

Otimização Numérica

Prof. Wagner H. Bonat

Universidade Federal do Paraná
Departamento de Estatística
Laboratório de Estatística e Geoinformação



Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação
- 7 Recomendações

Métodos de otimização não-linear

- Os métodos são em geral categorizados baseado na dimensionalidade
 - Unidimensional: Golden Section search.
 - Multidimensional.
- Caso multidimensional, tem-se pelo menos quatro tipos de algoritmos
 - Não baseados em gradiente: Nelder-Mead;
 - Baseados em gradiente: Gradiente descendente e variações;
 - Baseados em hessiano: Newton e quasi-Newton (BFGS);
 - Algoritmos baseados em simulação e ideias genéticas: Simulating Annealing (SANN).
- A função genérica `optim()` em R fornece interface aos principais algoritmos de otimização.
- Vamos discutir as principais ideias por traz de cada tipo de algoritmo.
- Existe uma infinidade de variações e implementações.

Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent**
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação
- 7 Recomendações

Programação não-linear: Problemas unidimensionais

- Golden Section Search é o mais popular e muito eficiente.
- Algoritmo
 - 1 Define a razão de ouro $\psi = \frac{\sqrt{5}-1}{2} = 0.618$;
 - 2 Escolha um intervalo $[a, b]$ que contenha a solução;
 - 3 Avalie $f(x_1)$ onde $x_1 = a + (1 - \psi)(b - a)$ e compare com $f(x_2)$ onde $x_2 = a + \psi(b - a)$;
 - 4 Se $f(x_1) < f(x_2)$ continua a procura em $[a, x_1]$ caso contrário em $[x_2, b]$.
- Em R a função `optimize()` implementa este método.

```
args(optimize)
```

```
# function (f, interval, ..., lower = min(interval), upper = max(interval),
#   maximum = FALSE, tol = .Machine$double.eps^0.25)
# NULL
```

- Na função `optim()` esse método é chamado de Brent.

Exemplo: Otimização unidimensional

- Minimize a função $f(x) = |x - 2| + 2|x - 1|$.
- Implementando e otimizando.

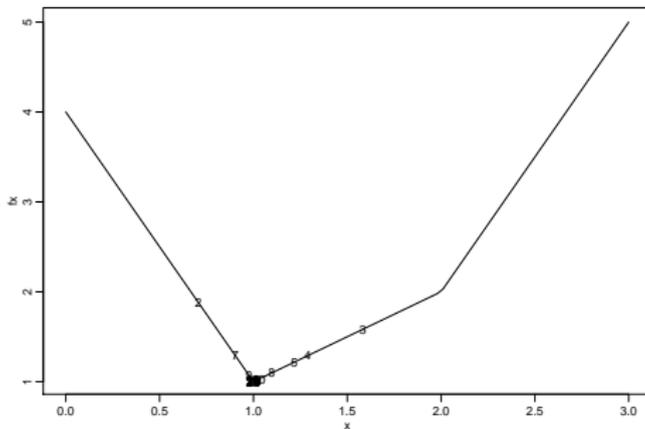
```
xx <- c()
fx <- function(x) {
  out <- abs(x-2) + 2*abs(x-1)
  xx <<- c(xx, x)
  return(out)
}
out <- optimize(f = fx, interval = c(-3,3))
out
```

```
# $minimum
# [1] 1.000021
#
# $objective
# [1] 1.000021
```

Exemplo: Otimização unidimensional

- Traço do algoritmo.

```
par(mfrow = c(1,1), mar=c(2.6, 3, 1.2, 0.5), mgp = c(1.6, 0.6, 0))
fx <- function(x) abs(x-2) + 2*abs(x-1)
plot(fx, 0, 3)
for(i in 1:length(xx)) {
  text(x = xx[i], y = fx(xx[i]), label = i)
}
```



Sumário

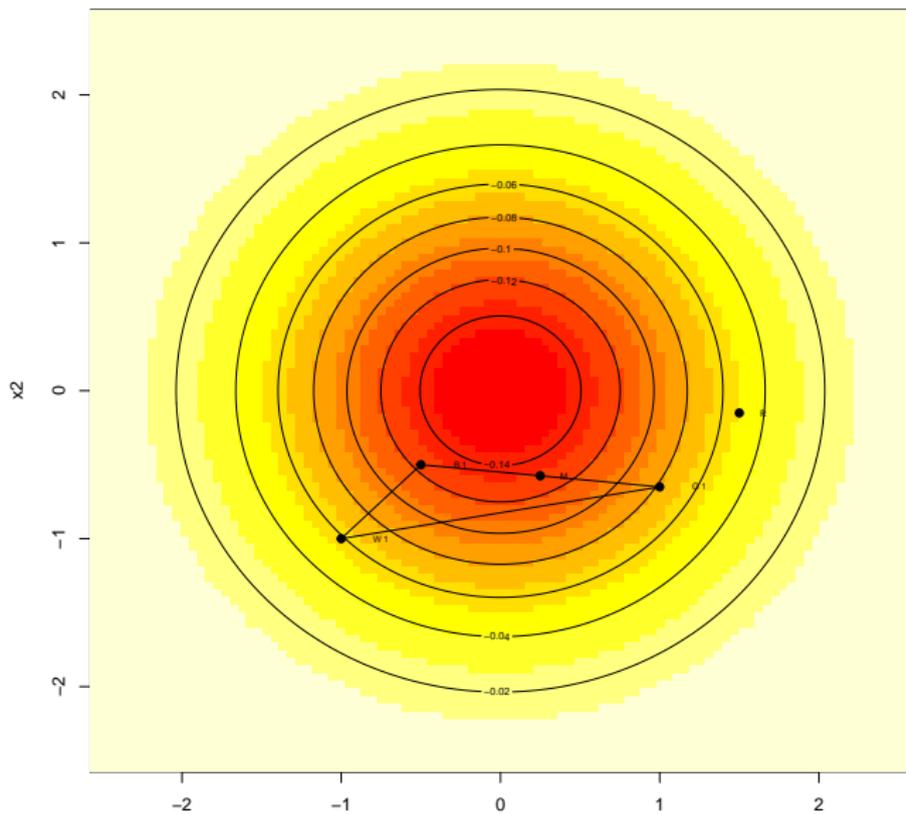
- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead**
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação
- 7 Recomendações

Método de Nelder-Mead (gradient free)

- Algoritmo de Nelder-Mead

- 1 Escolha um simplex com $n + 1$ pontos $p_1(x_1, y_1), \dots, p_{n+1}(x_{n+1}, y_{n+1})$, sendo n o número de parâmetros.
- 2 Calcule $f(p_i)$ e ordene por tamanho $f(p_1) \leq \dots \leq f(p_n)$.
- 3 Avalie se o melhor valor é bom o suficiente, se for, pare.
- 4 Delete o ponto com maior/menor $f(p_i)$ do simplex.
- 5 Escolha um novo ponto pro simplex.
- 6 Volte ao passo 2.

Algoritmo de Nelder-Mead: Ilustração



Algoritmo de Nelder-Mead: Escolhendo o novo ponto

- Ponto central do lado melhor (B):

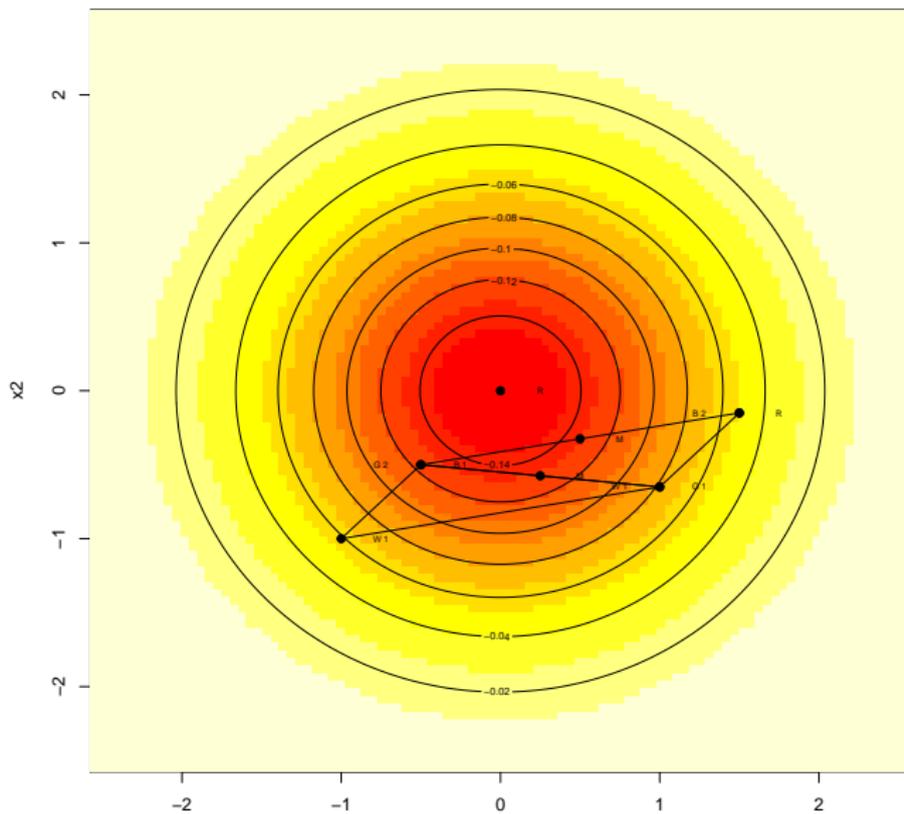
$$M = \frac{B + G}{2} = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right).$$

- Refletir o simplex para o lado BG.

$$R = M + (M - W) = 2M - W.$$

- Se a função em R é menor que em W movemos na direção correta.
 - 1 Opção 1: Faça $W = R$ e repita.
 - 2 Opção 2: Expandir usando o ponto $E = 2R - M$ e $W = E$, repita.
- Se a função em R e W são iguais contraia W para próximo a B, repita.
- A cada passo uma decisão lógica precisa ser tomada.

Algoritmo de Nelder-Mead: Ilustração



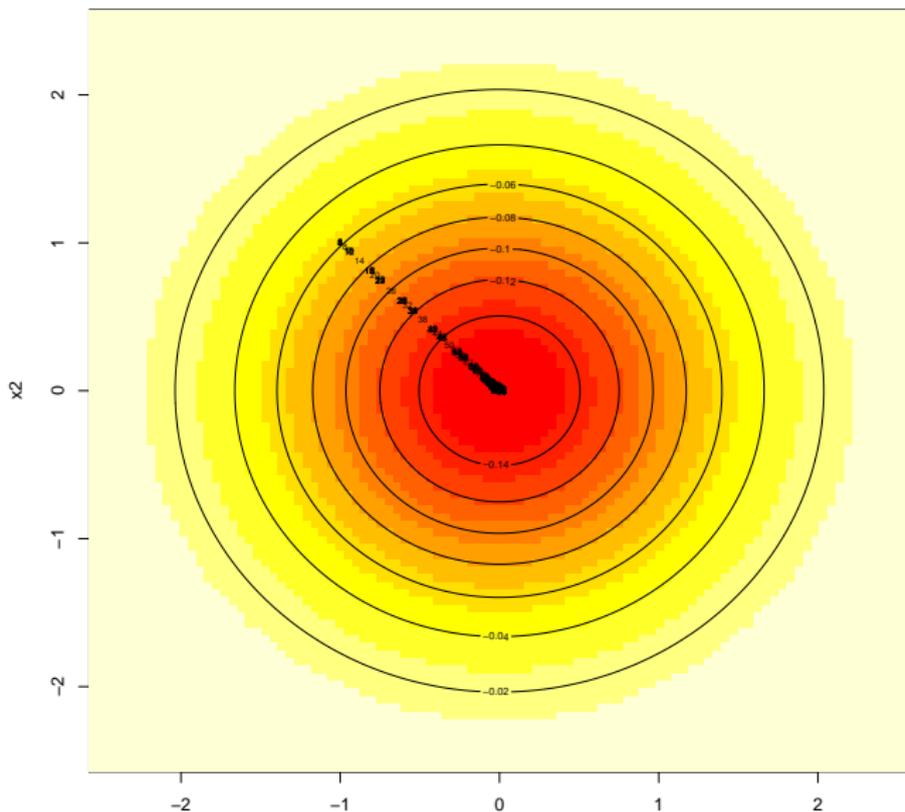
Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente**
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação
- 7 Recomendações

Métodos baseado em gradiente

- Use o gradiente de $f(x)$, ou seja, $f'(x)$ para obter a direção de procura.
 - 1 $f'(x)$ pode ser obtido analiticamente;
 - 2 $f'(x)$ qualquer aproximação numérica.
- A direção de procura s_n é o negativo do gradiente no último ponto.
- Passos básicos
 - 1 Calcule a direção de busca $-f'(x)$.
 - 2 Obtenha o próximo passo $x^{(n+1)}$ movendo com passo α_n na direção de $-f'(x)$.
 - 3 Tamanho do passo α_n pode ser fixo ou variável.
 - 4 Repita até $f'(x^i) \approx 0$ seja satisfeito.

Ilustração: Métodos baseado em gradiente



Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano**
- 6 Métodos baseados em simulação
- 7 Recomendações

Métodos baseado em hessiano

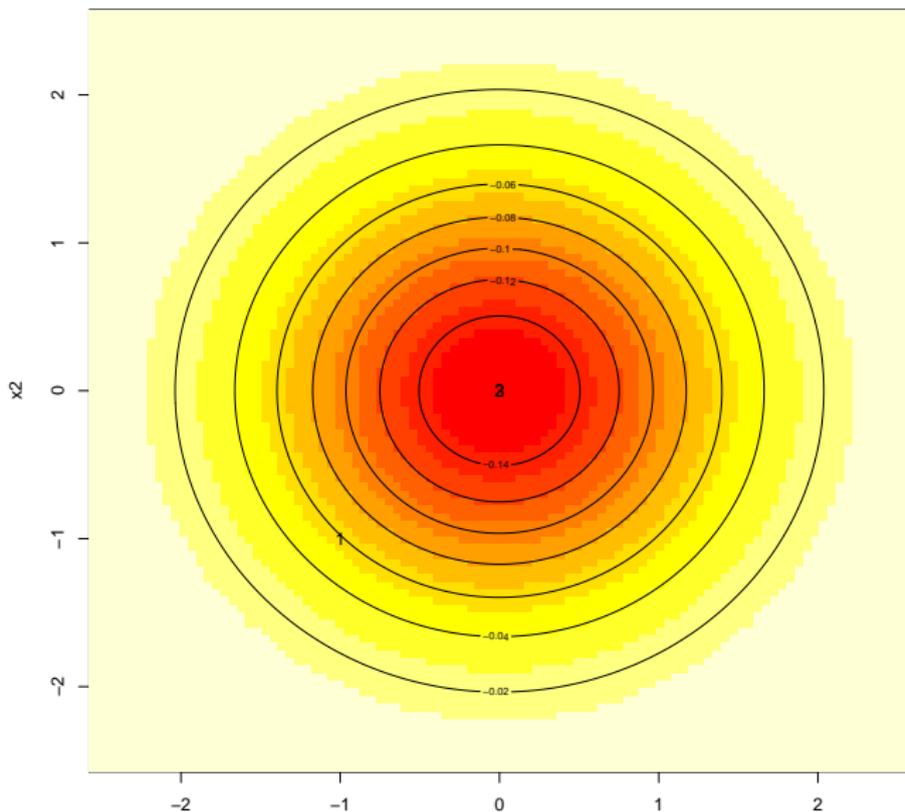
- Algoritmo de Newton-Raphson.
- Maximizar/minimizar uma função $f(x)$ é o mesmo que resolver a equação não-linear $f'(x) = 0$.
- Equação de iteração

$$x^{(i+1)} = x^{(i)} - \mathbf{J}(\mathbf{x}^{(i)})^{-1} f'(\mathbf{x}^{(i)}),$$

onde \mathbf{J} é a segunda derivada (hessiano) de $f(x)$.

- $\mathbf{J}(\mathbf{x}^{(i)})$ pode ser obtida analítica ou numericamente.
- $\mathbf{J}(\mathbf{x}^{(i)})$ pode ser aproximada por uma função mais simples de calcular.
- Métodos Quasi-Newton (mais famoso BFGS).

Ilustração: Métodos baseado em hessiano (Newton)



Métodos Quasi-Newton

- Métodos quasi-Newton tentam imitar o método de Newton.
- A equação de iteração é dada por

$$x^{(i+1)} = x^{(i)} - \alpha_i \mathbf{H}_i f'(x^{(i)}),$$

onde \mathbf{H}_i é alguma aproximação para o inverso do Hessiano.

- α_i é o tamanho do passo.
- Denote $\delta_i = x^{(i+1)} - x^{(i)}$ e $\gamma_i = f'(x^{(i+1)}) - f'(x^{(i)})$.
- Para obter \mathbf{H}_{i+1} o algoritmo impõe que

$$\mathbf{H}_{i+1} \gamma_i = \delta_i.$$

- Algoritmo DFP

$$\mathbf{H}_{i+1} = \mathbf{H}_i - \frac{\mathbf{H}_i \gamma_i \gamma_i^\top \mathbf{H}_i}{\gamma_i^\top \mathbf{H}_i \gamma_i} + \frac{\delta_i \delta_i^\top}{\delta_i^\top \gamma_i}.$$

Métodos Quasi-Newton

- Versão melhorada do DFP devido a Broyden, Fletcher, Goldfarb e Shanno (BFGS).
- Aproxima o hessiano por

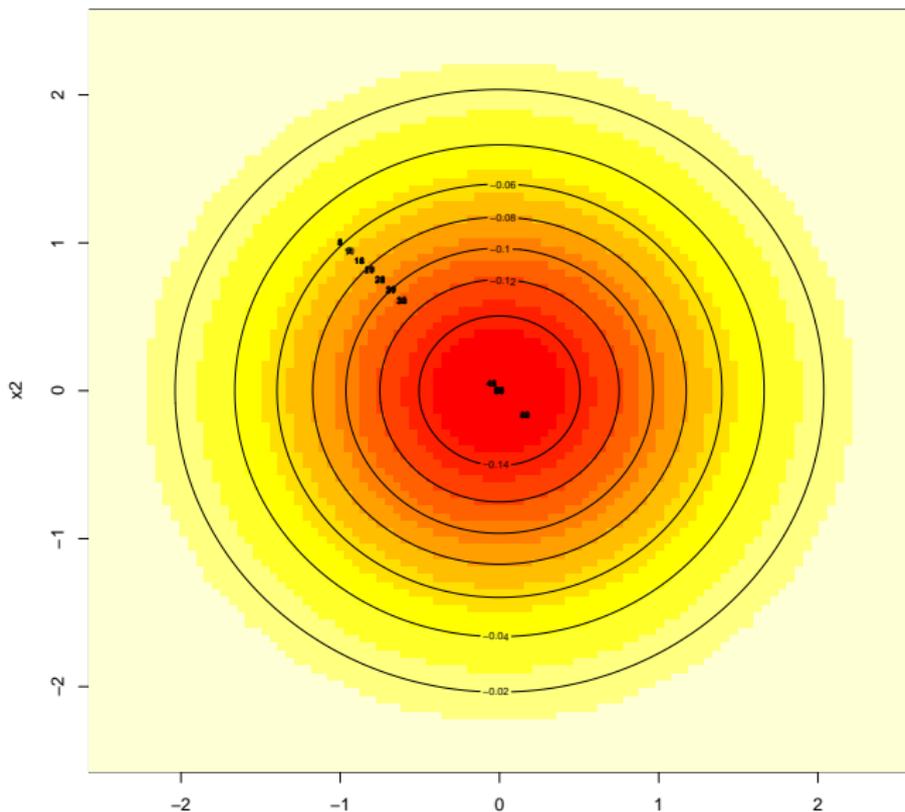
$$\mathbf{H}_{i+1} = \mathbf{H}_i - \frac{\delta_i \gamma_i^\top \mathbf{H}_i + \mathbf{H}_i \gamma_i \delta_i^\top}{\delta_i^\top \gamma_i} + \left(1 + \frac{\gamma_i^\top \mathbf{H}_i \gamma_i}{\delta_i^\top \gamma_i} \right) \left(\frac{\delta_i \delta_i^\top}{\delta_i^\top \gamma_i} \right).$$

- Implementações modernas do BFGS usando *wolfe line search* para encontrar α_i .
- Considere $\psi(\alpha) = f(x^{(i)} - \alpha \mathbf{H}_i f'(x^{(i)}))$, encontre α_i tal que

$$\psi_i(\alpha_i) \leq \psi_i(0) + \mu \psi_i'(0) \alpha_i \quad \text{e} \quad \psi_i'(\alpha_i) \geq \eta \psi_i'(0),$$

onde μ e η são constantes com $0 < \mu \leq \eta < 1$.

Ilustração: Métodos baseado em hessiano (BFGS)



Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação**
- 7 Recomendações

Métodos baseado em simulação

- Algoritmo genérico (maximização):
 - 1 Gere uma solução aleatória (x_1);
 - 2 Calcule a função objetivo no ponto simulado $f(x_1)$;
 - 3 Gere uma solução na vizinhança (x_2) do ponto em (1);
 - 4 Calcule a função objetivo no novo ponto $f(x_2)$:
 - Se $f(x_2) > f(x_1)$ mova para x_2 .
 - Se $f(x_2) < f(x_1)$ TALVEZ mova para x_2 .
 - 5 Repita passos 3-4 até atingir algum critério de convergência ou número máximo de iterações.

Métodos baseado em simulação: Simulating annealing

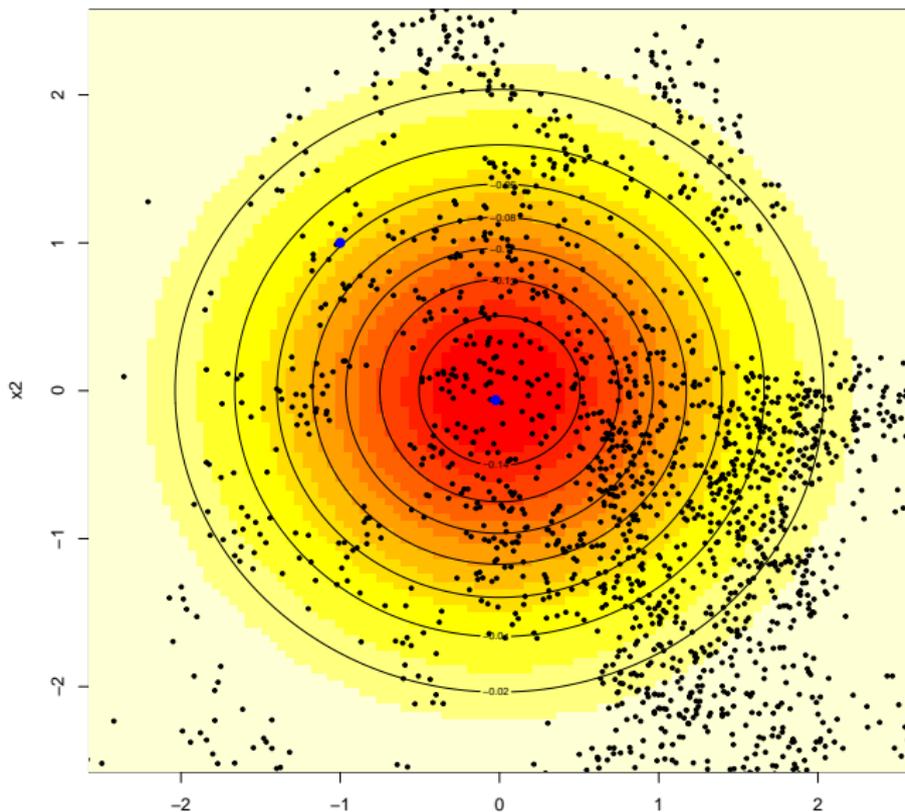
- Para decidir se um ponto x_2 quando $f(x_2) < f(x_1)$ será aceito, usa-se uma probabilidade de aceitação

$$a = \exp(f(x_2) - f(x_1))/T,$$

onde T é a *temperatura* (pense como um *tuning*).

- Se $f(x_2) > f(x_1)$ então $a > 1$, assim o x_2 será aceito com probabilidade 1.
- Se $f(x_2) < f(x_1)$ então $0 < a < 1$.
- Assim, x_2 será aceito se $a > U(0, 1)$.
- Amostrador de Metropolis no contexto de MCMC (*Markov Chain Monte Carlo*).

Ilustração: Métodos baseado em simulação (SANN)



Escolhendo o melhor método

- Método de Newton é o mais eficiente (menos iterações).
- Porém, cada iteração pode ser cara computacionalmente.
- Cada iteração envolve a solução de um sistema $p \times p$.
- Métodos quasi-Newton são eficientes, principalmente se o gradiente for obtido analiticamente.
- Quando a função é suave os métodos de Newton e quasi-Newton geralmente convergem.
- Métodos baseados apenas em gradiente são simples computacionalmente.
- Em geral precisam de *tuning* o que pode ser difícil na prática.
- Método de Nelder-Mead é simples e uma escolha razoável.
- Métodos baseados em simulação são ideal para funções com máximos/mínimos locais.
- Em geral são caros computacionalmente e portanto lentos.

Sumário

- 1 Programação não-linear
- 2 Método Golden-Search ou Brent
- 3 Método de Nelder-Mead
- 4 Métodos baseados em gradiente
- 5 Métodos baseados em hessiano
- 6 Métodos baseados em simulação
- 7 **Recomendações**

Escolhendo o melhor método

- Em R o pacote `optimx()` fornece funções para avaliar e comparar o desempenho de métodos de otimização.
- Exemplo: Minimizando a Normal bivariada.
- Escrevendo a função objetivo

```
fx <- function(xx){-dmvnorm(xx)}
```

Escolhendo o melhor método

- Comparando os diversos algoritmos descritos.

```
require(optimx)
```

```
# Loading required package: optimx
```

```
res <- optimx(par = c(-1,1), fn = fx,
              method = c("BFGS", "Nelder-Mead", "CG"))
res
```

#		p1	p2	value	fevals	gevals	niter	convcode
#	BFGS	-1.772901e-06	1.772901e-06	-0.1591549	13	11	NA	0
#	Nelder-Mead	1.134426e-04	-1.503306e-04	-0.1591549	55	NA	NA	0
#	CG	-8.423349e-06	8.423349e-06	-0.1591549	97	49	NA	0
#		kkt1	kkt2	xtime				
#	BFGS	TRUE	TRUE	0.005				
#	Nelder-Mead	TRUE	TRUE	0.003				
#	CG	TRUE	TRUE	0.016				

Algumas recomendações

- Otimização trata todos os parâmetros da mesma forma.
- Cuidado com parâmetros em escalas muito diferentes.
- Cuidado com parâmetros restritos.
- Recomendação: Torne todos os parâmetros irrestritos ou faça sua função a prova de erros.
- Use o máximo possível de resultados analíticos.
- Tutorial V.