Machine Learning and Photonics

Week 14

Ergun Simsek

University of Maryland Baltimore County
simsek@umbc.edu

Optimizers and Numerical Optimization

May 9, 2023

Overview

What is optimization? Why do we care?

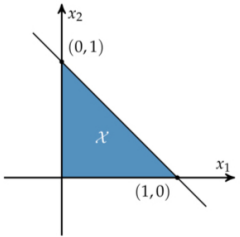
Optimization in engineering is the process of finding the best system design subject to a set of constraints.

A typical optimization problem is to

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in X \end{aligned}$$

A design point (x) can be represented as a vector of values corresponding to different design variables.

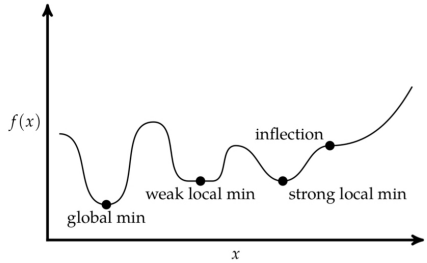
$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & f(x_1,x_2) \\ \text{subject to} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_1 + x_2 \leq 1 \end{aligned}$$



Why is $f'(x)$ important?

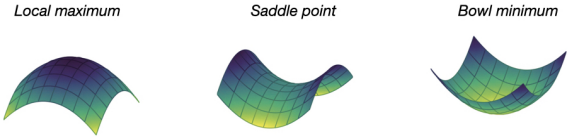
$f'(x) = 0$ is not a sufficient condition

Univariate:



- $f'(x^*) = 0$ and $f''(x^*) > 0$, then strong local minimum
- $f'(x^*) = 0$ and $f''(x^*) < 0$, then strong local maximum

Multivariate:



LOCAL DESCENT: A general approach to optimization

1. Check $f(\mathbf{x}^{(k)}) \overset{?}{<} \epsilon$ If it does, terminate; otherwise proceed to the next step.
2. Determine the descent direction $\mathbf{d}_k$ using local information such as the gradient (or Hessian for multivariate optimization).
3. Determine the step size or learning rate α_k.
4. Compute the next design point according to: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
5. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)}$, go back to step-1!

The choice of descent direction

- So we assumed that **d** is a valid descent direction,
- Then we said that we could use the line search method to obtain a sufficient decrease.
- Repeating it for many times, we hoped to arrive at the local minimum.

- How can we find a valid descent direction?

Steepest Descent

An intuitive choice for the descent direction is the direction of steepest descent (the direction opposite the gradient ∇f)

$$\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$$

where $\mathbf{x}^{(k)}$ is our design point at descent iteration k.

In gradient descent, we typically normalize the direction of steepest descent

$$\mathbf{d}^{(k)} = -\frac{\mathbf{g}^{(k)}}{||\mathbf{g}^{(k)}||}$$

Gradient descent

If we optimize the step size at each step, we have

$$\alpha^{(k)} = \arg\min_{\alpha} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$$

The optimization above implies that the directional derivative equals zero. Since

$$\nabla f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})^T \mathbf{d}^{(k)} = 0$$

$$\mathbf{d}^{(k+1)} = -\frac{\nabla f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})}{||\nabla f(\mathbf{x}^{(k)} + \alpha^{(k)})||}$$

If you multiply the above equation's both side with $\mathbf{d}^{(k)}$, you can see that these two consecutive directions are orthogonal.

$$(\mathbf{d}_{k+1})^T \mathbf{d}^k = 0$$

Conjugate gradient

Gradient descent can perform poorly in narrow valleys. The conjugate gradient method overcomes this issue by doing a small transformation.
When minimizing the quadratic functions:

$$\underset{\alpha}{\text{minimize}} : f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

is equivalent to solving the linear equation

$$Ax = b$$

where A is $N \times N$ symmetric and positive definite, and thus f has a unique local minimum.
When solving $\mathbf{Ax} = \mathbf{b}$, a powerful method is to find a sequence of N conjugate directions satisfying

$$(\mathbf{d}^{(i)})^T\mathbf{A}\mathbf{d}^{(j)} = 0 \quad (i \neq j), \tag{1}$$

To find the successive conjugate directions

One can start with the direction of steepest descent

$$\mathbf{d}^{(1)} = -\mathbf{g}^{(1)} \tag{2}$$

We then use line search to find the next design point. For quadratic functions $f = \frac{1}{2}x^T Ax - b^T x$, the step factor α can be computed as

$$\begin{aligned} \frac{\partial f(\mathbf{x}+\alpha d)}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\left[\frac{1}{2}(\mathbf{x}+\alpha\mathbf{d})^T\mathbf{A}(\mathbf{x}+\alpha\mathbf{d}) + \mathbf{b}^T(\mathbf{x}+\alpha\mathbf{d}) + c\right] \\ &= \mathbf{d}^T\mathbf{A}(\mathbf{x}+\alpha\mathbf{d}) + \mathbf{d}^T\mathbf{b} \\ &= \mathbf{d}^T(\mathbf{A}\mathbf{x}+\mathbf{b}) + \alpha\mathbf{d}^T\mathbf{A}\mathbf{d} \end{aligned} \tag{3}$$

Let the gradient be zero,

$$\alpha = -\frac{\mathbf{d}^T(\mathbf{A}\mathbf{x}+\mathbf{b})}{\mathbf{d}^T\mathbf{A}\mathbf{d}} \tag{4}$$

Then the update is

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)} \tag{5}$$

To find the successive conjugate directions (continued)

For the next step

$$\mathbf{d}^{(k+1)}) = -\mathbf{g}^{(k+1)} + \beta^{(k)}\mathbf{d}^{(k)} \tag{6}$$

where $\beta^{(k)}$ is a series of scalar parameters. Larger values of β indicate that the previous descent direction contributes strongly.
We solve β, from the followings

$$(\mathbf{d}^{(k+1))^T}\mathbf{A}\mathbf{d}^{(k)} = 0 \tag{7}$$

$$(-\mathbf{g}^{(k+1)} + \beta^{(k)}\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{d}^{(k)} = 0 \tag{8}$$

$$-\mathbf{g}^{(k+1)}\mathbf{A}\mathbf{d}^{(k)} + \beta^{(k)}(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{d}^{(k)} = 0 \tag{9}$$

$$\beta^{(k)} = \frac{(\mathbf{g}^{(k+1)})^T\mathbf{A}\mathbf{d}^{(k)}}{(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{d}^{(k)}} \tag{10}$$

The conjugate method is exact for quadratic functions. But it can be applied to non quadratic functions as well when the quadratic function is a good approximation.

To Approximate A and β

Unfortunately, we don't know the value of **A** that best approximate f around $\mathbf{x}^{(k)}$. So we need choose a way to compute β. In literature, there are two commonly used formulas.

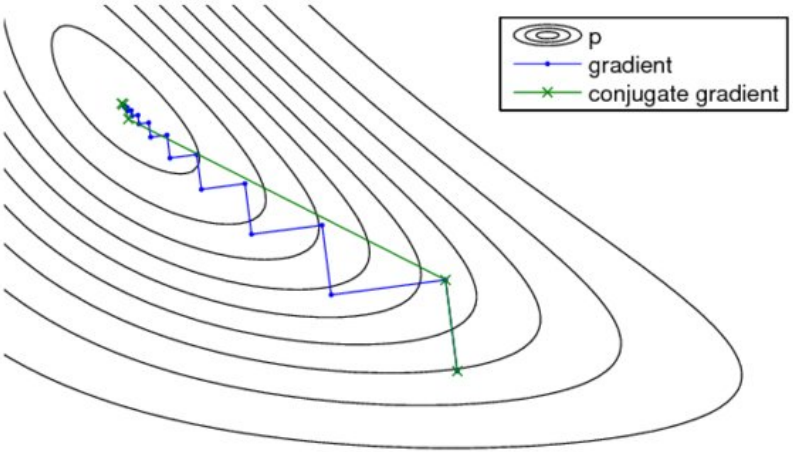
Fletcher-Reeves

$$\beta^k = \frac{g^{(k)T} g^{(k)}}{g^{(k-1)T} g^{(k-1)}}$$

Polak-Ribiere

$$\beta^k = \frac{g^{(k)T} (g^{(k)} - g^{(k-1)})}{g^{(k-1)T} g^{(k-1)}}$$

Comparison between Conjugate Gradient and Steepest Descent



Summary

- Gradient descent follows the direction of steepest descent
- Two consecutive search directions in gradient descent are orthogonal
- In conjugate gradient, the search directions are conjugate with respect to an approximate hessian.
- Both SD and CG work with the line search method

Gradient Methods with Varying Learning Rate

- Momentum
- Nesterov Momentum
- Adagrad
- RMSProp
- Adadelta
- Adam

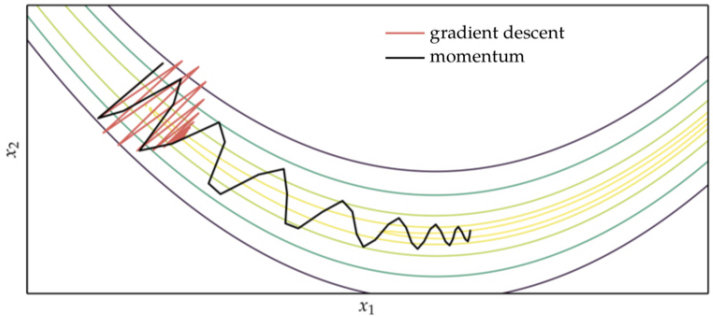
Momentum

Allow momentum to accumulate is one way to speed the progress.

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} + (1 - \beta) \nabla f(\mathbf{x}^{(k)}) \tag{11}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{v}^{(k+1)} \tag{12}$$

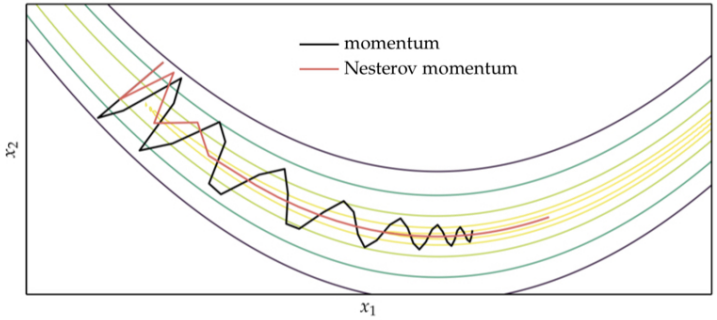
When β=0, it is gradient descent.



Nesterov Momentum

$$\mathbf{v}^{(k+1)} = \beta \mathbf{v}^{(k)} + (1-\beta)\nabla f(\mathbf{x}^{(k)} - \beta \mathbf{v}^{(k)}) \tag{13}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{v}^{(k+1)} \tag{14}$$



Adagrad: Adaptive subgradient method

Idea: A learning rate for each one in θ

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}}\mathbf{x}^{(k)}$$

$$s_i^{(k)} = \sum_{j=1}^{k} \left(g_i^{(j)} \right)^2$$

where ϵ is a small value on the order of 1e-8.

RMSProp and Adadelta

$$\mathbf{s}^{(k+1)} = \gamma \mathbf{s}^{(k)} + (1-\gamma)(\mathbf{g}^{(k)} \odot \mathbf{g}^{(k)}) \tag{15}$$

where γ is between 0 and 1, and usually is 0.9.

$$\begin{aligned} \mathbf{x}_i^{(k+1)} &= \mathbf{x}_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k)}}} \mathbf{g}_i^{(k)} \\ &= \mathbf{x}_i^{(k)} - \frac{\alpha}{\epsilon + \mathrm{RMS}(\mathbf{g}_i)} \mathbf{g}_i^{(k)} \end{aligned} \tag{16}$$

While in Adadelta, an exponentially decaying average is used,

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} - \frac{\mathrm{RMS}(\nabla_i)}{\epsilon + \mathrm{RMS}(\mathbf{g}_i)} \mathbf{g}_i^{(k)} \tag{17}$$

Adam

$$
\begin{aligned}
\mathbf{v}^{(k+1)} &= \gamma_v \mathbf{v}^{(k)} + (1-\gamma_v)\mathbf{g}^{(k)} \\
\mathbf{s}^{(k+1)} &= \gamma_s \mathbf{s}^{(k)} + (1-\gamma_s)\left(\mathbf{g}^{(k)} \odot \mathbf{g}^{(k)}\right) \\
\hat{\mathbf{v}}^{(k+1)} &= \mathbf{v}^{(k+1)} / (1-\gamma_v^{(k)}) \\
\hat{\mathbf{s}}^{(k+1)} &= \mathbf{s}^{(\mathbf{k+1})} / (1-\gamma_s^{(k)}) \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \alpha \hat{\mathbf{v}}^{(k+1)} / \left(\epsilon + \sqrt{\hat{\mathbf{s}}^{(k+1)}}\right)
\end{aligned}
\tag{18}
$$

Summary

- Descent methods with momentum build up progress in favorable directions
- A wide variety of accelerated descent methods use special techniques to speed up descent

Direct methods without gradient

Direct methods rely solely on the objective function f. They are usually called

- zero-orther
- black box
- pattern search
- derivative free

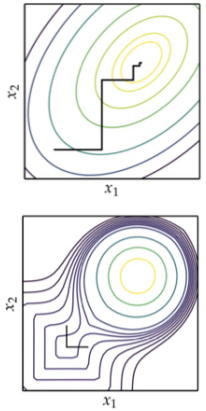
Cyclic Coordinate Search

Alternate coordinate directions for its line search.

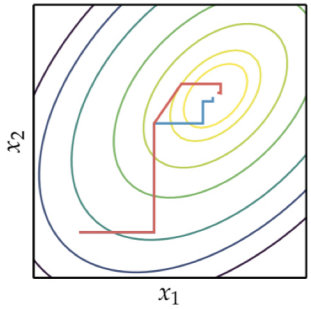
$$\mathbf{x}^{(2)} = \arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \mathbf{x}_2^{(1)}, \mathbf{x}_3^{(1)}, \cdots, \mathbf{x}_n^{(1)}) \qquad (19)$$

Then, it moves to the next coordinate,

$$\mathbf{x}^{(3)} = \arg\min_{\mathbf{x}_2} f(\mathbf{x}_1^{(2)}, \mathbf{x}_2, \mathbf{x}_3^{(2)}, \cdots, \mathbf{x}_n^{(2)}) \qquad (20)$$



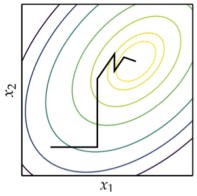
Acceleration



Powell's method

Maintain a list of search directions $\boldsymbol{u}^1, \cdots, \boldsymbol{u}^n$s.

$$\boldsymbol{x}^{i+1} \leftarrow \text{line search}(f, \boldsymbol{x}^i, \boldsymbol{u}^i) \text{ for all} i i n 1, \cdots, n$$
$$\boldsymbol{u}^{i+1} \leftarrow \boldsymbol{u}^{i+1}$$
$$\text{for all} i i n 1, \cdots, n-1$$
$$\boldsymbol{u}^n \leftarrow \boldsymbol{x}^{n+1} - \boldsymbol{x}^n$$

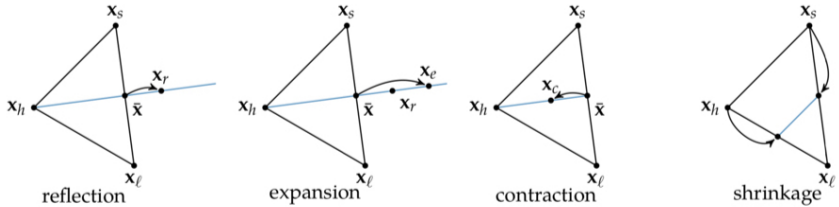


Nelder-Mead Simplex Method

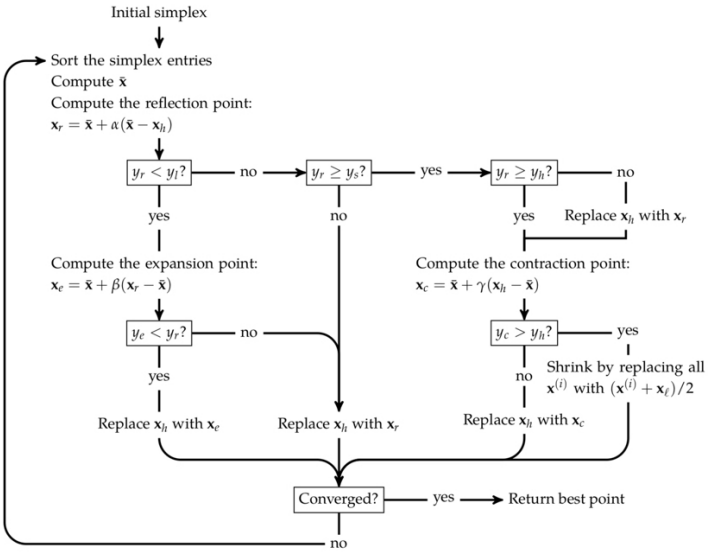
Use a simplex to traverse the space in search of a minimum. A simplex is a $n+1$-vertices polyhedron in n-dimensional space.

- x_h, pt of highest f,
- x_s, pt of 2nd highest f,
- x_l, pt of lowest f,
- $\bar{x}$, mean pt excluding x_h.

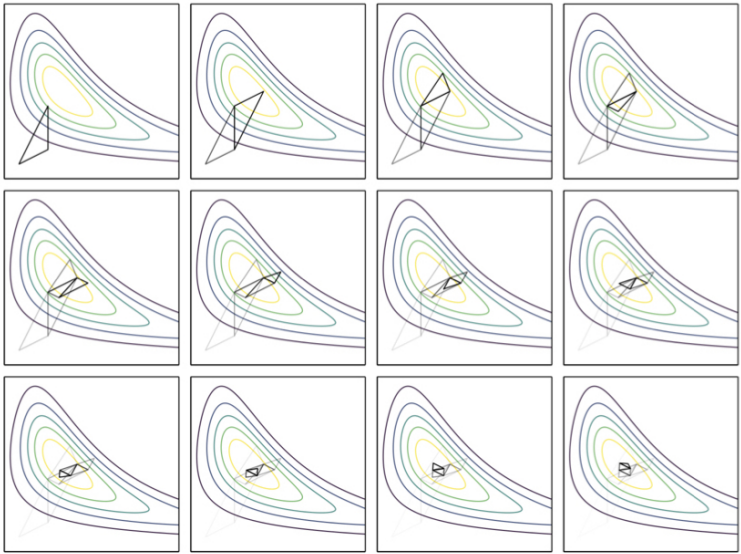
- Reflection. $x_r = \bar{x} + (\bar{x} - x_h)$,
- Expansion. $x_e = \bar{x} + 2(x_r - \bar{x})$,
- Contraction. $x_c = \bar{x} + 0.5(x_h - \bar{x})$,
- Shrinkage, halving the distance to x_l.



Nelder-Mead Simplex Algorithm



Nelder-Mead Simplex method in practice



Simulated Annealing

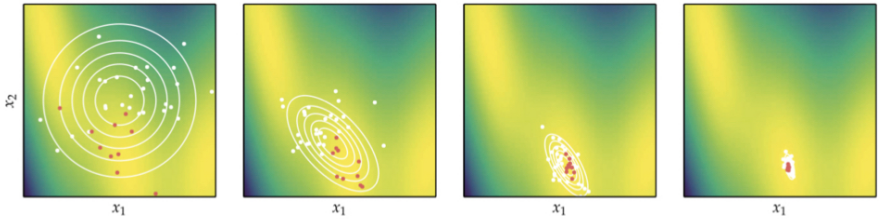
Use Temperature to control the degree of stochasticity during the randomized search.
At every iteration, a candidate transition from $\boldsymbol{x}$ to $\boldsymbol{x'}$ is sampled from a transition distribution T and is accepted with probability

$$\begin{cases} 1 & \text{if } \Delta y \leq 0 \\ \min(\exp(-\Delta y / t), 1) & \text{if } \Delta y > 0 \end{cases}$$

where $\Delta y = f(\boldsymbol{x}) - f(\boldsymbol{x'})$

Cross-Entropy Method

$$\boldsymbol{\mu}^{(k+1)} = \frac{1}{m_{\text{elite}}} \sum_{i=1}^{m_{\text{elite}}} \mathbf{x}^{(k)}$$
$$\Sigma^{(k+1)} = \frac{1}{m_{\text{elite}}} \sum_{i=1}^{m_{\text{elite}}} (\mathbf{x}^{(k)} - \boldsymbol{\mu}^{(k+1)})(\mathbf{x}^{(k)} - \boldsymbol{\mu}^{(k+1)})^T \quad (21)$$



Covariance Matrix Adaptation

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \Sigma)$$

Sort the designs according to their objective function values, i.e., $f(x^1) \leq f(x^2) \leq \cdots \leq f(x^m)$.

Calculate the new mean vector

$$\boldsymbol{\mu}^{k+1} \leftarrow \sum_{i=1}^{m} w_i \boldsymbol{x}^i$$

$$\sum_{i}^{m} w_i = 1 \quad w_1 > w_2 > \cdots > w_m > 0$$

Covariance Matrix Adaptation

The recommended weighting is obtained by

$$w'_i = \ln \frac{m+1}{2} - \ln i \text{ for } i \in \{1, \cdots, m\}$$

to obtain $\boldsymbol{w} = \boldsymbol{w}' / \sum_i w'_i$.
The step size is updated using a cumulative $\boldsymbol{p}_\sigma$ that tracks steps over time

$$\begin{aligned}
\boldsymbol{p}_\sigma^1 &= 0 \\
\boldsymbol{p}_\sigma^{k+1} &\leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}(\Sigma^k)^{-1/2}\sigma_w \\
\mu_{\text{eff}} &= \frac{1}{\sum_i w_i^2} \\
\sigma_w &= \sum_{i=1}^{m_{\text{elite}}} w_i \sigma^i \text{ for } \sigma^i = \frac{\boldsymbol{x}^i - \boldsymbol{\mu}^k}{\sigma^k}
\end{aligned}$$

Covariance Matrix Adaptation

The new step size is

$$\sigma^{k+1} \leftarrow \sigma^k \exp\left(\frac{c_\sigma}{d_\sigma}\left[\frac{||\boldsymbol{p}_\sigma||}{\mathbb{E}||\mathcal{N}(0,\boldsymbol{I})||} - 1\right]\right)$$

where $\mathbb{E}$ is the expected length of a vector drawn from Gaussian distribution.

$$\mathbb{E}||\mathcal{N}(0,\boldsymbol{I})|| = \sqrt{2}\frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} \approx \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right)$$

$$c_\sigma = (\mu_{\text{eff}} + 2)/(n + \mu_{\text{eff}} + 5)$$
$$d_\sigma = 1 + 2\max(0, \sqrt{\mu_{\text{eff}} - 1)/(n+1)} - 1) + c_\sigma$$

Covariance Matrix Adaptation

The covariance matrix is updated as follows

$$\boldsymbol{p}_{\Sigma}^{1} = 0$$
$$\boldsymbol{p}_{\Sigma}^{k+1} \leftarrow (1 - c\Sigma)\boldsymbol{p}_{\Sigma}^{\boldsymbol{k}} + h_{\sigma}\sqrt{c\Sigma(2 - c_{\Sigma})\mu_{\text{eff}}}\sigma_{w}$$

where

$$h_{\sigma} = \begin{cases} 1 & if \frac{||\boldsymbol{p}_{\Sigma}||}{(1-c_{\sigma}^{2k+1})} < (1.4 + \frac{2}{n+1})\mathbb{E}||\mathcal{N}(0, \boldsymbol{I})|| \\ 0 & \text{otherwise} \end{cases}$$

The update requires the adjusted weights $\boldsymbol{w}$:

$$w_{i}^{0} = \begin{cases} w_{i} & \text{if } w_{i} \geq 0 \\ \frac{nw_{i}}{||\Sigma^{-1/2}\delta^{i}||^{2}} & \text{otherwise} \end{cases}$$

Covariance Matrix Adaptation

The The covariance update is then

$$\Sigma^{k+1} \leftarrow [1 + c_1 c_\sigma (1 - h_\sigma)(2 - c_\sigma) - c_1 - c_\mu]\Sigma^k + c_1 \boldsymbol{p}_\Sigma \boldsymbol{p}_\Sigma^T + c_\mu \sum_{i=1}^{\mu} w_i^0 \delta^i (\delta^i)^T$$

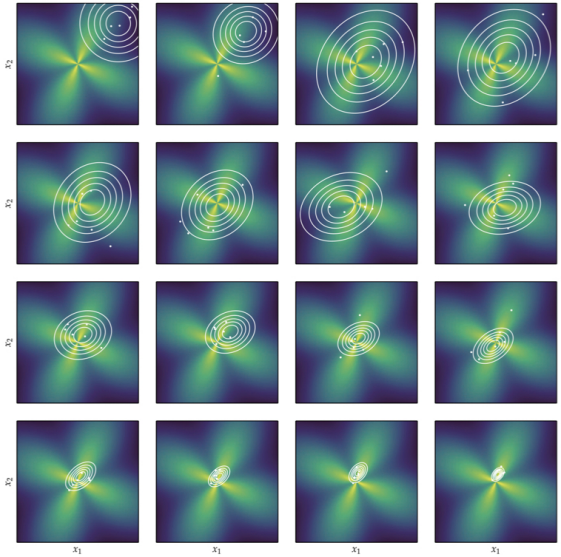
The constants have the following recommended values

$$c_\Sigma = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n}$$

$$c_1 = \frac{2}{(n + 1.3)^2 + \mu_{\text{eff}}}$$

$$c_\mu = \min\left(1 - c_1, 2\frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \mu_{\text{eff}}}\right)$$

Covariance Matrix Adaptation



22

Summary

- Stochastic methods employ random numbers during the optimization process
- Simulated annealin guses a temperature that controls random exploration and which is reduced over time to converge on a local minimum.
- The cross-entropy method and evolution strategies maintain proposal distributions from which they sample in order to inform updates.
- Covariance matrix adaptation is a robust and sample-efficient optimizer that maintains a multivariate Gaussian proposal distribution with a full covariance matrix.

Population Methods

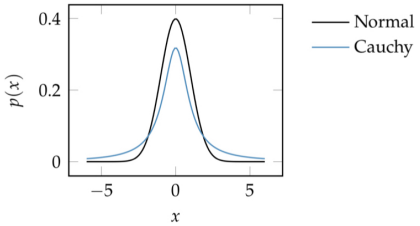
Typical steps

- Initialization
- Encoding
- Mutation
- Crossover
- Selection

Initialization

The following initialization strategies can be applied

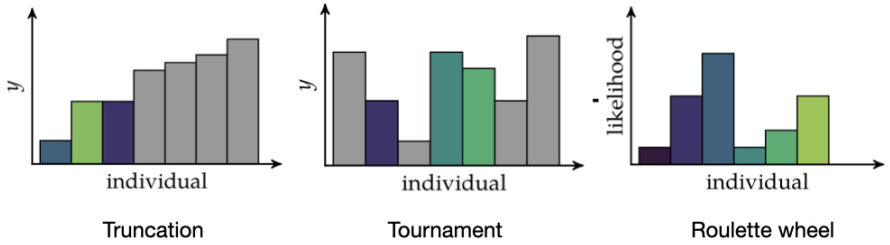
- Uniform distribution in a bounded region
- Multivariate normal distribution centered over a region of interest.
- The Cauchy distribution has an unbounded variance and can cover a much broader space.



Selection

Selection is the process of choosing chromosomes to use as parents for the next generation. For a population with m chromosomes, a selection method will produce a list of *m* parental pairs for the m children of the next generation. The selected pairs may contain duplicates.

- Truncation, random one from the best *k* truncation
- Tournament, the fittest out of *k* randomly chosen
- Roulette wheel, chosen with a probability proportional to the fitness



Particle Swarm Optimization

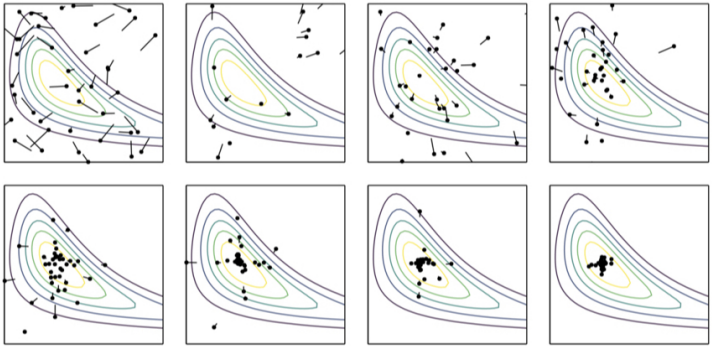
Introduce momentum to accelerate convergence toward minima.

$$\boldsymbol{x}^i \leftarrow \boldsymbol{x}^i + \boldsymbol{v}^i$$
$$\boldsymbol{v}^i \leftarrow w\boldsymbol{v}^i + c_1 r_1 (\boldsymbol{x}^i_{lbest} - x^i) + c_2 r_2 (\boldsymbol{x}_{gbest} - x^i)$$

where

- x_{lbest}: the current local best locations for the given population
- x_{gbest}: the global best locations
- w, c_1, c_2: empirical parameters
- r_1, r_2: random numbers drawn from $U(0, 1)$

PSO search



Summary

- Population methods use a collection of individuals in the design space to guide progression toward an optimum.
- Genetic algorithms leverage selection, crossover, and mutations to produce better subsequent generations.
- Particle swarm optimization and the firefly algorithm include rules and mechanisms for attracting design points to the best individuals in the population while maintaining suitable state space exploration.
- Population methods can be extended with local search approaches to improve convergence.