EM Algorithm, Stochastic Optimization

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

12 XII 2022 (R41)
Department of Computer and Information Science
Linköping University

Stochastic and combinatorial optimization

- So far: Unconstrained optimization
  - Predictor variables are continuous
  - Response function is differentiable
- We discussed Steepest descent, Newton, BFGS, CG
- But: predictors can be discrete
  (scheduling problems, travelling salesman)
- But: outcome can be discrete, noisy or multi–modal

Stochastic and combinatorial optimization

Given a (large) set of states S, find

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

- Exhaustive search (shortest path algorithm)
- Often exhaustive search is NP–hard (TSP)
- Alternative: stochastic methods
  random search

Simulated annealing

Motivation from physics: cooling of metal

- Parameters:
  Energy of metal
  (decreasing, but not strictly monotonic)
  Temperature (decreasing)

- Aim: find global minimum energy

Simulated annealing

1: Initialize $k = 1$, θ_0, $T(1) = T_0$, $i = 0$
2: **while** $k < k_{max} + 1$ **do**
3: generate new state ξ, compute $\Delta f = f(\xi) - f(\theta)$
4: **if** $\Delta f < 0$ **then**
5: accept ξ $(\theta_k = \xi)$
6: **else**
7: accept ξ $(\theta_k = \xi)$ with $P(\Delta f, T(k))$
8: **end if**
9: $k = k + 1$
{WE CAN ALSO HAVE A SEARCH LOOP FOR EACH $T(k)$ level}
10: **end while**

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equations of state calculations by fast computing machines. J. Chem. Phys. 21:1087-1092, 1953.

Simulated annealing

- `https://www.youtube.com/watch?v=iaq_Fpr4KZc`

- Generating new state:
  - Continuous: choose a new point a (random) distance from the current one
  - Discrete: similar or some rearrangement

- Selection probability: e.g $\exp(-\Delta f(x)/T)$: decreasing with $f(x)$, increasing with T

- Temperature function: constant, proportional to k, or

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

Remember: A smaller value is better than one on the path to the global minimum! Always keep track of smallest found.

Simulated annealing: TSP example

Assume constant temperature

1: Choose initial configuration $(Town_1, \ldots, Town_n)$
2: $k = 1$
3: **while** $k < k_{max} + 1$ **do**
4: Generate new configuration by rearrangement,

$$(1,2,3,4,5,6,7,8,9) \rightarrow (1,6,5,4,3,2,7,8,9)$$
$$(1,2,3,4,5,6,7,8,9) \rightarrow (1,7,8,2,3,4,5,6,9)$$

5: Measure difference in path length (Δf) between old and new configuration
6: **if** shorter path found **then**
7: accept it
8: **else**
9: accept it with probability $P(\Delta f)$
10: **end if**
11: $k++$
12: **end while**

Genetic algorithm

- Inspiration from evolutionary theory: survival of the fittest
- Variables=genotypes
- Observation=organism, characterized by genetic code
- State space=population of organisms
- Objective function=fitness of organism

New points are obtained from old points by crossover and mutation, the population only retains the fittest organisms (with better objective function).

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

Genetic algorithm

Encoding points

1. Enumerate each element of the state space, S
2. Code for observation i is binary representation of i (or something else)

Mutation and recombination rules

Crossover: $(1010\ 1110, 1100\ 0110) \longrightarrow 1010\ 0110$

Inversion: $11001011 \longrightarrow 11010011$

Mutation: $11010111 \longrightarrow 1101\mathbf{1}111$

Clone: $11010111 \longrightarrow 11010111$

Genetic algorithm

1: Initialize $i = 0$, population $\mathbf{P}_0$ and calculate fitness
2: **while** end criteria not met **do**
3: Choose individuals $\mathbf{T}_i$ from $\mathbf{P}_i$ for reproduction
4: $\mathbf{T}'_i$ = crossover between individuals from $\mathbf{T}_i$
5: $\mathbf{O}_i$ = randomly inverte, mutate, clone $\mathbf{T}'_i$
6: Calculate fitness and obtain $\mathbf{P}_i$ from $\mathbf{O}_i$
7: $i = i + 1$
8: **end while**

ALWAYS KEEP BEST INDIVIDUAL

Genetic algorithm: TSP example

Encoding and crossover

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

  Parent 1: FAB|ECGD Parent 2: DEA|CGBF
  Child: FAB|CGBF Child: DEA|ECGD

Instead

1. Remove FAB from DEACGBF $\longrightarrow$ DECG.
   Child becomes FABDECG.
2. Second child will be by taking prefix from Parent 2:
   DEAFBCG

Genetic algorithm: Mutations

- If a population is small and only crossover: the input domain becomes limited and may converge to a local minimum.
- Large initial populations are computationally heavy.
- Mutations allow one to explore more of S: jump out of local minimum.
- In TSP: mutation move a city in the tour to another position.
- Reproduction: Among m tours selected at step 2, two best are selected for reproduction, two worst replaced by children.
- If m is large, some tours might never be parents, global solution may be missed. Random chance of reproduction?
- Mutation probability is usually small (unless you want to jump wildly)

EM algorithm

Fundamental algorithm of computational statistics!

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

The data's (**both** $\mathbf{Y}$'s and $\mathbf{Z}$'s) distribution depends on some parameters θ.

AIM: Find MLE of θ.

- All data is known: Apply unconstrained optimization (discussed in Lecture 2)
- Unobserved data
  - **Sometimes** it is possible to look at the marginal distribution of the observed data.
  - Otherwise: **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

732A90_ComputationalStatisticsHT2022_Lecture06codeSlide15.R

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
```

Notice: can be done by studying marginal distribution of observed data.

EM algorithm: Applications

Mixture models Z is a latent variable, $P(Z = k) = \pi_k$

- Mixed data comes from different sources (e.g. for regression, classification)

- Clustering
  1. Density in each cluster is normally distributed.
  2. Cluster label is latent (we do not know what are the chances an observation is from the given cluster)

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

Direct MLE leads to numerical problems.
Introduce latent class variables and use EM.

EM algorithm: Gaussian mixtures

1: Initialize $r = 0$, $(\vec{\mu}_1^0, \ldots, \vec{\mu}_K^0)$, $(\mathbf{\Sigma}_1^0, \ldots, \mathbf{\Sigma}_K^0)$ **input**: data $\vec{x}_1, \ldots, \vec{x}_n$; K clusters

2: **while** end criteria not met **do**

3: **E-step** ($\overline{z}_{jk}$: *responsibility* of cluster k explaining $\vec{x}_j$)

$$\overline{z}_{jk} = \frac{\mathcal{N}\left(x_j | \vec{\mu}_k^r, \mathbf{\Sigma}_k^r\right)}{\sum_{i=1}^{K} \pi_i \mathcal{N}\left(\vec{x}_j | \vec{\mu}_i^r, \mathbf{\Sigma}_i^r\right)}, \quad j = 1, \ldots, n$$

4: **M–step** for $k = 1, \ldots, K$

$$n_k = \sum_{j=1}^{n} \overline{z}_{jk}, \qquad \pi_k^r = n_k / n,$$

$$\vec{\mu}_k^r = (n_k)^{-1} \sum_{j=1}^{n} \overline{z}_{jk} \vec{x}_j, \quad \mathbf{\Sigma}_k^r = (n_k)^{-1} \sum_{j=1}^{n} \overline{z}_{jk} (\vec{x}_j - \vec{\mu}_k^r)(\vec{x}_j - \vec{\mu}_k^r)^T$$

5: evalulate log–likelihood

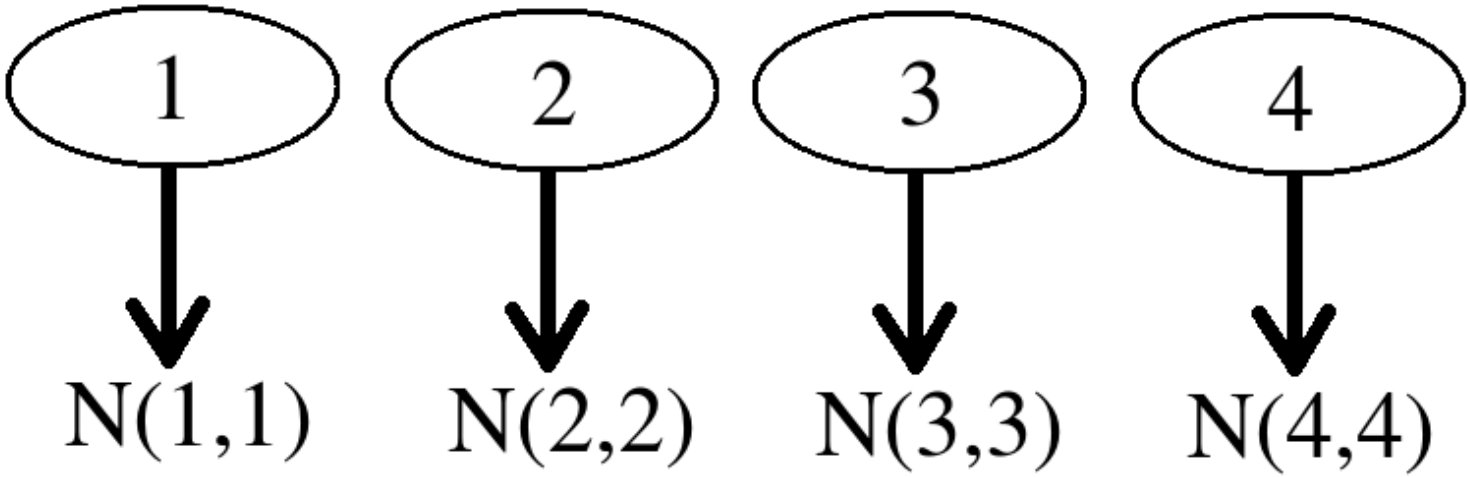
$$\log \mathcal{L}(\{\pi_k^r, \vec{\mu}_k^r, \mathbf{\Sigma}_k^r\}_{k=1}^K) = \sum_{j=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k^r \mathcal{N}\left(\vec{x}_j | \vec{\mu}_k^r, \mathbf{\Sigma}_k^r\right) \right)$$

6: $r = r + 1$

7: **end while**

See also Ch. 7.10.7, *EM for Mixture of Gaussians*, O. Simeon, *Machine Learning for Engineers*, Cambridge University Press, 2023 (there supervised log–loss optimized)

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

Random walk over the state space in search of minimum

1. Follow decreasing path
2. **BUT** with a certain probability go to higher values, to avoid local minima traps.
3. **Never forget** best found conformation!

4. Simulated annealing, Genetic algorithm, **EM algorithm**,

   Stochastic gradient descent (see 2016 slides)