Optimization of Energy Policies Using Direct Value Search

Jérémie Decock, Jean-Joseph Christophe, Olivier Teytaud

Optimization of Energy Policies

Using Direct Value Search

Jeremie Decock

Jean-Joseph Christophe

Olivier Teytaud

Inria, Artelys

May 12, 2014

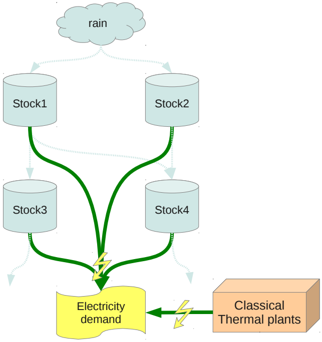
Introduction

- Optimization of Energy Policies
- with *Direct Value Search* (DVS)
  - Linear Programming
  - Direct Policy Search

Overview

Power systems problems we try to solve. . .

Unit commitment



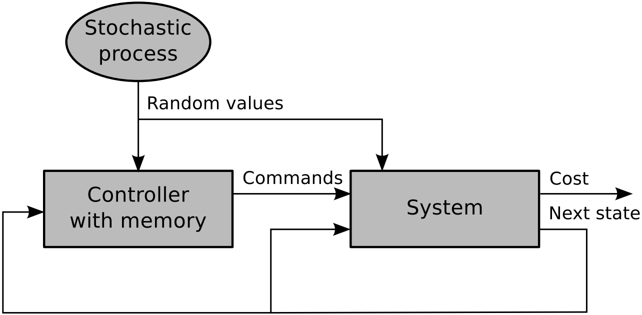
A short reminder

- A multi-stage problem
- Energy demand (forecast)
- Energy production:
  - Hydroelectricity (N water *stocks*)
  - Thermal plants
- Water flow through stock links

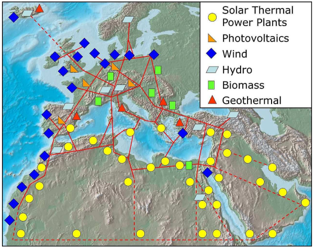
Problem

Which production unit should be used at time t to satisfy the demand with the lowest possible cost ?

Stochastic Control



POST (ADEME)



High scale investment studies (e.g. Europe + North Africa)

- Long term (2030 - 2050)
- Huge (non-stochastic) uncertainties
  - Future technologies
  - Future laws
  - ...
- Investment problem
  - Interconnections
  - Storage
  - Smart grids
  - Power plants
  - ...

Issues and methods

Issues

- Limited forecast (e.g. demand, weather, ...)
- Renewable energies increase production variability
- Transportation introduces constraints

Methods

- Can't assume Markovian process
  - Weather (influences production and demand)
  - ...
- Avoid simplified models to avoid model errors
  - Convex value function
  - Linear transition function
  - ...

Most classical solutions

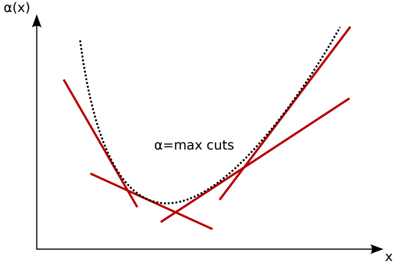
Stochastic Dual Dynamic Programming

Decompose the problem in instant cost + future cost as:

$$\min_x \underbrace{cost(x)}_{\text{instant cost}} + \underbrace{\alpha(x)}_{\text{Bellman Value}}$$

Approximate $\alpha(.)$ with Bender cuts.

Problems: It needs convexity of $\alpha(.)$, a markovian process and not too many state variables.



Direct Value Search

Direct Policy Search

Goal

Finds “good” parameters for a given parametric policy $\pi_{\boldsymbol{\theta}}$: *states* $\rightarrow$ *controls*.

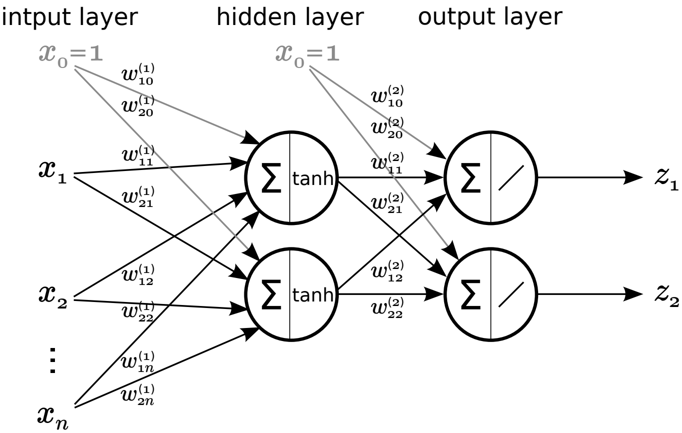
Requires a parametric controller (e.g. neural network)

Principle

Optimize the parameters on simulations (Noisy Black-Box Optimization).

Parametric policies π_{θ}

Neural Networks: $\boldsymbol{\theta} = \left(w_{10}^{(1)}, \ldots, w_{mn}^{(1)}, w_{10}^{(2)}, \ldots, w_{km}^{(2)}\right)^T$



Summary

	Pros	Cons
S(D)DP	large constrained $\mathcal{U}$	not anytime
	polynomial time decision making	convex problems only (SDDP)
	asymptotically find the optimum	small $\mathcal{S}$
		markovian random process
DPS	anytime	slow on large $\mathcal{U}$
	large $\mathcal{S}$	hardly handles decision constraints
	works with non linear functions	
	no random process constraint	

Direct Value Search

Merge both approaches

Direct Value Search

An overview

Like Bellman decomposition: present cost and futur state valorization

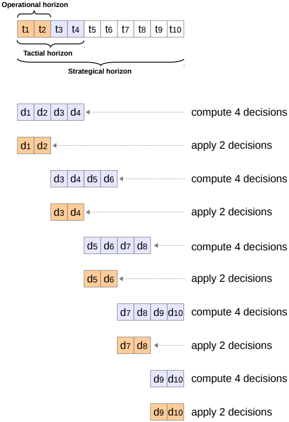
$$\Pi(\mathbf{s}_t) = \arg\min_{\mathbf{u}_t} \mathrm{cost}(\mathbf{u}_t) + V(\mathbf{s}_{t+1})$$
$$V(\mathbf{s}_{t+1}) = \underbrace{\boldsymbol{\alpha}_t.\mathbf{s}_{t+1}}_{\text{LP}}$$
$$\boldsymbol{\alpha}_t = \underbrace{\pi_{\boldsymbol{\theta}}(\mathbf{s}_t)}_{\text{not LP}}$$

- Given $\boldsymbol{\theta}$, decision making solved as a LP
- Non-linear mapping for choosing the parameters of the LP from the current state

Requires the optimization of $\boldsymbol{\theta}$ (noisy black-box optimization problem)

Direct Value Search

Recourse planning



- Decisions $\mathbf{u}_{1\ldots k} := (\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_k)$ are optimized (tactical horizon)
- Only $\mathbf{u}_{1\ldots h} := (\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_h)$ are applied (operational horizon)
- The system is now in a new state at stage h
- Decisions $\mathbf{u}_{h\ldots h+k}$ are optimized (tactical horizon)
- Only $\mathbf{u}_{h\ldots 2h}$ are applied (operational horizon)
- ...

Direct Value Search

Assumptions

We assume we know $\boldsymbol{\iota}_{t\ldots k}$ the random realizations from current stage to tactical horizon.

Direct Value Search

Step 1 (offline): compute $\pi_{\boldsymbol{\theta}}(.)$

Build parametric policy $\pi_{\boldsymbol{\theta}}$

Require:
a parametric policy $\pi_{\boldsymbol{\theta}}(.)$ where $\pi_{\boldsymbol{\theta}}$ is a mapping from $\mathcal{S}$ to $\mathcal{U}$,
a Stochastic Decision Process SDP,
an initial state **s**

Ensure:
a parameter $\hat{\boldsymbol{\theta}}$ leading to a policy $\pi_{\hat{\boldsymbol{\theta}}}(.)$

Find a parameter $\hat{\theta}$ minimizing the expectation of the following fitness function
$\boldsymbol{\theta} \mapsto \text{Simulate}(\mathbf{s}, \text{SDP}, \pi_{\boldsymbol{\theta}})$
with a given non-linear noisy optimization algorithm (e.g. SA-ES, CMA-ES, ...)

return $\hat{\boldsymbol{\theta}}$

Direct Value Search

Step 1 (offline): compute $\pi_{\theta}(.)$

Simulate(s_0, SDP, π_{θ})

$c \leftarrow 0$
for $t \leftarrow t_0,\ t_0 + h,\ t_0 + 2h,\ \ldots,\ T$ **do**
  $\boldsymbol{\iota}_{t\ldots k} \leftarrow$ get_random_realizations(.)

  if $t + k - 1 < T$ **then**
    $\boldsymbol{\alpha} \leftarrow \pi_{\boldsymbol{\theta}}(\mathbf{s}_t^+)$
    $\mathbf{u}_{t\ldots k} \leftarrow \arg\min_{\mathbf{u}}\ \text{cost}\left(\mathbf{u}_{t\ldots k},\ \boldsymbol{\iota}_{t\ldots k},\ \mathbf{s}_t\right)\ -\ \boldsymbol{\alpha}_s^\top\ .\ \mathbf{s}_{t+k-1}$
  else
    $\mathbf{u}_{t\ldots k} \leftarrow \arg\min_{\mathbf{u}}\ \text{cost}\left(\mathbf{u}_{t\ldots k},\ \boldsymbol{\iota}_{t\ldots k},\ \mathbf{s}_t\right)$
  end if

  $c \leftarrow c + \text{SDP\_cost}(\mathbf{s}_t,\ \mathbf{u}_{t\ldots h},\ \boldsymbol{\iota}_{t\ldots h})$
  $\mathbf{s}_{t+h} \leftarrow \text{SDP\_transition}(\mathbf{s}_t,\ \mathbf{u}_{t\ldots h},\ \boldsymbol{\iota}_{t\ldots h})$
end for

return c

Direct Value Search

Step 2 (online): use $\pi_\theta(.)$ to solve the actual problem

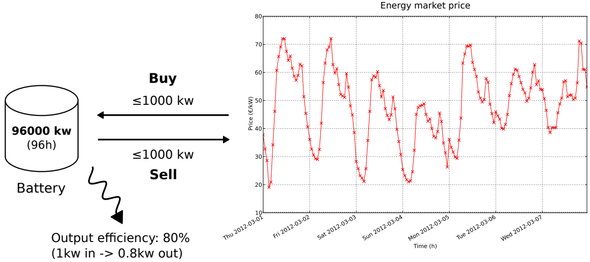
The offline optimisation of π_θ can be stopped at any time.

Then

- we have π_θ, an approximation of state's marginal value
- we can use it to solve the actual problem, the same manner as in *Simulate*

Experimental results

A simple test case



Goal: find a policy which maximise gains

Buy (and stock) when the market price is low, sell when the market price is high.

- The market price is stochastic.
- 10 constrained batteries.

A simple test case

- The state vector $\mathbf{s}^{+}$ is the stock level of the 10 batteries and additional information (4 handcrafted time-dependent auxiliary inputs);
- 10 decision variables have to be made at each time step (i.e. the quantity of energy to buy or sell for each batteries);
- we work in maximization, costs are replaced by rewards.

Baselines used for comparison

Recourse planning without final valorization

Bellman values are considered to be null ($\alpha = \mathbf{0}$)

$$u(x,t) = \arg\min_{u_t} \min_{u_{t+1},\dots,u_{t+k-1}} \mathbb{E} c_t + \dots + c_{t+k-1}$$

Recourse planning with constant marginal valorization

This is a linear approximation of Bellman values. Bellman values are considered to be independent of the current state (α is constant)

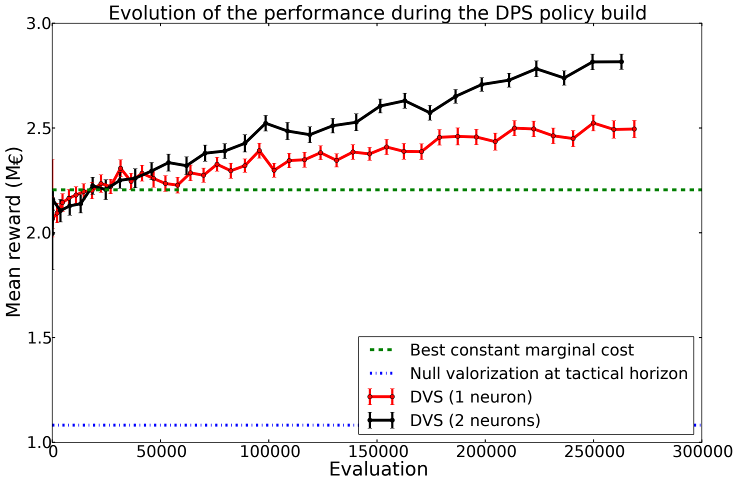
$$u(x,t) = \arg\min_{u_t} \min_{u_{t+1},\dots,u_{t+k-1}} \mathbb{E} c_t + \dots + c_{t+k-1} + \alpha . x_{t+k-1}$$

We look for the best constant marginal valorization α.

DVS setup

- Parametric policy $\pi_{\boldsymbol{\theta}}$ = a neural network with N neurons in a single hidden layer and weights vector θ;
- θ is optimized by maximizing $\theta \mapsto \text{Simulate}(\theta)$ with a Self-Adaptive Evolution Strategy (SA-ES).

Results


Evolution of the performance during the DPS policy build
Mean reward (M€)
Evaluation
3.0
2.5
2.0
1.5
1.0
0
50000
100000
150000
200000
250000
300000
Best constant marginal cost
Null valorization at tactical horizon
DVS (1 neuron)
DVS (2 neurons)

Conclusion

Conclusion

Still rather preliminary (a little tested) but promising

- forecasts naturally included in optimization
- anytime algorithm (users immediately get approximate results)
- no convexity constraints
- room for detailed simulations (e.g. with very small time scale, for volatility)
- no random process constraints (not Markov)
- can handle large state spaces (as DPS)
- can handle large action spaces (as SDP)

Can work on the “real” problem, without “cast”

Future work

- add relevant information in the state vector
  - e.g. moving average or regression analysis on the price
- optimize parametric policies with something else than SAES
  - Fabian
  - Newton
  - ...
- test DVS on a more challenging problem
  - more decision variables
  - more timesteps
  - non-convex Bellman values
  - ...
- parallelization

Bibliography

- *NEWAVE versus ODIN: comparison of stochastic and deterministic models for the long term hydropower scheduling of the interconnected brazilian system*. M. Zambelli et al., 2011.
- *Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC*. D. Bertsekas, 2005. (MPC = deterministic forecasts)
- Astrom 1965
- *Renewable energy forecasts ought to be probabilistic!* P. Pinson, 2013 (WIPFOR talk)
- *Training a neural network with a financial criterion rather than a prediction criterion* Y. Bengio, 1997 (quite practical application of direct policy search, convincing experiments)

Thank you for your attention

Questions ?

References I

- S. Astete-Morales, J. Liu, and O. Teytaud, *log-log convergence for noisy optimization*, Proceedings of EA 2013, LLNCS, Springer, 2013, p. accepted.
- alexander shapiro, *Analysis of stochastic dual dynamic programming method*, European Journal of Operational Research **209** (2011), no. 1, 63–72.
- R. Bellman, *Dynamic programming*, Princeton Univ. Press, 1957.
- Yoshua Bengio, *Using a financial training criterion rather than a prediction criterion*, CIRANO Working Papers 98s-21, CIRANO, 1998.

References II

- Dimitri P. Bertsekas, *Dynamic programming and suboptimal control: A survey from ADP to MPC*, Eur. J. Control **11** (2005), no. 4-5, 310–334.
- H.-G. Beyer, *The theory of evolutions strategies*, Springer, Heidelberg, 2001.
- D.P. Bertsekas and J.N. Tsitsiklis, *Neuro-dynamic programming*, Athena Scientific, 1996.
- Zhihao Cen, J. Frederic Bonnans, and Thibault Christel, *Energy contracts management by stochastic programming techniques*, Annals of Operations Research **200** (2011), no. 1, 199–222 (Anglais), RR-7289 RR-7289.

References III

- Gérard Cornuéjols, *Revival of the gomory cuts in the 1990's*, Annals OR **149** (2007), no. 1, 63–66.
- Z. L. Chen and W. B. Powell, *Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse*, J. Optim. Theory Appl. **102** (1999), no. 3, 497–524.
- Christopher J. Donohue and John R. Birge, *The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse*, Algorithmic Operations Research **1** (2006), no. 1.

References IV

Vaclav Fabian, *Stochastic approximation of minima with improved asymptotic speed*, Ann. Math. Statist. **38** (1967), no. 1, 191–200.

S. Hong, P.O. Malaterre, G. Belaud, and C. Dejean, *Optimization of irrigation scheduling for complex water distribution using mixed integer quadratic programming (MIQP)*, HIC 2012 – 10th International Conference on Hydroinformatics (Hamburg, Allemagne), IAHR, 2012, pp. p. – p. (Anglais).

References V

Verena Heidrich-Meisner and Christian Igel, *Hoeffding and bernstein races for selecting policies in evolutionary direct policy search*, ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning (New York, NY, USA), ACM, 2009, pp. 401–408.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos, *Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)*, Evolutionary Computation **11** (2003), no. 1, 1–18.

Holger Heitsch and Werner Römisch, *Scenario tree reduction for multistage stochastic programs*, Computational Management Science **6** (2009), no. 2, 117–133.

References VI

- L. G. Khachiyan, *A polynomial algorithm in linear programming*, Doklady Akademii Nauk SSSR **244** (1979), 1093–1096.
- K. Linowsky and A. B. Philpott, *On the convergence of sampling-based decomposition algorithms for multistage stochastic programs*, J. Optim. Theory Appl. **125** (2005), no. 2, 349–366.
- A.B. Philpott and Z. Guan, *On the convergence of stochastic dual dynamic programming and related methods*, Operations Research Letters **36** (2008), no. 4, 450 – 455.
- P. Pinson, *Renewable energy forecasts ought to be probabilistic*, 2013, WIPFOR seminar, EDF.

References VII

- W.-B. Powell, *Approximate dynamic programming*, Wiley, 2007.
- M. V. F. Pereira and L. M. V. G. Pinto, *Multi-stage stochastic optimization applied to energy planning*, Math. Program. **52** (1991), no. 2, 359–375.
- A. Ruszczynski, vol. 10, ch. Decomposition methods, North-Holland Publishing Company, Amsterdam, 2003.
- S. Uryasev and R.T. Rockafellar, *Optimization of conditional value-at-risk*, Research report (University of Florida. Dept. of Industrial & Systems Engineering), Department of Industrial & Systems Engineering, University of Florida, 1999.

Direct Value Search

Step 1 (offline): compute $\pi_{\boldsymbol{\theta}}(.)$

Build parametric policy $\pi_{\boldsymbol{\theta}}$

Require:

a parametric policy $\pi_{\boldsymbol{\theta}}(.)$ where $\pi_{\boldsymbol{\theta}}$ is a mapping from $\mathcal{S}$ to $\mathcal{U}$,
a Stochastic Decision Process SDP,
an initial state **s**

Ensure:

a parameter $\hat{\boldsymbol{\theta}}$ leading to a policy $\pi_{\hat{\boldsymbol{\theta}}}(.)$

Find a parameter $\hat{\theta}$ minimizing the expectation of the following fitness function
$\boldsymbol{\theta} \mapsto \text{Simulate}(\mathbf{s}, \text{SDP}, \pi_{\boldsymbol{\theta}})$
with a given non-linear noisy optimization algorithm (e.g. SA-ES, CMA-ES, ...)

return $\hat{\boldsymbol{\theta}}$

Direct Value Search

Step 1 (offline): compute $\pi_\theta(.)$

Simulate(s_0, **SDP**, π_θ)

$c \leftarrow 0$
for $t \leftarrow t_0,\ t_0 + h,\ t_0 + 2h,\ \ldots,\ T$ **do**
$\boldsymbol{\iota}_{t\ldots k} \leftarrow$ get_random_realizations(.)

if $t + k - 1 < T$ **then**
$\boldsymbol{\alpha} \leftarrow \pi_\theta(\mathbf{s}_t^+)$
$\boldsymbol{\alpha}_s \leftarrow \text{scale}(\boldsymbol{\alpha}, \mathbf{n}_f)$
$\mathbf{u}_{t\ldots k} \leftarrow \arg\min_{\mathbf{u}}\ \text{cost}\left(\mathbf{u}_{t\ldots k},\ \boldsymbol{\iota}_{t\ldots k},\ \mathbf{s}_t\right) - \boldsymbol{\alpha}_s^\top \cdot \mathbf{s}_{t+k-1}$
else
$\mathbf{u}_{t\ldots k} \leftarrow \arg\min_{\mathbf{u}}\ \text{cost}\left(\mathbf{u}_{t\ldots k},\ \boldsymbol{\iota}_{t\ldots k},\ \mathbf{s}_t\right)$
end if

$c \leftarrow c + \text{SDP\_cost}(\mathbf{s}_t,\ \mathbf{u}_{t\ldots h},\ \boldsymbol{\iota}_{t\ldots h})$
$\mathbf{s}_{t+h} \leftarrow \text{SDP\_transition}(\mathbf{s}_t,\ \mathbf{u}_{t\ldots h},\ \boldsymbol{\iota}_{t\ldots h})$
end for

return c

Direct Value Search

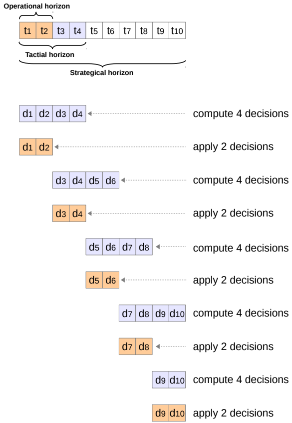
Step 2 (online): use $\pi_\theta(.)$ to solve the actual problem

The offline optimisation of π_θ can be stopped at any time.

Then

- we have π_θ, an approximation of state's marginal value
- we can use it to solve the actual problem, the same manner as in *Simulate*

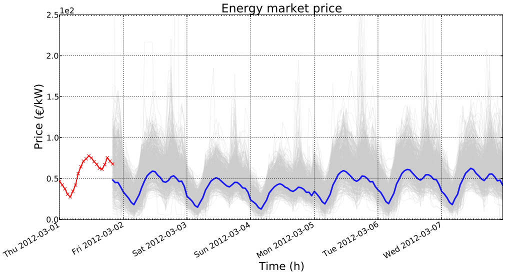
Recourse planning (closed loop)



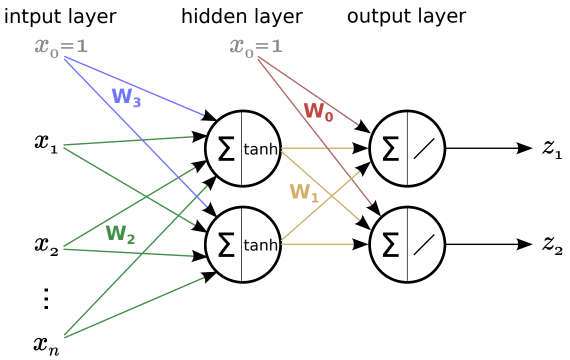
Recourse planning:

- Decisions $u_1, u_2, \dots u_{\tau_t}$ are optimized (tactical horizon)
- Only $u_1, u_2, \dots u_{\tau_o}$ are applied (operational horizon)
- The system is now in a new state at stage τ_o
- Decisions $u_{\tau_o}, u_{\tau_o+1}, \dots u_{\tau_o+\tau_t}$ are optimized (tactical horizon)
- Only $u_{\tau_o}, u_{\tau_o+1}, \dots u_{\tau_o+\tau_o}$ are applied (operational horizon)
- ...

MPC example


Energy market price
1e2
2.5
2.0
1.5
1.0
0.5
0.0
Price (€/kW)
Thu 2012-03-01
Fri 2012-03-02
Sat 2012-03-03
Sun 2012-03-04
Mon 2012-03-05
Tue 2012-03-06
Wed 2012-03-07
Time (h)

Parametric policies



$$\mathbf{z} = \mathbf{W_0} + \mathbf{W_1} \tanh(\mathbf{W_2}\ \mathbf{x} + \mathbf{W_3})$$

Parametric policies

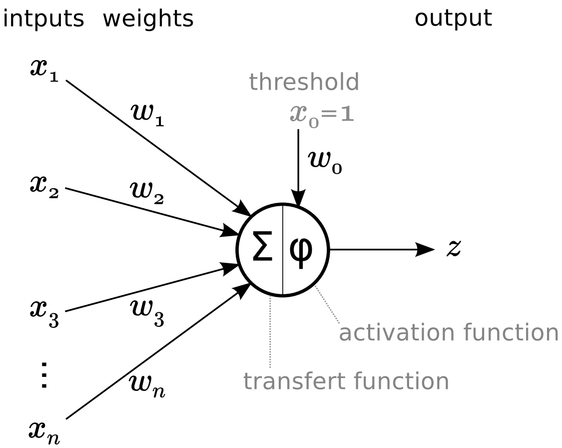
$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{pmatrix}, \mathbf{W_3} = \begin{pmatrix} w_{10}^{(1)} \\ w_{20}^{(1)} \\ \vdots \\ w_{m0}^{(1)} \end{pmatrix}, \mathbf{W_0} = \begin{pmatrix} w_{10}^{(2)} \\ w_{20}^{(2)} \\ \vdots \\ w_{k0}^{(2)} \end{pmatrix},$$

$$\mathbf{W_2} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & \cdots & w_{1n}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & \cdots & w_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}^{(1)} & w_{m2}^{(1)} & \cdots & w_{mn}^{(1)} \end{pmatrix}, \mathbf{W_1} = \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} & \cdots & w_{1m}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & \cdots & w_{2m}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1}^{(2)} & w_{k2}^{(2)} & \cdots & w_{km}^{(2)} \end{pmatrix}.$$

$$\mathbf{z} = \mathbf{W_0} + \mathbf{W_1} \tanh(\mathbf{W_2}\ \mathbf{x} + \mathbf{W_3})$$

Artificial Neurons

Artificial neuron



Self-adaptive Evolution Strategy (SA-ES) with revaluations

Require:
$K > 0$, $\lambda > \mu > 0$, a dimension $d > 0$, τ (usually $\tau = \frac{1}{\sqrt{2d}}$).

Initialize parent population $P_\mu = \{(\mathbf{x}_1, \sigma_1), (\mathbf{x}_2, \sigma_2), \ldots, (\mathbf{x}_\mu, \sigma_\mu)\}$
with $\forall i \in \{1, \ldots, \mu\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $\sigma_i = 1$.

while stop condition **do**

Generate the offspring population $P_\lambda = \{(\mathbf{x}'_1, \sigma'_1), (\mathbf{x}'_2, \sigma'_2), \ldots, (\mathbf{x}'_\lambda, \sigma'_\lambda)\}$
where each individual is generated by:

1. *Select* (randomly) ρ parents from P_μ.
2. *Recombine* the ρ selected parents to form a recombinant individual $(\mathbf{x}', \sigma')$.
3. *Mutate* the strategy parameter: $\sigma' \leftarrow \sigma' e^{\tau \mathcal{N}(0,1)}$.
4. *Mutate* the objective parameter: $\mathbf{x}' \leftarrow \mathbf{x}' + \sigma' \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{1})$.

Select the new parent population P_μ taking the μ best form $P_\lambda \cup P_\mu$.

end while

Fabian

Input: an initial $x_1 = 0 \in \mathbb{R}^d$, $\frac{1}{2} > \gamma > 0$, $a > 0$, $c > 0$, $m \in \mathbb{N}$, weights $w_1 > \dots > w_m$ summing to 1, scales $1 \geq u_1 > \dots > u_m > 0$.
$n \leftarrow 1$
while (true) **do**
  Compute $\sigma_n = c/n^\gamma$.
  Evaluate the gradient g at x_n by finite differences, averaging over $2m$ samples per axis:

$$\begin{aligned}
\forall i,j \in \{1,\dots,d\} \times \{1\dots m\},\ x_n^{(i,j)+} &= x_n + u_j e_i, \\
\forall i,j \in \{1,\dots,d\} \times \{1\dots m\},\ x_n^{(i,j)-} &= x_n - u_j e_i, \\
\forall i \in \{1,\dots,d\}, g^{(i)} &= \frac{1}{2\sigma_n} \sum_{j=1}^{m} w_j \left(f(x_n^{(i,j)+}) - f(x_n^{(i,j)-}) \right).
\end{aligned}$$

  Apply $x_{n+1} \leftarrow x_n - \frac{a}{n} g$
  $n \leftarrow n + 1$
end while

Fabian's stochastic gradient algorithm with finite differences. Several variants have been defined, in particular versions in which only one point (or a constant number of points, independently of the dimension) is evaluated at each iteration.