Statistical Theory and Modeling (ST2601)
Lecture 9 - Statistical Information. Numerical Maximum Likelihood.

Mattias Villani

Department of Statistics
Stockholm University

mattiasvillani.com @matvil @matvil mattiasvillani

Overview

- The information in a likelihood function
- Maximum likelihood estimator in large samples
- Numerical maximum likelihood

The Big Picture

- Four stages of learning **modern statistical learning**.

1. **Understand probability and statistical models**.
2. **Mathematical derivation** of the MLE
   - Write up the log-likelihood $\ell(\theta)$
   - Calculate derivative $\ell'(\theta)$
   - Solve for the MLE from $\ell'(\theta) = 0$
3. **Numerical optimization** for the MLE
   - Code up the log-likelihood $\ell(\theta)$
   - Use **automatic differentiation** to find $\ell'(\theta)$
   - Solve for the MLE using a **numerical optimizer**.
4. **Probabilistic programming languages** (**PPL**) (Stan etc)
   - Express the statistical model, almost like in textbooks.
   - Let the framework to all the work for you.

- Learning 1 $\rightarrow$ 2 $\rightarrow$ 3$\rightarrow$ 4 makes Stage 4 understandable and explainable to clients, helps in debugging, allows you to go beyond models in manual.
- Stage 3 gives freedom when PPL doesn't do what you need.

iid Poisson model in Stan

- **IID Poisson** model

$$Y_1, \ldots, Y_N | \lambda \overset{\text{iid}}{\sim} \text{Pois}(\lambda)$$

```
data {
  int<lower = 0> N;
  int<lower = 0> y[N];
}
parameters {
  real<lower = 0> lambda;
}
model {
  lambda ~ gamma(1, 1);
  y ~ poisson(lambda);
}
generated quantities {
  int<lower = 0> y_tilde = poisson_rng(lambda);
}
```

Negative Binomial regression in Turing.jl

```
# Negative binomial regression
@model function negbinomialReg(y, X, τ, μ₀, σ₀)
    p = size(X,2)
    β ~ filldist(Normal(0, τ), p)
    λ = exp.(X*β)
    ψ ~ LogNormal(μ₀, σ₀)
    n = length(y)
    for i in 1:n
        y[i] ~ NegativeBinomial(ψ, ψ/(ψ + λ[i]))
    end
end
```

How informative is my data about the parameters?

- **Probabilistic model** $p(Y_1, \ldots, Y_n|\theta)$ with **parameter** θ.
- Example: iid Poisson model:

$$Y_1, \ldots, Y_N|\lambda \overset{\text{iid}}{\sim} \text{Pois}(\lambda)$$

- **Likelihood function** (assuming independent data)

$$L(\theta) = p(y_1, \ldots, y_n|\theta) = \prod_{i=1}^{n} p(y_i|\theta)$$

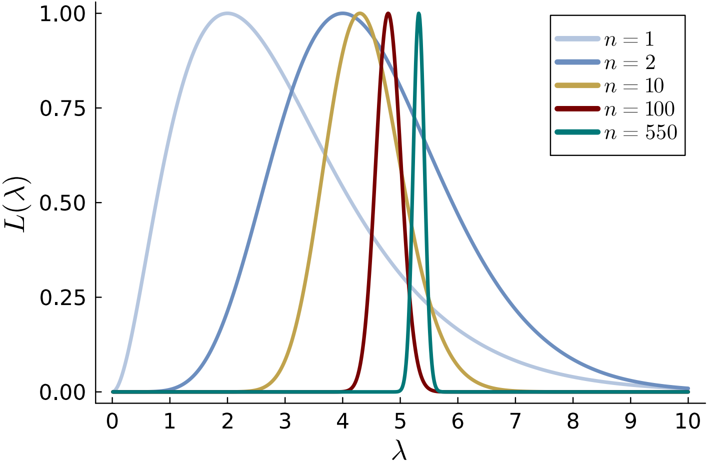
- **Log-likelihood function**

$$\ell(\theta) = \log L(\theta)$$

- **Observed information**: **Given a dataset** $y_1, \ldots, y_n$, how much information is there about θ?
- **Expected information**: **Before collecting data**, how information can I expected to get about θ?

Observed information

- **Given a dataset**, how much information is there about θ?
- **How peaked** is the likelihood function?



Observed information

- The **second derivative** measures the curvature of a function

$$f''(x) = \frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) = \frac{\mathrm{d}}{\mathrm{d}x} f'(x)$$

- **Observed information** from n observations

$$J_n(\hat{\theta}) = -\ell''(\hat{\theta})$$

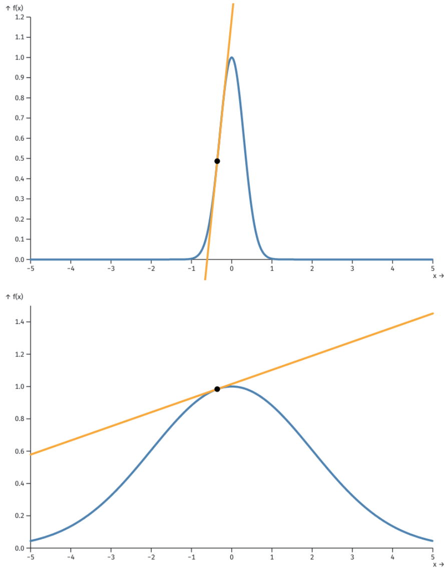
  where $\hat{\theta}$ is the maximum likelihood estimate (MLE).
- Often written as

$$\left. \frac{\mathrm{d}^2}{\mathrm{d}\theta^2} \ell(\theta) \right|_{\theta=\hat{\theta}}$$

- Why a negative sign? The second derivative is negative at the maximum. We like information to be a positive number.

- Why the **log**-likelihood function? Suggested by likelihood theory. Log-likelihood is approx quadratic in large samples.

Second derivative measures curvature



Observed information in the iid Poisson model

- IID Poisson model

$$Y_1, \ldots, Y_N | \lambda \overset{\text{iid}}{\sim} \text{Pois}(\lambda)$$

- Log-likelihood

$$\ell(\lambda) = \sum_{i=1}^{n} \log p(y_i|\lambda)$$

- Log density for ith observation

$$\log p(y_i|\lambda) = \log\left(\frac{\lambda^{y_i} e^{-\lambda}}{y_i!}\right) = y_i \log(\lambda) - \lambda - \log(y_i!)$$

- Log-likelihood

$$\begin{aligned}\ell(\lambda) &= \sum_{i=1}^{n} \left(y_i \log(\lambda) - \lambda - \log(y_i!)\right) \\ &= \log(\lambda) \sum_{i=1}^{n} y_i - n\lambda - \sum_{i=1}^{n} \log(y_i!)\end{aligned}$$

Observed information in the iid Poisson model

- Log-likelihood (constants in orange)

$$\ell(\lambda) = \log(\lambda)\sum_{i=1}^{n} y_i - n\lambda - \sum_{i=1}^{n}\log(y_i!)$$

- **First derivative**

$$\ell'(\lambda) = \frac{\sum_{i=1}^{n} y_i}{\lambda} - n$$

- Solving $\ell'(\lambda) = 0$ gives the MLE $\hat{\lambda} = \frac{\sum_{i=1}^{n} y_i}{n} = \bar{y}$.
- **Second derivative**

$$\ell''(\lambda) = -\frac{\sum_{i=1}^{n} y_i}{\lambda^2} = -\frac{n\bar{y}}{\lambda^2}$$

- **Observed information**

$$\mathcal{J}_n(\hat{\lambda}) = -\ell''(\hat{\lambda}) = \frac{n\bar{y}}{\hat{\lambda}^2} = \frac{n\bar{y}}{\bar{y}^2} = \frac{n}{\bar{y}}$$

- Information grows linearly in sample size n. Always true for iid

.

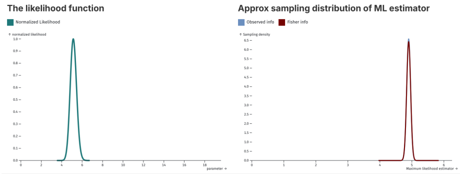
Expected information

- The observed information varies from dataset to dataset.
- **Expected information**: **Before collecting data**, how information can I expected to get about θ?
- The expected information **over all possible datasets**

$$\mathcal{I}_n(\theta) = \mathbb{E}\left(\mathcal{J}_n(\theta)\right)$$

  $\mathcal{J}_n(\theta)$ is the observed information from dataset $(Y_1, \ldots, Y_n)$.
- Also called the **Fisher information**.
- Designing experiments and data collection (**active learning**).
- Also in **sampling distribution of the MLE** in large samples.

Likelihood and Information - widget



Maximum likelihood estimator in large samples

- **Sampling distribution of the MLE** in large datasets

$$\hat{\theta}_n \overset{\text{approx}}{\sim} N\left(\theta, \frac{1}{\mathcal{I}_n(\theta)}\right) \text{ for large } n$$

- MLE is **asymptotically unbiased** (as $n \to \infty$).
- **Asymptotically efficient** (lowest possible variance among unbiased estimators. **Cramér-Rao lower bound**).
- In large samples $\mathcal{I}_n(\theta) \approx \mathcal{J}_n(\hat{\theta})$, so we can use $\mathcal{J}_n(\hat{\theta})$.
- The beauty: a computer can compute $\hat{\theta}_n$ and $\mathcal{J}_n(\hat{\theta}_n)$. 😍

Maximum likelihood in large samples - example

- **MLE sampling distribution for large n**

$$\hat{\theta}_n \overset{\text{approx}}{\sim} N\left(\theta, \frac{1}{\mathcal{J}_n(\hat{\theta}_n)}\right) \text{ for large } n$$

- IID Poisson model

$$\hat{\lambda}_n(Y_1, \ldots, Y_n) \overset{\text{approx}}{\sim} N\left(\lambda, \frac{1}{\mathcal{J}_n(\hat{\lambda}_n)}\right) = N\left(\lambda, \frac{1}{n/\bar{y}}\right) = N\left(\lambda, \frac{\bar{y}}{n}\right)$$

- In this case we can compute the **true sampling variance**

$$\mathbb{V}(\hat{\lambda}) = \mathbb{V}(\bar{Y}) = \frac{\mathbb{V}(Y_i)}{n} = \frac{\lambda}{n}$$

since variance = mean for Poisson.

Multi-parameter case

- Multi-parameter models: $p(y_1, \dots, y_n|\boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)^\top$ is a vector with parameters.
- Example: Negative binomial with $\boldsymbol{\theta} = (r, \mu)^\top$

$$Y_1, \dots, Y_N | r, \mu \overset{\text{iid}}{\sim} \text{NegBin}(r, \mu)$$

- Example: Two-dimensional **observed information matrix**)

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} = \begin{pmatrix} \frac{\partial^2}{\partial \theta_1^2} \ell(\theta_1, \theta_2) & \frac{\partial^2}{\partial \theta_1 \partial \theta_2} \ell(\theta_1, \theta_2) \\ \frac{\partial^2}{\partial \theta_2 \partial \theta_1} \ell(\theta_1, \theta_2) & \frac{\partial^2}{\partial \theta_2^2} \ell(\theta_1, \theta_2) \end{pmatrix}$$

- **Sampling distribution of the MLE** in large samples

$$\hat{\boldsymbol{\theta}}_n \overset{\text{approx}}{\sim} N\left(\boldsymbol{\theta}, \mathcal{J}_n^{-1}(\hat{\boldsymbol{\theta}}_n)\right) \text{ for large } n$$

where N is the multivariate normal distribution and $\mathcal{J}_n^{-1}(\hat{\boldsymbol{\theta}}_n)$ is the matrix inverse of $\mathcal{J}_n(\hat{\boldsymbol{\theta}}_n)$.

Numerical optimization

- Find the maximum of a function $f(x)$

$$x_{\max} = \arg\max_{x \in \mathcal{X}} f(x)$$

  involves **solving for x in $f'(x) = 0$**.

- **Gradient ascent**: find $x_{\max}$ by iterating until convergence:

$$x_{k+1} = x_k + \underbrace{\gamma}_{\text{learning rate}} \cdot \underbrace{f'(x_k)}_{\text{gradient}}$$

- **Newton-Raphson**: find $x_{\max}$ by iterating until convergence:

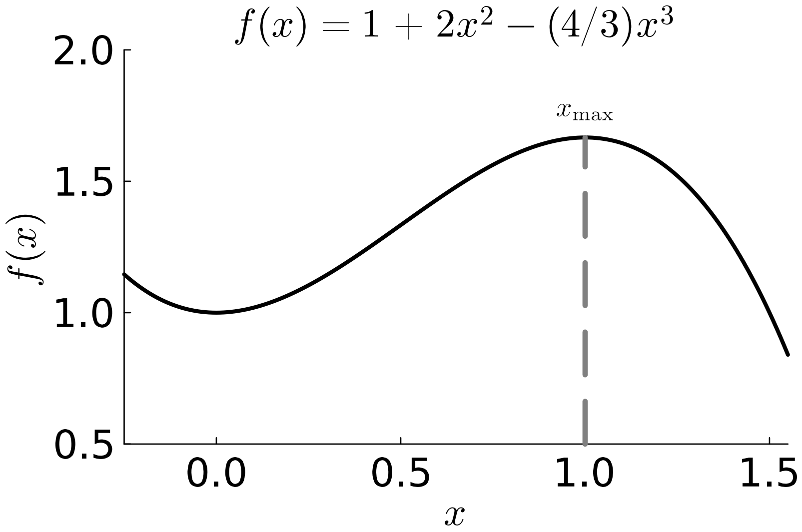
$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

- Multi-dimensional MLE solves $\ell'(\boldsymbol{\theta}) = 0$. Newton-Raphson:

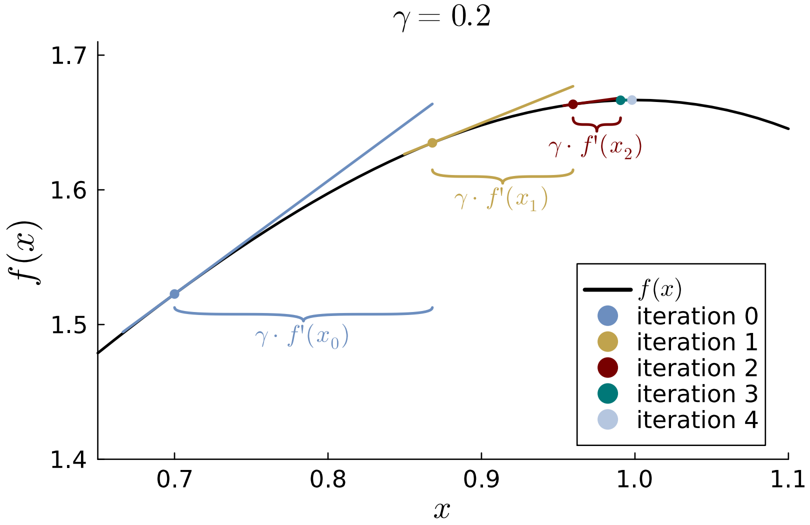
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \underbrace{\mathcal{J}^{-1}(\boldsymbol{\theta}_k)}_{\text{inverse hessian}} \cdot \underbrace{f'(\boldsymbol{\theta}_k)}_{\text{gradient}}$$

- Inverse Hessian cheaply computed from gradients (BFGS).

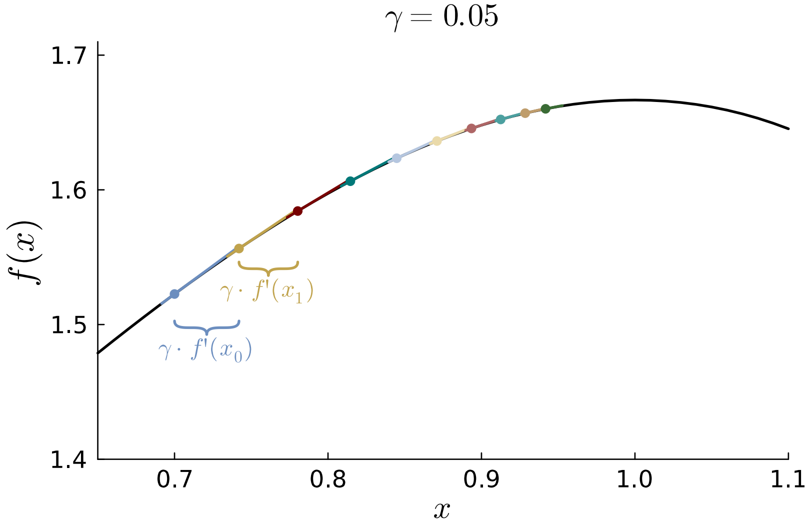
Finding the maximum



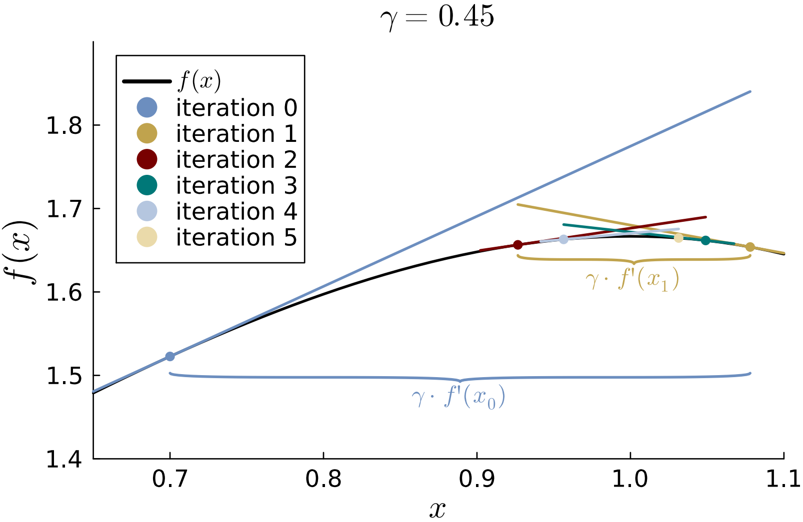
Gradient ascent - good learning rate



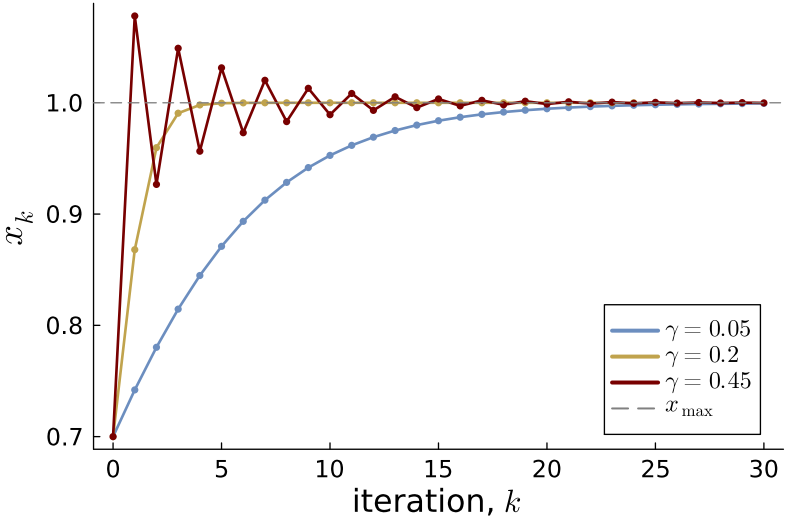
Gradient ascent - too small learning rate



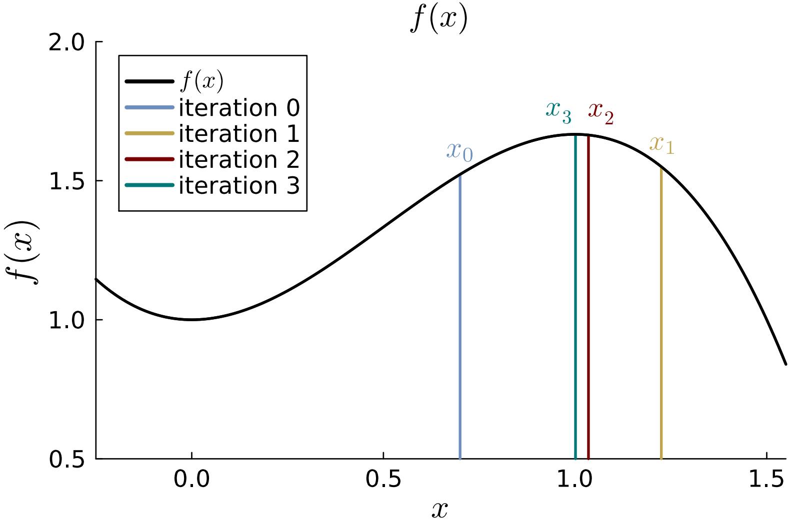
Gradient ascent - too large learning rate



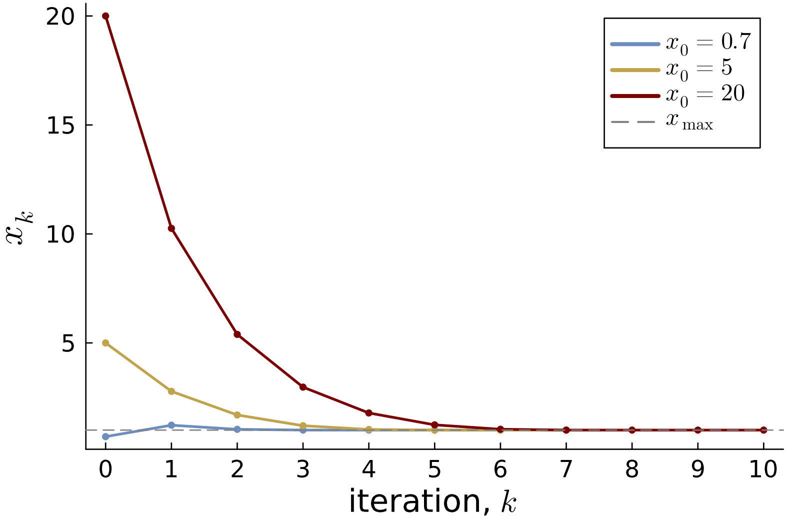
Gradient ascent convergence



Newton-Raphson



Newton-Raphson convergence



Optim in R

```
# Function to maximized. Depends on parameter a
myFunc <- function(x, a){
  funcVal = 1 - (x/a-2)^2
  return (funcVal)
}

# Run optim
a = 1
initVal <- c(3)       # initial guess
OptimResults <- optim(initVal, myFunc, gr=NULL, a,
                      method=c("BFGS"),
                      control=list(fnscale=-1),
                      hessian=TRUE)

OptimResults$par     # maximizer
OptimResults$value   # maximum function value
OptimResults$hessian # second derivative
```