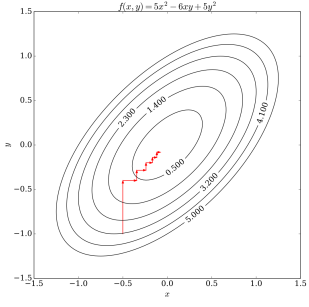
Optimization in Machine Learning

Coordinate descent

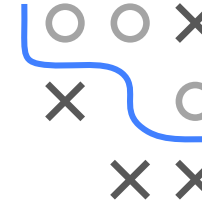


Learning goals

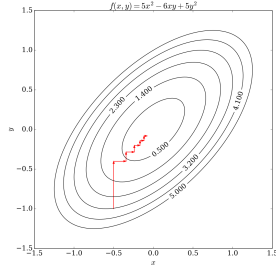
- Axes as descent direction
- CD on linear model and LASSO
- Soft thresholding

COORDINATE DESCENT

- **Assumption:** Objective function not differentiable
- **Idea:** Instead of gradient, use coordinate directions for descent
- First: Select starting point $\boldsymbol{x}^{[0]} = (x_1^{[0]}, \ldots, x_d^{[0]})$
- Step t: Minimize f along x_i for each dimension i for fixed $x_1^{[t]}, \ldots, x_{i-1}^{[t]}$ and $x_{i+1}^{[t-1]}, \ldots, x_d^{[t-1]}$:

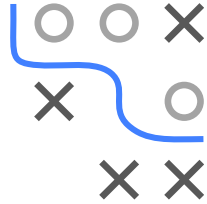


fix axis and optimize then take the next axis



Source: Wikipedia (Coordinate descent)

COORDINATE DESCENT



- Minimum is determined with (exact / inexact) line search
- Order of dimensions can be any permutation of $\{1, 2, \ldots, d\}$
- **Convergence:**
  - f convex differentiable
  - f sum of convex differentiable and *convex separable* function:

(we don't care about axis order)

$$f(\mathbf{x}) = g(\mathbf{x}) + \sum_{i=1}^{d} h_i(x_i),$$

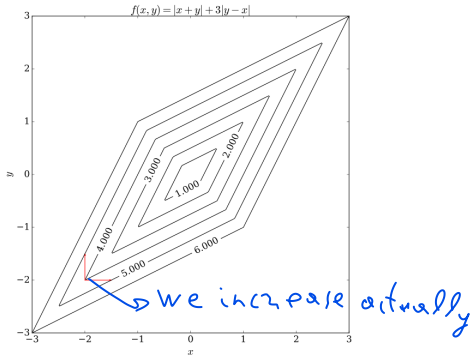
h_i only depends on x_i

where g convex differentiable and h_i convex

COORDINATE DESCENT

Not convergence in general for convex functions.

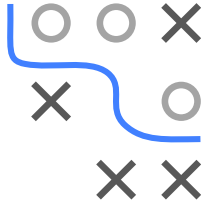
Counterexample:



Source: Wikipedia (Coordinate descent)

EXAMPLE: LINEAR REGRESSION

Minimize LM with L2-loss via CD:



$$\min g(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right)^2 = \min_{\boldsymbol{\theta}} \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2$$

where $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{n \times d}$ with columns $\mathbf{x}_1, \ldots, \mathbf{x}_d \in \mathbb{R}^n$.

Assume: Scaled data, i.e., $\mathbf{X}^\top \mathbf{X} = I_d$ (just to get intuition)

is this the same as standard scaling?

Then:

$$g(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta} - \mathbf{y}^\top \mathbf{X}\boldsymbol{\theta}$$

$$\overset{(*)}{=} \frac{1}{2}\mathbf{y}^\top \mathbf{y} + \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta} - \mathbf{y}^\top \sum_{k=1}^{d} \mathbf{x}_k \theta_k$$

$^{(*)}$ $\mathbf{X}\boldsymbol{\theta} = \mathbf{x}_1\theta_1 + \mathbf{x}_2\theta_2 + \cdots + \mathbf{x}_d\theta_d = \sum_{k=1}^{d} \mathbf{x}_k \theta_k$

EXAMPLE: LINEAR REGRESSION

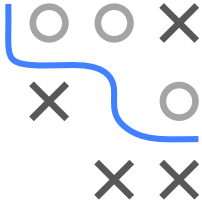
- Exact CD update in direction j:

$$\frac{\partial g(\boldsymbol{\theta})}{\partial \theta_j} = \theta_j - \mathbf{y}^\top \mathbf{x}_j$$

- By solving $\frac{\partial g(\boldsymbol{\theta})}{\partial \theta_j} = 0$, we get

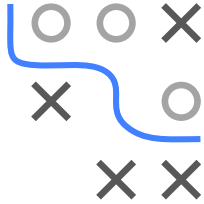
$$\theta_j^* = \mathbf{y}^\top \mathbf{x}_j$$

- **Repeat** this update for all θ_j



SOFT THRESHOLDING

Minimize LM with L2-loss and L1 regularization via CD:



$$\min_{\boldsymbol{\theta}} h(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda\|\boldsymbol{\theta}\|_1$$

Note that $h(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top\mathbf{y} + \frac{1}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} - \sum_{k=1}^{d}(\mathbf{y}^\top\mathbf{x}_k\theta_k + \lambda|\theta_k|)$

Assume (again): $\mathbf{X}^\top\mathbf{X} = I_d$.
Since $|\cdot|$ is not differentiable, distinguish three cases:

- **Case 1:** $\theta_j > 0$. Then $|\theta_j| = \theta_j$ and

$$0 = \frac{\partial h(\boldsymbol{\theta})}{\partial \theta_j} = \theta_j - \mathbf{y}^\top\mathbf{x}_j + \lambda \qquad \Leftrightarrow \qquad \theta^*_{j,\text{LASSO}} = \theta^*_j - \lambda$$

- **Case 2:** $\theta_j < 0$. Then $|\theta_j| = -\theta_j$ and

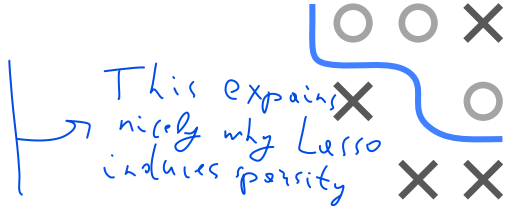
$$0 = \frac{\partial h(\boldsymbol{\theta})}{\partial \theta_j} = \theta_j - \mathbf{y}^\top\mathbf{x}_j - \lambda \qquad \Leftrightarrow \qquad \theta^*_{j,\text{LASSO}} = \theta^*_j + \lambda$$

- **Case 3:** $\theta_j = 0$

SOFT THRESHOLDING

We can write the solution as:

$$\theta^*_{j,\text{LASSO}} = \begin{cases} \theta^*_j - \lambda & \text{if } \theta^*_j > \lambda \\ \theta^*_j + \lambda & \text{if } \theta^*_j < -\lambda \\ 0 & \text{if } \theta^*_j \in [-\lambda, \lambda], \end{cases}$$

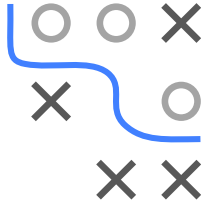


This operation is called **soft thresholding**.

Coefficients for which the solution to the unregularized problem are smaller than a threshold, $|\theta^*_j| < \lambda$, are shrinked to zero.

Note: Derivation of soft thresholding operator not trivial (subgradients)

CD FOR STATISTICS AND ML



Why is it being used?

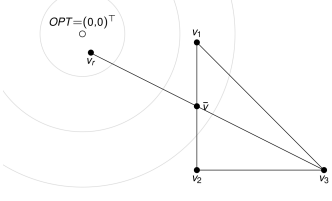
- Easy to implement
- Scalable: no storage/operations on large objects, just current point
  $\Rightarrow$ Good implementation can achieve state-of-the-art performance
- Applicable for non-differentiable (but convex separable) objectives

Examples:

- Lasso regression, Lasso GLM, graphical Lasso
- Support Vector Machines
- Regression with non-convex penalties

Optimization in Machine Learning

Nelder-Mead method

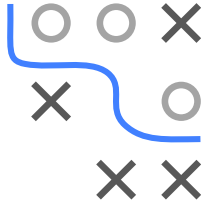


Learning goals

- General idea
- Reflection, expansion, contraction
- Advantages & disadvantages
- Examples

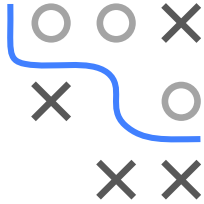
NELDER-MEAD METHOD

- Derivative-free method $\Rightarrow$ heuristic
- Generalization of bisection in d-dimensional space
- Based on d-simplex, defined by $d + 1$ points:
  - $d = 1$ interval
  - $d = 2$ triangle
  - $d = 3$ tetrahedron
  - $\cdots$



NELDER-MEAD METHOD

A version of the **Nelder-Mead** method:

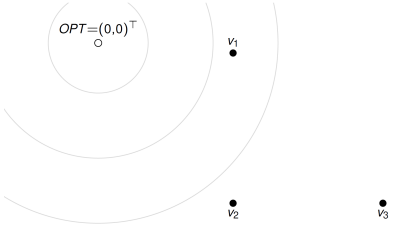


Initialization: Choose $d + 1$ random, affinely independent points $\mathbf{v}_i$ ($\mathbf{v}_i$ are vertices: corner points of the simplex/polytope).

1. **Order**: Order points according to ascending function values

$$f(\mathbf{v}_1) \leq f(\mathbf{v}_2) \leq \ldots \leq f(\mathbf{v}_d) \leq f(\mathbf{v}_{d+1}).$$

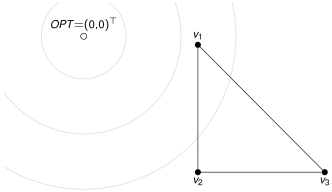
with $\mathbf{v}_1$ best point, $\mathbf{v}_{d+1}$ worst point.

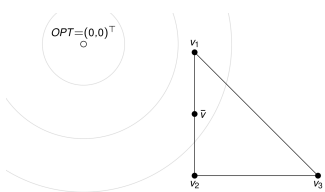


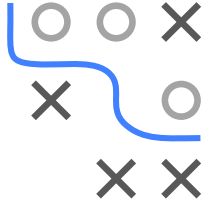
NELDER-MEAD METHOD

❷ Compute **centroid** without worst point

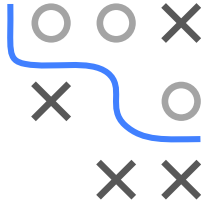
$$\bar{\mathbf{v}} = \frac{1}{d}\sum_{i=1}^{d} \mathbf{v}_i.$$







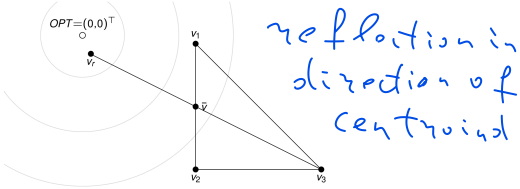
NELDER-MEAD METHOD



❸ **Reflection:** Compute reflection point

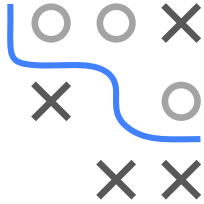
$$\mathbf{v}_r = \bar{\mathbf{v}} + \rho(\bar{\mathbf{v}} - \mathbf{v}_{d+1}),$$

with $\rho > 0$. Compute $f(\mathbf{v}_r)$.



Note: Default value for reflection coefficient: $\rho = 1$

NELDER-MEAD METHOD



Distinguish three cases:

- **Case 1:** $f(\mathbf{v}_1) \leq f(\mathbf{v}_r) < f(\mathbf{v}_d)$

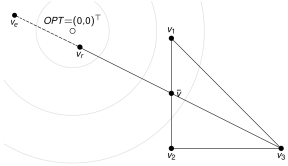
  $\Rightarrow$ Accept $\mathbf{v}_r$ and discard $\mathbf{v}_{d+1}$

- **Case 2:** $f(\mathbf{v}_r) < f(\mathbf{v}_1)$

  $\Rightarrow$ **Expansion:**

  $$\mathbf{v}_e = \bar{\mathbf{v}} + \chi(\mathbf{v}_r - \bar{\mathbf{v}}), \quad \chi > 1.$$

  We discard $\mathbf{v}_{d+1}$ and except the better of $\mathbf{v}_r$ and $\mathbf{v}_e$.



Note: Default value for expansion coefficient: $\chi = 2$

if good direction try to go more

NELDER-MEAD METHOD

- **Case 3:** $f(\mathbf{v}_r) \geq f(\mathbf{v}_d)$

  $\Rightarrow$ **Contraction:**

  $$\mathbf{v}_c = \bar{\mathbf{v}} + \gamma(\mathbf{v}_{d+1} - \bar{\mathbf{v}})$$

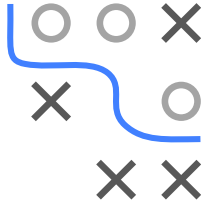
  with $0 < \gamma \leq 1/2$.

  - If $f(\mathbf{v}_c) < f(\mathbf{v}_{d+1})$, accept $\mathbf{v}_c$.
  - Otherwise, shrink **entire** simplex (**Shrinking**):

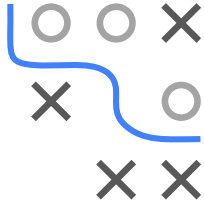
  $$\mathbf{v}_i = \mathbf{v}_1 + \sigma(\mathbf{v}_i - \mathbf{v}_1) \quad \forall i$$

  Note: Default values for contraction and shrinking coefficient: $\gamma = \sigma = 1/2$

4. **Repeat** all steps until stopping criterion met.



NELDER-MEAD



Advantages:

- No gradients needed
- Robust, often works well for non-differentiable functions.

Drawbacks:

- Relatively slow (not applicable in high dimensions)
- Not each step improves solution, only mean of corner values is reduced.
- No guarantee for convergence to local optimum / stationary point.

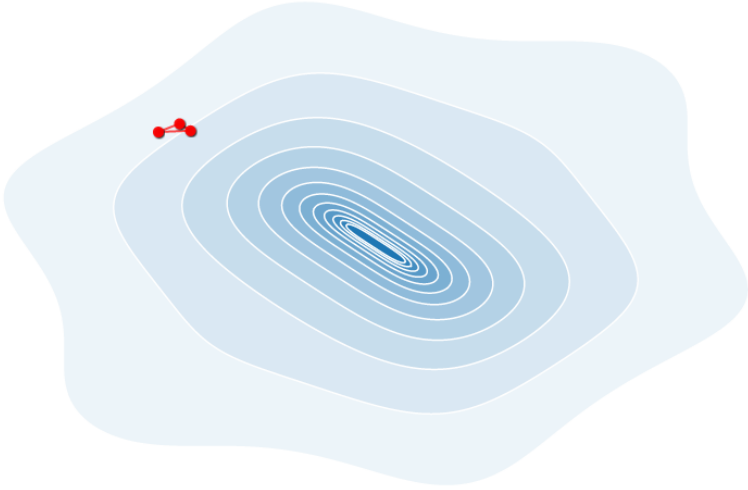
Visualization:

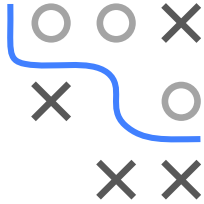
`http://www.benfrederickson.com/numerical-optimization/`

Note: Nelder-Mead is default method of `R` function `optim()`. If gradient is available and cheap, L-BFGS is preferred.

NELDER-MEAD VISUALIZATION IN 2D

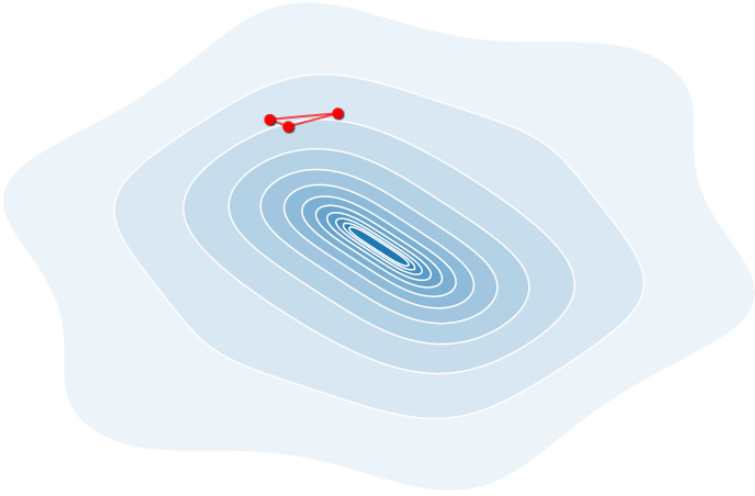
$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$

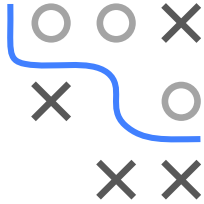




NELDER-MEAD VISUALIZATION IN 2D

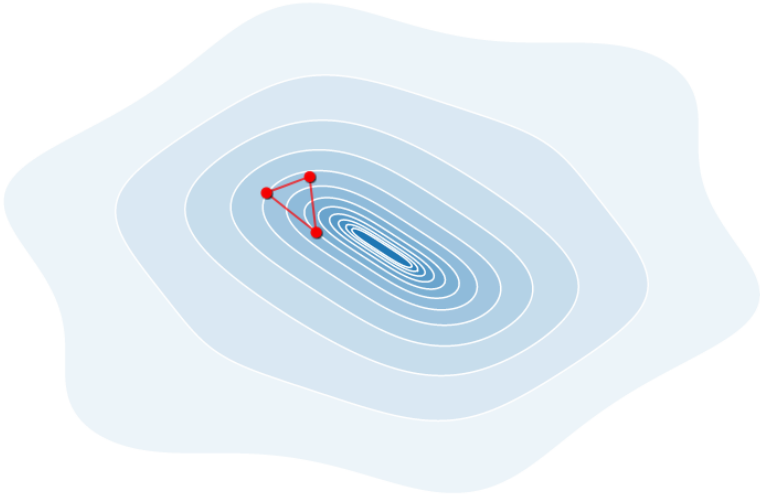
$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$

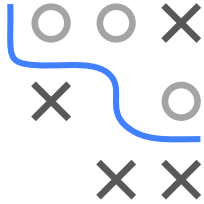




NELDER-MEAD VISUALIZATION IN 2D

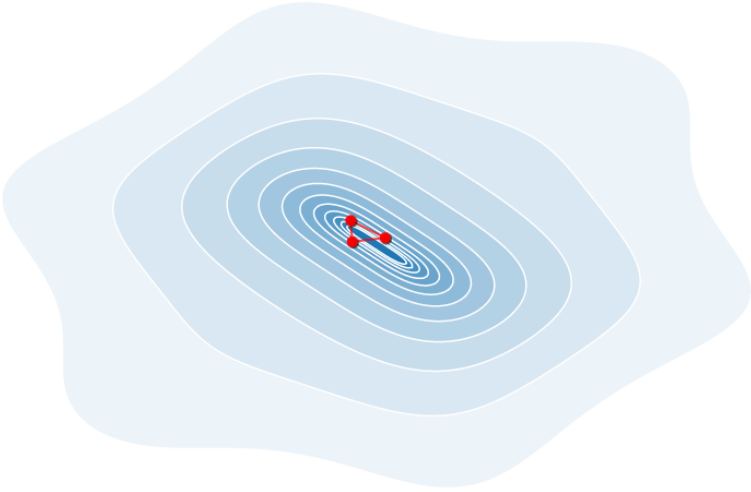
$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$

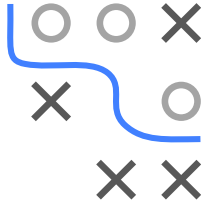




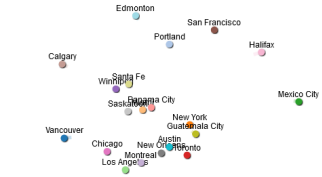
NELDER-MEAD VISUALIZATION IN 2D

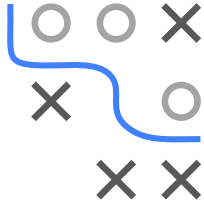
$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$





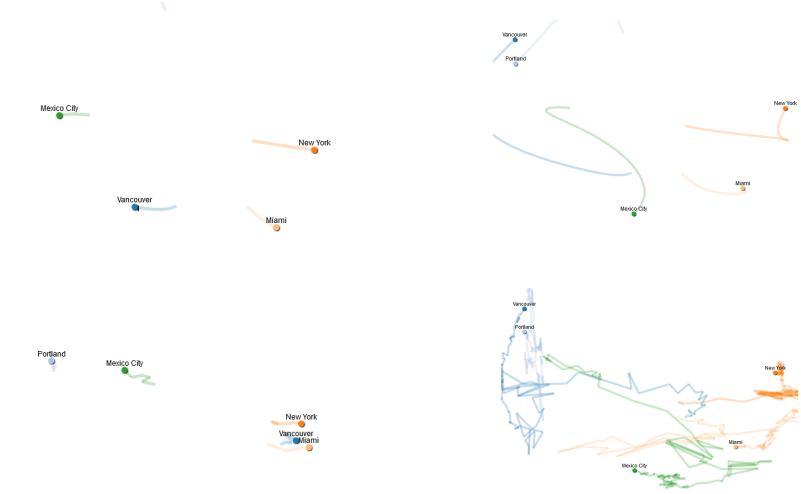
NELDER-MEAD VS. GD

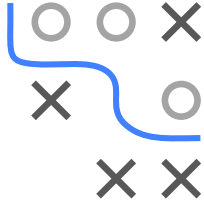




Nelder-Mead in multiple dimensions: Organize points (US cities) to keep predefined mutual distances. For 10 cities, gradient descent (top) converges well for a suitable learning rate. Nelder-Mead (bottom) fails to converge, even after many iterations.

NELDER-MEAD VS. GD

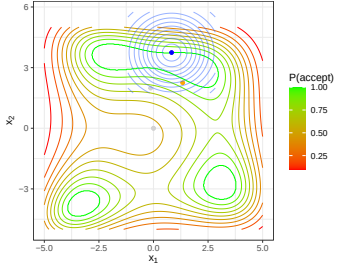




Even for only 5 cities, Nelder-Mead (bottom) performs poorly. However, gradient descent (top) still works.

Optimization in Machine Learning

Simulated Annealing

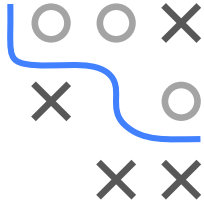


Learning goals

- Motivation
- Metropolis algorithm
- Simulated Annealing

INTRODUCTION

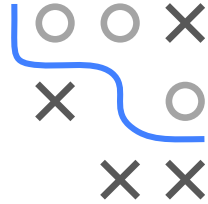
Heuristics for the optimization of complex (multivariate, non-linear, non-convex) objective functions

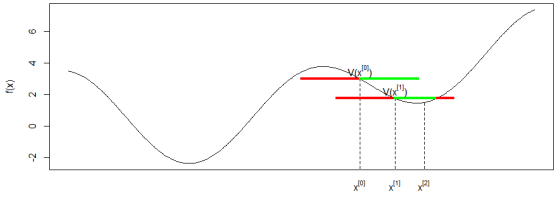


- Procedure for finding good solutions to complex problems.
- Does not guarantee optimal/best result (global optimum), but usually good solutions.
- Goal for complex optimization problems: avoid “getting stuck” in local optima.
- Is often used for difficult discrete problems as well.
- Local search strategy with random option to accept worse values.

SIMPLE STOCHASTIC LOCAL SEARCH

- Given is a multivariate objective function $f(\mathbf{x})$
- Define a local neighborhood area $V(\mathbf{x})$ for a given $\mathbf{x}$
- Sample proposal $\mathbf{x}^{[t+1]}$ uniformly at random from neighborhood $V(\mathbf{x}^{[t]})$
- Calculate $f(\mathbf{x}^{[t+1]})$
- If $\Delta f = f(\mathbf{x}^{[t+1]}) - f(\mathbf{x}^{[t]}) < 0$, $\mathbf{x}^{[t+1]}$ is accepted as new solution, otherwise a new proposal from neighborhood is sampled.

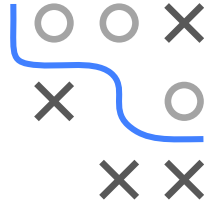




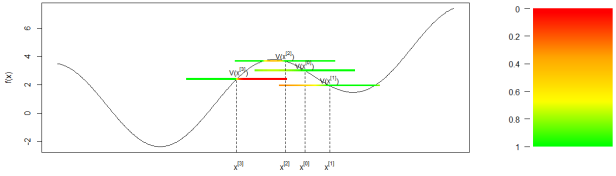
just select a random point nearby if better pick it.

Simple stochastic local search: Acceptance (green) and rejection range (red)

METROPOLIS ALGORITHM

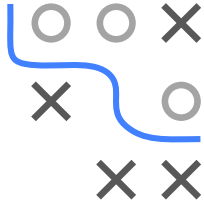


- Simple stochastic local search strongly depends on $\mathbf{x}^{[0]}$ and the neighborhood.
  $\Rightarrow$ Danger of ending up in local minima
- **Idea:** allow worse candidates with some probability
- **Metropolis:** accept candidates from previous rejection range ($\Delta f > 0$) with probability $\mathbb{P}(\text{accept} \mid \Delta f) = \exp(-\Delta f / T)$
- T denotes "temperature"



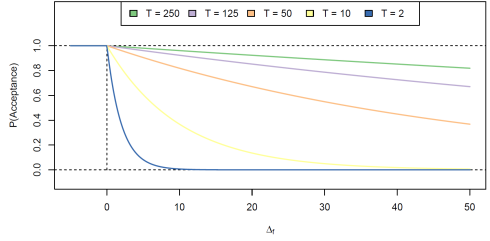
Simulated annealing: Colors correspond to $\mathbb{P}(\text{accept})$

METROPOLIS ALGORITHM

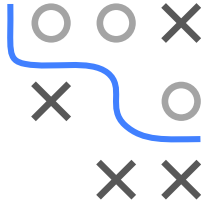


- Parameter T describes temperature/progress of the system
- High temperatures correspond to high probability of accepting worse **x**
- Local minima can be escaped, but no convergence can be achieved at *constant* temperature
- We come across an important principle of optimization:

exploration (high T) vs. exploitation (low T)



SIMULATED ANNEALING



- Start with high temperature to **explore** whole space
- Slowly reduce temperature to converge
  $\Rightarrow$ Sequence of descending temperatures $T^{[t]}, t \in \mathbb{N}$
- Procedure is called **simulated annealing**
- Temperature is often kept constant several iterations in a row to explore the space, then multiplied by coefficient $0 < c < 1$:

$$T^{[t+1]} = c \cdot T^{[t]}$$

- Other strategies possible, for example:

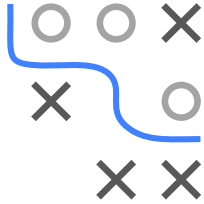
$$T^{[t]} = T^{[0]} \left(1 - \frac{t}{t_{\max}}\right)$$

Choosing neighborhood:

- Many different strategies. Strongly depends on objective function.

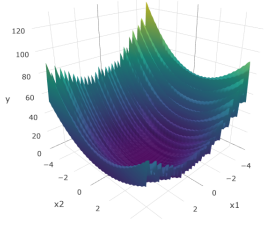
ANALOGY TO METALLURGY

- **Simulated annealing** draws analogy between a cooling process (e.g. a metal or liquid) and an optimization problem.
- If cooling of a liquid material (amount of atoms) is too fast, it solidifies in suboptimal configuration, slow cooling produces crystals with optimal structure (minimum energy stage).
- Consider atoms of the liquid as a system with many degrees of freedom, analogy to optimization problem of a multivariate function
- Minimum energy stage corresponds to optimum of objective function.



Optimization in Machine Learning

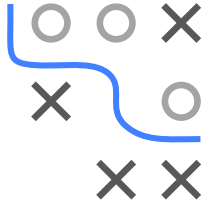
Multi-Start Optimization



Learning goals

- Multimodal functions
- Basins of Attractions
- Simple multi-start procedure

MOTIVATION



- So far: derivative-free methods for *unimodal* objective function (exception: simulated annealing)
- With multimodal objective functions, methods converge to **local minima**.
- Optimum found may differ for different starting values $\mathbf{x}^{[0]}$

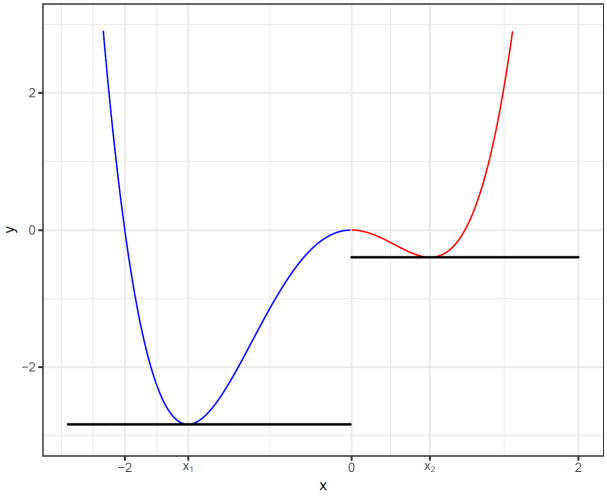
Attraction areas:

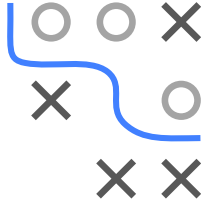
- Let $f_1^*, \ldots, f_k^*$ be local minimum values of f with $f_i^* \neq f_j^* \quad \forall i \neq j$.
- Notation: $A(\mathbf{x}^{[0]})$ denotes result of algorithm A started at $\mathbf{x}^{[0]}$
- Then: Set

$$\mathcal{A}(f_i^*, A) = \{\mathbf{x} : A(\mathbf{x}) = f_i^*\}$$

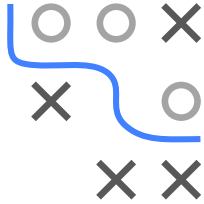
  is called *attraction area*/*basin of attraction* of f_i^* for algorithm A

ATTRACTION AREAS





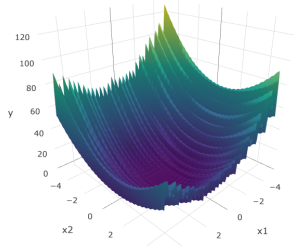
MULTI-STARTS



Levy function:

$$f(\mathbf{x}) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(2\pi x_2)]$$

- Global minimum: $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 1)^\top$
- Optimize f by BFGS method with random starting point in $[-2, 2]^2$ and collect result
- Repeat 100 times

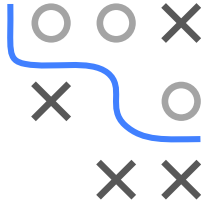


Distribution of results (y values):

```
## Min.     1st Qu.    Median    Mean     3rd Qu.    Max.
## 0.0000   0.1099     0.5356    2.4351   1.9809     18.3663
```

MULTI-STARTS

Idea: use multiple starting points $\mathbf{x}^{[1]}, \ldots, \mathbf{x}^{[k]}$ for algorithm A

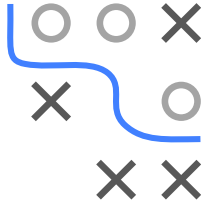


Algorithm Multistart optimization

Given: optimization algorithm $A(\cdot)$, $f : \mathcal{S} \mapsto \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$
$k = 0$
repeat
    Draw starting point $\mathbf{x}^{[k]}$ from $\mathcal{S}$ (e.g. uniform if $\mathcal{S}$ is of finite volume)
    if k = 0 **then** $\hat{\mathbf{x}} = \mathbf{x}^{[0]}$
    end if
    Initialize algorithm with start value $\mathbf{x}^{[k]} \Rightarrow \tilde{\mathbf{x}} = A(\mathbf{x}^{[k]})$
    if $f(\tilde{\mathbf{x}}) < f(\hat{\mathbf{x}})$ **then** $\hat{\mathbf{x}} = \tilde{\mathbf{x}}$
    end if
    k = k + 1
until Stop criterion fulfilled
return $\hat{\mathbf{x}}$

MULTI-STARTS

BFGS with Multistart gives us the true minimum of the Levy function:



```
iters = 20 # number of starts
xbest = c(runif(1, -2, 2), runif(1, -2, 2))

for (i in 1:iters) {
x1 = runif(1, -2, 2)
x2 = runif(1, -2, 2)
res = optim(par = c(x1, x2), fn = f, method = "BFGS")
}

if (res$value < f(xbest)) {
xbest = res$par
}

xbest
## [1] 1 1
```