Computing Unit 6: Optimization and Root Finding

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Kurt Hornik

EQUIS ACCREDITED AACSB ACCREDITED ASSOCIATION OF MBAS ACCREDITED

Outline

- Basics
- Root finding
- Optimization

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Optimization: for $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, find $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$ (if a max is sought) or $f(x^*) = \min_{x \in X} f(x)$ (if a min is sought).

Simplest case where max/min are attained: life can be more complicated.

If $X = \mathbb{R}^n$: unconstrained optimization problem.

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Optimization: for $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, find $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$ (if a max is sought) or $f(x^*) = \min_{x \in X} f(x)$ (if a min is sought).

Simplest case where max/min are attained: life can be more complicated.

If $X = \mathbb{R}^n$: unconstrained optimization problem.

If x^* is a root of f: $\Rightarrow$ minimizes $\|f(x)\|^2$ over the domain of f.

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Optimization: for $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, find $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$ (if a max is sought) or $f(x^*) = \min_{x \in X} f(x)$ (if a min is sought).

Simplest case where max/min are attained: life can be more complicated.

If $X = \mathbb{R}^n$: unconstrained optimization problem.

If x^* is a root of f: $\Rightarrow$ minimizes $\|f(x)\|^2$ over the domain of f.

If x^* is a local optimum of f and this is smooth: $\Rightarrow$ x^* is a critical point of f, i.e., $\nabla f(x^*) = 0$.

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Optimization: for $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, find $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$ (if a max is sought) or $f(x^*) = \min_{x \in X} f(x)$ (if a min is sought).

Simplest case where max/min are attained: life can be more complicated.

If $X = \mathbb{R}^n$: unconstrained optimization problem.

If x^* is a root of f: $\Rightarrow$ minimizes $\|f(x)\|^2$ over the domain of f.

If x^* is a local optimum of f and this is smooth: $\Rightarrow$ x^* is a critical point of f, i.e., $\nabla f(x^*) = 0$.

Can find roots via optimization, and optimize via root finding!

Basics

Root finding: for $f : \mathbb{R}^n \to \mathbb{R}^m$, find roots x^* such that $f(x^*) = 0$.

Optimization: for $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$, find $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$ (if a max is sought) or $f(x^*) = \min_{x \in X} f(x)$ (if a min is sought).

Simplest case where max/min are attained: life can be more complicated.

If $X = \mathbb{R}^n$: unconstrained optimization problem.

If x^* is a root of f: $\Rightarrow$ minimizes $\|f(x)\|^2$ over the domain of f.

If x^* is a local optimum of f and this is smooth: $\Rightarrow$ x^* is a critical point of f, i.e., $\nabla f(x^*) = 0$.

Can find roots via optimization, and optimize via root finding!

Note that the f for optimization always is a *scalar* function!

Outline

- Basics
- Root finding
- Optimization

Root finding

- Solve systems of linear equations ⇒ numerical linear algebra (2 more weeks)

Root finding

- Solve systems of linear equations ⇒ numerical linear algebra (2 more weeks)
- Find roots of polynomials: `polyroot()`.

Root finding

- Solve systems of linear equations ⇒ numerical linear algebra (2 more weeks)
- Find roots of polynomials: `polyroot()`.
- Find roots of continuous univariate functions: `uniroot()`.
  This is based on the *bisection method*.

Root finding

- Solve systems of linear equations ⇒ numerical linear algebra (2 more weeks)
- Find roots of polynomials: `polyroot()`.
- Find roots of continuous univariate functions: `uniroot()`.
  This is based on the *bisection method*.
- Already encountered Newton’s method for root finding, which is a *fixed-point iteration* method.
  Interestingly, no function for this in base R.

Bisection method

Suppose f is continuous with a sign change in $[a, b]$ (i.e., $f(a)$ and $f(b)$ have opposite signs).

Then f must have at least one root in $[a, b]$.

Bisection method

Suppose f is continuous with a sign change in $[a, b]$ (i.e., $f(a)$ and $f(b)$ have opposite signs).

Then f must have at least one root in $[a, b]$.

Can be found by the following iteration: Starting with $a_0 = a$ and $b_0 = b$,

- Take the current interval $[a_n, b_n]$.
- If $b_n - a_n$ is small enough; done. Otherwise, compute the midpoint $x_n = (a_n + b_n)/2$.
- If f has a sign change in $[a_n, x_n]$, take this as the next interval; otherwise take $[x_n, b_n]$.

Bisection method

In each step, halves the interval (length).

After n steps, interval length is $(b-a)/2^n$.

Bisection method

In each step, halves the interval (length).

After n steps, interval length is $(b-a)/2^n$.

So the interval converges to a root of f “exponentially fast”.

Bisection method

In each step, halves the interval (length).

After n steps, interval length is $(b-a)/2^n$.

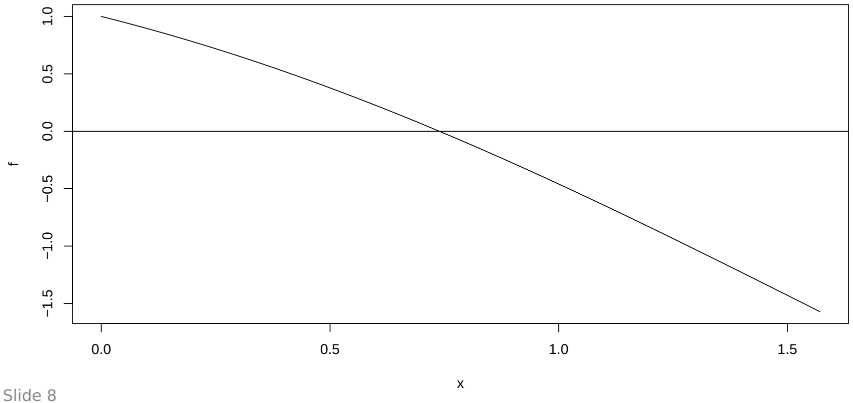
So the interval converges to a root of f “exponentially fast”.

Hence the computational complexity is *logarithmic* in the precision (interval length) sought.

Example

The function $f(x) = \cos(x) - x$ has a root in $[0, \pi/2]$:

```
R> f <- function(x) cos(x) - x
R> plot(f, 0, pi / 2); abline(h = 0)
```



Example

```
R> uniroot(f, c(0, pi / 2))

$root
[1] 0.7390839

$f.root
[1] 2.059443e-06

$iter
[1] 5

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05
```

Fixed-point iteration

Suppose g is continuous and the iteration

$$x_{n+1} = g(x_n)$$

gives a sequence x_n with (finite) limit x^*. What can we say about x^*?

Fixed-point iteration

Suppose g is continuous and the iteration

$$x_{n+1} = g(x_n)$$

gives a sequence x_n with (finite) limit x^*. What can we say about x^*?

By continuity,

$$x^* = \lim_n x_{n+1} = \lim_n g(x_n) = g\left(\lim_n x_n\right) = g(x^*)$$

Thus x^* must be a *fixed point* of g, i.e., solve $x^* = g(x^*)$.

Fixed-point iteration

Suppose g is continuous and the iteration

$$x_{n+1} = g(x_n)$$

gives a sequence x_n with (finite) limit x^*. What can we say about x^*?

By continuity,

$$x^* = \lim_n x_{n+1} = \lim_n g(x_n) = g\left(\lim_n x_n\right) = g(x^*)$$

Thus x^* must be a *fixed point* of g, i.e., solve $x^* = g(x^*)$.

Can use to find roots of f via fixed points of $g(x) = x + f(x)$? Not quite so simple . . .

Fixed-point iteration

If g is smooth about x^*,

$$x_{n+1} - x^* = g(x_n) - g(x^*) \approx g'(x^*)(x_n - x^*),$$

Fixed-point iteration

If g is smooth about x^*,

$$x_{n+1} - x^* = g(x_n) - g(x^*) \approx g'(x^*)(x_n - x^*),$$

So convergence somehow needs g to be *attractive* at x^*, i.e., $|g'(x^*)| < 1$.

Otherwise, the approximation error $x_n - x^*$ would not get reduced.

Fixed-point iteration

If g is smooth about x^*,

$$x_{n+1} - x^* = g(x_n) - g(x^*) \approx g'(x^*)(x_n - x^*),$$

So convergence somehow needs g to be *attractive* at x^*, i.e., $|g'(x^*)| < 1$.

Otherwise, the approximation error $x_n - x^*$ would not get reduced.

In the attractive case, approximation errors go to zero exponentially fast.

Fixed-point iteration

Prime example: Newton's method. This uses

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Fixed-point iteration

Prime example: Newton's method. This uses

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Clearly,

$$x = g(x) \Rightarrow \frac{f(x)}{f'(x)} = 0 \Rightarrow f(x) = 0.$$

Fixed-point iteration

Prime example: Newton's method. This uses

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Clearly,

$$x = g(x) \Rightarrow \frac{f(x)}{f'(x)} = 0 \Rightarrow f(x) = 0.$$

Also,

$$g'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}.$$

Fixed-point iteration

Hence, if x^* is a fixed point of g: $g'(x^*) = 0$!

In a sense, such fixed points are “very attractive”.

Fixed-point iteration

Hence, if x^* is a fixed point of g: $g'(x^*) = 0$!

In a sense, such fixed points are “very attractive”.

Convergence to x^* is actually “locally quadratic” in the sense that

$$|x_{n+1} - x^*| \le \text{const}\, |x_n - x^*|^2.$$

(Note the confusing terminology: approximation errors go to zero faster than exponentially fast!).

Outline

- Basics
- Root finding
- Optimization

Optimization with base R

- `optimize()` optimizes a univariate function based on golden section search.

Optimization with base R

- `optimize()` optimizes a univariate function based on golden section search.
- `nlm()` optimizes functions via Newton-type algorithms (i.e., via finding critical points)

Optimization with base R

- `optimize()` optimizes a univariate function based on golden section search.
- `nlm()` optimizes functions via Newton-type algorithms (i.e., via finding critical points)
- `optim()` provides several optimization methods: Nelder-Mead, quasi-Newton and conjugate-gradient algorithms, and simulated annealing.

Optimization with base R

- `optimize()` optimizes a univariate function based on golden section search.
- `nlm()` optimizes functions via Newton-type algorithms (i.e., via finding critical points)
- `optim()` provides several optimization methods: Nelder-Mead, quasi-Newton and conjugate-gradient algorithms, and simulated annealing.
- `nls()` solves non-linear least squares problems
- `constrOptim()` minimizes a function subject to linear inequality constraints using an adaptive barrier algorithm.

Much more in add-on packages . . . and other courses.

Here, we really only show the tip of the iceberg, and some basics.

Golden section search

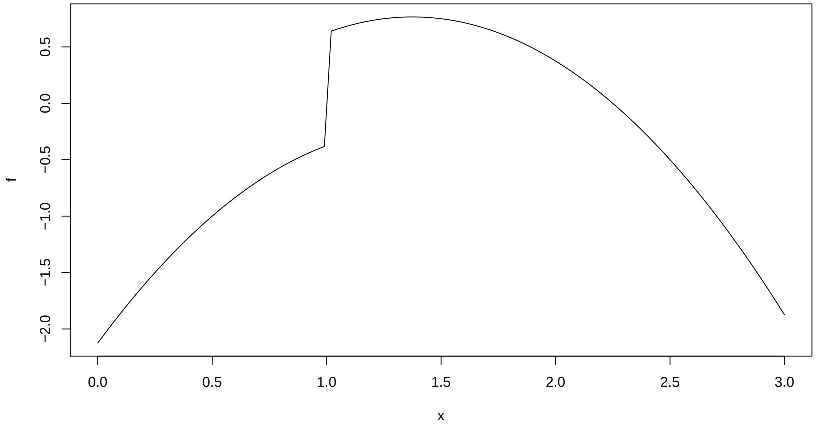
A function f is *unimodal* (over some interval $[a, b]$) if there is an m in the interval such that

$$f \text{ is increasing for } a \leq x < m \text{ and decreasing for } m < x \leq b.$$

In this case, clearly f has its maximum at m, and no local maxima.

f does not have to be continuous. Example:

Golden section search



Golden section search

Golden section search is based on “trisection”.

One starts with an interval $[a, b]$ containing the maximum, and adds two more points x_1 and x_2 to cut into three subintervals:

$$a < x_1 < x_2 < b.$$

Golden section search

Golden section search is based on “trisection”.

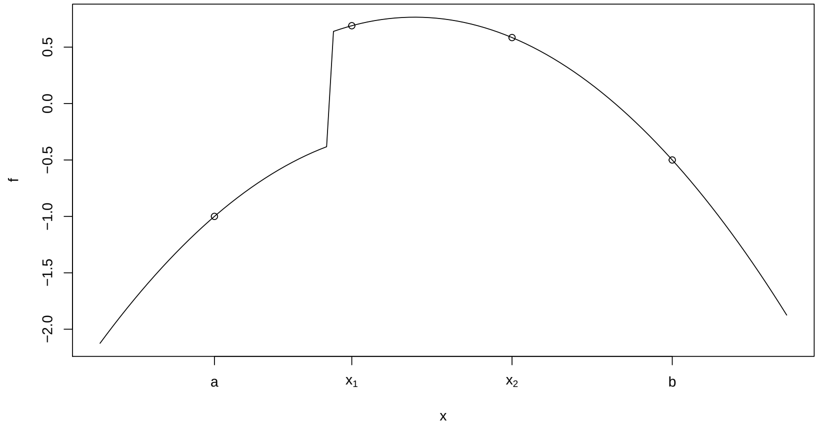
One starts with an interval $[a, b]$ containing the maximum, and adds two more points x_1 and x_2 to cut into three subintervals:

$$a < x_1 < x_2 < b.$$

If $f(x_1) > f(x_2)$, the max cannot be in $[x_2, b]$, and the new interval is $[a, x_2]$.

If $f(x_1) < f(x_2)$, the max cannot be in $[a, x_1]$, and the new interval is $[x_1, b]$.

Golden section search



Golden section search

Golden section search is based on the following idea:

- Intervals are always partitioned in constant proportions:

$$x_1 = a + \gamma_1(b - a), \qquad x_2 = a + \gamma_2(b - a)$$

Golden section search

Golden section search is based on the following idea:

- Intervals are always partitioned in constant proportions:

$$x_1 = a + \gamma_1(b-a), \qquad x_2 = a + \gamma_2(b-a)$$

  The γ_i are chosen in a way that one of the previous x_i points (and hence its function value $f(x_i)$) can be re-used.

Why? Function evaluation can be very expensive.

Golden section search

If $f(x_1) > f(x_2)$, the new interval is $[a' = a, b' = x_2]$, and the new x points are

$$x'_1 = a + \gamma_1(x_2 - a), \qquad x'_2 = a + \gamma_2(x_2 - a)$$

One of these must be x_1, and clearly $x'_1 < x_1$, so we must have $x'_2 = x_1$.

Golden section search

If $f(x_1) > f(x_2)$, the new interval is $[a' = a, b' = x_2]$, and the new x points are

$$x'_1 = a + \gamma_1(x_2 - a), \qquad x'_2 = a + \gamma_2(x_2 - a)$$

One of these must be x_1, and clearly $x'_1 < x_1$, so we must have $x'_2 = x_1$.

Thus,

$$a + \gamma_1(b - a) = x_1 = x'_2 = a + \gamma_2(x_2 - a) = a + \gamma_2^2(b - a)$$

from which $\gamma_1 = \gamma_2^2$.

Golden section search

If $f(x_1) < f(x_2)$, the new interval is $[a' = x_1, b' = b]$, and the new x points are

$$x'_1 = x_1 + \gamma_1(b - x_1), \qquad x'_2 = x_1 + \gamma_2(b - x_1)$$

One of these must be $x_2 = (1-\gamma_2)a + \gamma_2 b$. As clearly $x'_2 = (1-\gamma_2)x_1 + \gamma_2 b > x_2$, we must have $x_2 = x'_1$. Thus:

Golden section search

If $f(x_1) < f(x_2)$, the new interval is $[a' = x_1, b' = b]$, and the new x points are

$$x_1' = x_1 + \gamma_1(b - x_1), \qquad x_2' = x_1 + \gamma_2(b - x_1)$$

One of these must be $x_2 = (1 - \gamma_2)a + \gamma_2 b$. As clearly $x_2' = (1 - \gamma_2)x_1 + \gamma_2 b > x_2$, we must have $x_2 = x_1'$. Thus:

$$\begin{aligned} a + \gamma_2(b - a) &= x_1 + \gamma_1(b - x_1) \\ &= (1 - \gamma_1)x_1 + \gamma_1 b \\ &= a + (1 - \gamma_1)(x_1 - a) + \gamma_1(b - a) \\ &= a + (1 - \gamma_1)\gamma_1(b - a) + \gamma_1(b - a) \end{aligned}$$

from which $\gamma_2 = 2\gamma_1 - \gamma_1^2$.

Golden section search

Looks complicated? We have

$$\gamma_2^2 = \gamma_1, \qquad \gamma_1^2 = 2\gamma_1 - \gamma_2.$$

Golden section search

Looks complicated? We have

$$\gamma_2^2 = \gamma_1, \qquad \gamma_1^2 = 2\gamma_1 - \gamma_2.$$

Subtracting:

$$\gamma_2^2 - \gamma_1^2 = \gamma_1 - (2\gamma_1 - \gamma_2) = \gamma_2 - \gamma_1.$$

Golden section search

Looks complicated? We have

$$\gamma_2^2 = \gamma_1, \qquad \gamma_1^2 = 2\gamma_1 - \gamma_2.$$

Subtracting:

$$\gamma_2^2 - \gamma_1^2 = \gamma_1 - (2\gamma_1 - \gamma_2) = \gamma_2 - \gamma_1.$$

As $\gamma_1 < \gamma_2$ and $\gamma_2^2 - \gamma_1^2 = (\gamma_2 - \gamma_1)(\gamma_2 + \gamma_1)$, we get

$$\gamma_2 + \gamma_1 = 1.$$

Golden section search

Substituting now gives $\gamma_2^2 = \gamma_1 = 1 - \gamma_2$, so that γ_2 solves

$$\gamma_2^2 + \gamma_2 - 1 = 0.$$

Looks familiar?

Golden section search

Substituting now gives $\gamma_2^2 = \gamma_1 = 1 - \gamma_2$, so that γ_2 solves

$$\gamma_2^2 + \gamma_2 - 1 = 0.$$

Looks familiar?

Roots are $-1/2 \pm \sqrt{5/4}$, so that

$$\gamma_2 = \frac{\sqrt{5}-1}{2} = \frac{\sqrt{5}-1}{2}\frac{\sqrt{5}+1}{\sqrt{5}+1} = \frac{2}{\sqrt{5}+1} = \frac{1}{\phi} \approx 0.61803$$

is the reciprocal value of the golden ratio ϕ!

Similarly,

$$\gamma_1 = 1 - \gamma_2 = 1 - \frac{1}{\phi} = 1 - (\phi - 1) = 2 - \phi \approx 0.38197.$$

Golden section search

Good to know (so that we can impress friends and family).

Golden section search

Good to know (so that we can impress friends and family).

One can show that using the golden section proportions provides optimal shrinkage of the interval length.

Golden section search

Good to know (so that we can impress friends and family).

One can show that using the golden section proportions provides optimal shrinkage of the interval length.

Text book and lecture notes contain a simple implementation.

Golden section search

Good to know (so that we can impress friends and family).

One can show that using the golden section proportions provides optimal shrinkage of the interval length.

Text book and lecture notes contain a simple implementation.

In real life, just use `optimize()`.

Golden section search

Good to know (so that we can impress friends and family).

One can show that using the golden section proportions provides optimal shrinkage of the interval length.

Text book and lecture notes contain a simple implementation.

In real life, just use `optimize()`.

With the function we used as example:

```
R> f <- function(x) - (x - 0.5) * (x - 2.25) + ifelse(x < 1, -1, 0)
```

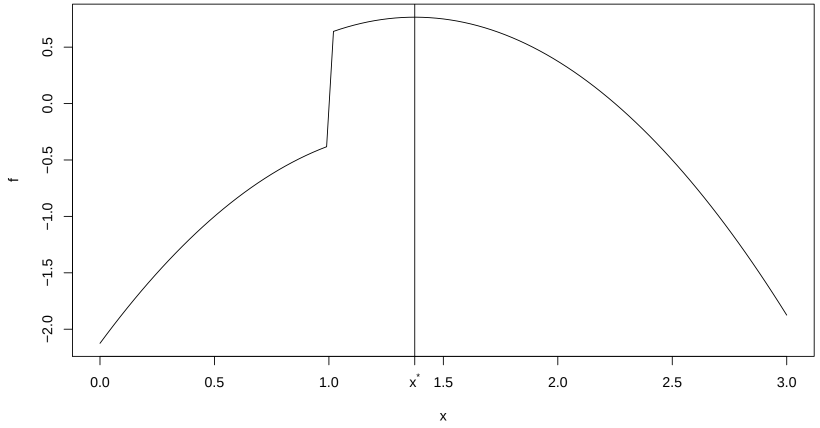
Golden section search

```
R> optimize(f, c(0, 3), maximum = TRUE)

$maximum
[1] 1.375

$objective
[1] 0.765625
```

Golden section search



Golden section search

Things to note:

- Golden section search is only guaranteed to work if f is unimodal (for max).

Golden section search

Things to note:

- Golden section search is only guaranteed to work if f is unimodal (for max).
- By default, `optimize` does minimization and not maximization.

Golden section search

Things to note:

- Golden section search is only guaranteed to work if f is unimodal (for max).
- By default, `optimize` does minimization and not maximization.
- `optimize` only works for functions $f : \mathbb{R} \to \mathbb{R}$. Does not need derivatives: blessing or curse?

Example: MLE for the Poisson distribution

The density of the Poisson distribution with parameter λ at (non-negative integer) x is

$$\frac{\lambda^x}{x!} e^{-\lambda}$$

Example: MLE for the Poisson distribution

The density of the Poisson distribution with parameter λ at (non-negative integer) x is

$$\frac{\lambda^x}{x!} e^{-\lambda}$$

Suppose we observe a sample $x_1, \ldots, x_n$ from a Poisson distribution with unknown λ. How can we estimate λ from that sample?

Example: MLE for the Poisson distribution

The density of the Poisson distribution with parameter λ at (non-negative integer) x is

$$\frac{\lambda^x}{x!} e^{-\lambda}$$

Suppose we observe a sample $x_1, \dots, x_n$ from a Poisson distribution with unknown λ. How can we estimate λ from that sample?

We'll learn how to do this in the Statistics 2 course.

Example: MLE for the Poisson distribution

The preferred method is *maximum likelihood estimation* (MLE): one forms the likelihood function

$$L(\lambda|x_1,\ldots,x_n) = \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}$$

and maximizes this over λ.

Usually, one actually takes the log-likelihood because this is more convenient to work with.

Example: MLE for the Poisson distribution

Up to an additive constant, the log-likelihood is

$$LL(\lambda|x_1,\dots,x_n) = \sum_{i=1}^{n}(x_i \log(\lambda) - \lambda) = s(x_1,\dots,x_n)\log(\lambda) - n\lambda$$

with $s(x_1,\dots,x_n) = \sum_{i=1}^{n} x_i$.

Example: MLE for the Poisson distribution

Up to an additive constant, the log-likelihood is

$$LL(\lambda|x_1,\dots,x_n) = \sum_{i=1}^{n}(x_i \log(\lambda) - \lambda) = s(x_1,\dots,x_n)\log(\lambda) - n\lambda$$

with $s(x_1,\dots,x_n) = \sum_{i=1}^{n} x_i$.

So for the MLE we have to find $\max_{\lambda>0} s\log(\lambda) - n\lambda$.

The MLE is obviously given by $\hat{\lambda} = s/n = \bar{x}$.

Example: MLE for the Poisson distribution

Up to an additive constant, the log-likelihood is

$$LL(\lambda|x_1,\dots,x_n) = \sum_{i=1}^{n}(x_i \log(\lambda) - \lambda) = s(x_1,\dots,x_n)\log(\lambda) - n\lambda$$

with $s(x_1,\dots,x_n) = \sum_{i=1}^{n} x_i$.

So for the MLE we have to find $\max_{\lambda>0} s\log(\lambda) - n\lambda$.

The MLE is obviously given by $\hat{\lambda} = s/n = \bar{x}$.

Nice as we can do math to find a simple formula.

Example: MLE for the Poisson distribution

Numerically, we can simply do

```
R> mle_pois <- function(x) {
+     LL <- function(lambda) {
+         sum(x) * log(lambda) - length(x) * lambda
+     }
+     optimize(LL, lower = 0, upper = max(x), maximum = TRUE)
+ }
```

(Some finite upper bound is needed.)

Example: MLE for the Poisson distribution

To illustrate:

```
R> x <- rpois(100, 3.2)
R> mle_pois(x)

$maximum
[1] 3.660001

$objective
[1] 108.8715

R> ## Compare to
R> mean(x)

[1] 3.66
```

Multivariate optimization

Newton-type algorithms use the following idea.

Suppose $f : X \to \mathbb{R}$ with $X \subseteq \mathbb{R}^n$ is twice continuously differentiable on the interior of X and attains its max there at some x^*.

Then x^* must be a critical point:

$$\nabla f(x^*) = 0,$$

where

$$\nabla f(x) = \left[\frac{\partial f}{\partial \xi_1}(x), \ldots, \frac{\partial f}{\partial \xi_n}(x) \right]', \qquad x = (\xi_1, \ldots, \xi_n)'$$

is the gradient (and the prime denotes transposition).

Multivariate optimization

We can try finding roots of the gradient using a multivariate version of Newton's method.

First, close to x_n we have

$$\nabla f(x) \approx L_n(x) = \nabla f(x_n) + H_f(x_n)(x - x_n),$$

where

$$H_f(x) = \left[\frac{\partial^2 f}{\partial \xi_i \partial \xi_j}(x) \right]_{1 \le i,j \le n}$$

is the *Hessian* of f at x, i.e., the matrix of all second partial derivatives of f.

Details in the math course.

Multivariate optimization

Now solve $L_n(x) = 0$. Mathematically, this gives the linear system

$$H_f(x_n)(x - x_n) = -\nabla f(x_n)$$

which has solution

$$x - x_n = -H_f(x_n)^{-1}\nabla f(x_n).$$

Multivariate optimization

Now solve $L_n(x) = 0$. Mathematically, this gives the linear system

$$H_f(x_n)(x - x_n) = -\nabla f(x_n)$$

which has solution

$$x - x_n = -H_f(x_n)^{-1}\nabla f(x_n).$$

Thus, one gets the iteration

$$x_{n+1} = x_n - H_f(x_n)^{-1}\nabla f(x_n).$$

Multivariate optimization

Now solve $L_n(x) = 0$. Mathematically, this gives the linear system

$$H_f(x_n)(x - x_n) = -\nabla f(x_n)$$

which has solution

$$x - x_n = -H_f(x_n)^{-1}\nabla f(x_n).$$

Thus, one gets the iteration

$$x_{n+1} = x_n - H_f(x_n)^{-1}\nabla f(x_n).$$

That's the basic idea. Things are really more complicated, but fortunately we can simply use functions like `optim()` or `nlm()`.

Multivariate optimization

Things to note:

- This needs f smooth and the first and second partials of f. We can either provide these in addition to f, or have R approximate these numerically.

Multivariate optimization

Things to note:

- This needs f smooth and the first and second partials of f. We can either provide these in addition to f, or have R approximate these numerically.
- This can only find some local optimum of f, which may not be the global one.

Multivariate optimization

Things to note:

- This needs f smooth and the first and second partials of f. We can either provide these in addition to f, or have R approximate these numerically.
- This can only find some local optimum of f, which may not be the global one.
- Things will always be fine if we maximize a concave function.

Example: MLE for the normal distribution

The density of the normal distribution with parameters μ and σ^2 at x is given by

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Example: MLE for the normal distribution

The density of the normal distribution with parameters μ and σ^2 at x is given by

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Suppose we observe a sample $x_1, \ldots, x_n$ from a normal distribution with unknown μ and σ^2. How can we estimate μ and σ^2 from that sample?

Example: MLE for the normal distribution

The likelihood of a sample $x_1, \ldots, x_n$ is

$$L(\mu, \sigma^2 | x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x_i - \mu)^2}{2\sigma^2} \right)$$

Example: MLE for the normal distribution

The likelihood of a sample $x_1, \ldots, x_n$ is

$$L(\mu, \sigma^2 | x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Up to an additive constant, the log-likelihood is

$$LL(\mu, \sigma^2) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

Example: MLE for the normal distribution

The likelihood of a sample $x_1, \dots, x_n$ is

$$L(\mu, \sigma^2 | x_1, \dots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x_i - \mu)^2}{2\sigma^2} \right)$$

Up to an additive constant, the log-likelihood is

$$LL(\mu, \sigma^2) = -\frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2.$$

This has partials

$$\frac{\partial LL}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu), \qquad \frac{\partial LL}{\partial \sigma^2} = -\frac{n}{2} \frac{1}{\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^{n} (x_i - \mu)^2.$$

Example: MLE for the normal distribution

This has a unique critical point given by

$$\hat{\mu} = \bar{x}, \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where $\bar{x}$ is the mean of x, which thus gives the MLEs.

Note that the MLE of σ^2 is not the sample variance!

Example: MLE for the normal distribution

This has a unique critical point given by

$$\hat{\mu} = \bar{x}, \qquad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

where $\bar{x}$ is the mean of x, which thus gives the MLEs.

Note that the MLE of σ^2 is not the sample variance!

Nice as we can do math to find a simple formula.

Example: MLE for the normal distribution

Numerically, we can simply minimize twice the negative log-likelihood:

```
R> mle_norm <- function(x, p0) {
+     nLL <- function(p) {
+         mu <- p[1]
+         sigmasq <- p[2]
+         length(x) * log(sigmasq) + sum((x - mu) ^ 2) / sigmasq
+     }
+     optim(p0, nLL)
+ }
```

This needs some initial estimate of (μ, σ^2).

Example: MLE for the normal distribution I

To illustrate (note that in R the normal distribution is parametrized by the standard deviation σ and not the variance σ^2):

```
R> x <- rnorm(100, 0.5, 2)
R> mle_norm(x, c(0, 1))

$par
[1] 0.6005534 4.5651644

$value
[1] 251.826

$counts
function gradient
      53       NA

$convergence
```

Example: MLE for the normal distribution II

```
[1] 0

$message
NULL

R> ## Compare to
R> c(mean(x), (1 - 1 / length(x)) * var(x))

[1] 0.6003242 4.5642757
```

Example: MLE for the normal distribution

By default, this actually uses Nelder-Mead, which

> *uses only function values and is robust but relatively slow. It will work reasonably well for non-differentiable functions.*

Example: MLE for the normal distribution

By default, this actually uses Nelder-Mead, which

> *uses only function values and is robust but relatively slow. It will work reasonably well for non-differentiable functions.*

Could also do quasi-Newton: `optim(method = "BFGS")`, which

> *uses function values and gradients to build up a picture of the surface to be optimized.*

(No explicit computation of second partials.)

Example: MLE for the normal distribution

Alternatively, we can use `nlm()` which employs a Newton-type method (and always minimizes), without explicitly providing gradients and Hessians):

```
R> nLL2 <- function(p, x) {
+     mu <- p[1]
+     sigmasq <- p[2]
+     length(x) * log(sigmasq) + sum((x - mu) ^ 2) / sigmasq
+ }
```

Example: MLE for the normal distribution

Alternatively, we can use `nlm()` which employs a Newton-type method (and always minimizes), without explicitly providing gradients and Hessians):

```
R> nLL2 <- function(p, x) {
+     mu <- p[1]
+     sigmasq <- p[2]
+     length(x) * log(sigmasq) + sum((x - mu) ^ 2) / sigmasq
+ }
```

Starting from $(0, 1)$ again:

Example: MLE for the normal distribution

```
R> nlm(nLL2, c(0, 1), x)

$minimum
[1] 251.826

$estimate
[1] 0.6003227 4.5642620

$gradient
[1] -4.237677e-05 -5.506546e-05

$code
[1] 1

$iterations
[1] 15
```