^+ + R_i^-}{2},$$
- Variance of a pair:
$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+\,,\,R_i^-\right)}{4},$$
- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+\,,\,R_i^-\right) \leq 0$

Antithetic Variates

- Problem: Estimate $I = \mathbb{E}[R]$.
- Monte-Carlo:
$$I_n = \frac{1}{n}(R_1 + \ldots + R_n)$$
- Variance is $(1/n)\,\mathrm{Var}\,[R]$
- Assume $n = 2k$, consider
$$I_n^A = \frac{1}{k}\sum_{i=1}^{k}\frac{R_i^+ + R_i^-}{2},$$
- Variance of a pair:
$$\mathrm{Var}\left[\frac{R_i^+ + R_i^-}{2}\right] = \frac{\mathrm{Var}\left[R_i^+\right] + \mathrm{Var}\left[R_i^-\right] + 2\mathrm{Cov}\left(R_i^+\,,\,R_i^-\right)}{4},$$
- Make R_i^+, R_i^- "antithetic": $\mathrm{Cov}\left(R_i^+\,,\,R_i^-\right) \leq 0$

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y: Original random deal
  - Y': Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1), f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^{id}: Original random deck
  - $Y^{\square}$: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1), f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^{id}: Original random deck
  - Y^{rev}: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\text{Cov}\left(f(Y; W_1),\, f(Y'; W_2)\right) \approx -\text{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^+: Original random deck
  - Y^-: Reverse the hands of the players
- Distributions of $f(Y; W_1)$ $-f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1), f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^+: Original random deck
  - Y^-: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1),\, f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^+: Original random deck
  - Y^-: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1),\, f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^+: Original random deck
  - Y^-: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1), f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Antithetic Variates – Example

- $f = f(\theta; Y, W)$
- Y – external choices, W – internal choices of players
- Example (Y) in poker: Cards
- "Antithetic dealing":
  - Y^+: Original random deck
  - Y^-: Reverse the hands of the players
- Distributions of $f(Y; W_1) - f(Y'; W_2)$ are aligned:

$$\mathrm{Cov}\left(f(Y; W_1), f(Y'; W_2)\right) \approx -\mathrm{Var}\left[f(Y; W)\right]$$

- Practice: 20-fold reduction of variance in Omaha Hi-Lo

Outline

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Omeha Hi-Lo: The game

- small blind (5$)/big blind (10$)
- each player (2-10) gets 4 hole cards
- betting (1st round)
- flop: 3 community cards
- betting (2nd round)
- turn: 1 community card
- betting (3rd round)
- river: 1 community card
- betting (4th round)
- showdown

Betting

- 3 actions: fold, check/call, bet/raise
- raise: +1$ in 1st and 2nd round, +2$ in 3rd and 4th round
- maximum of 4 raises allowed per round
- betting is over when all the active players have placed equal amounts in the pot

Betting

- 3 actions: fold, check/call, bet/raise
- raise: +1$ in 1st and 2nd round, +2$ in 3rd and 4th round
- maximum of 4 raises allowed per round
- betting is over when all the active players have placed equal amounts in the pot

Betting

- 3 actions: fold, check/call, bet/raise
- raise: +1$ in 1st and 2nd round, +2$ in 3rd and 4th round
- maximum of 4 raises allowed per round
- betting is over when all the active players have placed equal amounts in the pot

Betting

- 3 actions: fold, check/call, bet/raise
- raise: +1$ in 1st and 2nd round, +2$ in 3rd and 4th round
- maximum of 4 raises allowed per round
- betting is over when all the active players have placed equal amounts in the pot

Showdown

- high hand and low hand (2 hole cards + 3 community cards)
- High: the best high hand wins (at least) half of the pot
- Low: the best low hand (if any) wins half of the pot
- low hand: A to 8; the lowest high card wins

Showdown

- high hand and low hand (2 hole cards + 3 community cards)
- High: the best high hand wins (at least) half of the pot
- Low: the best low hand (if any) wins half of the pot
- low hand: A to 8; the lowest high card wins

Showdown

- high hand and low hand (2 hole cards + 3 community cards)
- High: the best high hand wins (at least) half of the pot
- Low: the best low hand (if any) wins half of the pot
- low hand: A to 8; the lowest high card wins

Showdown

- high hand and low hand (2 hole cards + 3 community cards)
- High: the best high hand wins (at least) half of the pot
- Low: the best low hand (if any) wins half of the pot
- low hand: A to 8; the lowest high card wins

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s, a) = \mathbb{E}[w(C, I)R - R_1 \mid s, a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \text{argmax}_a Q(s, a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

Action-selection

- s – information state of player
- a – action (fold, call, raise)
- $Q(s,a) = \mathbb{E}[w(C,I)R - R_1 \mid s,a]$
  - C – random card configuration
  - I – set of active players at showdown
  - R – final size of the pot
  - R_1 – McRaise's final contribution
- $\pi^*(s) = \mathrm{argmax}_a Q(s,a)$

EC assumption

- EC $\equiv$ "Everyone's Checking"
- $R = \Pi(s, a)$
- $R_1 = B(s, a)$
- $\hat{Q}(s, a) = \mathbb{E}_{\mathrm{EC}}[w(C, I)]\Pi(s, a) - B(s, a)$
- $\overline{w}(s, a) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mathrm{EC}}[w(C, I)]$:

Winning Chances

EC assumption

- EC $\equiv$ "Everyone's Checking"
- $R = \Pi(s, a)$
- $R_1 = B(s, a)$
- $\hat{Q}(s, a) = \mathbb{E}_{\mathrm{EC}}[w(C, I)]\Pi(s, a) - B(s, a)$
- $\overline{w}(s, a) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mathrm{EC}}[w(C, I)]$:

Winning Chances

EC assumption

- EC $\equiv$ "Everyone's Checking"
- $R = \Pi(s, a)$
- $R_1 = B(s, a)$
- $\hat{Q}(s, a) = \mathbb{E}_{\mathrm{EC}}[w(C, I)]\Pi(s, a) - B(s, a)$
- $\overline{w}(s, a) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mathrm{EC}}[w(C, I)]$:

Winning Chances

EC assumption

- EC $\equiv$ "Everyone's Checking"
- $R = \Pi(s, a)$
- $R_1 = B(s, a)$
- $\hat{Q}(s, a) = \mathbb{E}_{\mathrm{EC}}[w(C, I)]\Pi(s, a) - B(s, a)$
- $\overline{w}(s, a) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mathrm{EC}}[w(C, I)]$:

Winning Chances

EC assumption

- EC $\equiv$ "Everyone's Checking"
- $R = \Pi(s, a)$
- $R_1 = B(s, a)$
- $\hat{Q}(s, a) = \mathbb{E}_{\mathrm{EC}}[w(C, I)]\Pi(s, a) - B(s, a)$
- $\overline{w}(s, a) \stackrel{\mathrm{def}}{=} \mathbb{E}_{\mathrm{EC}}[w(C, I)]$:

Winning Chances

Estimating $\hat{Q}$

$$\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s).$$

Bayes-rule:

$$p(c|s) = \frac{p(s|c)p(c)}{p(s)}$$

How to compute $p(c|s)$? W.I.S.

Estimating $\hat{Q}$

$$\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s).$$

Bayes-rule:

$$p(c|s) = \frac{p(s|c)p(c)}{p(s)}$$

How to compute $p(c|s)$? W.I.S.

Estimating $\hat{Q}$

$$\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s).$$

Bayes-rule:

$$p(c|s) = \frac{p(s|c)p(c)}{p(s)}$$

How to compute $p(c|s)$? W.I.S.

Weighted Importance Sampling

$$\mathbb{E}[f(X)] = \sum_x f(x)p(x)$$

$$Y_i \sim q(\cdot)$$

$$S_n = \frac{\sum_{i=1}^{n} f(Y_i)\lambda(Y_i)}{\sum_{i=1}^{n} \lambda(Y_i)},$$

$$\lambda(Y_i) \propto p(Y_i)/q(Y_i)$$

Weighted Importance Sampling

$$\mathbb{E}[f(X)] = \sum_x f(x)p(x)$$

$$Y_i \sim q(\cdot)$$

$$S_n = \frac{\sum_{i=1}^{n} f(Y_i)\lambda(Y_i)}{\sum_{i=1}^{n} \lambda(Y_i)},$$

$$\lambda(Y_i) \propto p(Y_i)/q(Y_i)$$

Weighted Importance Sampling

$$\mathbb{E}[f(X)] = \sum_x f(x)p(x)$$
$$Y_i \sim q(\cdot)$$

$$S_n = \frac{\sum_{i=1}^{n} f(Y_i)\lambda(Y_i)}{\sum_{i=1}^{n} \lambda(Y_i)},$$
$$\lambda(Y_i) \propto p(Y_i)/q(Y_i)$$

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRAISE
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRAISE
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Estimating $\hat{Q}$

- $\overline{w}(s,a) = \mathbb{E}[w(C, I(s))|s,a] = \sum_c w(c, I(s))p(c|s)$.
- Estimate $\overline{w}$ by WIS: sufficient to compute $p(s|c)$
- $p(s|c) = p(\pi_1(s)|c) \prod_{t=1}^{m} p_{i_t}(a_t(s)|s_t, c_{i_t}(c))$
  - $\pi_1(s)$ – private cards of McRaise
  - p_i – action-selection model of player i
  - s_t – public history up to stage t
- Opponent model:
  - Collect actions within the same round
  - Actions within a round are not independent (player's have memories within rounds)

Outline

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

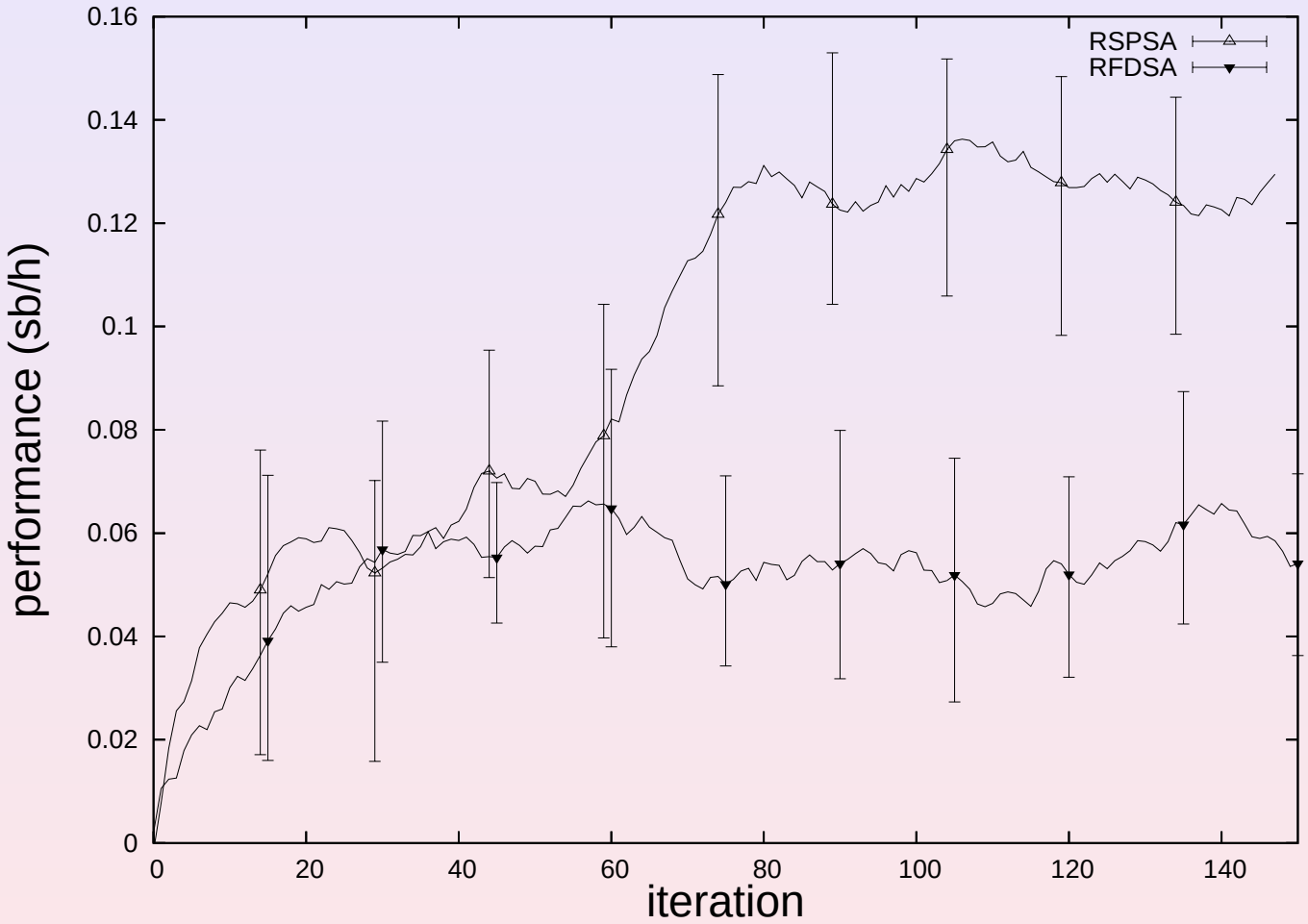
- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable

Tuning the opponent model

- 2-players, one is McRaise, other is “tuned”
- Tuning 6 parameters of the opponent model
- Compare non-asymptotic performance of RSPSA and (R)FDSA
- Initialization: original parameter settings
- Common random numbers: synchronized rngs (same deck)
- Antithetic dealing
- Batch-size of approx. 10,000 games
- Deck reuse
- Note: Objective function is non-differentiable


RSPSA
RFDSA
performance (sb/h)
iteration
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0
20
40
60
80
100
120
140

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

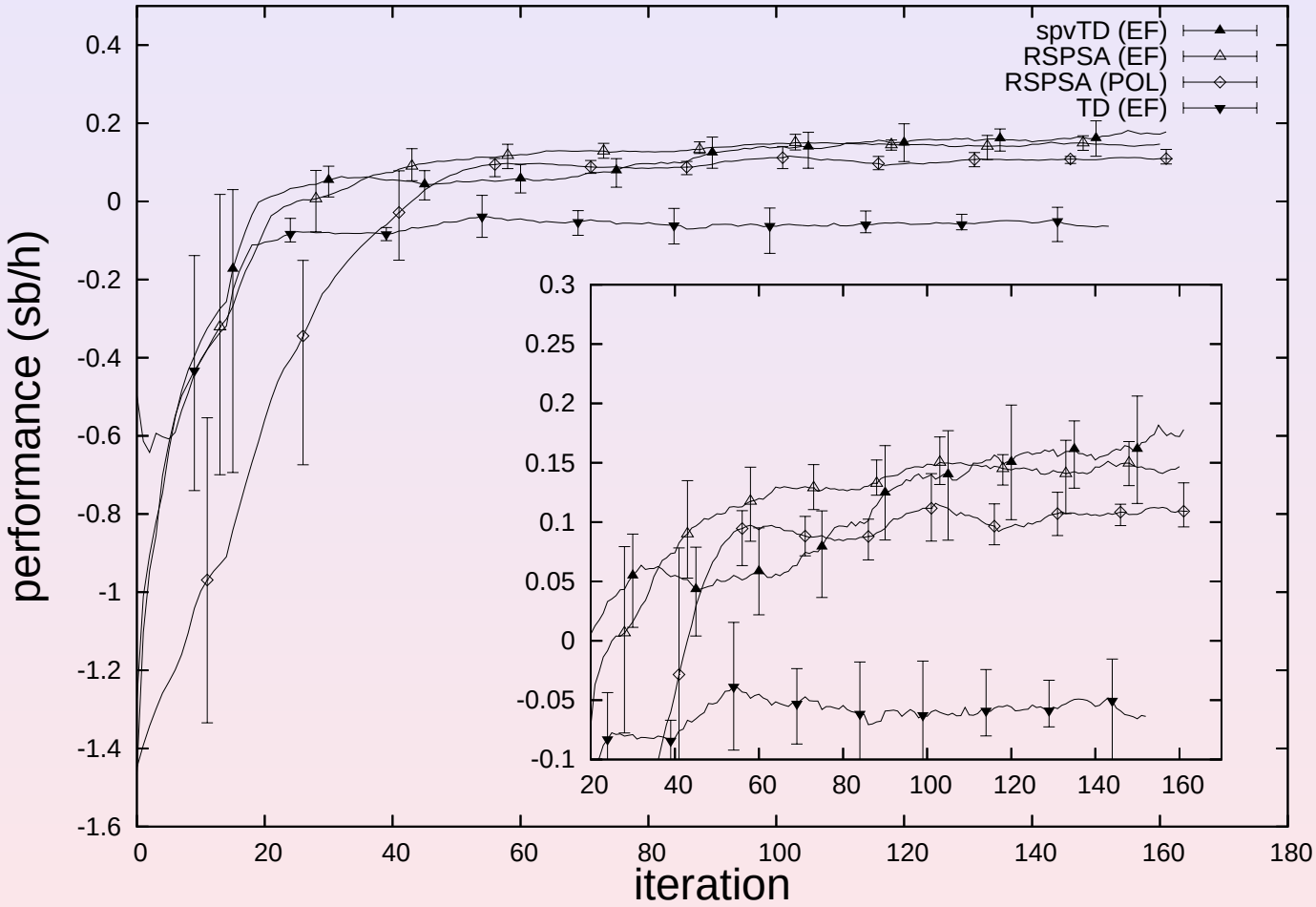
- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP

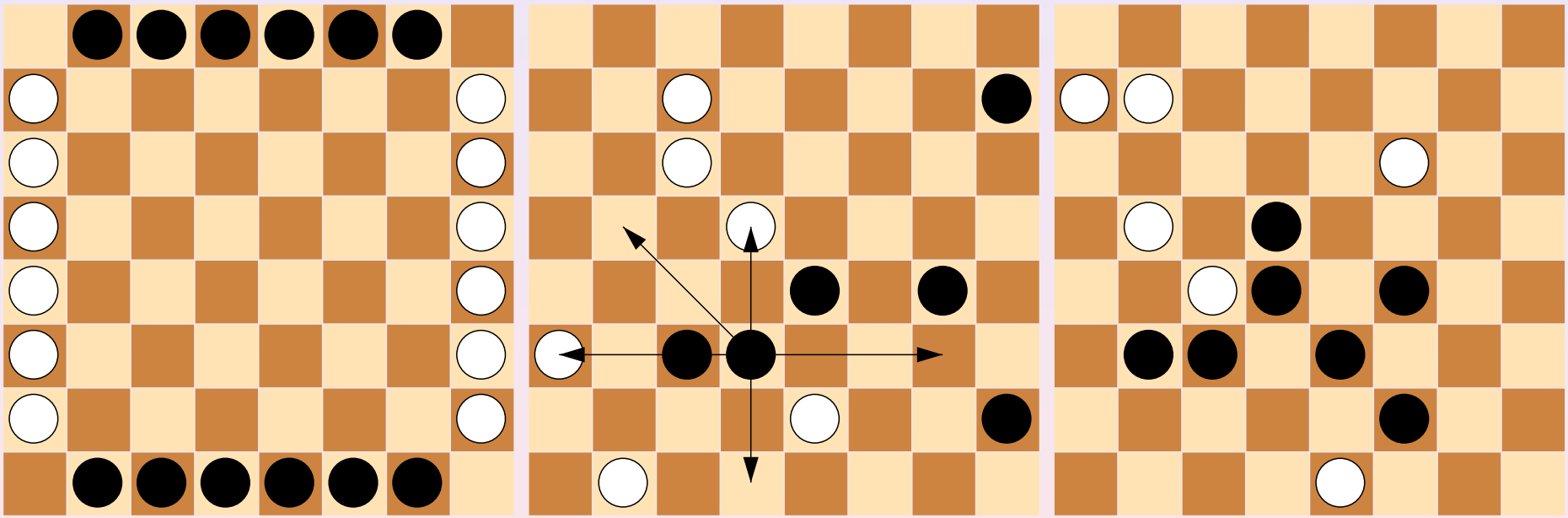
Learning policies and evaluation functions

- RSPSA
  - neural network representing after-state value-function
  - Action-network representing $p(a|s)$
  - Objective function is differentiable, but the exact gradient is intractable
- TD-learning
  - neural neural representing after-state value-function
  - with or without supervised initialization
  - TD(λ) combined with RPROP


spvTD (EF)
RSPSA (EF)
RSPSA (POL)
TD (EF)
performance (sb/h)
iteration
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
0
20
40
60
80
100
120
140
160
180
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
20
40
60
80
100
120
140
160

Outline

Lines of Actions



MIA4++

- Winner of the 8th and 9th Computer Olympiad
- Standard search-based program
- Tuned in the experiments: realisation probabilities (277 weights)

MIA4++

- Winner of the 8th and 9th Computer Olympiad
- Standard search-based program
- Tuned in the experiments: realisation probabilities (277 weights)

MIA4++

- Winner of the 8th and 9th Computer Olympiad
- Standard search-based program
- Tuned in the experiments: realisation probabilities (277 weights)

Outline

Tuning the Realization-Probability weights

- RSPSA
- Performance evaluation
  - Playing 500 games: 5 different opponents (different EF) starting from 50 fixed positions with both colours
  - The score of the selected moves in 5,000 positions (the score is given by a deeper search)

Tuning the Realization-Probability weights

- RSPSA
- Performance evaluation
  - Playing 500 games: 5 different opponents (different EF) starting from 50 fixed positions with both colours
  - The score of the selected moves in 5,000 positions (the score is given by a deeper search)

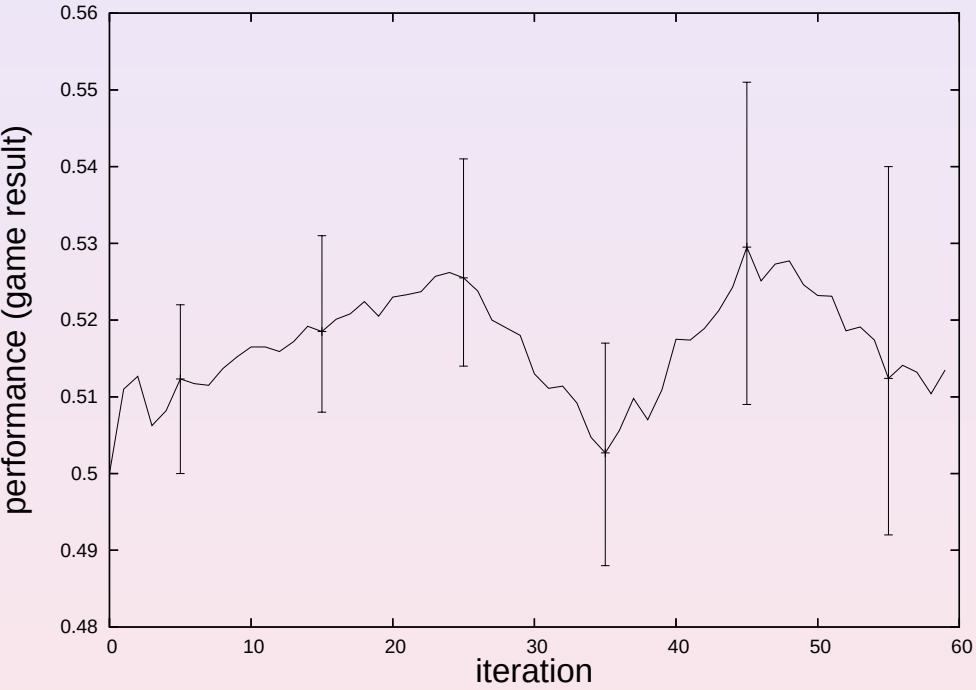
Tuning the Realization-Probability weights

- RSPSA
- Performance evaluation
  - Playing 500 games: 5 different opponents (different EF) starting from 50 fixed positions with both colours
  - The score of the selected moves in 5,000 positions (the score is given by a deeper search)

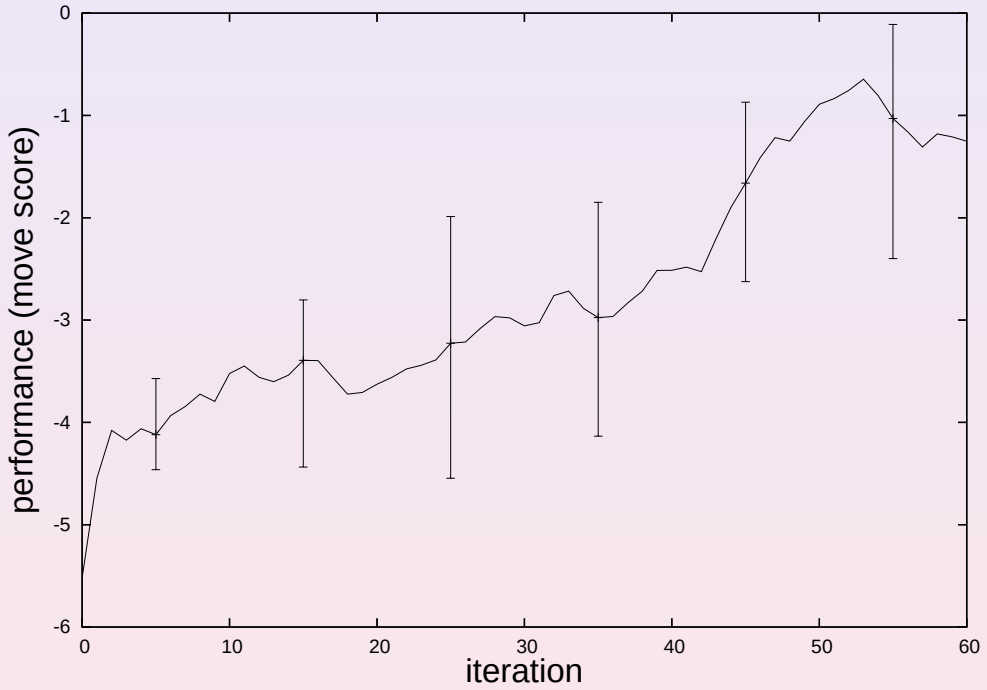
Tuning the Realization-Probability weights

- RSPSA
- Performance evaluation
  - Playing 500 games: 5 different opponents (different EF) starting from 50 fixed positions with both colours
  - The score of the selected moves in 5,000 positions (the score is given by a deeper search)

game results



move scores



Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)
  - Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Conclusions

- SPSA: Not necessarily worse than analytic gradient (LR-methods)

  Sometimes the only choice
- RSPSA algorithm = SPSA+RPROP
- Enhancements:
  - Common random numbers: same asymptotic rate as for LR-methods
  - Number of perturbations to be kept at maximum
  - Use antithetic variables
- Experiments in poker and LOA: RSPSA looks competitive with alternative optimization algorithms

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- MCRAISE:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...

Future Work

- Make batch-size dependent on the step-sizes
- Deterministic perturbation sequences
- Alternative step-size tuning
- McRaise:
  - Evaluate
  - Opponent model selection based on bandits
  - Better opponent modelling
  - Reduce predictability
  - ...