GLM Extensions

Mento Carlo Optimization

Junyi Zhou

Indiana University

April 6, 2019

Outline

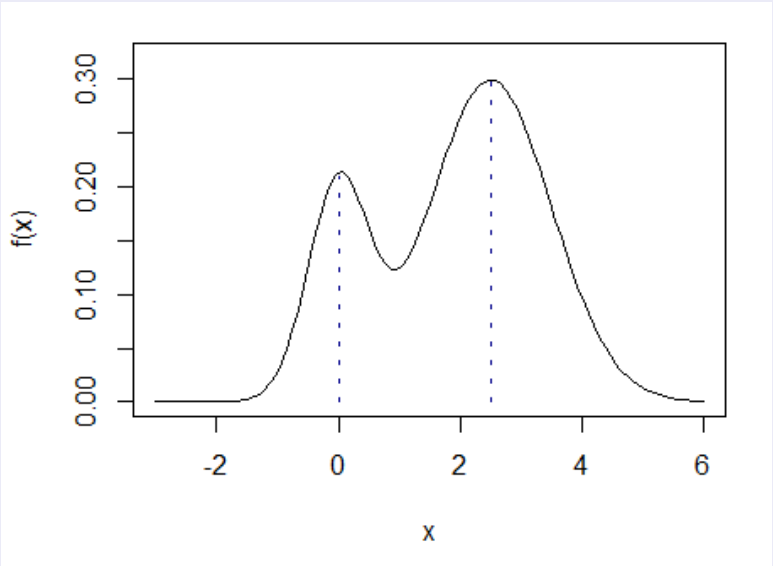
Global Example

Data Generation

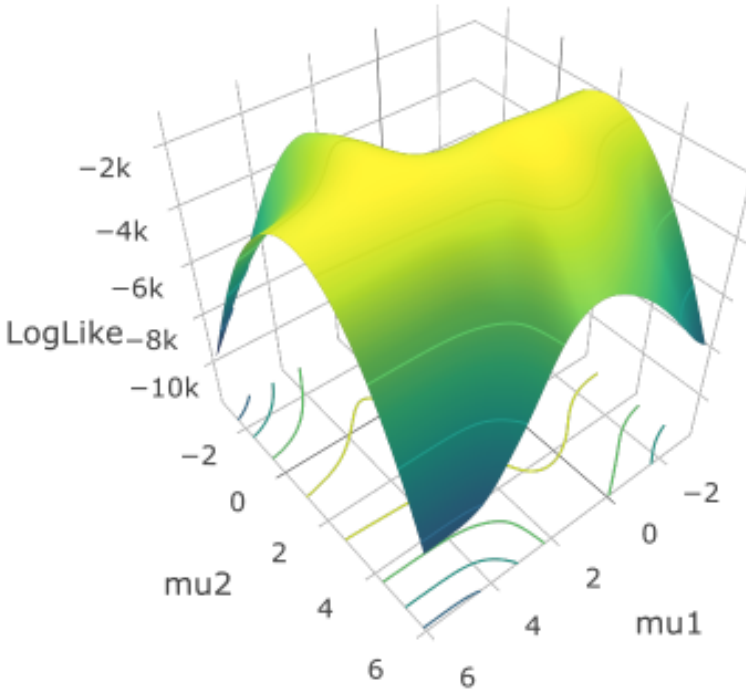
- We sample 800 data from a mixture model

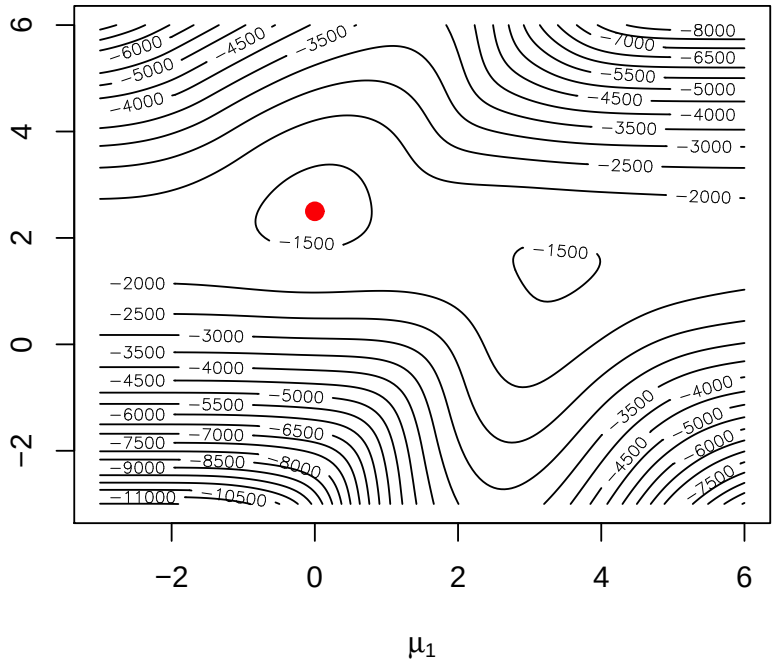
$$\frac{1}{4}\mathcal{N}(\mu_1, 0.5) + \frac{3}{4}\mathcal{N}(\mu_2, 1) \tag{1}$$

- Based on the samples, we would like to estimate μ_1 and μ_2
- Density Plot ($\mu_1 = 0, \mu_2 = 2.5$)



Global Example (Con't)





- log-Likelihood function

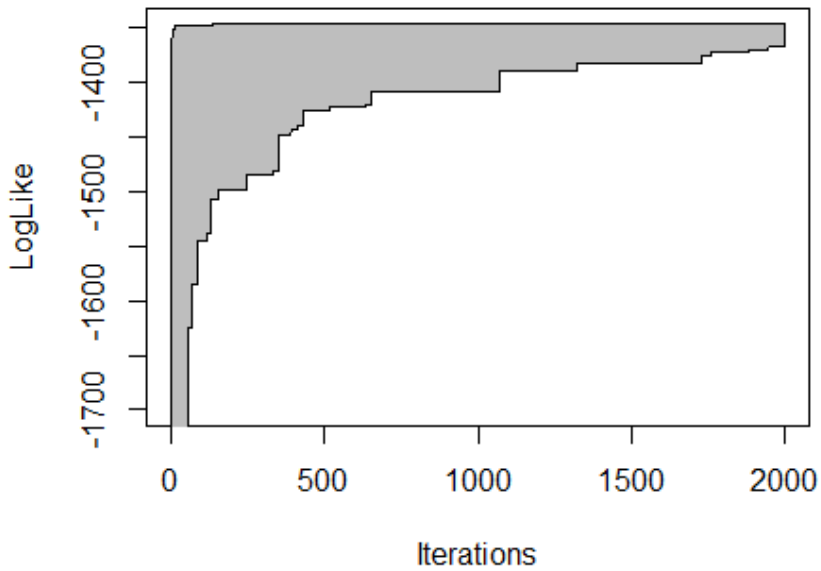
$$logL(\theta|x) \propto \sum_i log\{\pi \exp(-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}) + (1-\pi)\exp(-\frac{(x_i-\mu_2)^2}{2\sigma_2^2})\} \quad (2)$$

- Our purpose is to find out the optimum point (maximum) on the objective function (logLikelihood function $h(\theta|X)$)

Stochastic Search

Basic Solution

- Purpose is to find out the optimum
- Simulate points over Θ until sufficiently high $h(\theta)$ is observed, or reach certain stopping criterion
- Compare to grid search?
- Blind Search, i.e. does not rely on $h(\theta)$



Range of Maximum Found among 500 Trials

Stochastic Search

Gradient Descent Related Methods

- Gradient Descent (Hill-Climbing)

$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X) \tag{3}$$

- Newton-Raphson (a special case)

$$\theta^{(j+1)} = \theta^{(j)} - [\nabla^2 h(\theta^{(j)}|X)]^{-1} \nabla h(\theta^{(j)}|X) \tag{4}$$

- Stochastic Gradient Descent (SGD)

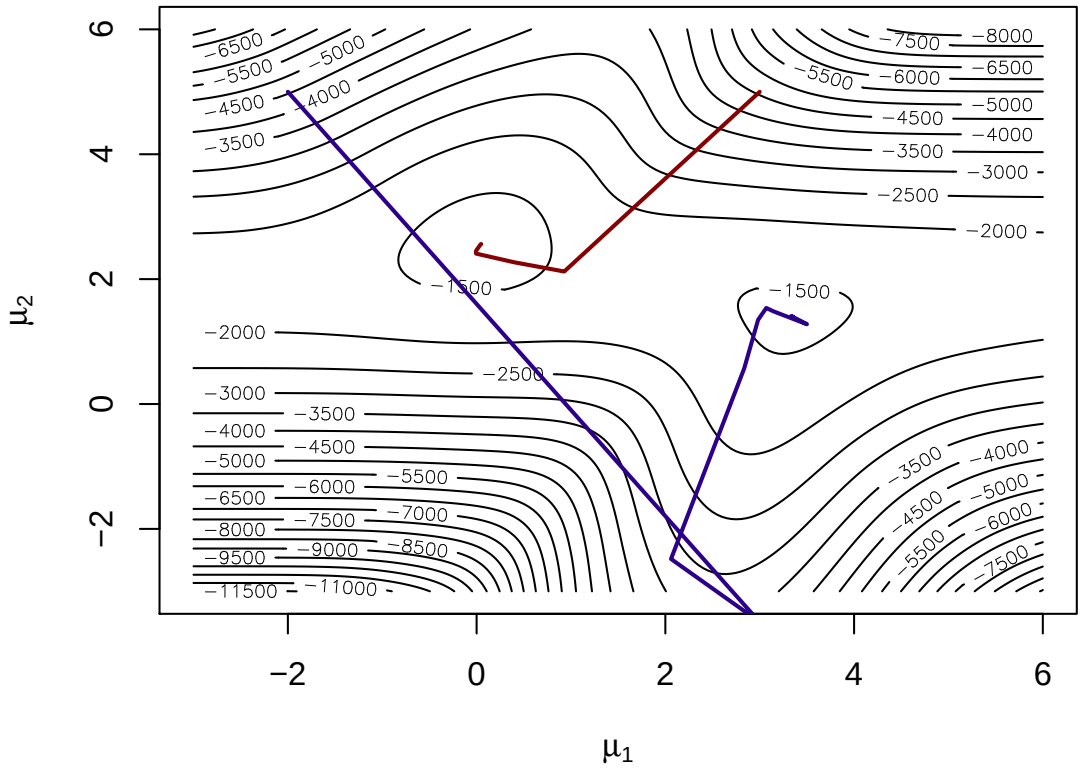
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i) \tag{5}$$

- Mini-Batch SGD

$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_{(i:i+n)}) \tag{6}$$

Newton-Raphson Solutions

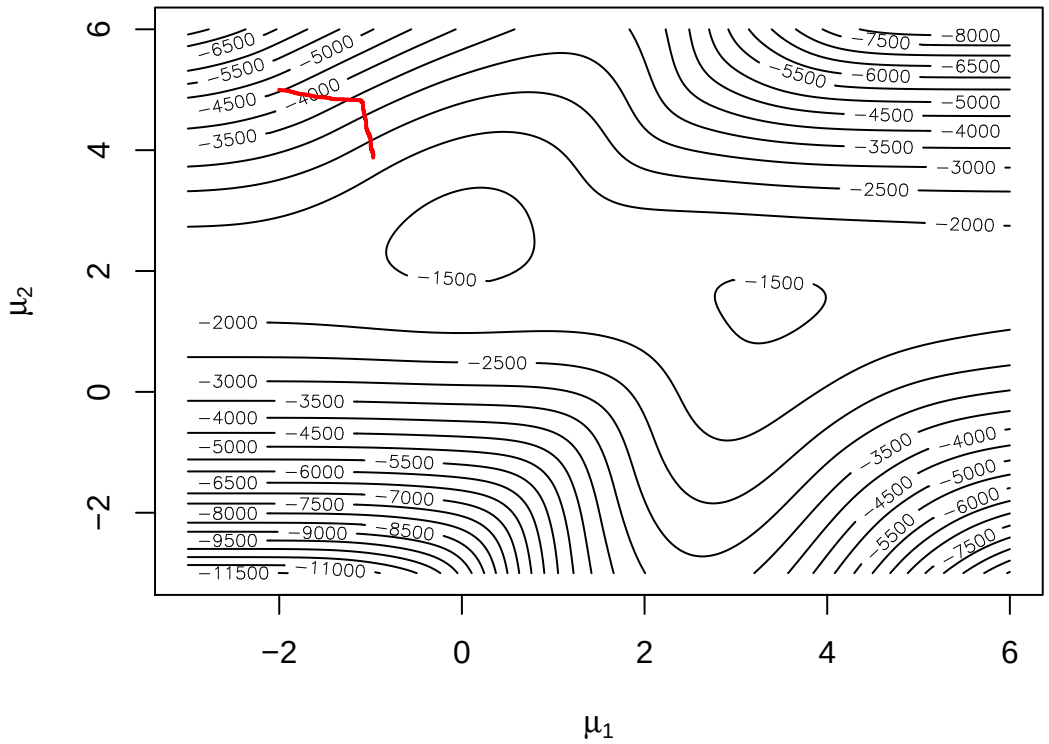
$$\theta^{(j+1)} = \theta^{(j)} - [\nabla^2 h(\theta^{(j)}|X)]^{-1} \nabla h(\theta^{(j)}|X)$$



- Fast and Aggressive (likely to diverge)
- Deterministic, i.e. path is decided given an initial point
- Get trapped at saddle points

Stochastic Gradient Descent (SGD) Solutions

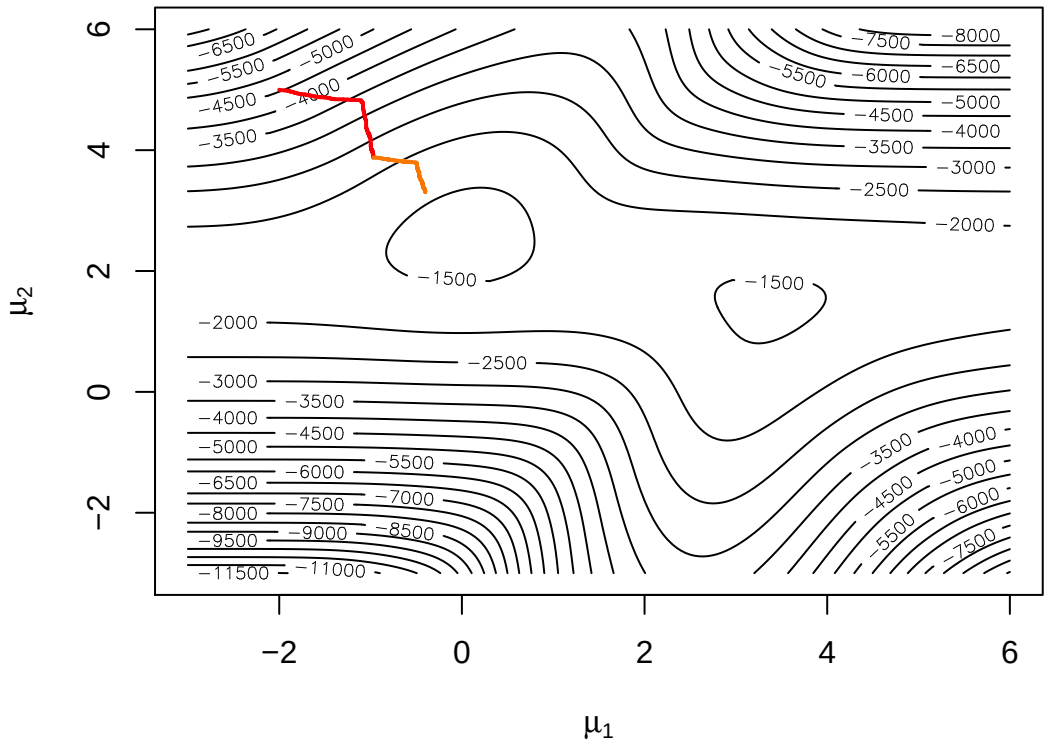
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Gradient Descent (SGD) Solutions

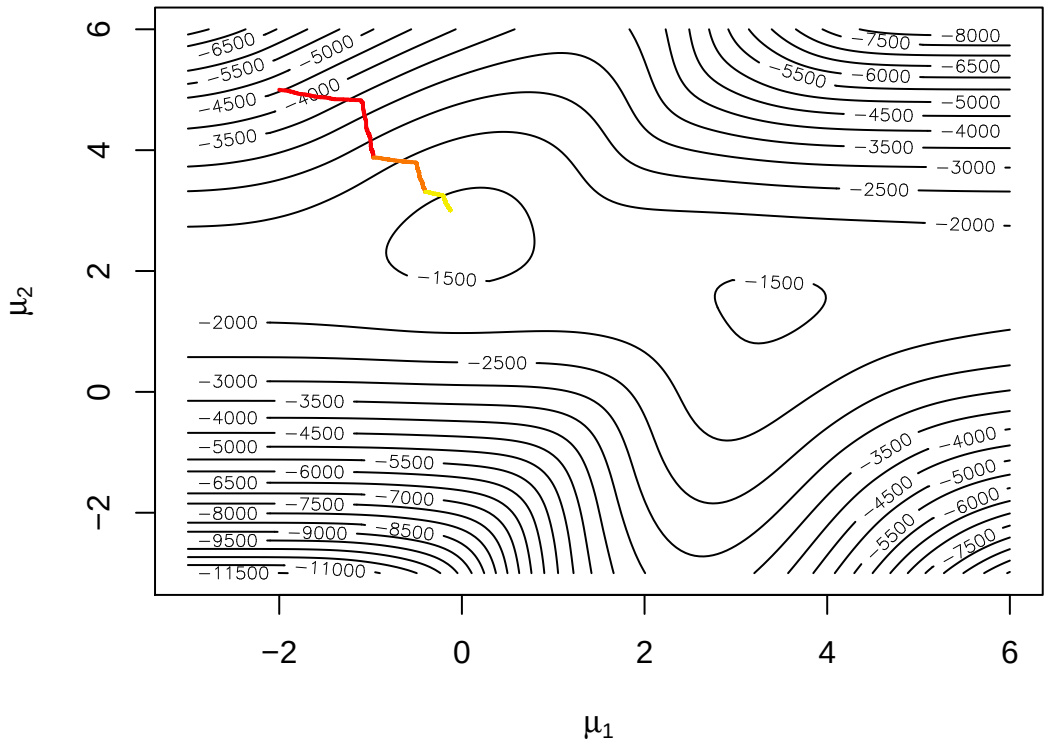
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Gradient Descent (SGD) Solutions

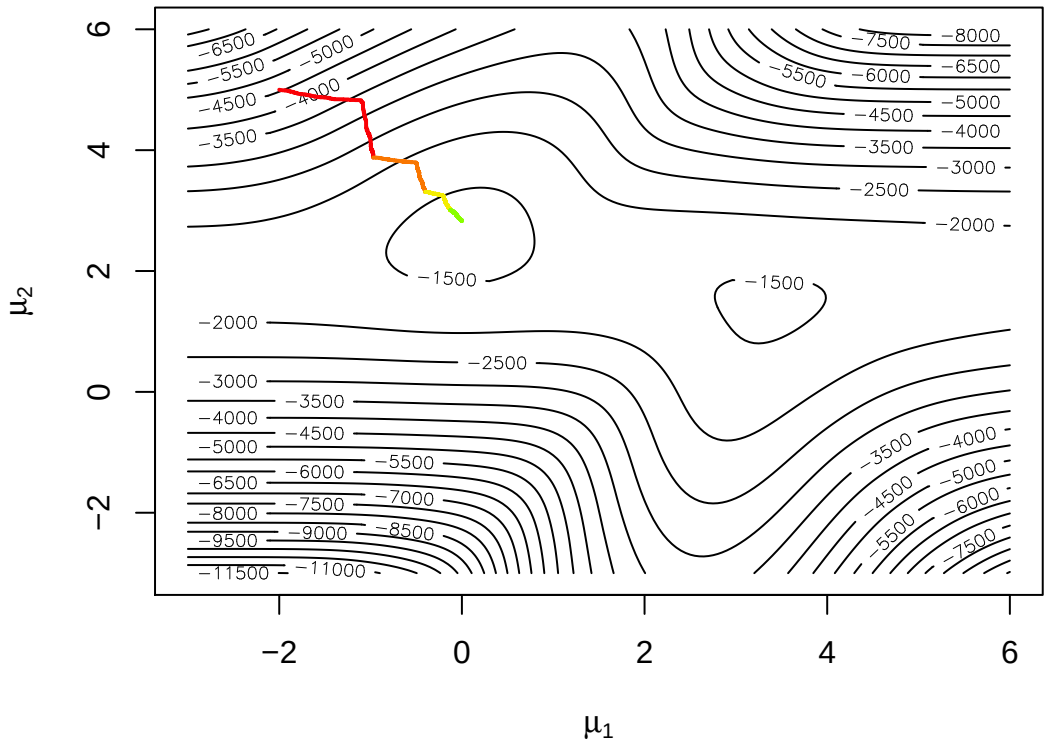
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Gradient Descent (SGD) Solutions

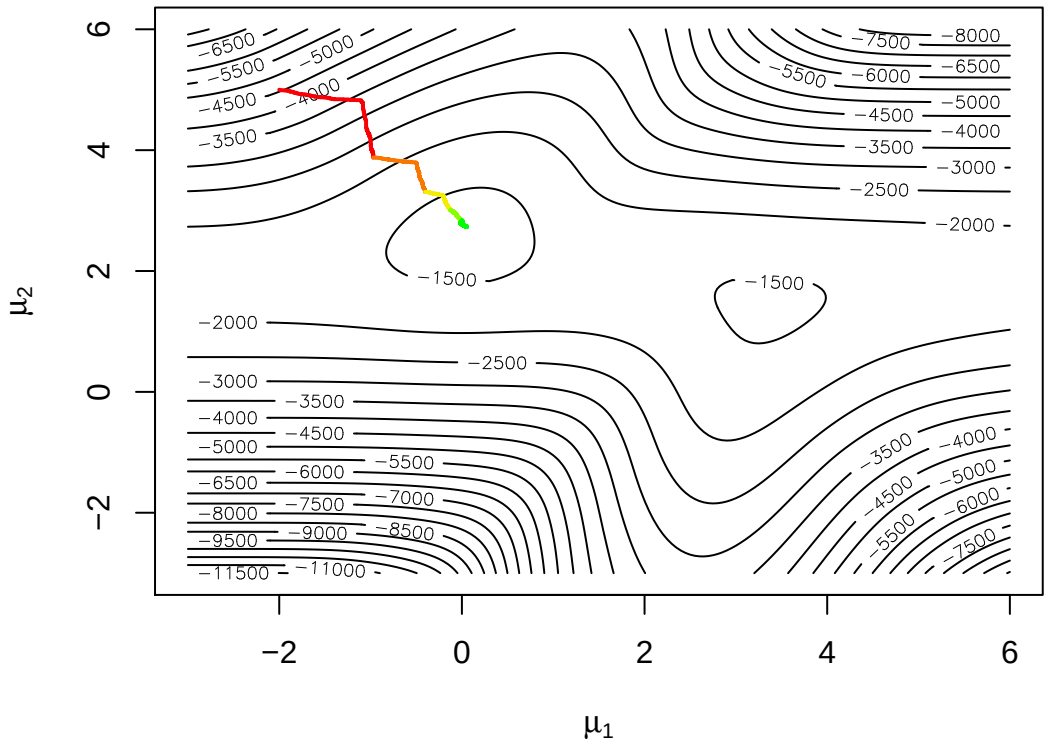
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Gradient Descent (SGD) Solutions

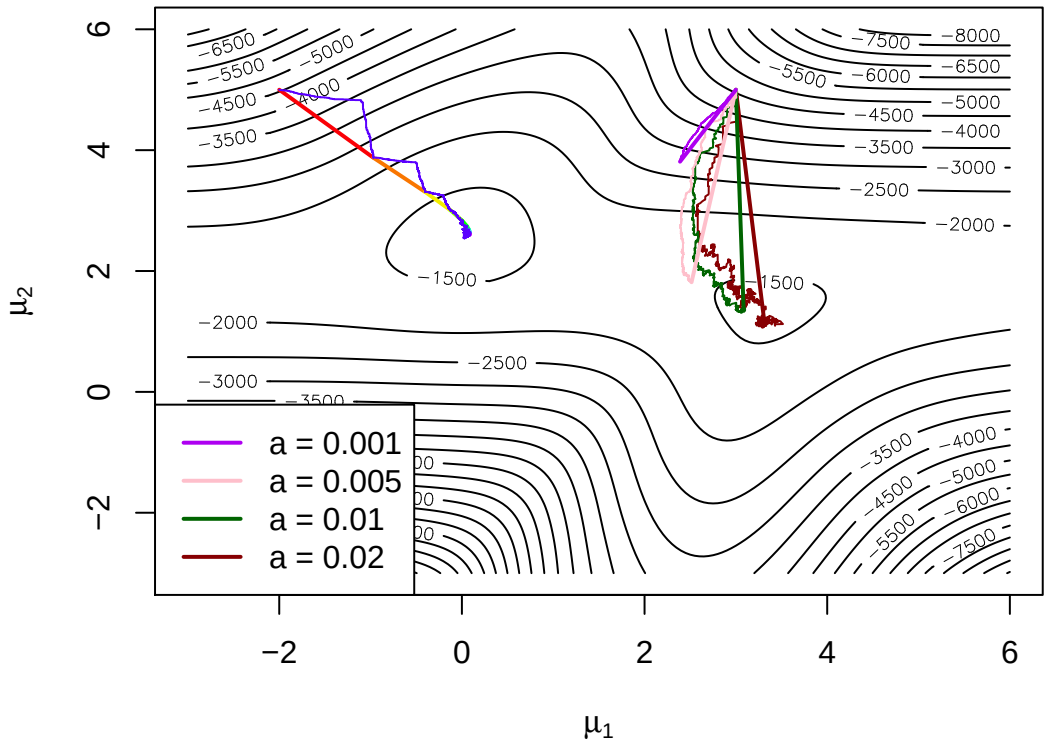
$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Gradient Descent (SGD) Solutions

$$\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X_i)$$



- Slow and Conservative (behaviour depends on α_j, how?)
- Stochasticity comes from order of input data, but path is determined once order is fixed
- Have chance to reach global optimum (same start points as in NR method)

Stochastic Search

Mini-batch SGD Solutions

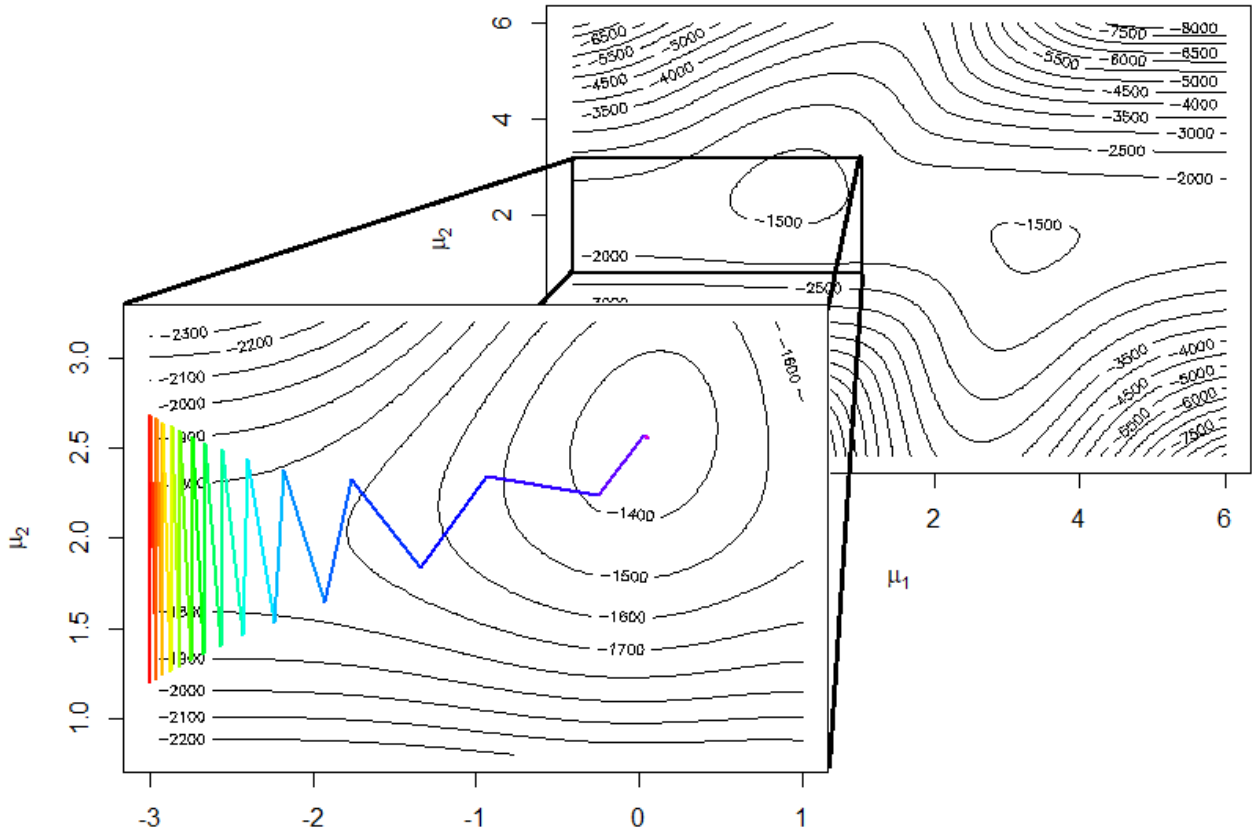
- Similar to SGD, but more stable, efficient than SGD
- Simulation study of the relationship between learning rate and performance

Learning Rate	0.1	0.05	0.02	0.005	0.002
Mini-Batch	NA	82.5%	63.5%	14.5%	0.0%
SGD	32.5%	29.0%	20.5%	5.0%	0.0%

Table: Probability of Reaching Global Maximum Start from $(3, 5)$

Gradient Descent Optimization Algorithm

Example of tradional Gradient Descent



- Recall $\theta^{(j+1)} = \theta^{(j)} + \alpha_j \nabla h(\theta^{(j)}|X)$
- Fixed learning rate $\alpha = 0.0025$
- Zigzag behaviour on the mountain ridge

Gradient Descent Optimization Algorithm

Momentum Based Methods

- Momentum
  - SGD has trouble navigating on mountain ridge (ravines)
  - Momentum add a fraction γ of past time step to current update step to solve this problem

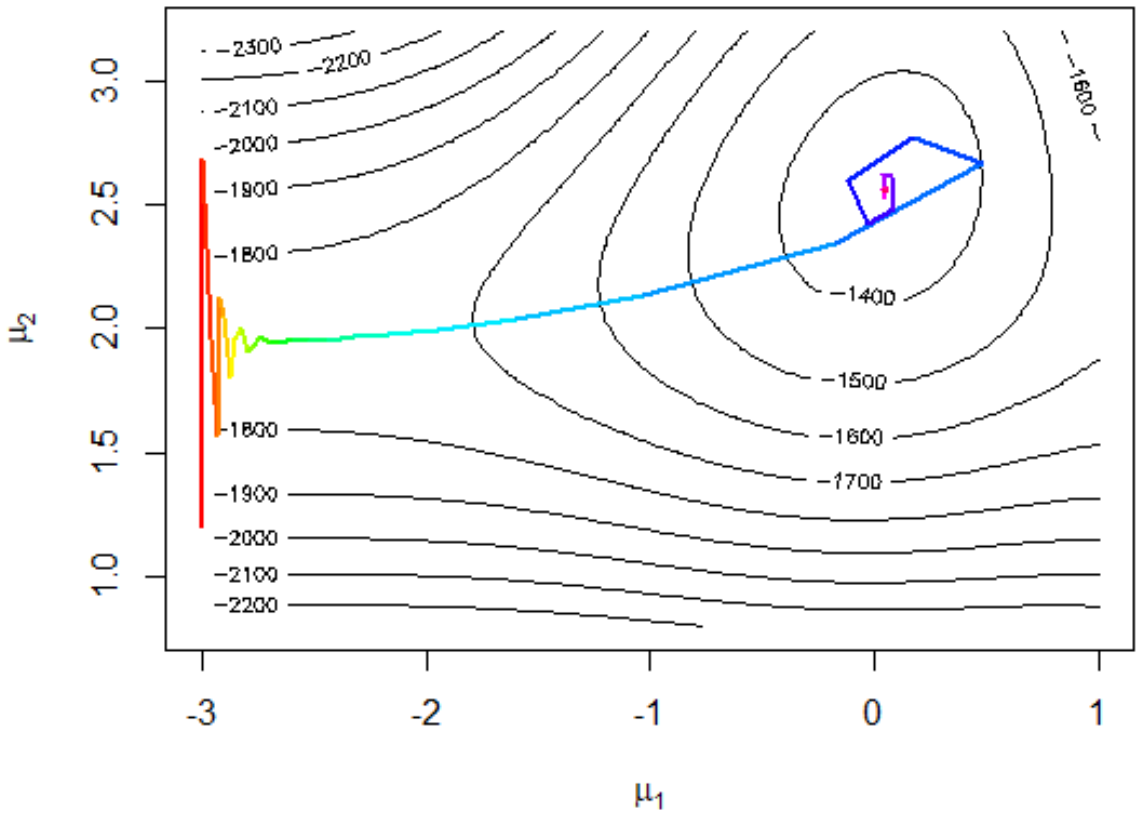
$$v_j = \gamma v_{j-1} + \alpha_j \nabla h(\theta^{(j)}) \tag{7}$$

$$\theta^{(j+1)} = \theta^{(j)} + v_j \tag{8}$$

  - Momentum helps to escape from saddle points
- Nesterov Accelarated Gradient
  - Account the next step into current step
  - Explore it if interested

Gradient Descent Optimization Algorithm

Gradient Descent with Momentum



- Fixed learning rate $\alpha = 0.0025$ and $\gamma = 0.5$
- Mild zigzag behaviour with less oscillation; faster
- Behaviour around the optimum?

Gradient Descent Optimization Algorithm

Adaptive Learning Rate Based Algorithm

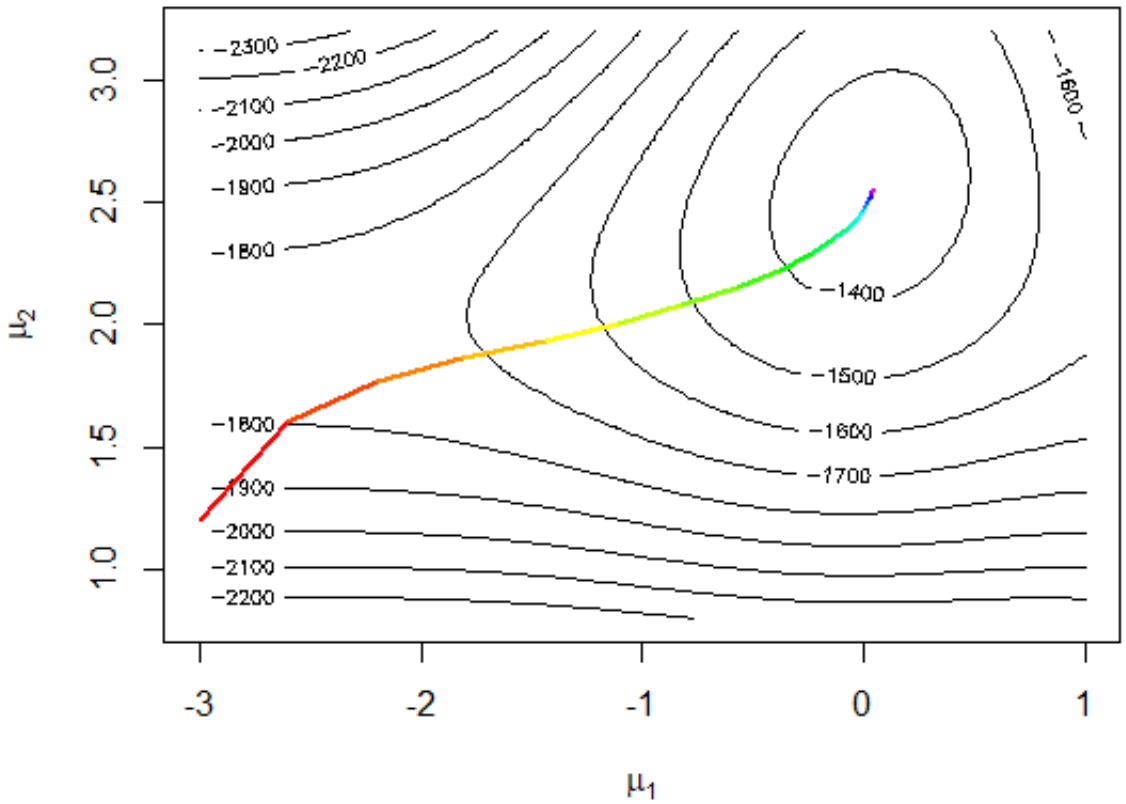
- AdaGrad
  - Finding out a good question-specific learning rate is not a piece of cake
  - AdaGrad adapts the learning rate and different parameters can have different learning rate
  - Larger updates for parameters associated with infrequent features, and vice versa

$$\theta^{(j+1)} = \theta^{(j)} + \alpha(G_j + \epsilon I_p)^{-1/2}\nabla h(\theta^{(j)}) \tag{9}$$

    where $G_j = diag(\sum_{t=1}^{j}(\nabla_i h_t)^2)$
  - Do not need to tune α any longer
- Adadelta
- RMSprop
- Adam
- AdaMax

Gradient Descent Optimization Algorithm

AdaGrad Solutions



- Fixed $\alpha = 0.4$ (usually do not need to tune a lot) and $\epsilon = 1 \times 10^{-8}$
- No zigzag
- Learn fast at beginning, but can be extremely slow after tons of iterations

Stochastic Search

Stochastic Gradient Method

- Stochastic Gradient (notice the name)

$$\nabla h(\theta^{(j)}) \approx \frac{h(\theta^{(j)} + \beta_j \zeta_j) - h(\theta^{(j)} - \beta_j \zeta_j)}{2\beta_j} \zeta_j = \frac{\Delta h(\theta^{(j)}, \beta_j \zeta_j)}{2\beta_j} \zeta_j \quad (10)$$

$$\theta^{(j+1)} = \theta^{(j)} + \frac{\alpha_j}{2\beta_j} \Delta h(\theta^{(j)}, \beta_j \zeta_j) \zeta_j \quad (11)$$

  where ζ_j is a unit sphere with random direction

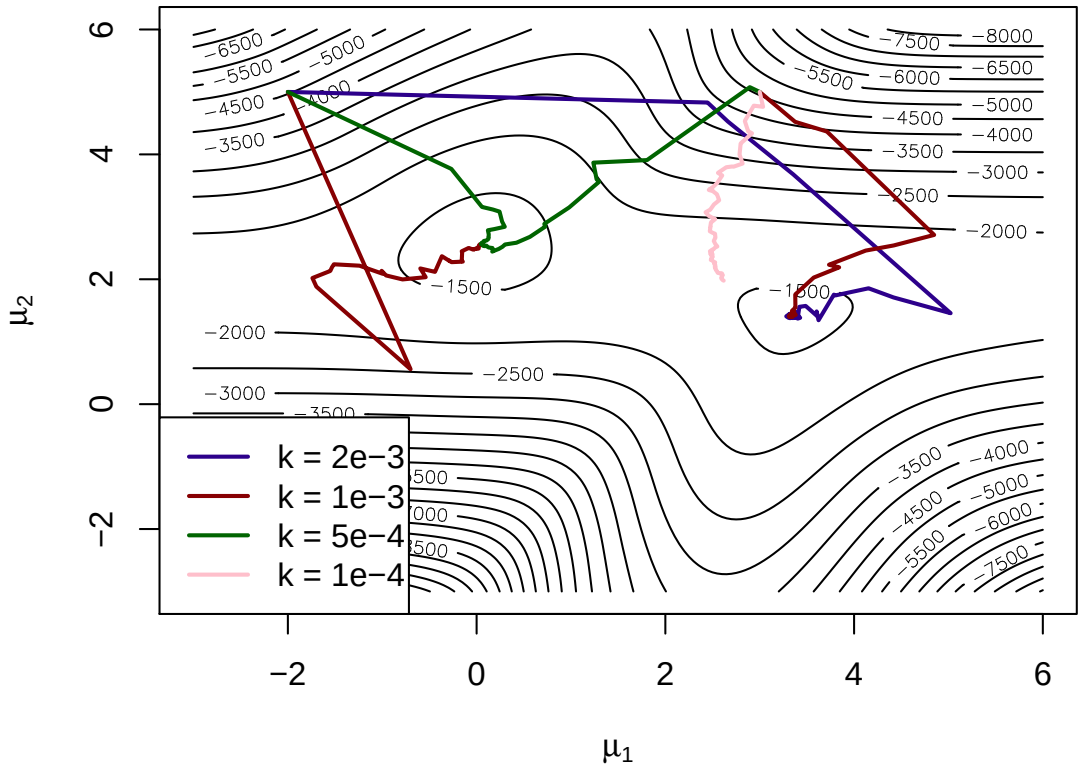
- Need to tune α_j and β_j (too much to tune)

$$\theta^{(j+1)} = \theta^{(j)} + k \Delta h(\theta^{(j)}, \beta_j \zeta_j) \zeta_j \quad (12)$$

- How much role does $h(\theta)$ play?

Stochastic Search

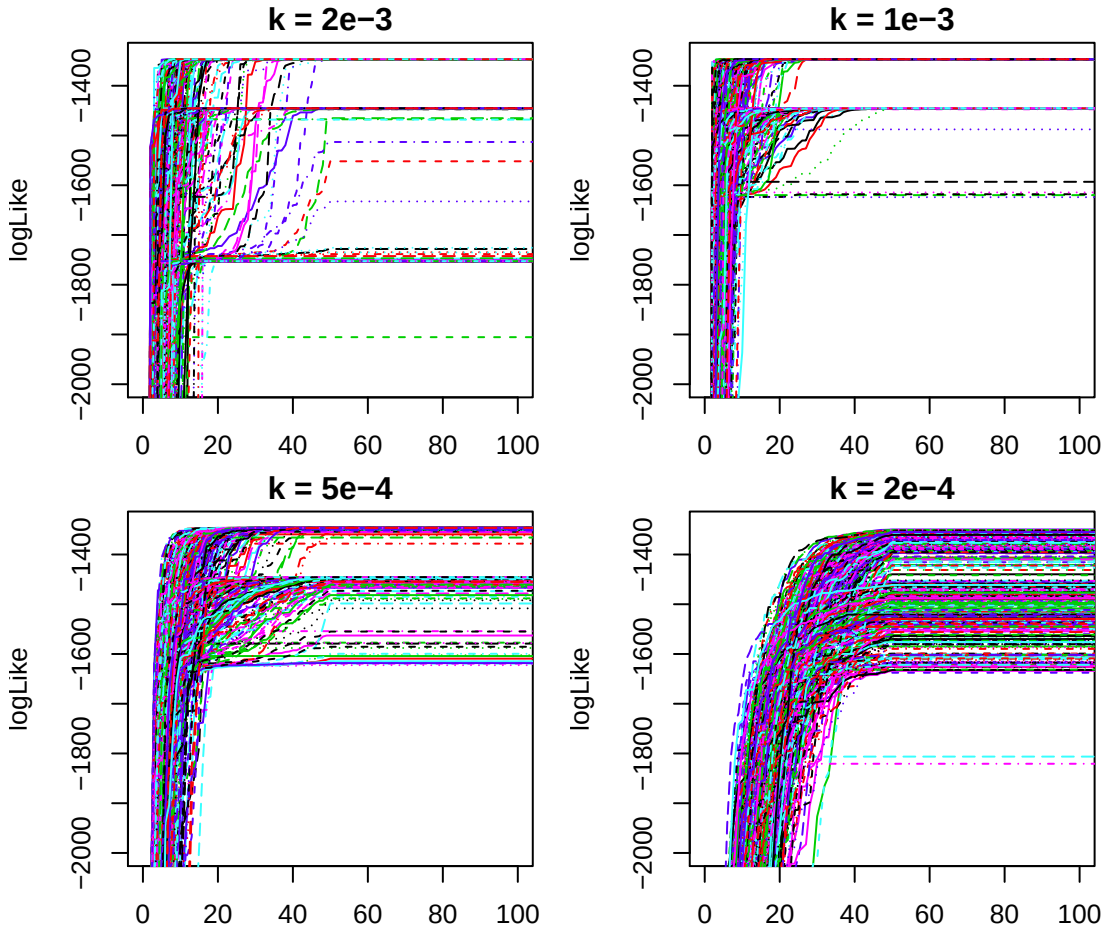
Stochastic Gradient Solutions



- Set $\beta_j = 1/\sqrt{log(1+j)}$
- Behaviour quite depends on k and β_j
- Have chance to find global optimum

Stochastic Search

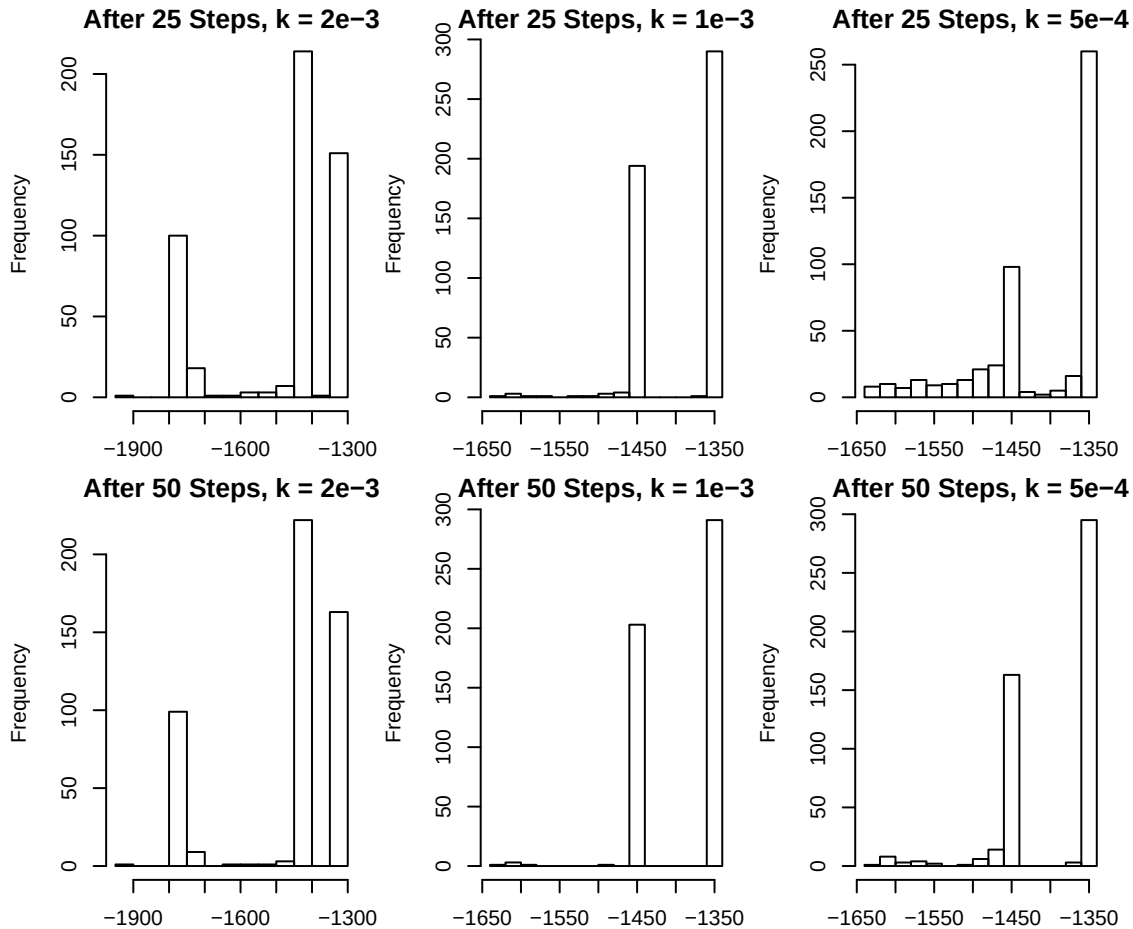
Stochastic Gradient Simulation Results



- 500 traces
- Can be trapped at other places

Stochastic Search

Stochastic Gradient Simulation Results



- Converge very fast
- Sensitive to k

Stochastic Search

Simulated Annealing

- A method coming from simulating the physical process of heating a material and then slowly lowering the temperature to increase its ductility and hardness
- Atoms within a solid material reaches equilibrium state during slowly cooling process
- Rationale is based on Boltzmann-Gibbs distribution

$$p_i = \frac{e^{-\frac{\varepsilon_i}{kT}}}{\sum_j e^{-\frac{\varepsilon_j}{kT}}} \propto e^{-\frac{\varepsilon_i}{kT}} \tag{13}$$

  where ε_i is the energy of the i^{th} state
- Under certain temperature, lower energy level has higher probability
- When cooling down, system loses energy so that prone to lower energy states
- Cooling slowly makes system have higher chance to reach equilibrium

Stochastic Search

Simulated Annealing Algorithm

- Given a random perturbance $\beta_j \zeta_j$, where $||\zeta_j|| = 0$

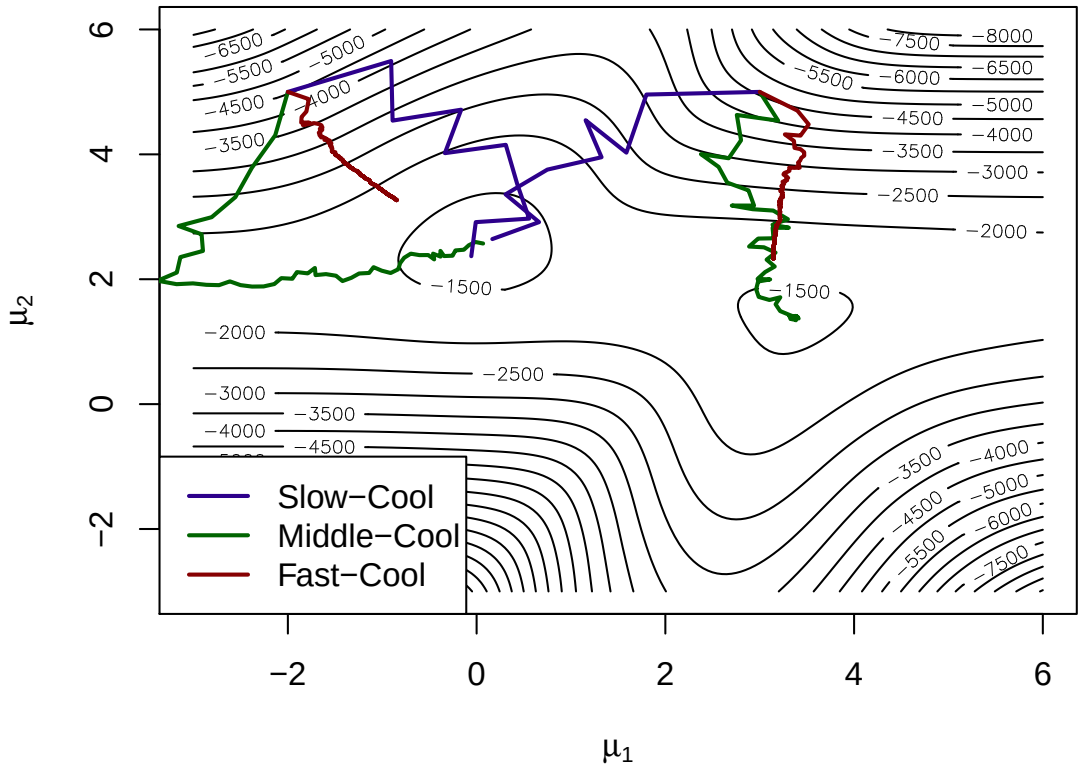
$$\theta^{(j+1)} = \begin{cases} \theta^{(j)} + \beta_j \zeta_j & \text{with probability } \rho = \min\{exp(-\Delta\varepsilon_j / T_j), 1\}, \\ \theta^{(j)} & \text{with probability } 1 - \rho, \end{cases} \tag{14}$$

  where $\Delta\varepsilon_j = h(\theta^{(j)} + \beta_j \zeta_j) - h(\theta^{(j)})$

- T_j represents temperature at j^{th} iterations, and can be decrease in $1/log(1+j)$ or $1/(1+j)^2$
- β_j also need to be carefully adjusted, e.g. $\sqrt{T_j}$
- Very different from Stochastic Gradient?
- How much role does $h(\theta)$ play?
- Works for either unconstrained or constrained optimization problems

Stochastic Search

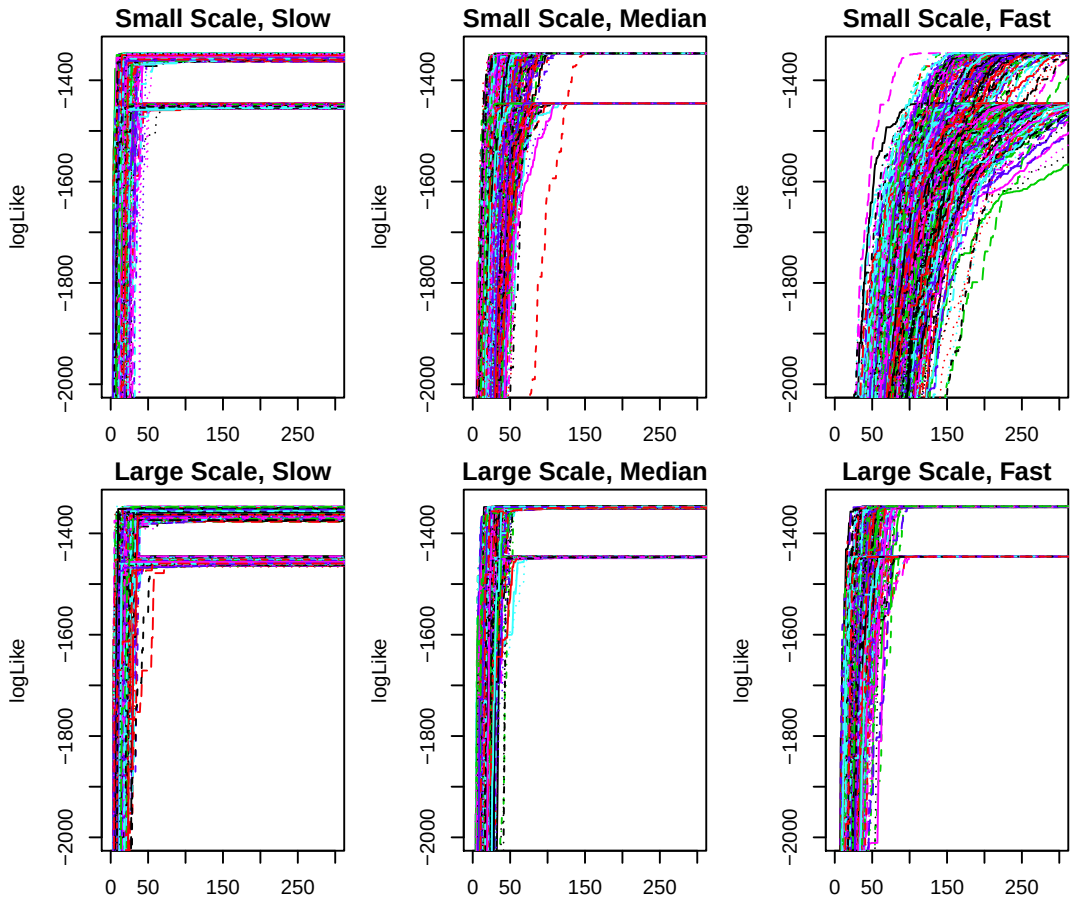
Simulated Annealing Solutions



- Slow: $T_j = 1/log(1+j)$; Middle: $T_j = 1/(1+j)$; Fast: $T_j = 1/(1+j)^2$
- If cooling too fast, it is hard to reach 'equilibrium'

Stochastic Search

Simulated Annealing Simulation Studies



(Compare with the previous one?)

- Small Scale: $\beta_j = T_j^{0.5}$; Large Scale: $\beta_j = T_j^{0.3}$
- Note that the stopping position may not be the best result

Stochastic Search

Simulated Annealing Simulation Studies (Con't)

Cool-Down Speed	Slow	Median	Fast
Small Scale	55.2%	50.6%	36.0%
Large Scale	54.8%	53.4%	46.2%

Table: Probability of Reaching Global Maximum Start from $(3, 5)$

- Same parameter settings as before
- There is a limitation
- Note the correlation between scale and temperature
- What else do you think?

Is there any better way?

EM Algorithm

- Observed log-likelihood function

$$l^o(\theta|x) \propto \sum_i log\{\pi exp(-(x_i-\mu_1)^2/2\sigma_1^2)+(1-\pi)exp(-(x_i-\mu_2)^2/2\sigma_2^2)\} \tag{15}$$

- Complete log-likelihood function

$$l^c(\theta|x,\delta) \propto -\sum_i \delta_i(x_i-\mu_1)^2/\sigma_1^2 - \sum_i (1-\delta_i)(x_i-\mu_2)^2/\sigma_2^2 \tag{16}$$

- To accomplish E-step, we need to figure out $E(\delta_i|x,\theta^{(k)})$

$$E(\delta_i|x,\theta^{(k)}) = \frac{\pi\phi((x_i-\mu_1)/\sigma_1)}{\pi\phi((x_i-\mu_1)/\sigma_1)+(1-\pi)\phi((x_i-\mu_2)/\sigma_2)} \tag{17}$$

- Then we could figure out the iterative updates of θ

$$\mu_1^{(k+1)} = \sum_i x_i E^{(k)}(\delta_i) / \sum_i E^{(k)}(\delta_i) \tag{18}$$

$$\mu_2^{(k+1)} = \sum_i x_i [1-E^{(k)}(\delta_i)] / \sum_i [1-E^{(k)}(\delta_i)] \tag{19}$$

Complete Likelihood vs. Observed Likelihood

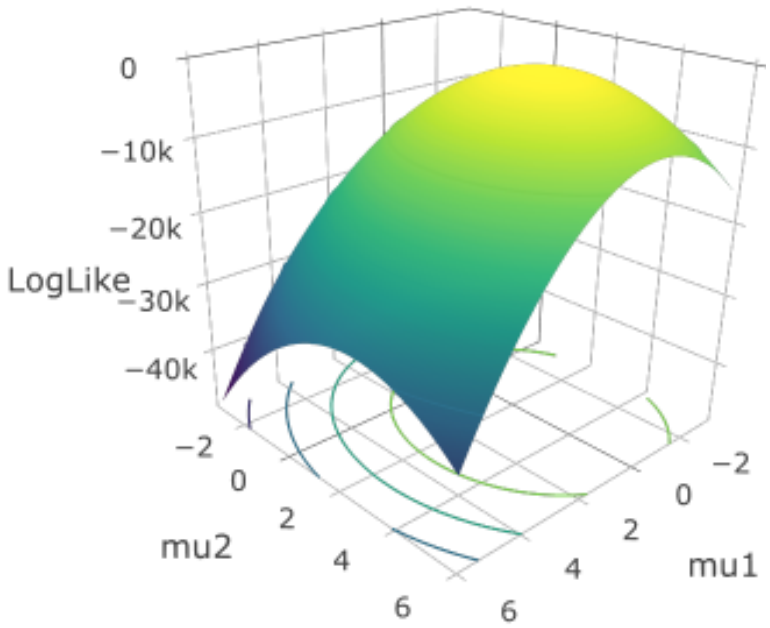


Figure: Complete Likelihood

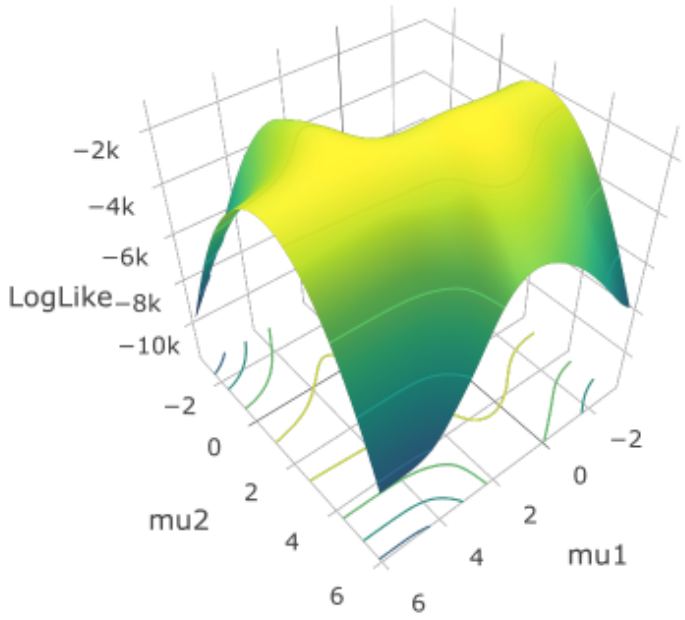
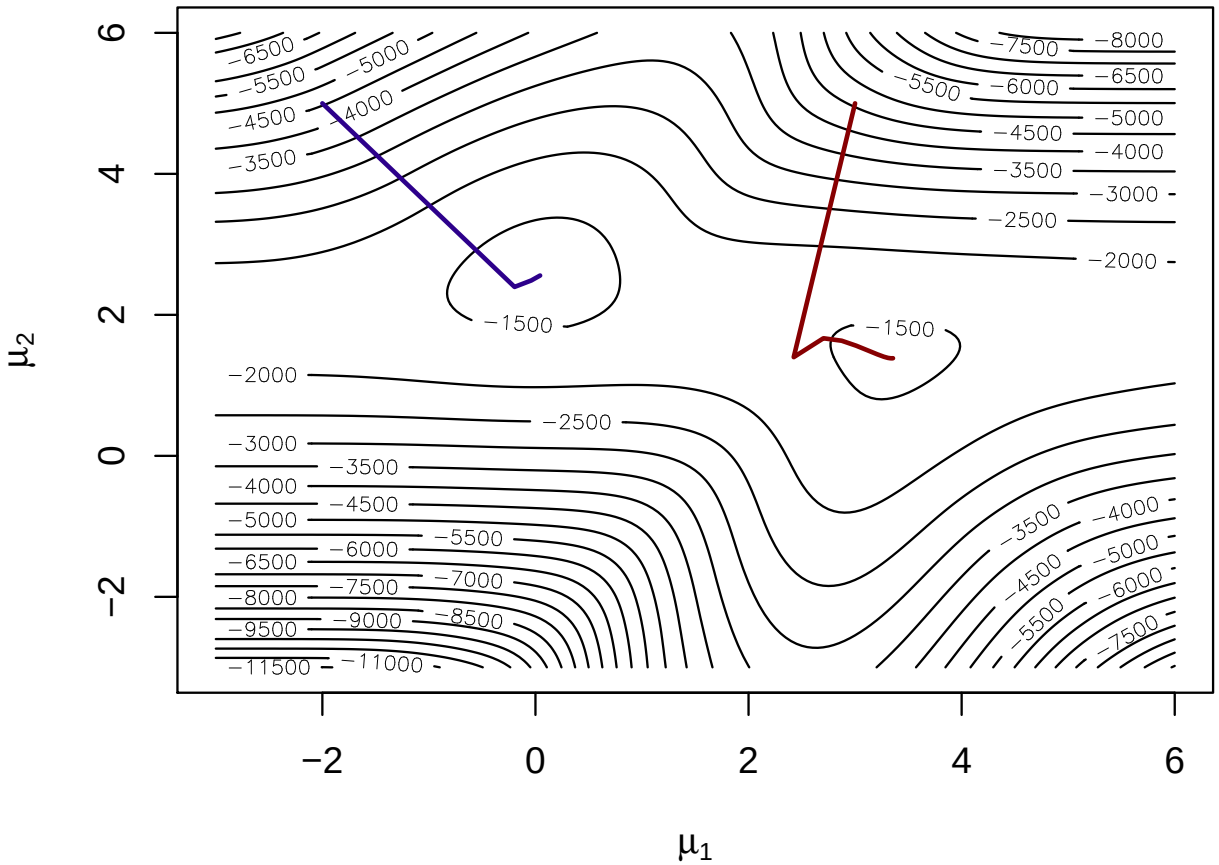


Figure: Observed Likelihood

EM Solutions



Stochastic Approximation

Monte Carlo EM (MCEM)

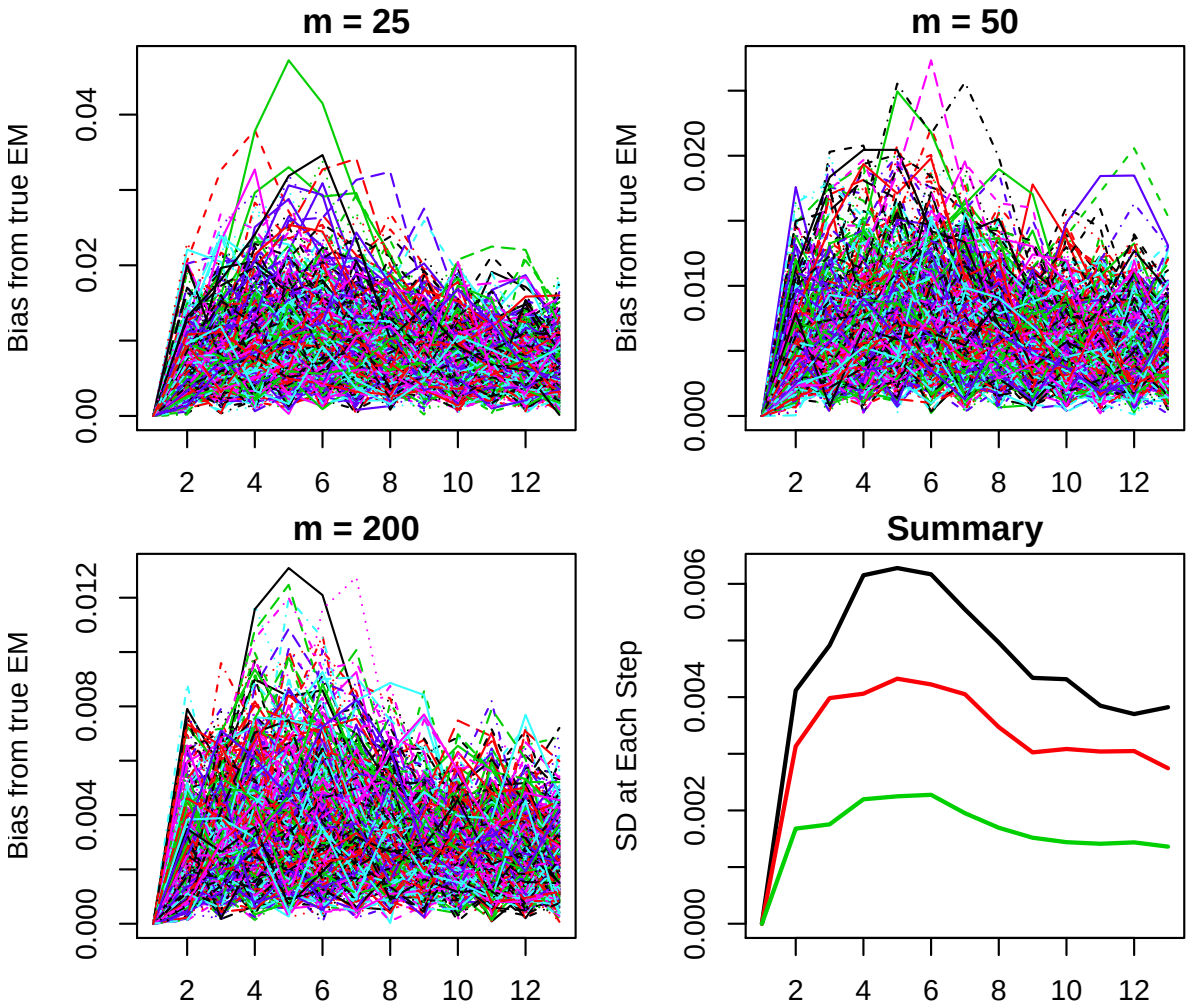
- Sometimes the E-step is not straight forward, so we use MC to approximate the expectation
- If we know the distribution of the unobserved part, we could sample unobserved variable from its distribution and figure out the expectation. E.g.

$$\delta_i \sim Binom(1, \frac{\pi\phi((x_i-\mu_1)/\sigma_1)}{\pi\phi((x_i-\mu_1)/\sigma_1)+(1-\pi)\phi((x_i-\mu_2)/\sigma_2)}) \quad (20)$$

$$E(\delta_i|x,\theta^{(k)}) = \frac{1}{m}\sum_{i=1}^{m}\delta_i \quad (21)$$

Stochastic Approximation

Monte Carlo EM (MCEM)



- Check the variance of distance between MCEM and true EM step by step