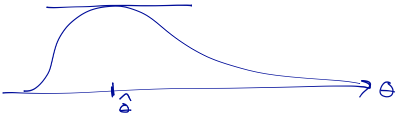
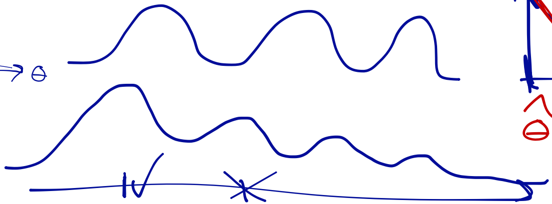
Optimisation for parameter estimation

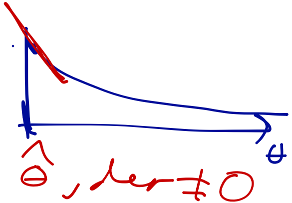
- In statistics, we often want to find the combination $\widehat{\boldsymbol{\theta}}$ of parameter values $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ that maximises the *likelihood function* $L(\boldsymbol{y}; \boldsymbol{\theta})$.
- In special cases, we can use analysis to find closed form expressions for $\widehat{\boldsymbol{\theta}}$.
  Example: If $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ are independent observations of $y_i \sim \mathsf{N}(\mu, \sigma^2)$, the likelihood is

$$L(\boldsymbol{y}; \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - \mu)^2}{2\sigma^2}\right]$$

which is maximised by $\widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\widehat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{\mu})^2$.



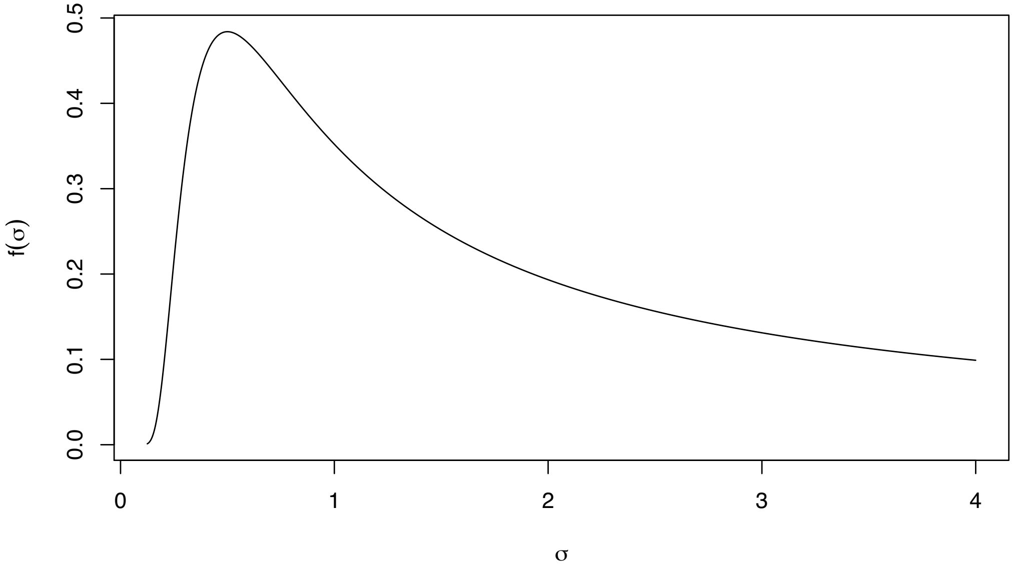




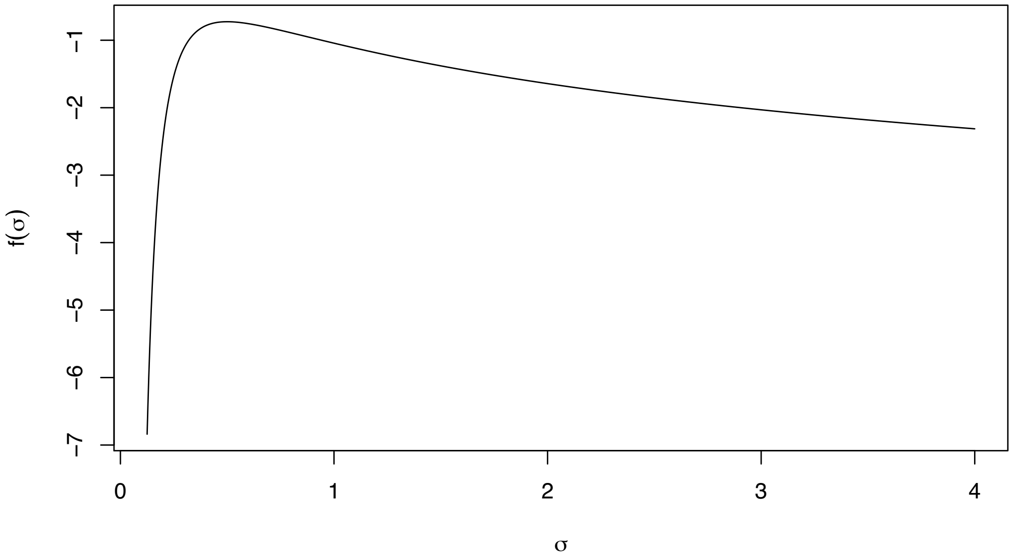
Optimisation for parameter estimation

- For more complicated models, it may be difficult or impossible to find a solution by hand.
- Example: Change the example model by letting σ^2 depend on a covariate, e.g. as $\log(\sigma_i) = \theta_1 + x_i\theta_2$.
  Now, the y_i are independent realisations from $y_i \sim \mathsf{N}(\mu, \sigma_i^2)$.
- This *log-linear* model for the standard deviation doesn't provide a simple/analytical maximum likelihood solution to finding $\widehat{\theta}_1$ and $\widehat{\theta}_2$.
- We need *numerical* optimisation methods!
- We usually convert the likelihood function into a related *target function* $f(\boldsymbol{\theta})$ that is then *minimised*.

Target function $f(\sigma) = L(\boldsymbol{y}; \sigma)$; has inflexion points

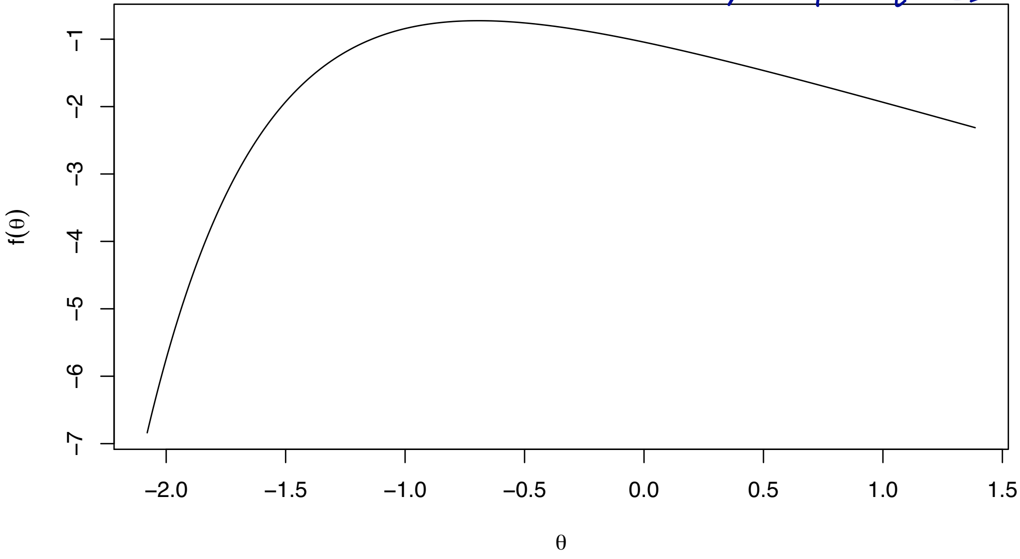


Target function $f(\sigma) = \log L(\boldsymbol{y}; \sigma)$; is very skewed



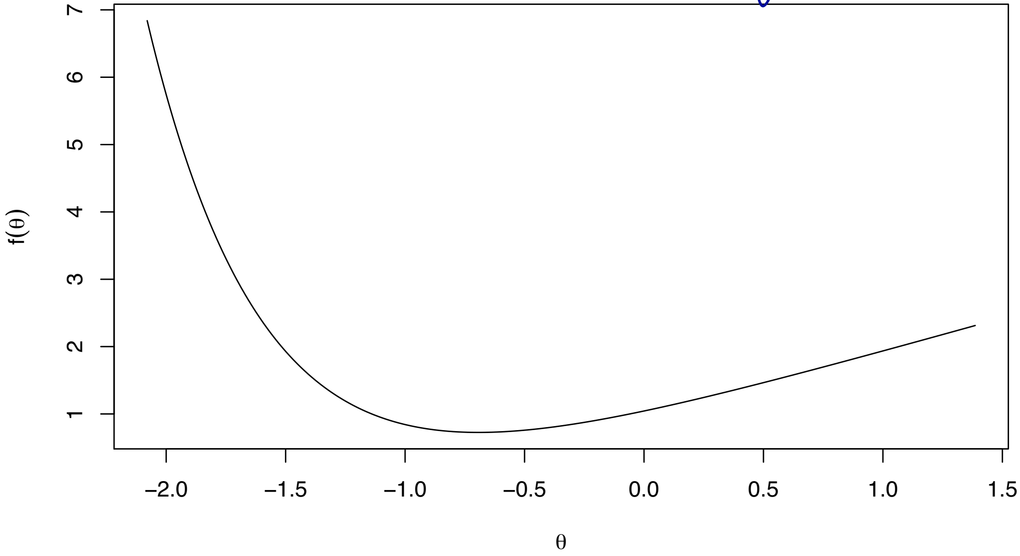
Target function $f(\theta) = \log L(\boldsymbol{y}; \sigma = \mathrm{e}^{\theta})$; have theory for *minimisation*

$\sigma_i = \exp(\theta_1 + x_i \theta_2)$



Target function $f(\theta) = -\log L(\boldsymbol{y}; \sigma = e^{\theta})$; the *negative log-likelihood*

"negated"



Searching for a minimum

Let $\boldsymbol{g}(\boldsymbol{\theta})$ be the gradient vectors of $f(\cdot)$, i.e. $g_j(\boldsymbol{\theta}) = \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_j}$. At a minimum, $\|\boldsymbol{g}(\boldsymbol{\theta})\| = 0$.

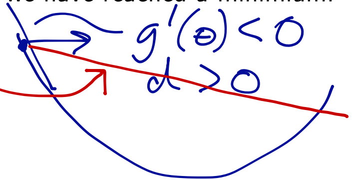
Local minimum search algorithm

Start at some $\boldsymbol{\theta}^{[0]}$, and iterate over $\boldsymbol{\theta}^{[k]}$, $k = 0, 1, 2, \ldots$:

1. Find a *descent direction* vector $\boldsymbol{d}^{[k]}$ from $\boldsymbol{\theta}^{[k]}$. This means that $\boldsymbol{g}(\boldsymbol{\theta}^{[k]})^\top \boldsymbol{d}^{[k]} < 0$.
2. Perform a *line search*, by finding a *step scaling* $\alpha_k > 0$ such that the new f value is sufficiently improved:
   $f(\boldsymbol{\theta}^{[k]} + \alpha_k \boldsymbol{d}^{[k]}) < f(\boldsymbol{\theta}^{[k]}) + \epsilon\, \alpha_k \boldsymbol{g}(\boldsymbol{\theta}^{[k]})^\top \boldsymbol{d}^{[k]}$
   for some fixed $0 < \epsilon < 1$ (can be small, e.g. 10^{-3})
3. Let $\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^{[k]} + \alpha_k \boldsymbol{d}^{[k]}$

Terminate the iteration when either no improvement is found or we have reached a minimium:

$$
\begin{aligned}
k &> \text{maximum allowed iteration steps}, \\
\|\boldsymbol{\theta}^{[k+1]} - \boldsymbol{\theta}^{[k]}\| &< \text{tol}_x, \\
f(\boldsymbol{\theta}^{[k]}) - f(\boldsymbol{\theta}^{[k+1]}) &< \text{tol}_f, \text{ or} \\
\|\boldsymbol{g}(\boldsymbol{\theta}^{[k]})\| &< \text{tol}_g.
\end{aligned}
$$



A simple and practical line search method

Given a valid descent direction $\boldsymbol{d}^{[k]}$ we know that an $\alpha_k > 0$ exists, such that the function value at $\boldsymbol{\theta}^{[k]} + \alpha_k \boldsymbol{d}^{[k]}$ is lower than at the starting point, $f(\boldsymbol{\theta}^k)$.

A simple *inexact line search* method:

1. Let $\alpha_k = 1$.
2. Stop if $f(\boldsymbol{\theta}^{[k]} + \alpha_k \boldsymbol{d}^{[k]}) < f(\boldsymbol{\theta}^{[k]}) + \epsilon\, \alpha_k \boldsymbol{g}(\boldsymbol{\theta}^{[k]})^\top \boldsymbol{d}^{[k]}$.
3. Otherwise, divide α_k by 2, and go back to step 2.

Provided that $0 < \epsilon < 1$, this iteration will terminate.

Gradient descent direction with adaptive step length

The most basic descent direction is the reverse gradient:

$$\boldsymbol{d}^{[k]} = -\gamma_k \boldsymbol{g}(\boldsymbol{\theta}^{[k]})/\|\boldsymbol{g}(\boldsymbol{\theta}^{[k]})\|,$$

where $\gamma_k > 0$ is the *proposed step length*. A fixed γ_k is inefficient.
In the first step, let $\gamma_0 = 1$, and for $k = 1, 2, \ldots$,
if $\alpha_{k-1} = 1$ (the proposed length was OK), let $\gamma_k = 3\gamma_{k-1}/2$ (try a longer step next time),
otherwise $\gamma_k = \alpha_{k-1}\gamma_{k-1}$ (reuse the latest actual accepted step length).

Newton optimisation

A second order Taylor series approximation of $f(\cdot)$ contains useful information about the size shape of the target function.

Let $\boldsymbol{H}(\boldsymbol{\theta})$ be the second order derivative matrix (or *Hessian*) of $f(\cdot)$, with $H_{ij}(\boldsymbol{\theta}) = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j}$.

Newton search direction

The quadratic approximation of $f(\cdot)$,

$$f(\boldsymbol{\theta} + \boldsymbol{d}) \approx f(\boldsymbol{\theta}) + \boldsymbol{g}(\boldsymbol{\theta})^\top \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^\top \boldsymbol{H}(\boldsymbol{\theta})\boldsymbol{d},$$

constructed at $\boldsymbol{\theta}^{[k]}$, is minimised by taking a step

$$\boldsymbol{d}^{[k]} = -\boldsymbol{H}(\boldsymbol{\theta}^{[k]})^{-1}\boldsymbol{g}(\boldsymbol{\theta}^{[k]})$$

if the Hessian is positive definite (has strictly positive eigenvalues).

Bad news: The Hessian $\boldsymbol{H}$ is not always positive definite far away from the minimum.
Good news: Replacing $\boldsymbol{H}$ by *any* positive definite matrix leads to a descent direction.
Solution: Find practical approximations to $\boldsymbol{H}$ that are positive definite. Example: *BFGS*

Quasi-Newton method example: BFGS

One of the most popular *Quasi-Newton* methods requires only function values and gradients.

Broyden-Fletcher-Goldfarb-Shanno (BFGS)

The BFGS method can be formulated to work directly with an approximation to $\boldsymbol{H}(\boldsymbol{\theta})^{-1}$, removing the need to solve a linear system to find the descent direction.

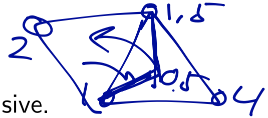
1. Let $\boldsymbol{B}^{[0]}$ be a guess of $\boldsymbol{H}(\boldsymbol{\theta}^{[0]})^{-1}$.
2. For each k, compute the step $\boldsymbol{a}_k = \alpha_k \boldsymbol{d}^{[k]}$ from line search using the search direction $\boldsymbol{d}^{[k]} = -\boldsymbol{B}^{[k]}\boldsymbol{g}(\boldsymbol{\theta}^{[k]})$.
3. Compute $\boldsymbol{b}_k = \boldsymbol{g}(\boldsymbol{\theta}^{[k]} + \boldsymbol{a}_k) - \boldsymbol{g}(\boldsymbol{\theta}^{[k]})$, and update the $\boldsymbol{B}$ matrix:

$$\boldsymbol{B}^{[k+1]} = \boldsymbol{B}^{[k]} + \frac{\boldsymbol{a}_k^\top \boldsymbol{b}_k + \boldsymbol{b}_k^\top \boldsymbol{B}^{[k]} \boldsymbol{b}_k}{(\boldsymbol{a}_k^\top \boldsymbol{b}_k)^2} \boldsymbol{a}_k \boldsymbol{a}_k^\top - \frac{\boldsymbol{B}^{[k]} \boldsymbol{b}_k \boldsymbol{a}_k^\top + \boldsymbol{a}_k \boldsymbol{b}_k^\top \boldsymbol{B}^{[k]}}{\boldsymbol{a}_k^\top \boldsymbol{b}_k}$$

   The equations guarantee that $\boldsymbol{B}^{[k]}$ stays positive definite.

The initial $\boldsymbol{B}^{[0]}$ is often chosen to be either proportional to an identity matrix, or a diagonal matrix based on the diagonal elements of $\boldsymbol{H}(\boldsymbol{\theta}^{[0]})$, which costs only around twice as much as a single gradient calculation.

Computational cost considerations



- Computing f, $\boldsymbol{g}$, and $\boldsymbol{H}$ is often expensive.
- When using finite differences, the cost of $\boldsymbol{g}$ is proportional to m, and the cost of $\boldsymbol{H}$ is proportional to m^2.
- We want to balance the cost per iteration with the number of iterations required to reach the minimum.
- It's usually not worth computing the actual second order derivatives, unless we can find a closed form positive definite Hessian approximation.
- From the theory of negative log-likelihood functions it is known that the *expected Hessian*, $\widetilde{\boldsymbol{H}}(\boldsymbol{\theta}) = \mathsf{E}_{\boldsymbol{y}|\boldsymbol{\theta}}[\boldsymbol{H}(\boldsymbol{\theta})]$ is always positive definite. Using this in place of the *observed Hessian* $\boldsymbol{H}(\boldsymbol{\theta})$ is called *Fisher Scoring*.
- Conclusion:
  - For smooth target functions with cheap gradients, BFGS or Fisher Scoring is preferable
  - For less smooth target functions or expensive gradients, the Simplex method is preferable (robust and uses only f values, see Computer Lab 2; this is also the default method in `optim()` in R).
- In Computer Lab 2, you will experiment with different target functions and optimisation methods in a graphical interactive R tool.

Fisher Scoring example

Let $y_i \sim \mathsf{N}(\mu(\boldsymbol{\theta}), \sigma(\boldsymbol{\theta})^2)$ (independent), $\mu(\boldsymbol{\theta}) = \theta_1$, $\sigma(\boldsymbol{\theta}) = \mathrm{e}^{\theta_2}$. The negative log-likelihood is

$$f(\theta_1, \theta_2) = \frac{n}{2} \log(2\pi) + n\theta_2 + \frac{1}{2\mathrm{e}^{2\theta_2}} \sum_{i=1}^{n} (y_i - \theta_1)^2$$

The gradient elements are

$$g_1(\theta_1, \theta_2) = -\frac{1}{\mathrm{e}^{2\theta_2}} \sum_{i=1}^{n} (y_i - \theta_1), \qquad g_2(\theta_1, \theta_2) = n - \frac{1}{\mathrm{e}^{2\theta_2}} \sum_{i=1}^{n} (y_i - \theta_1)^2$$

The observed and expected Hessian elements are

$$H_{11}(\theta_1, \theta_2) = \frac{n}{\mathrm{e}^{2\theta_2}}, \quad H_{12}(\theta_1, \theta_2) = \frac{2}{\mathrm{e}^{2\theta_2}} \sum_{i=1}^{n} (y_i - \theta_1), \quad H_{22}(\theta_1, \theta_2) = \frac{2}{\mathrm{e}^{2\theta_2}} \sum_{i=1}^{n} (y_i - \theta_1)^2,$$

$$\widetilde{H}_{11}(\theta_1, \theta_2) = \frac{n}{\mathrm{e}^{2\theta_2}}, \quad \widetilde{H}_{12}(\theta_1, \theta_2) = 0, \quad \widetilde{H}_{22}(\theta_1, \theta_2) = 2n.$$

The expected Hessian is diagonal with positive elements (so clearly positive definite) and much cheaper to compute, since the observations are not involved!