Stochastic Search using the Natural Gradient

Efficient Natural Evolution Strategies (eNES)

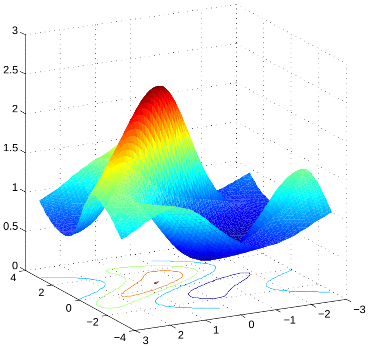
Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber
{yi,daan,tom,juergen}@idsia.ch

IDSIA, Galleria 2, Manno 6928, Switzerland

June 17th, 2009

Blackbox Optimization

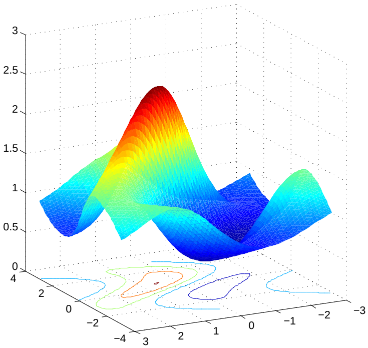
Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



Blackbox Optimization

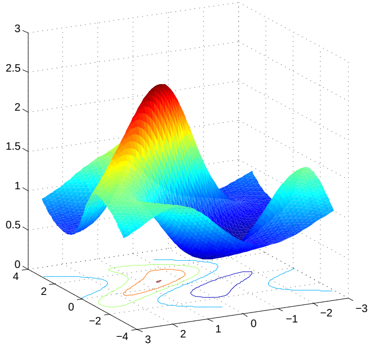
Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.

Challenge:



Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.

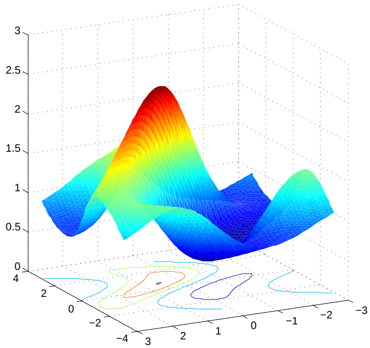


Challenge:

- Complex fitness landscapes.

Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.

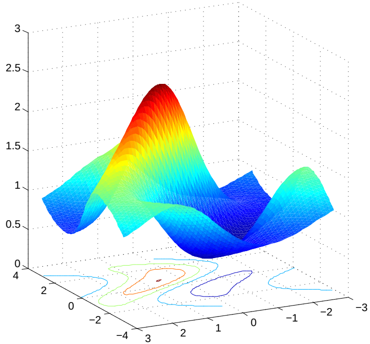


Challenge:

- Complex fitness landscapes.
  - Local optimas, saddle points, etc.

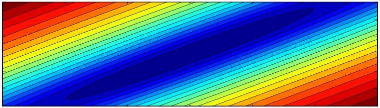
Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



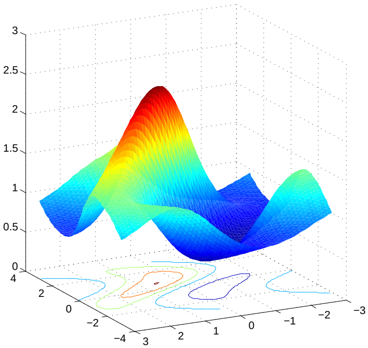
Challenge:

- Complex fitness landscapes.
  - Local optimas, saddle points, etc.
  - Highly non-isotropic (ill-shaped) local behavior.



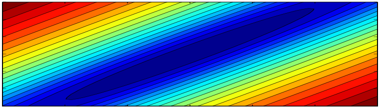
Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



Challenge:

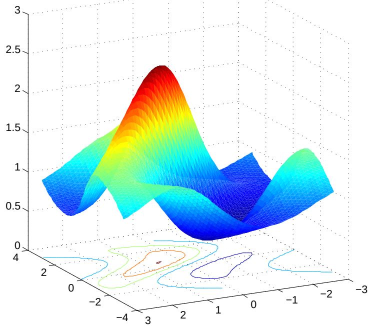
- Complex fitness landscapes.
  - Local optimas, saddle points, etc.
  - Highly non-isotropic (ill-shaped) local behavior.



  - Correlation between all dimensions.

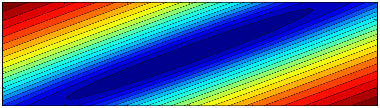
Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



Challenge:

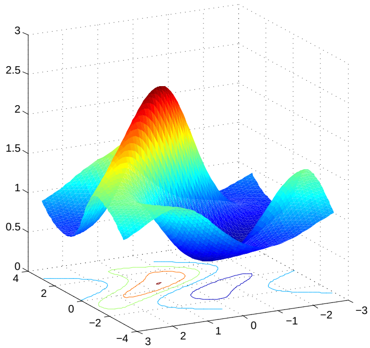
- Complex fitness landscapes.
  - Local optimas, saddle points, etc.
  - Highly non-isotropic (ill-shaped) local behavior.



  - Correlation between all dimensions.
- Expensive fitness evaluations.

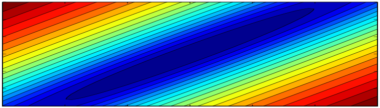
Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



Challenge:

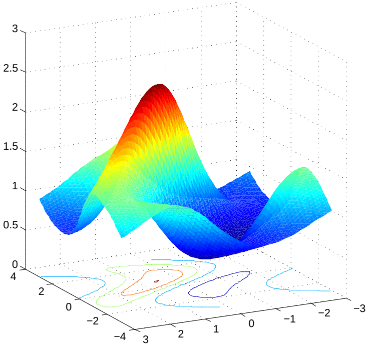
- Complex fitness landscapes.
  - Local optimas, saddle points, etc.
  - Highly non-isotropic (ill-shaped) local behavior.



  - Correlation between all dimensions.
- Expensive fitness evaluations.
- High dimensionality, d up to hundreds.

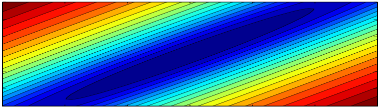
Blackbox Optimization

Goal: Maximizing some unknown 'fitness' function $f(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$.



Challenge:

- Complex fitness landscapes.
  - Local optimas, saddle points, etc.
  - Highly non-isotropic (ill-shaped) local behavior.



  - Correlation between all dimensions.
- Expensive fitness evaluations.
- High dimensionality, d up to hundreds.

Powerful methods are required to solve such problems.

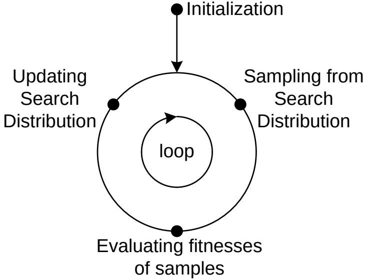
Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

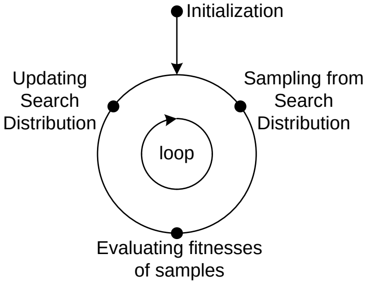
Typical flow of stochastic search algorithm:

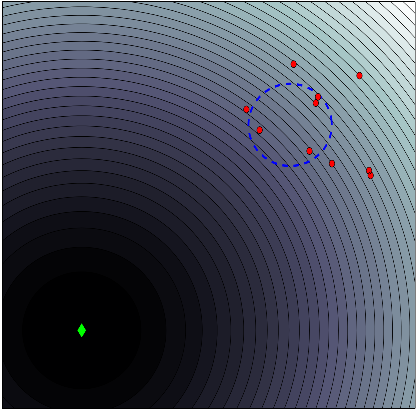


Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Typical flow of stochastic search algorithm:

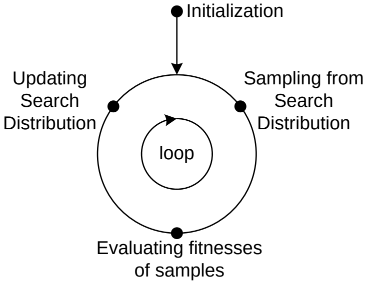


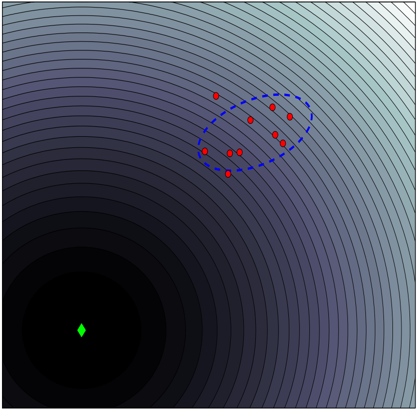


Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Typical flow of stochastic search algorithm:

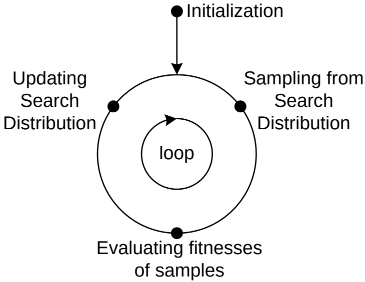


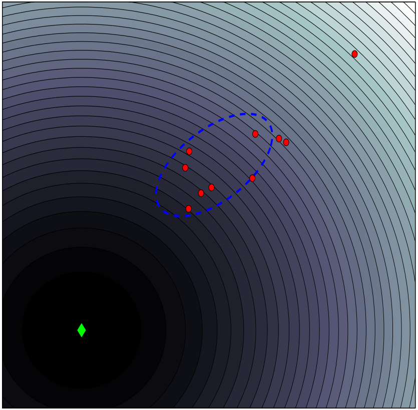


Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Typical flow of stochastic search algorithm:

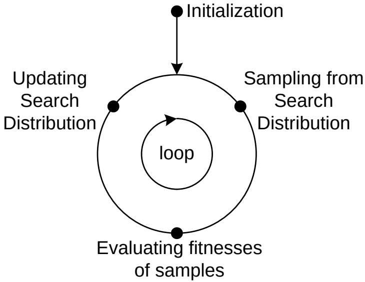


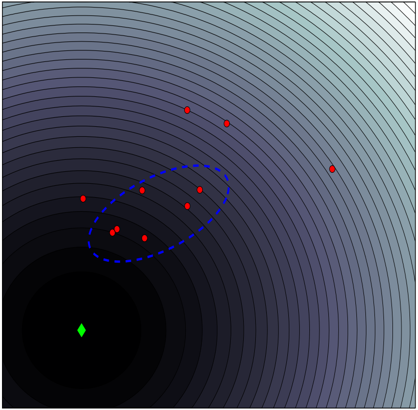


Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Typical flow of stochastic search algorithm:

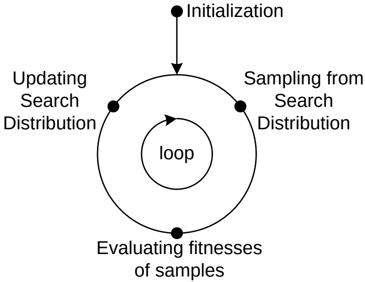


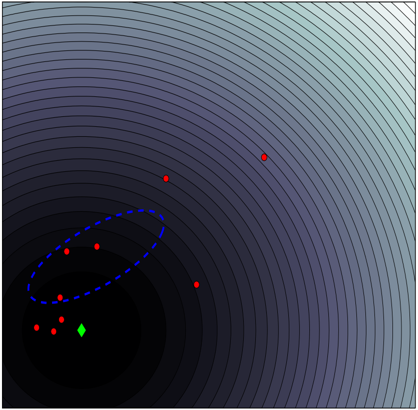


Stochastic Search Algorithms

Basic idea: Optimization by using population of samples.

Typical flow of stochastic search algorithm:





Stochastic Gradient Ascent

Let $p\left(\cdot|\theta\right)$ be the search distribution. We want to update θ towards better expected fitness:

$$J\left(\theta\right) = \mathbb{E}\left[f|\theta\right] = \int f\left(\mathbf{z}\right) p\left(\mathbf{z}|\theta\right) d\mathbf{z}.$$

Stochastic Gradient Ascent

Let $p\left(\cdot|\theta\right)$ be the search distribution. We want to update θ towards better expected fitness:

$$J\left(\theta\right) = \mathbb{E}\left[f|\theta\right] = \int f\left(\mathbf{z}\right) p\left(\mathbf{z}|\theta\right) d\mathbf{z}.$$

- The most straight forward way is by gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J\left(\theta\right).$$

Stochastic Gradient Ascent

Let $p(\cdot|\theta)$ be the search distribution. We want to update θ towards better expected fitness:

$$J(\theta) = \mathbb{E}[f|\theta] = \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z}.$$

- The most straight forward way is by gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta).$$

- We can compute the 'vanilla' gradient as

$$\begin{aligned}\nabla_\theta J(\theta) &= \int f(\mathbf{z})\, \nabla_\theta p(\mathbf{z}|\theta)\, d\mathbf{z} \\ &= \int f(\mathbf{z})\, \frac{p(\mathbf{z}|\theta)}{p(\mathbf{z}|\theta)} \nabla_\theta p(\mathbf{z}|\theta)\, d\mathbf{z} \quad \text{(log-likelihood trick)} \\ &= \mathbb{E}\left[f(\mathbf{z})\, \nabla_\theta \log p(\mathbf{z}|\theta)\, |\theta\right].\end{aligned}$$

Stochastic Gradient Ascent

- Using the Monte-Carlo estimation

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}\left[f(\mathbf{z})\,\nabla_\theta \log p(\mathbf{z}|\theta)\,|\theta\right] \\ &\simeq \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{z}_i)\,\nabla_\theta \log p(\mathbf{z}_i|\theta) = \frac{1}{n}\mathbf{G}\mathbf{f},\end{aligned}$$

with

$$\mathbf{G} = \left[\nabla_\theta \log p(\mathbf{z}_1|\theta) \ldots \nabla_\theta \log p(\mathbf{z}_n|\theta)\right],$$

$$\mathbf{f} = \left[f(\mathbf{z}_1) \ldots f(\mathbf{z}_n)\right]^\top.$$

Stochastic Gradient Ascent

- Using the Monte-Carlo estimation

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}\left[f(\mathbf{z})\,\nabla_\theta \log p(\mathbf{z}|\theta)\,|\theta\right] \\ &\simeq \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{z}_i)\,\nabla_\theta \log p(\mathbf{z}_i|\theta) = \frac{1}{n}\mathbf{G}\mathbf{f},\end{aligned}$$

with

$$\mathbf{G} = \left[\nabla_\theta \log p(\mathbf{z}_1|\theta) \ldots \nabla_\theta \log p(\mathbf{z}_n|\theta)\right],$$
$$\mathbf{f} = \left[f(\mathbf{z}_1) \ldots f(\mathbf{z}_n)\right]^\top.$$

Now the problem is to compute $\nabla_\theta \log p(\mathbf{z}|\theta)$. A closed form derivation can be obtained if $p(\mathbf{z}|\theta)$ is a Gaussian distribution.

The Gaussian Search Distribution

The Gaussian search distribution is given by

$$p\left(\mathbf{z}|\theta\right) = \mathcal{N}\left(\mathbf{z}|\mathbf{x}, \mathbf{C}\right).$$

The Gaussian Search Distribution

The Gaussian search distribution is given by

$$p(\mathbf{z}|\theta) = \mathcal{N}(\mathbf{z}|\mathbf{x}, \mathbf{C}).$$

- We use the parameter set $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, with $\mathbf{A}$ being the Cholesky decomposition of $\mathbf{C}$, i.e., $\mathbf{A}$ is an upper triangular matrix (UTM) and $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$.

The Gaussian Search Distribution

The Gaussian search distribution is given by

$$p\left(\mathbf{z}|\theta\right) = \mathcal{N}\left(\mathbf{z}|\mathbf{x}, \mathbf{C}\right).$$

- We use the parameter set $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, with $\mathbf{A}$ being the Cholesky decomposition of $\mathbf{C}$, i.e., $\mathbf{A}$ is an upper triangular matrix (UTM) and $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$.
    - No redundancy in θ since $\mathbf{C}$ is symmetric.

The Gaussian Search Distribution

The Gaussian search distribution is given by

$$p\left(\mathbf{z}|\theta\right) = \mathcal{N}\left(\mathbf{z}|\mathbf{x}, \mathbf{C}\right).$$

- We use the parameter set $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, with $\mathbf{A}$ being the Cholesky decomposition of $\mathbf{C}$, i.e., $\mathbf{A}$ is an upper triangular matrix (UTM) and $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$.
    - No redundancy in θ since $\mathbf{C}$ is symmetric.
- $\nabla_\theta \log p\left(\mathbf{z}|\theta\right)$ can be computed in closed form:

$$\nabla_{\mathbf{x}} \log p\left(\mathbf{z}|\theta\right) = \mathbf{C}^{-}\left(\mathbf{z}-\mathbf{x}\right)$$
$$\nabla_{\mathbf{A}} \log p\left(\mathbf{z}|\theta\right) = \mathbf{A}^{-\top}\left(\mathbf{z}-\mathbf{x}\right)\left(\mathbf{z}-\mathbf{x}\right)^\top \mathbf{C}^{-} - \mathrm{diag}\left(\mathbf{A}^{-}\right)$$

The Gaussian Search Distribution

The Gaussian search distribution is given by

$$p\left(\mathbf{z}|\theta\right) = \mathcal{N}\left(\mathbf{z}|\mathbf{x}, \mathbf{C}\right).$$

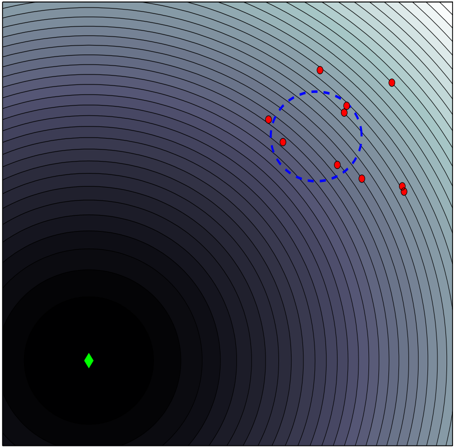
- We use the parameter set $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, with $\mathbf{A}$ being the Cholesky decomposition of $\mathbf{C}$, i.e., $\mathbf{A}$ is an upper triangular matrix (UTM) and $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$.
  - No redundancy in θ since $\mathbf{C}$ is symmetric.
- $\nabla_\theta \log p\left(\mathbf{z}|\theta\right)$ can be computed in closed form:

$$\begin{aligned}\nabla_{\mathbf{x}} \log p\left(\mathbf{z}|\theta\right) &= \mathbf{C}^{-}\left(\mathbf{z}-\mathbf{x}\right)\\ \nabla_{\mathbf{A}} \log p\left(\mathbf{z}|\theta\right) &= \mathbf{A}^{-\top}\left(\mathbf{z}-\mathbf{x}\right)\left(\mathbf{z}-\mathbf{x}\right)^\top \mathbf{C}^{-} - \mathrm{diag}\left(\mathbf{A}^{-}\right)\end{aligned}$$

- $\nabla_\theta^s J\left(\theta\right)$ can be computed from $\nabla_\theta \log p\left(\mathbf{z}_1|\theta\right) \ldots \nabla_\theta \log p\left(\mathbf{z}_1|\theta\right)$.

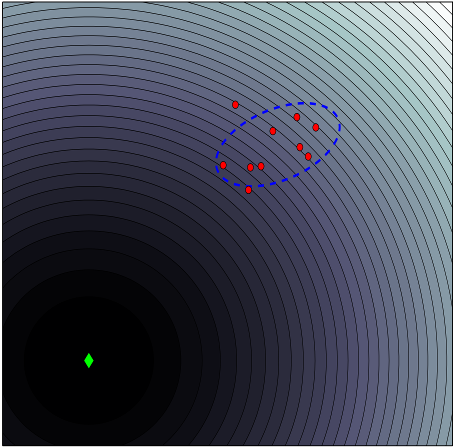
Stochastic Gradient Ascent

$$\theta \leftarrow \theta + \alpha \nabla_{\theta}^{s} J(\theta) = \theta + \frac{\alpha}{n}\mathbf{Gf}$$



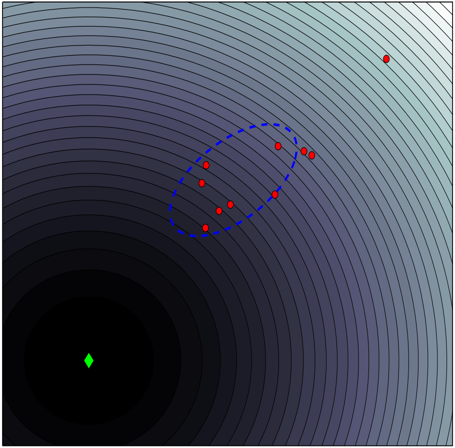
Stochastic Gradient Ascent

$$\theta \leftarrow \theta + \alpha \nabla_{\theta}^{s} J(\theta) = \theta + \frac{\alpha}{n} \mathbf{Gf}$$



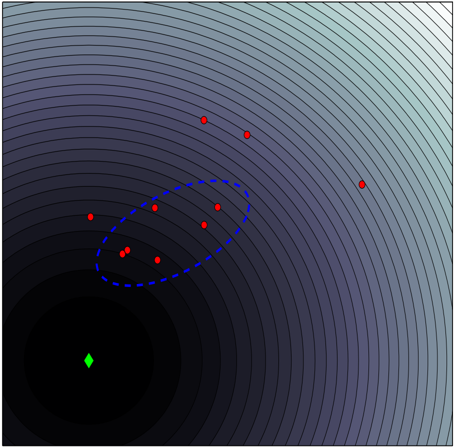
Stochastic Gradient Ascent

$$\theta \leftarrow \theta + \alpha \nabla_\theta^s J(\theta) = \theta + \frac{\alpha}{n} \mathbf{Gf}$$



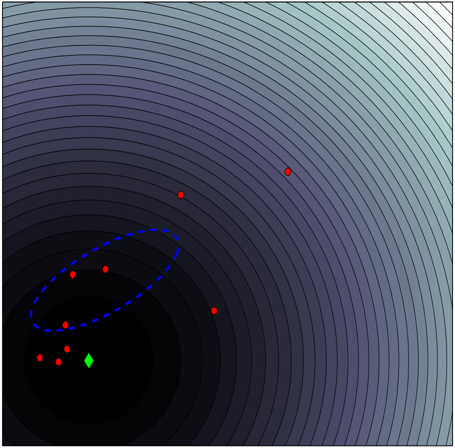
Stochastic Gradient Ascent

$$\theta \leftarrow \theta + \alpha \nabla_{\theta}^{s} J(\theta) = \theta + \frac{\alpha}{n}\mathbf{Gf}$$



Stochastic Gradient Ascent

$$\theta \leftarrow \theta + \alpha \nabla^{s}_{\theta} J\left(\theta\right) = \theta + \frac{\alpha}{n}\mathbf{Gf}$$



Novel Ideas in eNES

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.

2. The natural gradient is computed in an *Exact* and *Efficient* way.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.
4. Introducing *Optimal Fitness Baseline* to reduce the variance of gradient estimation.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.
4. Introducing *Optimal Fitness Baseline* to reduce the variance of gradient estimation.

Why Natural Gradient?

Why Natural Gradient?

- Vanilla gradient doesn't work:

Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.

Why Natural Gradient?

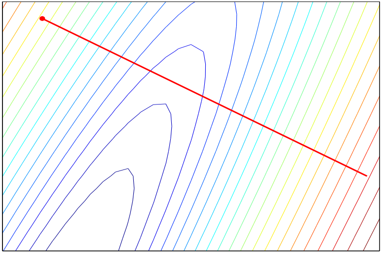
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.

Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

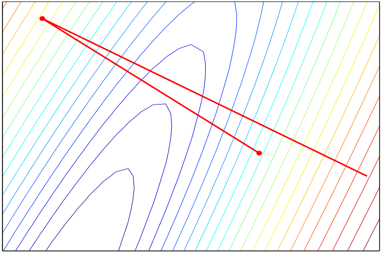
Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



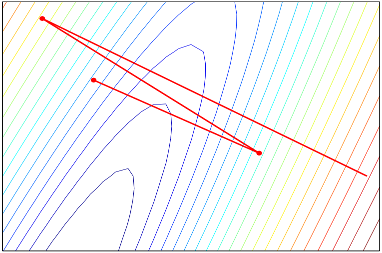
Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



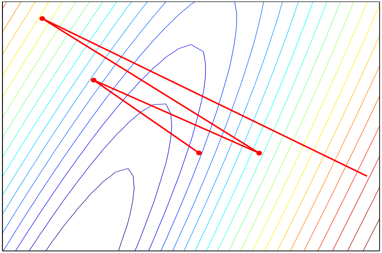
Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



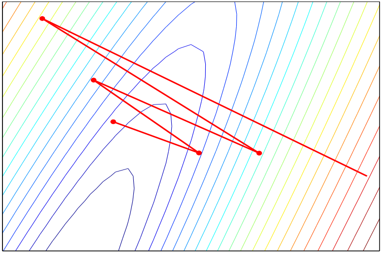
Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



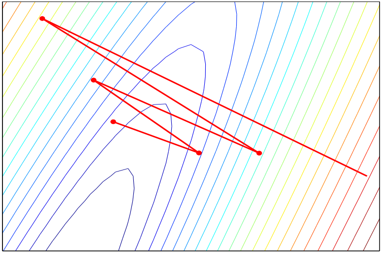
Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



Why Natural Gradient?

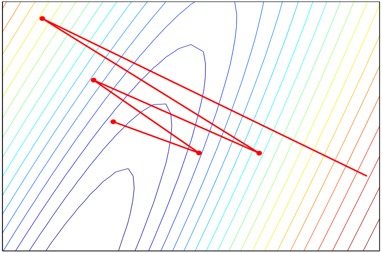
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



- Basic idea of natural gradient

Why Natural Gradient?

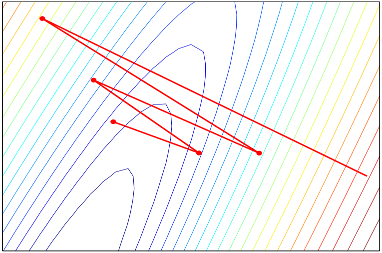
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.

Why Natural Gradient?

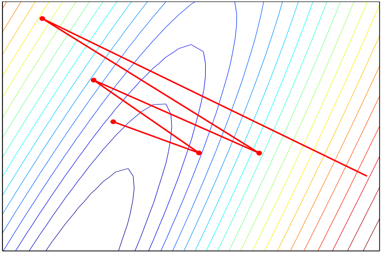
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.

Why Natural Gradient?

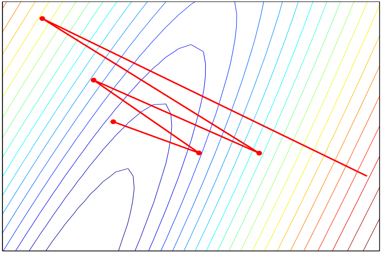
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



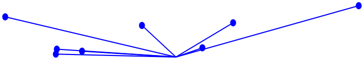
- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

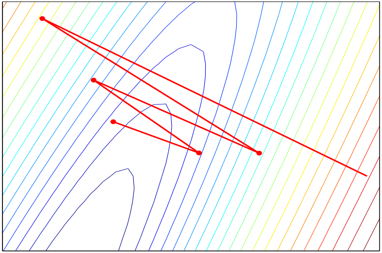


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

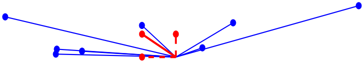


Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

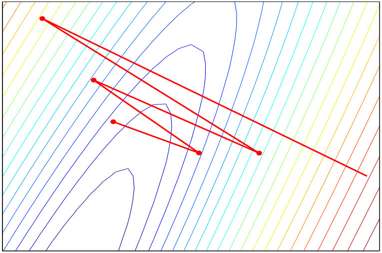


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

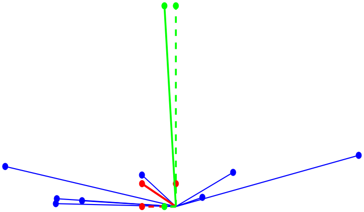


Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

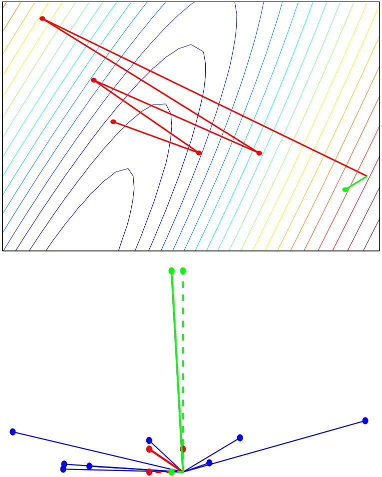


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.



Why Natural Gradient?

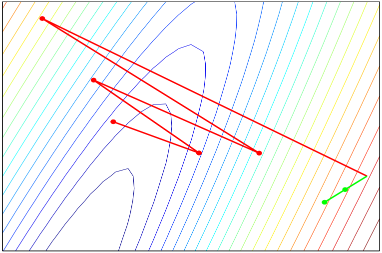
- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



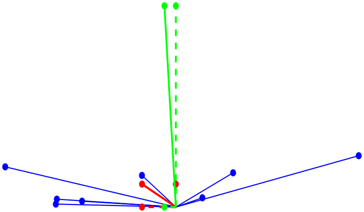
- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

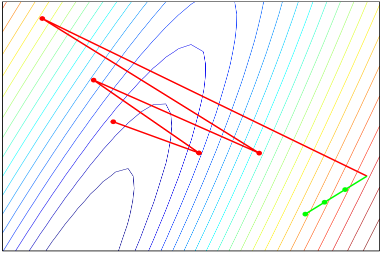


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

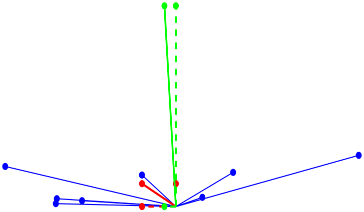


Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

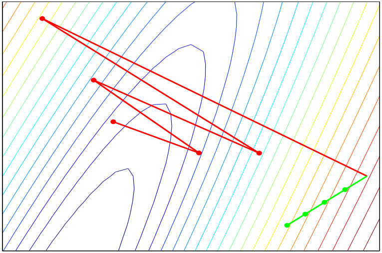


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

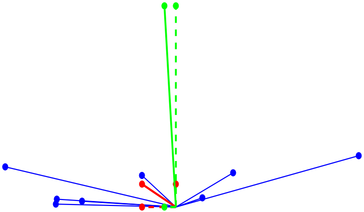


Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.

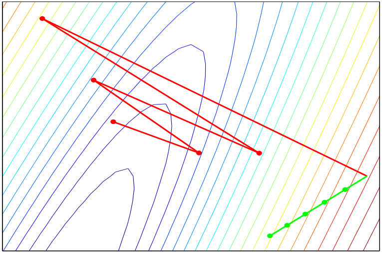


- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.

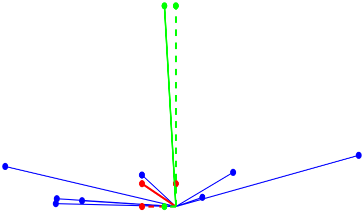


Why Natural Gradient?

- Vanilla gradient doesn't work:
  - Over-aggressive steps on ridges.
  - Too small steps on plateaus.
  - Slow or premature convergence, non-robust performance.



- Basic idea of natural gradient
  - Steepest ascent direction when considering correlations between elements in θ.
  - Re-weight gradient elements according to their uncertainties, resp.
  - Isotropic convergence on ill-shaped surface.



Formulation of Natural Gradient

Assume the distance between two adjacent distributions $p(\cdot|\theta)$ and $p(\cdot|\theta + \delta\theta)$ is defined by their KL divergence. The natural gradient $\tilde{\nabla}_\theta J(\theta)$ is given by the necessary condition

$$\mathbf{F}\tilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta).$$

Formulation of Natural Gradient

Assume the distance between two adjacent distributions $p(\cdot|\theta)$ and $p(\cdot|\theta + \delta\theta)$ is defined by their KL divergence. The natural gradient $\tilde{\nabla}_\theta J(\theta)$ is given by the necessary condition

$$\mathbf{F}\tilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta).$$

- $\mathbf{F}$ is the Fisher information matrix (FIM) of θ: (Intuitively, the normalized covariance of the gradient.)

$$\mathbf{F} = \mathbb{E}\left[\left(\nabla_\theta \log p(\mathbf{z}|\theta)\right)\left(\nabla_\theta \log p(\mathbf{z}|\theta)\right)^\top\right].$$

Formulation of Natural Gradient

Assume the distance between two adjacent distributions $p(\cdot|\theta)$ and $p(\cdot|\theta+\delta\theta)$ is defined by their KL divergence. The natural gradient $\tilde{\nabla}_\theta J(\theta)$ is given by the necessary condition

$$\mathbf{F}\tilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta).$$

- $\mathbf{F}$ is the Fisher information matrix (FIM) of θ: (Intuitively, the normalized covariance of the gradient.)

$$\mathbf{F} = \mathbb{E}\left[\left(\nabla_\theta \log p(\mathbf{z}|\theta)\right)\left(\nabla_\theta \log p(\mathbf{z}|\theta)\right)^\top\right].$$

- $\mathbf{F}$ may not be invertible.

Formulation of Natural Gradient

Assume the distance between two adjacent distributions $p\left(\cdot|\theta\right)$ and $p\left(\cdot|\theta+\delta\theta\right)$ is defined by their KL divergence. The natural gradient $\tilde{\nabla}_\theta J\left(\theta\right)$ is given by the necessary condition

$$\mathbf{F}\tilde{\nabla}_\theta J\left(\theta\right) = \nabla_\theta J\left(\theta\right).$$

- $\mathbf{F}$ is the Fisher information matrix (FIM) of θ: (Intuitively, the normalized covariance of the gradient.)

$$\mathbf{F} = \mathbb{E}\left[\left(\nabla_\theta \log p\left(\mathbf{z}|\theta\right)\right)\left(\nabla_\theta \log p\left(\mathbf{z}|\theta\right)\right)^\top\right].$$

- $\mathbf{F}$ may not be invertible.
- If $\mathbf{F}$ is invertable, we can compute the (estimated) natural gradient as

$$\tilde{\nabla}_\theta J\left(\theta\right) = \mathbf{F}^{-}\nabla_\theta J\left(\theta\right),\ \tilde{\nabla}^s_\theta J\left(\theta\right) = \mathbf{F}^{-}\nabla^s_\theta J\left(\theta\right).$$

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.
4. Introducing *Optimal Fitness Baseline* to reduce the variance of gradient estimation.

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

- $\mathbf{F}$ is indeed *invertible*.

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

- $\mathbf{F}$ is indeed *invertible*.
- $\mathbf{F}$ is a *block diagonal matrix*

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}^{-} & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{bmatrix}.$$

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

- $\mathbf{F}$ is indeed *invertible*.
- $\mathbf{F}$ is a *block diagonal matrix*

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}^{-} & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{bmatrix}.$$

  - $\mathbf{C}^{-}$ is the FIM for $\mathbf{x}$.

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

- $\mathbf{F}$ is indeed *invertible*.
- $\mathbf{F}$ is a *block diagonal matrix*

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}^- & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{bmatrix}.$$

  - $\mathbf{C}^-$ is the FIM for $\mathbf{x}$.
  - $\mathbf{F}_k$ is the FIM for $(n - k + 1$ non-zero elements in$)$ the k-th row of $\mathbf{A}$.

Property of FIM in the Gaussian Case

Let $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$. Our lucky findings:

- $\mathbf{F}$ is indeed *invertible*.
- $\mathbf{F}$ is a *block diagonal matrix*

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}^{-} & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{bmatrix}.$$

  - $\mathbf{C}^{-}$ is the FIM for $\mathbf{x}$.
  - $\mathbf{F}_k$ is the FIM for ($n - k + 1$ non-zero elements in) the k-th row of $\mathbf{A}$.
  - The FIM suggest a natural grouping of elements in θ. Groups are orthogonal with each other.

Property of FIM in the Gaussian Case

- $\mathbf{F}_k$ has the special form

$$\mathbf{F}_k = \begin{bmatrix} a_{k,k}^{-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \mathbf{D}_k,$$

with $\mathbf{D}_k$ being the $n - k + 1$ submatrix at the lower right corner of $\mathbf{C}^{-}$.

Property of FIM in the Gaussian Case

- $\mathbf{F}_k$ has the special form

$$\mathbf{F}_k = \begin{bmatrix} a_{k,k}^{-2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \mathbf{D}_k,$$

  with $\mathbf{D}_k$ being the $n - k + 1$ submatrix at the lower right corner of $\mathbf{C}^{-}$.
    - This special form permits a iterative algorithm to compute $\mathbf{F}_k^{-}$ from $\mathbf{F}_{k+1}^{-}$ with complexity $O\left(k^2\right)$.

Efficient Inverse of FIM

The computation of natural gradient requires the inverse of $\mathbf{F}$.

Efficient Inverse of FIM

The computation of natural gradient requires the inverse of $\mathbf{F}$.

- Naively, $\mathbf{F}$ is a matrix of size $O\left(d^2\right)$, so computing $\mathbf{F}^-$ requires $O\left(d^6\right)$.

Efficient Inverse of FIM

The computation of natural gradient requires the inverse of $\mathbf{F}$.

- Naively, $\mathbf{F}$ is a matrix of size $O\left(d^2\right)$, so computing $\mathbf{F}^-$ requires $O\left(d^6\right)$.
- We already find that $\mathbf{F}$ is block diagonal, so computing $\mathbf{F}^-$ requires $O\left(d^4\right)$.

Efficient Inverse of FIM

The computation of natural gradient requires the inverse of $\mathbf{F}$.

- Naively, $\mathbf{F}$ is a matrix of size $O\left(d^2\right)$, so computing $\mathbf{F}^-$ requires $O\left(d^6\right)$.
- We already find that $\mathbf{F}$ is block diagonal, so computing $\mathbf{F}^-$ requires $O\left(d^4\right)$.
- We can do better! Use the special form of each sub-block, the complexity is reduced to $O\left(d^3\right)$.

Efficient Inverse of FIM

The computation of natural gradient requires the inverse of $\mathbf{F}$.

- Naively, $\mathbf{F}$ is a matrix of size $O\left(d^2\right)$, so computing $\mathbf{F}^-$ requires $O\left(d^6\right)$.
- We already find that $\mathbf{F}$ is block diagonal, so computing $\mathbf{F}^-$ requires $O\left(d^4\right)$.
- We can do better! Use the special form of each sub-block, the complexity is reduced to $O\left(d^3\right)$.
- The estimated natural gradient is then computed as

$$\nabla_\theta^s J\left(\theta\right) = \frac{1}{n}\mathbf{F}^-\mathbf{Gf}.$$

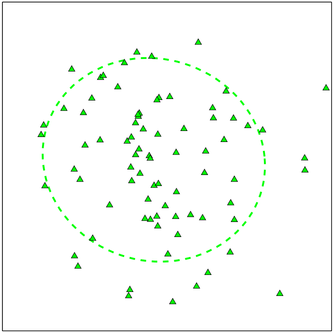
with complexity $O\left(d^3\right)$.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.
4. Introducing *Optimal Fitness Baseline* to reduce the variance of gradient estimation.

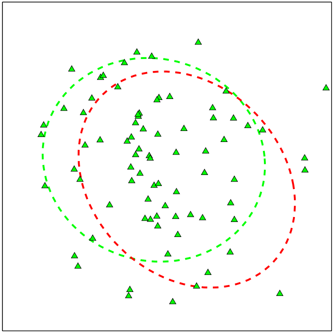
Importance Mixing

- At each cycle, we need to evaluate n new samples.



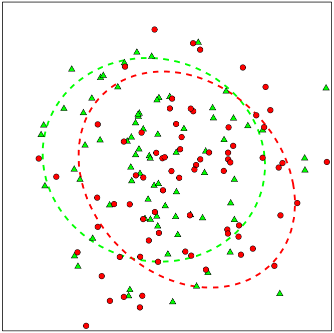
Importance Mixing

- At each cycle, we need to evaluate *n* new samples.
- It is common that the updated $\theta^{(t)}$ is close to $\theta^{(t-1)}$.



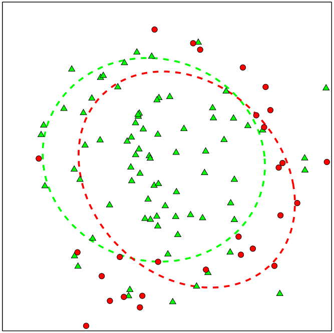
Importance Mixing

- At each cycle, we need to evaluate *n* new samples.
- It is common that the updated $\theta^{(t)}$ is close to $\theta^{(t-1)}$.
- Problem: Redundant fitness evaluations in overlapping high density area.



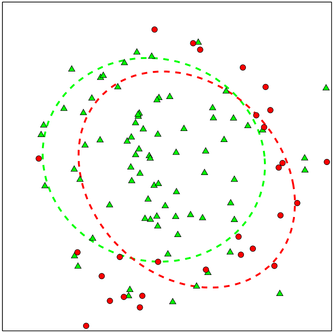
Importance Mixing

- At each cycle, we need to evaluate n new samples.
- It is common that the updated $\theta^{(t)}$ is close to $\theta^{(t-1)}$.
- Problem: Redundant fitness evaluations in overlapping high density area.
- Importance Mixing: Generate samples in less explored areas, while keeping the updated batch conformed to the new search distribution.



Importance Mixing

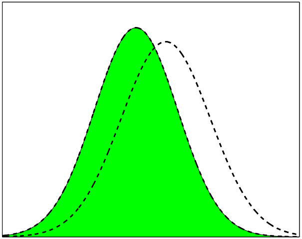
- At each cycle, we need to evaluate *n* new samples.
- It is common that the updated $\theta^{(t)}$ is close to $\theta^{(t-1)}$.
- Problem: Redundant fitness evaluations in overlapping high density area.
- Importance Mixing: Generate samples in less explored areas, while keeping the updated batch conformed to the new search distribution.



Reusing samples: fewer fitness evaluations.

Importance Mixing

Formally, importance mixing is carried out by two rejection samplings.

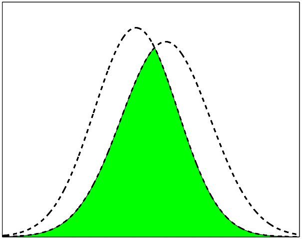


Importance Mixing

Formally, importance mixing is carried out by two rejection samplings.

- Forward pass: For each sample **z** from the previous batch, accept with probability

$$\min\left\{1, \frac{p\left(z|\theta^{(t)}\right)}{p\left(z|\theta^{(t-1)}\right)}\right\}.$$



Importance Mixing

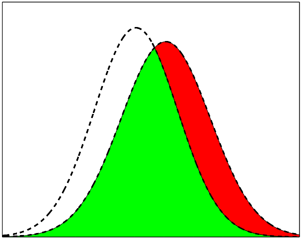
Formally, importance mixing is carried out by two rejection samplings.

- Forward pass: For each sample **z** from the previous batch, accept with probability

$$\min\left\{1, \frac{p\left(z|\theta^{(t)}\right)}{p\left(z|\theta^{(t-1)}\right)}\right\}.$$

- Backward pass: Accept newly generated sample **z** with probability

$$\max\left\{0, 1 - \frac{p\left(z|\theta^{(t-1)}\right)}{p\left(z|\theta^{(t)}\right)}\right\}$$



until batch size reached.

Novel Ideas in eNES

1. Use the *Natural Gradient* instead of the vanilla gradient.
2. The natural gradient is computed in an *Exact* and *Efficient* way.
3. Use *Importance Mixing* for reusing previously evaluated samples.
4. Introducing *Optimal Fitness Baseline* to reduce the variance of gradient estimation.

Optimal Fitness Baseline

A typical problem with the Monte-Carlo gradient estimation is that the variance is too big. The *fitness baseline* is introduced to reduce the variance.

$$\begin{aligned}\nabla_\theta J &= \nabla_\theta \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z} - \underbrace{\nabla_\theta \int b p(\mathbf{z}|\theta)\, d\mathbf{z}}_{=0} \\ &= \nabla_\theta \int [f(\mathbf{z}) - b]\, p(\mathbf{z}|\theta)\, d\mathbf{z},\end{aligned}$$

b is called the fitness baseline.

Optimal Fitness Baseline

A typical problem with the Monte-Carlo gradient estimation is that the variance is too big. The *fitness baseline* is introduced to reduce the variance.

$$\nabla_\theta J = \nabla_\theta \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z} - \underbrace{\nabla_\theta \int b p(\mathbf{z}|\theta)\, d\mathbf{z}}_{=0}$$

$$= \nabla_\theta \int [f(\mathbf{z}) - b]\, p(\mathbf{z}|\theta)\, d\mathbf{z},$$

b is called the fitness baseline.

- Adding the baseline b won't affect the expectation of $\nabla_\theta J$.

Optimal Fitness Baseline

A typical problem with the Monte-Carlo gradient estimation is that the variance is too big. The *fitness baseline* is introduced to reduce the variance.

$$\begin{aligned}\nabla_\theta J &= \nabla_\theta \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z} - \underbrace{\nabla_\theta \int b p(\mathbf{z}|\theta)\, d\mathbf{z}}_{=0} \\ &= \nabla_\theta \int [f(\mathbf{z}) - b]\, p(\mathbf{z}|\theta)\, d\mathbf{z},\end{aligned}$$

b is called the fitness baseline.

- Adding the baseline b won't affect the expectation of $\nabla_\theta J$.
- But it affects the *variance* of the estimation: For natural gradient

$$\mathbb{V}\left[\tilde{\nabla}_\theta J(\theta)\right] \propto b^2 \mathbb{E}\left[\mathbf{u}^\top \mathbf{u}\right] - 2b\mathbb{E}\left[\mathbf{u}^\top \mathbf{v}\right] + const$$

with

$$\mathbf{u} = \mathbf{F}^- \nabla_\theta \log p(\mathbf{z}|\theta)\,,\ \mathbf{v} = f(\mathbf{z})\, \mathbf{u}.$$

Optimal Fitness Baseline

- $\mathbb{V}\left[\tilde{\nabla}_\theta J\left(\theta\right)\right]$ is of quadratic form, we can minimize it. The *optimal fitness baseline* is given by

$$b^* = \frac{\mathbb{E}\left[\mathbf{u}^\top \mathbf{v}\right]}{\mathbb{E}\left[\mathbf{u}^\top \mathbf{u}\right]} \simeq \frac{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{v}_i}{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{u}_i}.$$

Optimal Fitness Baseline

- $\mathbb{V}\left[\tilde{\nabla}_\theta J\left(\theta\right)\right]$ is of quadratic form, we can minimize it. The *optimal fitness baseline* is given by

$$b^* = \frac{\mathbb{E}\left[\mathbf{u}^\top \mathbf{v}\right]}{\mathbb{E}\left[\mathbf{u}^\top \mathbf{u}\right]} \simeq \frac{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{v}_i}{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{u}_i}.$$

- The natural gradient is then estimated by

$$\tilde{\nabla}_\theta^s J\left(\theta\right) = \frac{1}{n}\mathbf{F}^- \mathbf{G}\left(\mathbf{f} - b^*\right).$$

Optimal Fitness Baseline

- $\mathbb{V}\left[\tilde{\nabla}_\theta J(\theta)\right]$ is of quadratic form, we can minimize it. The *optimal fitness baseline* is given by

$$b^* = \frac{\mathbb{E}\left[\mathbf{u}^\top \mathbf{v}\right]}{\mathbb{E}\left[\mathbf{u}^\top \mathbf{u}\right]} \simeq \frac{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{v}_i}{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{u}_i}.$$

- The natural gradient is then estimated by

$$\tilde{\nabla}_\theta^s J(\theta) = \frac{1}{n} \mathbf{F}^- \mathbf{G}\left(\mathbf{f} - b^*\right).$$

- Better: Different baselines b_j for different (groups of) parameter θ_j, further reducing the variance.

Optimal Fitness Baseline

- $\mathbb{V}\left[\tilde{\nabla}_\theta J(\theta)\right]$ is of quadratic form, we can minimize it. The *optimal fitness baseline* is given by

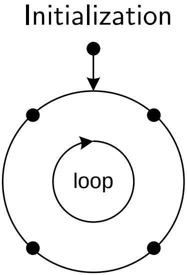
$$b^* = \frac{\mathbb{E}\left[\mathbf{u}^\top \mathbf{v}\right]}{\mathbb{E}\left[\mathbf{u}^\top \mathbf{u}\right]} \simeq \frac{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{v}_i}{\sum_{i=1}^n \mathbf{u}_i^\top \mathbf{u}_i}.$$

- The natural gradient is then estimated by

$$\tilde{\nabla}_\theta^s J(\theta) = \frac{1}{n}\mathbf{F}^- \mathbf{G}\left(\mathbf{f} - b^*\right).$$

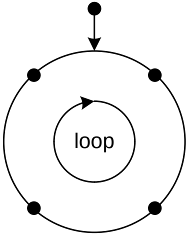
- Better: Different baselines b_j for different (groups of) parameter θ_j, further reducing the variance.
    - The block diagonal structure of **F** suggests using a *block fitness baseline*, where different baseline values are computed for orthogonal groups of parameters in θ.

Putting Things Together



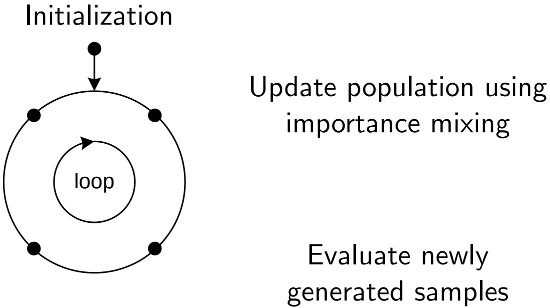
Putting Things Together

Initialization



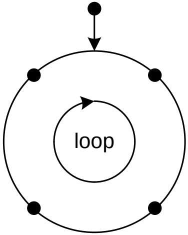
Update population using importance mixing

Putting Things Together



Putting Things Together

Initialization



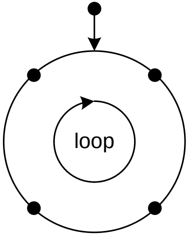
Update population using importance mixing

Evaluate newly generated samples

Compute optimal baseline $\mathbf{b}^*$ and $\tilde{\nabla}_\theta^s J(\theta)$

Putting Things Together

Initialization



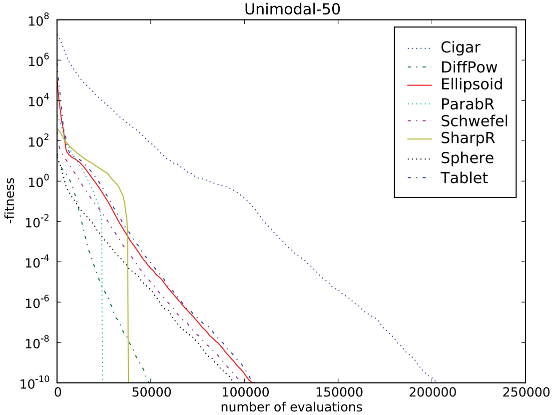
Update:
$\theta \leftarrow \theta + \alpha \tilde{\nabla}_{\theta}^{s} J(\theta)$

Update population using importance mixing

Compute optimal baseline $\mathbf{b}^*$ and $\tilde{\nabla}_{\theta}^{s} J(\theta)$

Evaluate newly generated samples

Empirical Results - Standard Blackbox Benchmarks



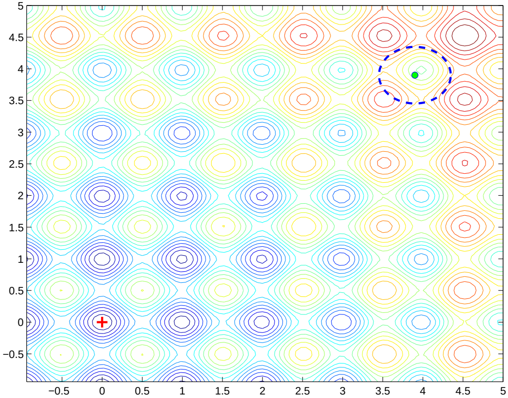
Empirical Results - Importance Mixing and Optimal Baseline

Percentage of runs that prematurely converged, while varying the type of fitness baseline used.

Baseline	premature convergence
None	52%
Uniform	50%
Block	0%

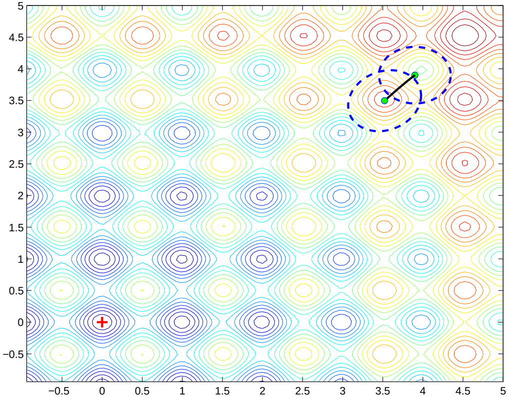
Importance Mixing reduces the number of fitness evaluations by a factor of $3 \sim 4$.

Empirical Results - Multimodal



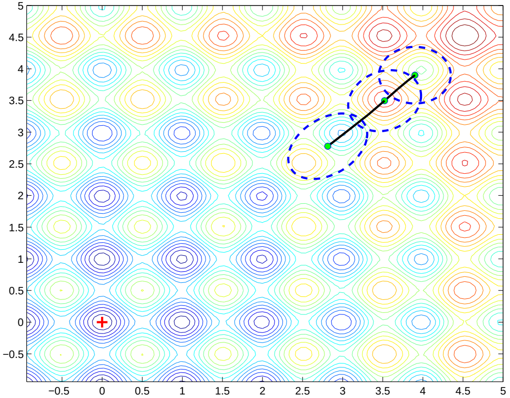
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



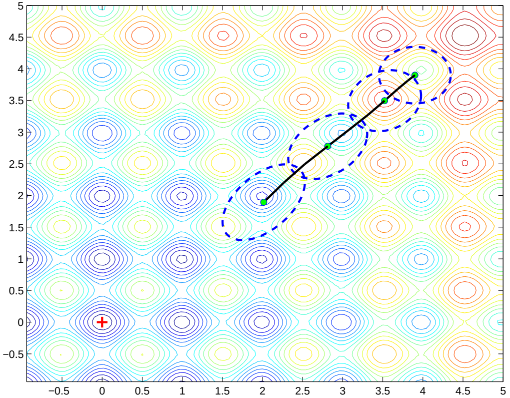
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



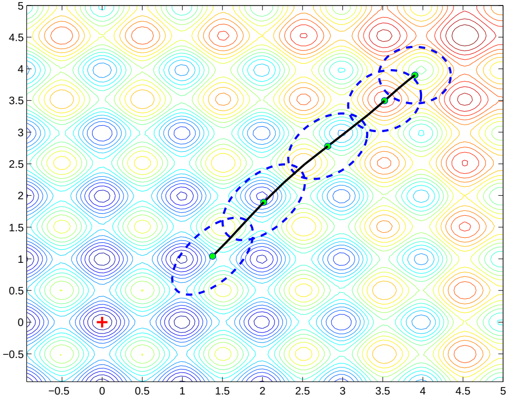
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



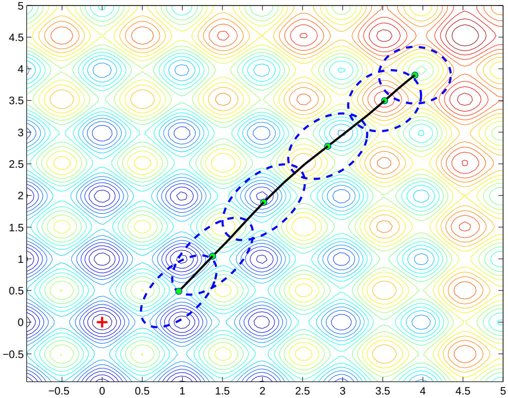
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



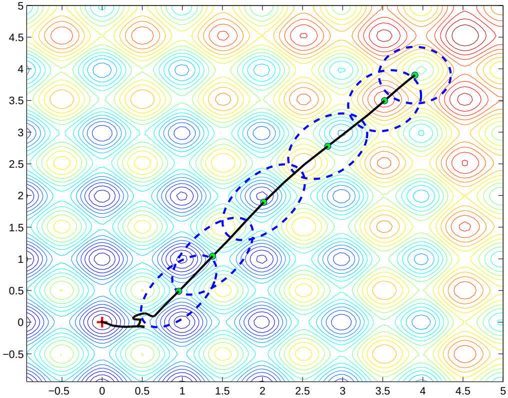
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



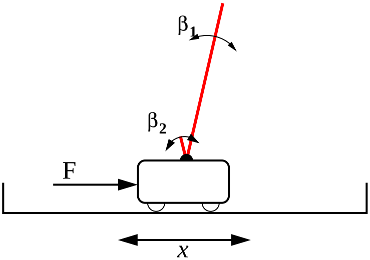
eNES is able to jump over deceptive local optima.

Empirical Results - Multimodal



eNES is able to jump over deceptive local optima.

Empirical Results - Double Pole Balancing



Non-Markovian double pole balancing, average numbers of evaluations.

Method	SANE	ESP	NEAT	CMA	CoSyNE	FEM	NES
Eval.	262, 700	7, 374	6, 929	3, 521	1, 249	2, 099	1, 753

Summary

Summary

- We derived a clear blackbox optimization algorithm from first principles.

Summary

- We derived a clear blackbox optimization algorithm from first principles.
- Derivation of exact Fisher information matrix.

Summary

- We derived a clear blackbox optimization algorithm from first principles.
- Derivation of exact Fisher information matrix.
- Efficient computation of the FIM inverse.

Summary

- We derived a clear blackbox optimization algorithm from first principles.
- Derivation of exact Fisher information matrix.
- Efficient computation of the FIM inverse.
- Importance mixing reduces the number of fitness evaluations.

Summary

- We derived a clear blackbox optimization algorithm from first principles.
- Derivation of exact Fisher information matrix.
- Efficient computation of the FIM inverse.
- Importance mixing reduces the number of fitness evaluations.
- Optimal fitness baselines reduces the variance of gradient estimation.

Summary

- We derived a clear blackbox optimization algorithm from first principles.
- Derivation of exact Fisher information matrix.
- Efficient computation of the FIM inverse.
- Importance mixing reduces the number of fitness evaluations.
- Optimal fitness baselines reduces the variance of gradient estimation.
- Competitive performance on standard benchmarks, including non-Markovian double pole balancing tasks.

Try it Out?

PyBrain

The Python Machine Learning Library

http://www.pybrain.org

Thank you!