Stochastic Optimization, EM Algorithm

732A90
Computational Statistics

Maryna Prus
(maryna.prus@liu.se)

Slides originally by Krzysztof Bartoszek

November 30, 2021
Department of Computer and Information Science
Linköping University

Stochastic and Combinatorial Optimization

- So far: Non-linear optimization
  - *Continuous* input variables, objective function
- Now: *Combinatorial* (discrete) optimization
  - *Discrete* input variables, objective function

    Example: Traveling Salesman Problem (TSP)

    $\rightarrow$ *see next slides*
- So far: Deterministic optimization methods
- Now: *Stochastic* optimization methods

Stochastic and Combinatorial Optimization

Given a *(discrete)* set of states S, find

$$\min_{s \in S} f(s)$$

- Exhaustive search sometimes possible
  - shortest path algorithm
- Often exhaustive search computationally expensive
  - large S, TSP, ...
- Alternative:

  Random search (stochastic methods)

  - Simulated annealing
  - Genetic algorithms
  - ...

Traveling Salesman Problem (TSP)

- Given
  - List of cities / towns:

    $\{c_1, c_2, \ldots, c_t\}$ or (simpler)

    $\{1, 2, \ldots, t\}$
  - Distances between each pair of cities:

    $d_{11}, d_{12}, \ldots d_{1t}, d_{22}, \ldots d_{2t}, \ldots, d_{t-1,t}$
- To find
  - *Shortest* possible route $s = (c_{i_1}, c_{i_2}, \ldots, c_{i_t})$

    that visits *each* city exactly *once*

    and *returns* to origin city ($f(s)$ - length of route s)

Simulated Annealing: Motivation

Motivation from physics: cooling of melted metal

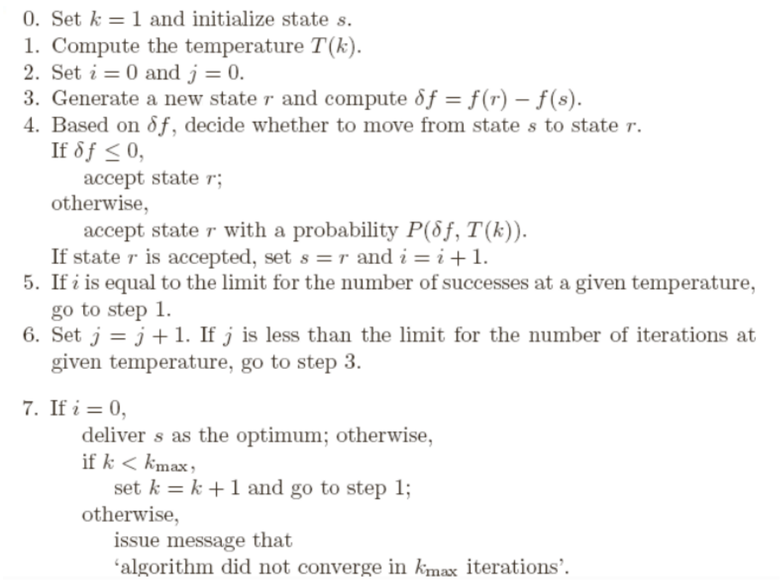
- Temperature high
  $\rightarrow$ molecules move randomly
- Temperature low
  $\rightarrow$ molecules have minimum potential energy

In algorithm:

"Temperature" (T) controls probability of moving uphill:

- "Temperature" high
  $\rightarrow$ probability of acceptance (for any point) high
- "Temperature" low
  $\rightarrow$ probability of acceptance (for any point) low
  $\rightarrow$ only downhill points accepted

Simulated Annealing: Algorithm



```
0. Set $k = 1$ and initialize state $s$.
1. Compute the temperature $T(k)$.
2. Set $i = 0$ and $j = 0$.
3. Generate a new state $r$ and compute $\delta f = f(r) - f(s)$.
4. Based on $\delta f$, decide whether to move from state $s$ to state $r$.
   If $\delta f \leq 0$,
       accept state $r$;
   otherwise,
       accept state $r$ with a probability $P(\delta f, T(k))$.
   If state $r$ is accepted, set $s = r$ and $i = i + 1$.
5. If $i$ is equal to the limit for the number of successes at a given temperature,
   go to step 1.
6. Set $j = j + 1$. If $j$ is less than the limit for the number of iterations at
   given temperature, go to step 3.

7. If $i = 0$,
       deliver $s$ as the optimum; otherwise,
       if $k < k_{\max}$,
           set $k = k + 1$ and go to step 1;
       otherwise,
           issue message that
           'algorithm did not converge in $k_{\max}$ iterations'.
```

Simulated Annealing: Details

- `https://www.youtube.com/watch?v=iaq_Fpr4KZc`
- Generating new state:
  - Continuous input: choose new point at (random) distance from current one
  - Discrete input: similar or some rearrangement
- Selection probability: e.g

  $$P(\delta f(x), T(k)) = \exp(-\delta f(x)/T(k))$$

  decreasing with $f(x)$, increasing with $T(k)$
- Temperature function: constant, proportional to k, or

  $$T(k+1) = b(k)T(k), \quad b(k) = (\log(k))^{-1}$$

Simulated Annealing: TSP Example

Assume constant temperature

```
1:  Choose initial configuration $(c_1, \ldots, c_n)$
2:  $k = 1$
3:  **while** $k < k_{max} + 1$ **do**
4:      Generate new configuration by rearrangement
            $$(1,2,3,4,5,6,7,8,9) \quad \rightarrow \quad (1,6,5,4,3,2,7,8,9)$$
5:      Measure difference in path length $(\delta f)$ between old and
        new configuration
6:      **if** shorter path found **then**
7:          accept it
8:      **else**
9:          accept it with probability $P(\delta f)$, i.e.
                $$\text{if } P(\delta f) \geq U,\ U \text{ from Unif}(0,1)$$
10:     **end if**
11:     $k = k + 1$
12: **end while**
```

Genetic Algorithm: Idea

- Idea from evolutionary theory: survival of the fittest
- States s = genotypes (genetic codes) $\rightarrow$ organisms
- State space S = population of organisms (all genetic codes)
- Objective function $f(s)$ = fitness of organism

New points are obtained from old points by recombination
$\rightarrow$ *crossover*, *mutation*, etc (see next slide)

Population only retains the fittest organisms
$\rightarrow$ with better objective function

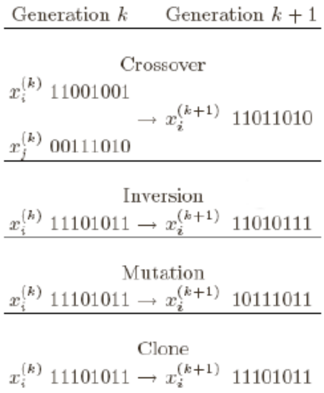
`https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications`

Genetic Algorithm

Encoding points

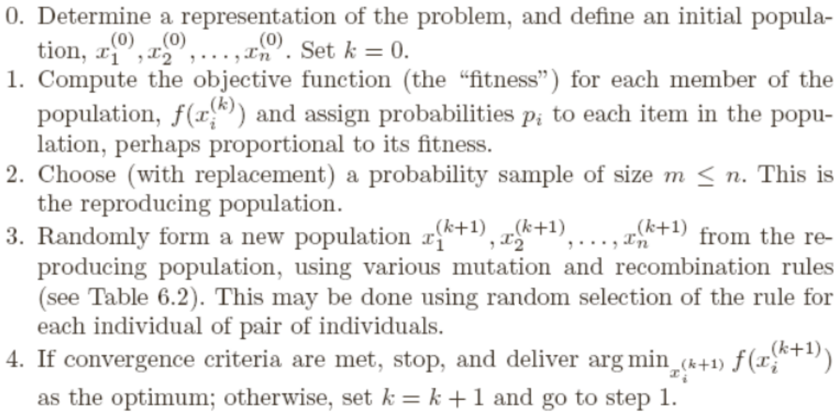
1. Enumerate each element of state space S
2. Code for state i - binary representation of i (or other)

Recombination rules:



Generation k	Generation $k+1$
Crossover	
$x_i^{(k)}$ 11001001	
	$\rightarrow x_i^{(k+1)}$ 11011010
$x_j^{(k)}$ 00111010	
Inversion	
$x_i^{(k)}$ 11101011 $\rightarrow$	$x_i^{(k+1)}$ 11010111
Mutation	
$x_i^{(k)}$ 11101011 $\rightarrow$	$x_i^{(k+1)}$ 10111011
Clone	
$x_i^{(k)}$ 11101011 $\rightarrow$	$x_i^{(k+1)}$ 11101011

Genetic Algorithm



0. Determine a representation of the problem, and define an initial population, $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$. Set $k = 0$.
1. Compute the objective function (the "fitness") for each member of the population, $f(x_i^{(k)})$ and assign probabilities p_i to each item in the population, perhaps proportional to its fitness.
2. Choose (with replacement) a probability sample of size $m \leq n$. This is the reproducing population.
3. Randomly form a new population $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$ from the reproducing population, using various mutation and recombination rules (see Table 6.2). This may be done using random selection of the rule for each individual of pair of individuals.
4. If convergence criteria are met, stop, and deliver $\arg\min_{x_i^{(k+1)}} f(x_i^{(k+1)})$ as the optimum; otherwise, set $k = k + 1$ and go to step 1.

Table 6.2 $\rightarrow$ table on previous slide

Genetic Algorithm: TSP Example

Encoding:

- Encode tours as $A_1, \ldots, A_n$

Crossover:

- Parent A1: 612|5374   Parent A2: 451|3726
- Child 1: 612|3726   Child 2: 451|5374

$\rightarrow$ Problem: same city twice!

Alternative:

- Remove 612 from 4513726 $\longrightarrow$ 4537
  Remove 451 from 6125374 $\longrightarrow$ 6237
- Child 1: 6124537   Child 2: 4516237

Genetic Algorithm: Discussion

- Small population and only crossover
  $\rightarrow$ input domain limited
  $\rightarrow$ may converge to local minimum
- Large initial population $\rightarrow$ computationally heavy
- Mutations allow to explore more of S
  $\rightarrow$ jump out of local minimum
- In TSP: mutation moves city to another position in tour
- Mutation probability usually small

EM Algorithm

Model depends on *observed* (known) data $\mathbf{Y}$
and *unobserved* (latent) data $\mathbf{Z}$

Distribution of both $\mathbf{Y}$ and $\mathbf{Z}$ depends on parameters θ

AIM: Find MLE of θ

- All data known

  $\rightarrow$ Apply unconstrained optimization (Lecture 2)
- Unobserved data

  $\rightarrow$ **EM algorithm**

EM Algorithm

Let

$$Q(\theta, \theta^k) = \int \log p(\mathbf{Y}, \mathbf{z}|\theta) p(\mathbf{z}|\mathbf{Y}, \theta^k) \mathrm{d}\mathbf{z} = \mathrm{E}\left[\mathrm{loglik}(\theta|\mathbf{Y}, \mathbf{Z})|\theta^k, \mathbf{Y}\right]$$

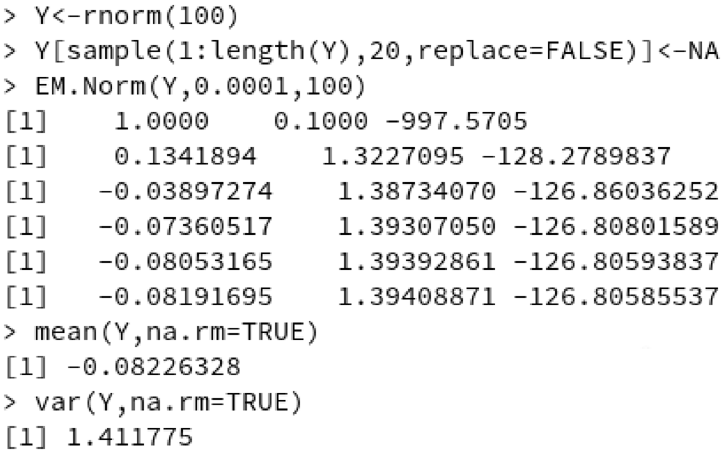
```
1: $k = 0,\ \theta^0 = \theta^0$
2: **while** Convergence not attained **and** $k < k_{max} + 1$ **do**
3:     **E–step**: Derive $Q(\theta, \theta^k)$
4:     **M–step**: $\theta^{k+1} = \mathrm{argmax}_\theta \ \ Q(\theta, \theta^k)$
5:     $k + +$
6: **end while**
```

Example: Normal data with missing values (but here analytical approach is also possible)

EM Algorithm: R

```r
floglik<-function(y, mu, sigma2, n){ -0.5*n*log(2*pi*sigma2)-0.5*sum
     ((y-mu)^2)/sigma2}

EM.Norm<-function(Y,eps,kmax){
     Yobs <- Y[!is.na(Y)];Ymiss <- Y[is.na(Y)]
     n <- length(c(Yobs, Ymiss));r <- length(Yobs)

     k<-1;muk<-1;sigma2k<-0.1

     llvalprev<-floglik(Yobs,muk,sigma2k,r);
     llvalcurr<-llvalprev+10+100*eps
     print(c(muk,sigma2k,llvalcurr))

     while ((abs(llvalprev-llvalcurr)>eps) && (k<(kmax+1))){
          llvalprev<-llvalcurr
          ## E-step
          EY<-sum(Yobs)+(n-r)*muk
          EY2<-sum(Yobs^2)+(n-r)*(muk^2+sigma2k)

          ## M-step
          muk<-EY/n
          sigma2k<-EY2/n-muk^2

          ## Compute log-likelihood
          llvalcurr<-floglik(Yobs,muk,sigma2k,r)
          k<-k+1

          print(c(muk,sigma2k,llvalcurr))
     }
}
```

EM Algorithm: R



```
> Y<-rnorm(100)
> Y[sample(1:length(Y),20,replace=FALSE)]<-NA
> EM.Norm(Y,0.0001,100)
[1]     1.0000     0.1000 -997.5705
[1]     0.1341894     1.3227095 -128.2789837
[1]    -0.03897274     1.38734070 -126.86036252
[1]    -0.07360517     1.39307050 -126.80801589
[1]    -0.08053165     1.39392861 -126.80593837
[1]    -0.08191695     1.39408871 -126.80585537
> mean(Y,na.rm=TRUE)
[1] -0.08226328
> var(Y,na.rm=TRUE)
[1] 1.411775
```

EM Algorithm: Applications

Mixture models

Z - latent variable: $P(Z = k) = \pi_k$

- Mixed data comes from different sources

  e.g. for regression, classification

- Clustering
  - Density in each cluster is normal *(Gaussian mixtures)*
  - Cluster label is latent

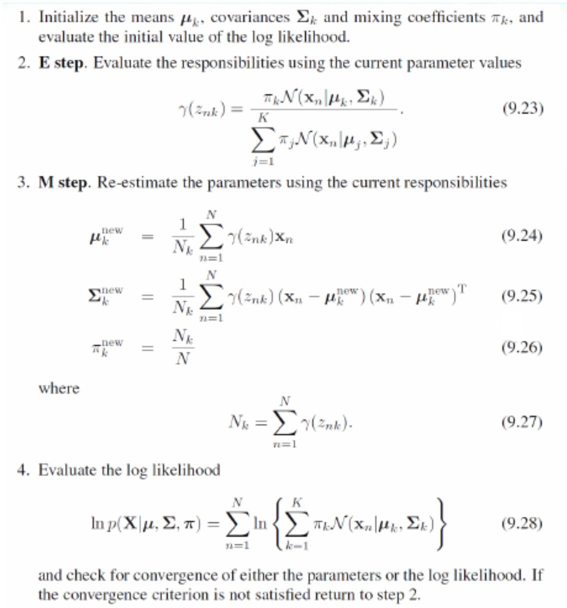
    $\rightarrow$ we do not know from which cluster is the observation
  - Mixture density

$$p(x) = \sum_{k=1}^{K} \pi_k \mathrm{N}\left(x|\vec{\mu}_k, \mathbf{\Sigma}_k\right)$$

Direct MLE leads to numerical problems

Introduce latent class variables and use EM

EM Algorithm: Gaussian Mixtures



1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

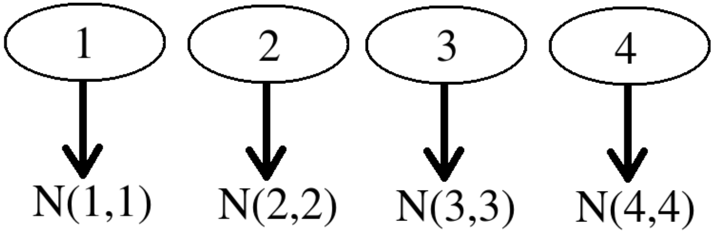
$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

$$E z_{nk} = \gamma(z_{nk})$$

Source: Pattern recognition by Bishop

Gaussian Mixtures: Example



$P(1) = P(2) = P(3) = P(4) = 0.25$

1. draw class $Z \in \{1,2,3,4\}$ uniformly
2. draw normal distribution $\mathcal{N}(Z,Z)$ with density $\phi_{Z,Z}(\cdot)$

We can write the mixture density as

$$f(x) = 0.25\phi_{1,1}(x) + 0.25\phi_{2,2}(x) + 0.25\phi_{3,3}(x) + 0.25\phi_{4,4}(x).$$

Summary

Computational solutions for

- Combinatorial / discrete optimization
  - Simulated annealing
  - Genetic algorithm
- MLE (in case of latent variables)
  - EM algorithm

...

Thank you for attention!