Gradient Descent Method in Solving Convex Optimization Problems

Presenter: Siyuan Lin & Gregory Matesi
Department of Mathematical and Statistical Sciences
University of Colorado, Denver
Fall 2019

General Introduction

- **Taylor Series**: Let $f(x)$ be any function infinitely differentiable around $x = x_0$.

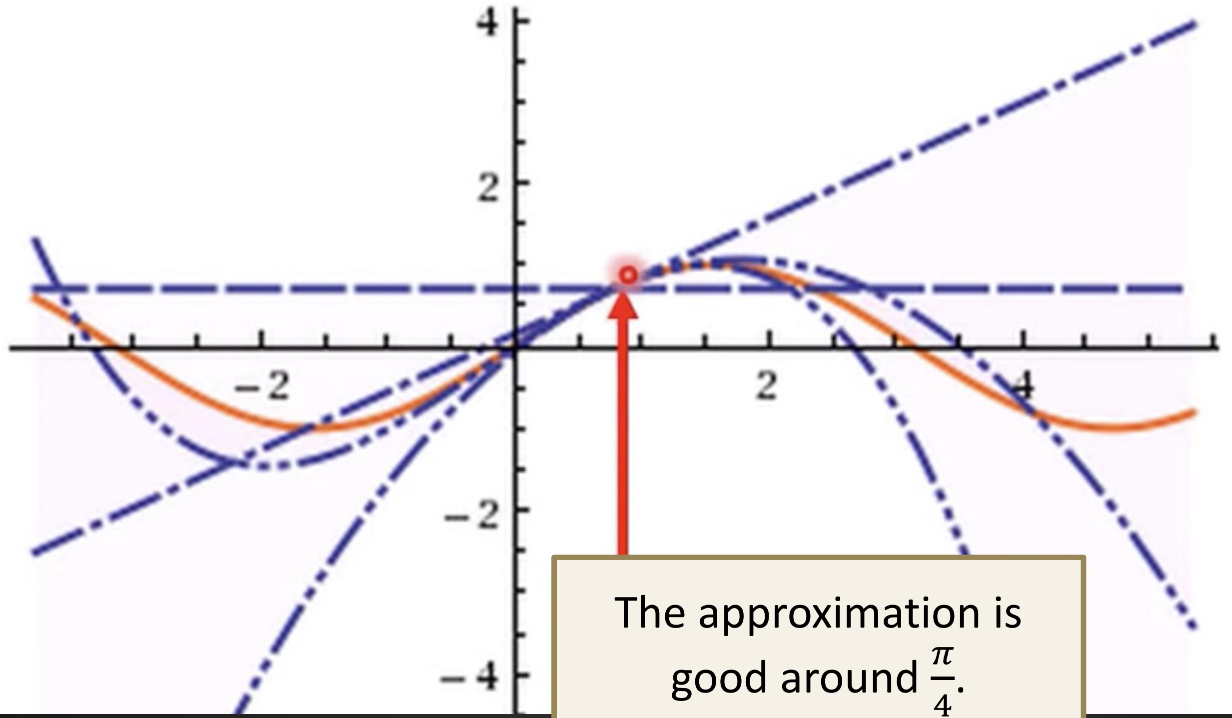
$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

$$= f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots$$

- When x is close to x_0 $\Rightarrow$ $f(x) \approx f(x_0) + f'(x_0)(x - x_0)$

E.g. Taylor series for $h(x)=\sin(x)$ around $x_0=\pi/4$

$$\sin(x)= \frac{1}{\sqrt{2}} + \frac{x-\frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^8}{40\,320\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^9}{362\,880\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^{10}}{3\,628\,800\sqrt{2}} + \ldots\ldots$$



The approximation is good around $\frac{\pi}{4}$.

Multivariable Taylor Series

- $f(x,y) = f(x_0,y_0) + \frac{\partial f(x_0,y_0)}{\partial x}(x - x_0) + \frac{\partial f(x_0,y_0)}{\partial y}(y - y_0)$

  + something related to$(x - x_0\ )^2$ and $(y - y_0\ )^2$+…

When x and y is close to x_0 and y_0

$$f(x,y) \approx f(x_0,y_0) + \frac{\partial f(x_0,y_0)}{\partial x}(x - x_0) + \frac{\partial f(x_0,y_0)}{\partial y}(y - y_0)$$

Consider the second order term, e.g. Newton’s Method

Formal Derivation

- Target Function

$$\min_x f(x)$$

- Gradient

$$\nabla f(x_0) = f'(x_0)$$

$-\nabla f(x_n)$: max-rate descending direction

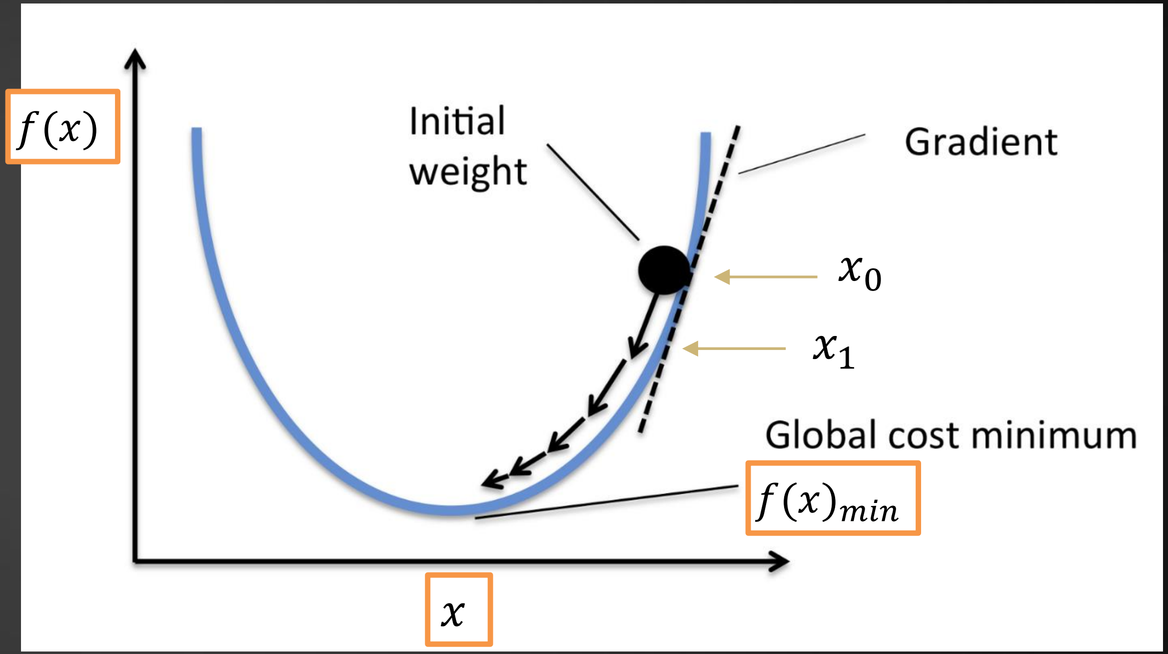
- Gradient Descent algorithm

$$x_{n+1} = x_n - \alpha_n \cdot \nabla f(x_n)$$

- Step Size

α_n(depends on the assumption of f)

Learning rate in machine learning



Pros of gradient descent

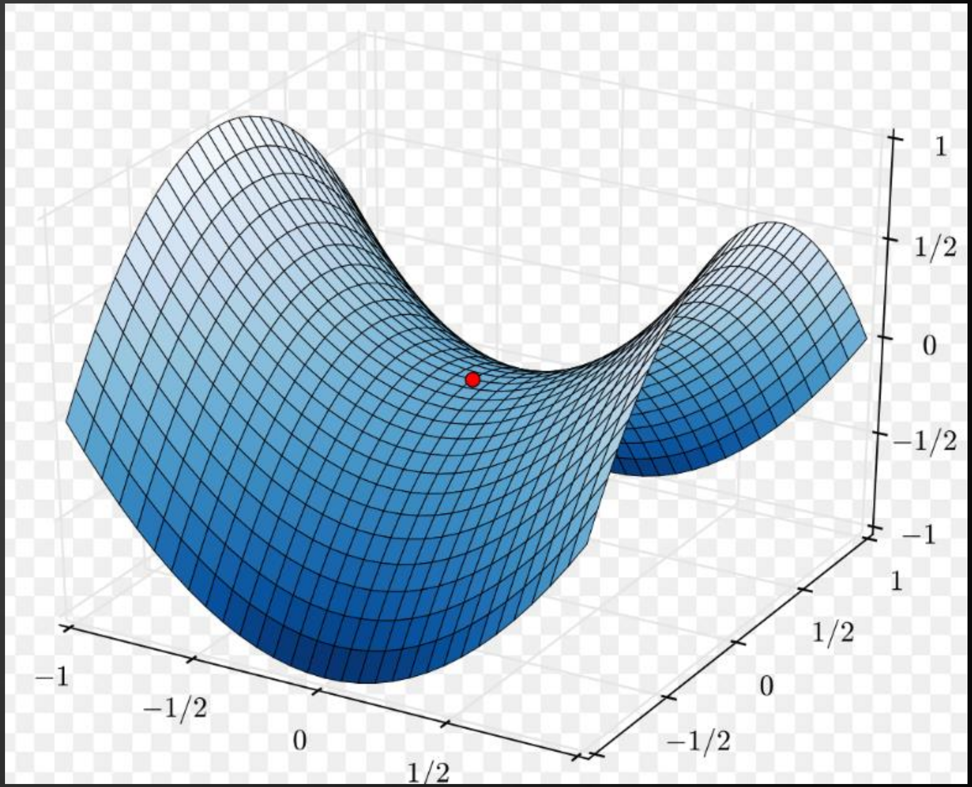
- Simple and intuitive;
  - Work under very few assumptions;
  - Work together with many other methods;

- Can be very fast for smooth objective functions,
  - i.e. well-conditioned and strongly convex.
  - when the function is convex:
    - local minima is global minima
    - converge to the global solution.

Limitations of Gradient Descent Method

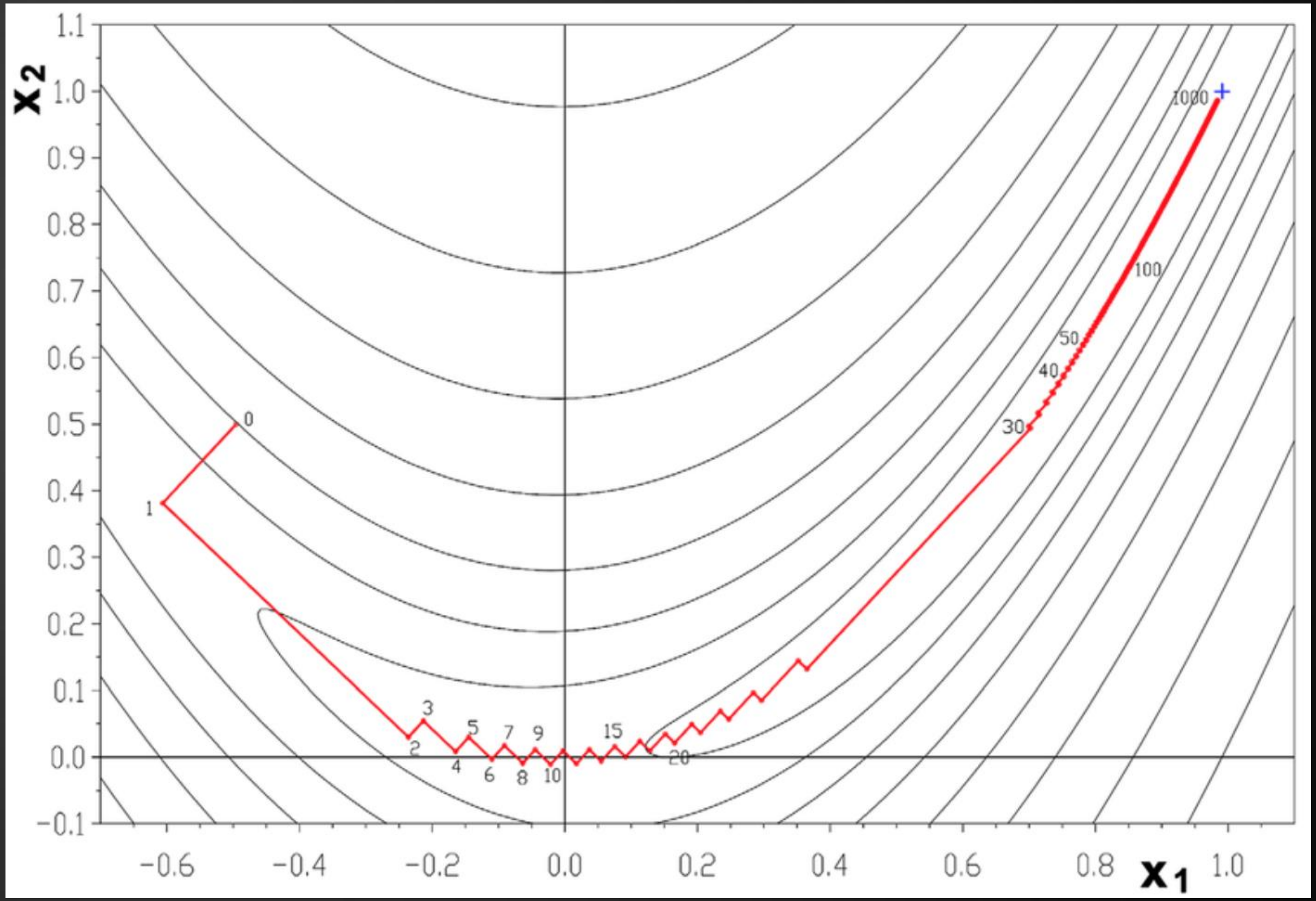
Zigzagging Issue

- When target function is non-convex
  - gradient descent increasingly 'zigzags
  - Newton‘s method is a better approach.



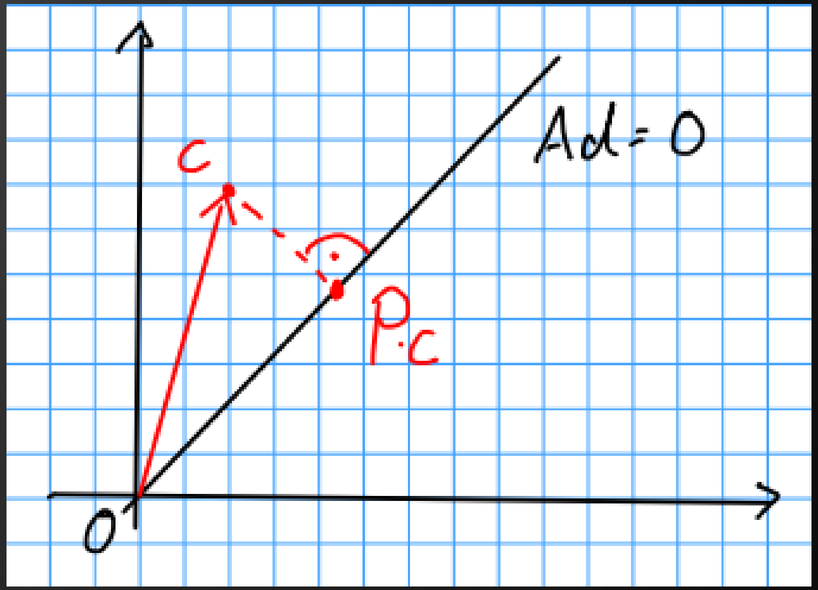
Limitations of Gradient Descent Method

- **Non-convergence Issue**
  - too small→ slow search
  - too big →cause overshooting and diverged iterations



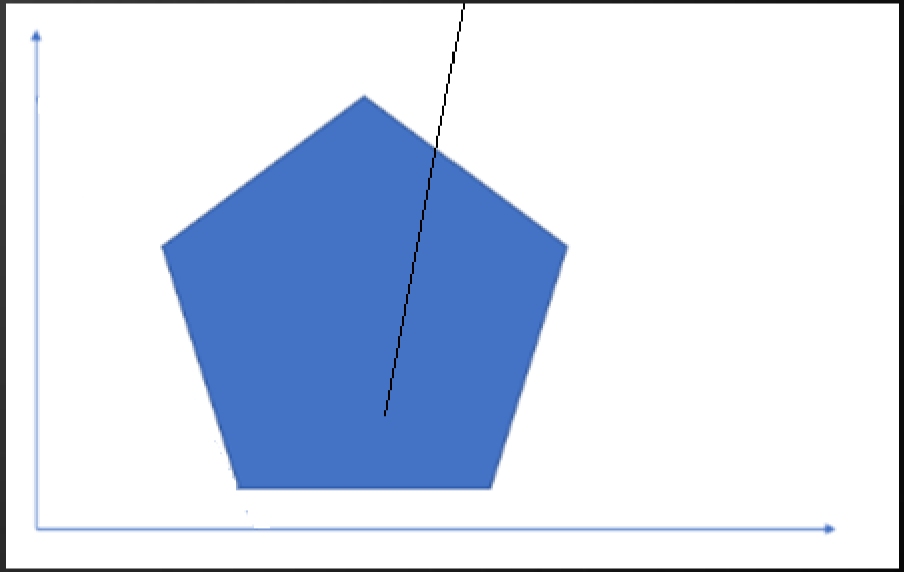
Limitations - Linear Objective function

- Project gradient into affine subspace
  - $A\Delta x = 0$
- Project gradient into first orthant
  - $c^T \Delta x \geq 0$



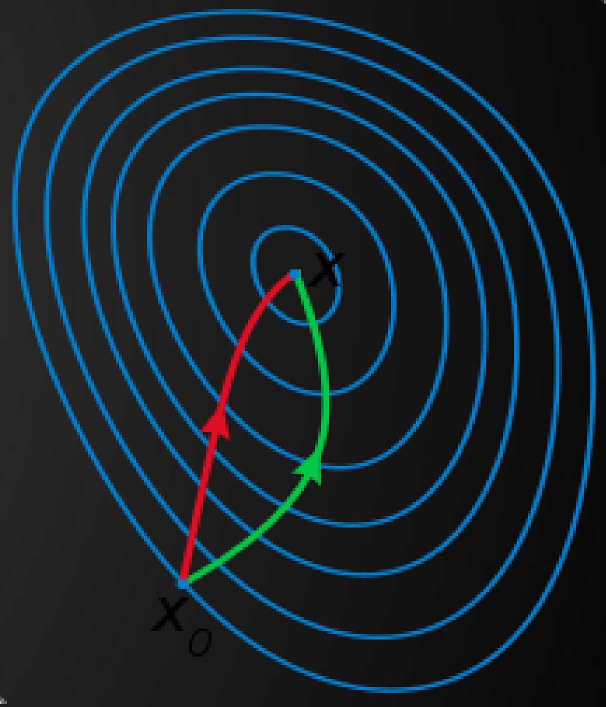
Limitations - Linear Objective function

- Gradient descent also does not perform well with a linear objective function
  - Projection of the gradient onto affine subspace
  - Projecting the gradient into the 1st orthant
    - Computationally expensive
  - Possible infeasible improving direction.



Newton Methods

- Rootfinder
  - $f'(x) = 0$
- Hessian
  - Square matrix of second order partial derivatives
  - Curvature information
- Quadratic convergence
  - In a neighborhood around a local optimum



Quadratic convergence: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec9.pdf

Newton Methods

- Derivative of second order Taylor expansion

$$f_T(x) = f_T(x_n + \Delta x) \approx f(x_n) + f'(x_n)\Delta \mathrm{x} + \frac{1}{2}\mathrm{f}''(\mathrm{x_n})\Delta \mathrm{x}^2$$

$$0 = \frac{d}{d\Delta x}\left(f(x_n) + f'(x_n)\Delta \mathrm{x} + \frac{1}{2}\mathrm{f}''(\mathrm{x_n})\Delta \mathrm{x}^2\right) = \mathrm{f}'(\mathrm{x_n}) + \mathrm{f}''(\mathrm{x_n})\Delta x$$

$$0 = \mathrm{f}'(\mathrm{x_n}) + \mathrm{f}''(\mathrm{x_n})\Delta x$$

Newton Methods

- Solve for Δx

$$\Delta x = -\frac{f'(x_n)}{f''(x_n)}$$

- Iterative process

$$x_{n+1} = x_n + \Delta x = x_n - \frac{f'(x_n)}{f''(x_n)}$$

- Multivariate

$$x_{n+1} = x_n - \gamma[Hf(x_n)]^{-1}\nabla f(x_n)$$

$$\gamma \in (0,1),\ \gamma = 1$$

Newton Methods

- Inverting Hessian can be computationally expensive
- Sometimes a surrogate will be used
  - Semi-newton methods

Implementation

- Most solvers use **semi-newton methods**
- **CONOPT** solver in AMPL.
- Generalized Reduced Gradient (GRG) algorithm
  - **Newton method** option
  - Sequential Quadratic Programming (SQP)
  - https://ampl.com/SOLVERS/conopt3.pdf

CONOPT

- Takes advantage of appearance of local linearity
  - when the function and constraint values are close to their linear approximations
  - Does not compute Jacobian for every iteration

CONOPT

- Lagrangian
  - Linear combination of the objective function with the constraints

$$\mathrm{L}(x,\lambda) = f(x) + \lambda g(x)$$

  - Hessian of the lagrangian multiplied by a vector

CONOPT

- nf: number of objective function evaluations
- ng: number of gradient evaluations
- nc: number of constraint evaluations

- nJ: number of evaluations of the Jacobian of the constraints
- nH: number of evaluations of the Hessian of the Lagrangian function
- **nHv: number of evaluations of the Hessian of the Lagrangian multiplied by a vector**

Example

- `option solver conopt, conopt_options 'lsesqp=1';`

Logical Switch Enabling SQP mode

```
conopt_options 'outlev=2'
```

```
CONOPT 3.17A: outlev=2

    The model has 5 variables and 2 constraints
    with 10 Jacobian elements, 5 of which are nonlinear.
    The Hessian of the Lagrangian has 5 elements on the diagonal,
    0 elements below the diagonal, and 5 nonlinear variables.

                   Pre-triangular equations:   0
                   Post-triangular equations:  1
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
        african  0.104805
     east_asian  0.264861
       european  0.17273
native_american  0.284324
    south_asian  0.173279
;
```

Example

```
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=0'; solve; display pi;
CONOPT 3.17A: lsesqp=0
CONOPT 3.17A: Locally optimal; objective 19.64638424
11 iterations; evals: nf = 18, ng = 11, nc = 18, nJ = 0, nH = 0, nHv = 0
pi [*] :=
         african  0.104805
      east_asian  0.264861
        european  0.17273
native_american  0.284324
     south_asian  0.173279
;

ampl:
ampl: reset; model mixtures.mod; data mixtures.dat; option solver conopt, conopt_options 'lsesqp=1'; solve; display pi;
CONOPT 3.17A: lsesqp=1
CONOPT 3.17A: Locally optimal; objective 19.64638424
10 iterations; evals: nf = 15, ng = 9, nc = 15, nJ = 0, nH = 0, nHv = 3
pi [*] :=
         african  0.104805
      east_asian  0.264861
        european  0.17273
native_american  0.284324
     south_asian  0.173279
;
```

Gradient Descent Method in Solving Optimization Problems

Example

```
param N >= 0;    # 20 genetic allele frequency observations
set ancestry;    # 5 global ancestries in the mixtures model

param a_tilde {1..N};   # 20x1 allele frequency from an observed heterogeneous ancestry
param A{1..N, ancestry};            # 20x5 matrix of allele frequencies

var pi {ancestry} >= 0; # propotion value for each ancestry in the mixtures model

minimize HA:
    sum {n in 1..N}  (sum {k in ancestry} ((A[n,k] * pi[k]) - a_tilde[n])**2) ;
    # sum of least squares mixtures model

subject to sumtoone:
    sum {k in ancestry} pi[k] = 1;  # proportions sum to 1
```

Example

```
set ancestry := african east_asian european native_american  south_asian;
param N := 20;

param a_tilde :=    1   0.66720042 2 0.17043184    3 0.06935845    4 0.42307358    5 0.77343482
                    6   0.20207072 7 0.76939016    8 0.06586837    9 0.32977999    10 0.34011927
                    11  0.21841629 12 0.09462661   13 0.11928382   14 0.70109397   15 0.60080761
                    16  0.98087062 17 0.88775104   18 0.39677821   19 0.22700411   20 0.62739267;

param A :   african     east_asian  european    native_american     south_asian     :=
        1   0.6352436   0.1385807   0.1665156   0.5659577           0.9041398
        2   0.2625085   0.2629710   0.4181271   0.7178834           0.7896494
        3   0.11973698  0.06742608  0.68754815  0.64800228          0.71269570
        4   0.1713013   0.7879743   0.4803838   0.3807989           0.9183643
        5   0.03612569  0.04574899  0.37353090  0.31219216          0.07174996
        6   0.37142610  0.08522377  0.08455566  0.52003942          0.53396940
        7   0.01295619  0.52020509  0.08344570  0.80204798          0.40205015
        8   0.2384410   0.8258287   0.4895211   0.6896276           0.7545883
        9   0.2051679   0.9381787   0.3756678   0.5106258           0.4657259
        10  0.02585700  0.21247118  0.18287401  0.55115948          0.09243381
        11  0.7119114   0.7760170   0.7929497   0.3597336           0.8863532
        12  0.005123027 0.104136581 0.666181950 0.022666776         0.745053036
        13  0.5524233   0.5716317   0.9709562   0.1433206           0.7371474
        14  0.2896559   0.8975506   0.4844245   0.9709875           0.2204392
        15  0.12560236  0.91613278  0.88561664  0.02557818          0.94784594
        16  0.8064603   0.5860706   0.3659275   0.8404793           0.3409812
        17  0.8327337   0.6031340   0.5502941   0.4890311           0.5284046
        18  0.3112420   0.7053747   0.5472243   0.1844714           0.8127638
        19  0.7856046   0.6434881   0.7781458   0.9617573           0.4538499
        20  0.2991599   0.8962980   0.8208560   0.6753767           0.1596549;
```