UNIVERSITY OF
COPENHAGEN

Optimization

Computational Statistics

Johan Larsson

Department of Mathematical Sciences, University of Copenhagen

March 31, 2026

Recap: Monte Carlo Methods

Standard Monte Carlo

Estimate $\mathrm{E}\, h(X)$ with $X \sim f$ by

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} h(X_i) \quad \text{with } X_i \overset{iid}{\sim} f.$$

Importance Sampling

Use the "trick" that

$$\mu = \int h(x)f(x)\ dx = \int h(x)\frac{f(x)}{g(x)}g(x)\ dx = \int h(x)w^*(x)g(x)\ dx$$

and estimate

$$\hat{\mu}_n^{IS} = \frac{1}{n}\sum_{i=1}^{n} h(X_i)w^*(X_i) \quad \text{with } X_i \overset{iid}{\sim} g.$$

Recap: Importance Sampling

Normalized Weights

When f is known only up to a normalizing constant, we can still use importance sampling by estimating normalized weights.

Picking a Good Importance Distribution

Pick an importance distribution g that covers the important parts of f, with heavy tails, and from which we can easily sample.

Today

Optimization

What is an optimization problem?

We learn about different kinds of optimization problems.

Convex Optimization

Convex optimization problems are a special class of optimization problems that are easier to solve.

Algorithms

We learn about gradient descent and Newton's method.

Optimization

Optimization

The process of trying to optimize an objective function f.

Typically, we are interested in either **minimizing** or **maximizing** the function.

Since minimizing f is equivalent to maximizing $-f$, we will focus on minimization.

Examples

- Fitting a generalized linear model by minimizing the negative log-likelihood
- Finding the shortest path in a network
- Minimizing variance in a portfolio of financial assets

Standard Form

In *standard form*, an optimization problem is written as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & h_j(x) = 0, \quad j = 1, \dots, k. \end{aligned}$$

where

- f is the objective function,
- g_i are inequality constraints, and
- h_j are equality constraints.

Solution

We write the solution as

$$x^* = \operatorname{argmin}_x f(x)$$

Smoothing Splines

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \left((\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\Omega} \boldsymbol{\beta} \right)$$

Finding the Hyperparameter for Smoothing Splines

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \left((\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\Omega} \boldsymbol{\beta} \right) \\ \text{subject to} \quad & \lambda \geq 0. \end{aligned}$$

Maximum Likelihood

$$\underset{\theta}{\text{minimize}}\,\big(-\log f(\theta \mid x)\big),$$

where $f(\theta \mid x)$ is the likelihood function.

Minimum-Variance Portfolio

$$\begin{array}{ll} \underset{w}{\text{minimize}} & w^T \Sigma w \\ \text{subject to} & \bar{p}^T w \geq r_{\min}, \\ & \mathbf{1}^T w = 1, \end{array}$$

where w are the asset weights, Σ is the covariance matrix of asset returns, $\bar{p}$ is the vector of expected returns, and $r_{\min}$ is the minimum acceptable return.

Example: Ordinary Least Squares (OLS) Regression

The problem is to

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad f(\beta) = \frac{1}{2}\|y - X\beta\|_2^2.$$

Solution

The solution is

$$\beta^* = \text{argmin}_{\beta \in \mathbb{R}^p} f(\beta).$$

Types of Optimization Problems

Convexity

- **Convex**
- Quasiconvex
- Nonconvex

Smoothness

- **Smooth**
- Nonsmooth

Constraints

- **Unconstrained**
- Constrained

Standard form:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & a_j^T x = b_j, \quad j = 1, \ldots, k, \end{array}$$

where f and g_i are convex.

Convex Functions

A function f is convex iff for all $x_1, x_2 \in \operatorname{dom} f$ and $\lambda \in [0, 1]$

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

This is known as **Jensen's inequality**.

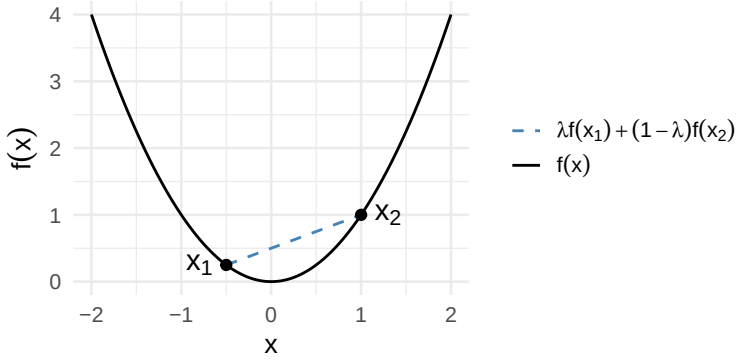


Figure 1: Jensen's inequality for a convex function.

First-Order Condition for Convexity

If f is **differentiable**, then the condition is equivalent to

$$f(x_1) \geq f(x_2) + \nabla f(x_2)^T (x_1 - x_2).$$

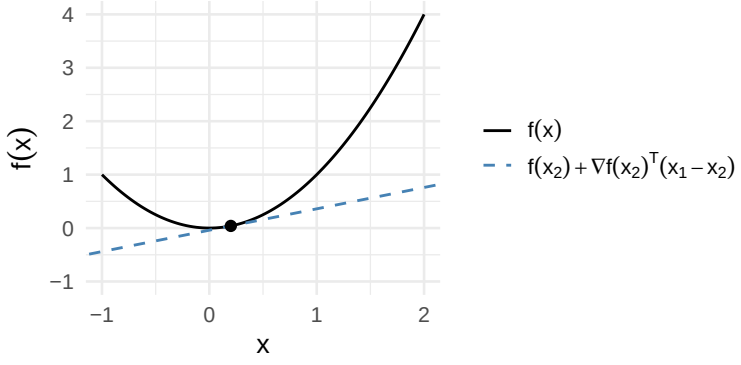


Figure 2: First-order condition for convexity.

Optimality

Any (local) minimum is **global** (but not necessarily unique).

First-Order Condition
For differentiable f, x^* is optimal iff

$$\nabla f(x^*)^T(x - x^*) \geq 0 \quad \text{for all} \quad x \in C,$$

where C is the feasible set.

Second-Order Condition
If f is twice-differentiable, then x^* is optimal iff

$$\nabla^2 f(x^*) \succeq 0.$$

Quasiconvex Optimization Problems

Standard Form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & A^T x = b, \end{array}$$

where g_i are convex but f is **quasiconvex**.

Examples

- Utility functions (economics)
- The floor function $f(x) = \lfloor x \rfloor$

Algorithms

Won't cover here, but for instance the **bisection** method.

Nonconvex Optimization Problems

- Generally hard to solve.
- No standard form (can be anything).
- Local minima are not necessarily global.

Examples

- Deep learning
- Hyperparameter tuning (cross-validation)
- Combinatorial problems

Algorithms

Won't cover here, but typically **stochastic algorithms** (next week) or **global optimization methods** (not in this course).

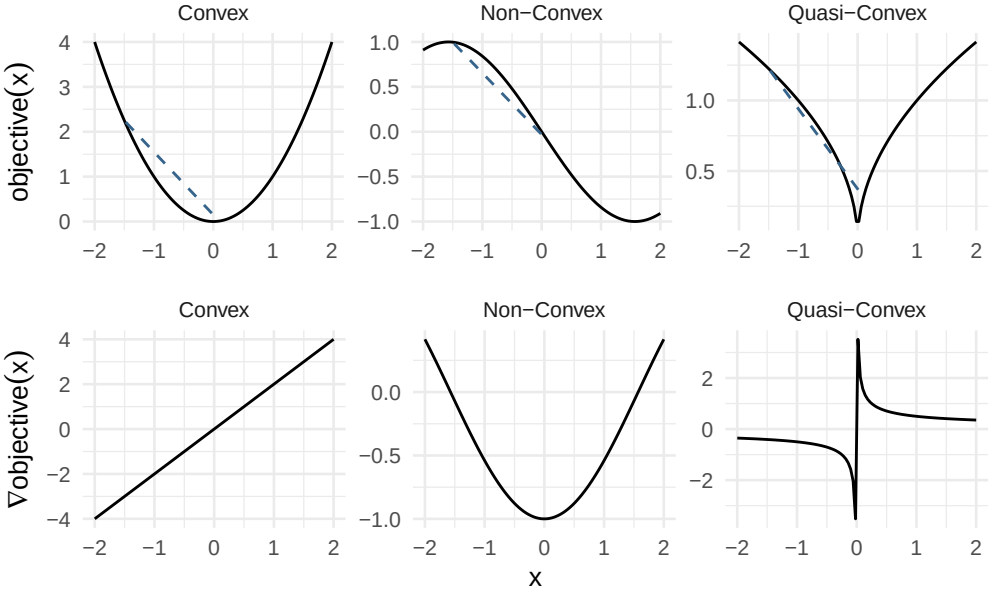


Figure 3: x^2 (convex), $\sin(x + \pi)$ (non-convex), and $\sqrt{|x|}$ (quasi-convex).

Solving Optimization Problems

Types of Optimization Methods

Direct (Exact) Methods

Provides a solution up to machine precision.

Examples: solving a linear system of equations

Iterative Optimization

Incrementally update optimization variable until some convergence criterion is met.

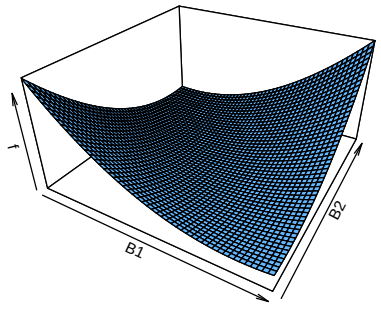
Examples: gradient descent, hill-climbing

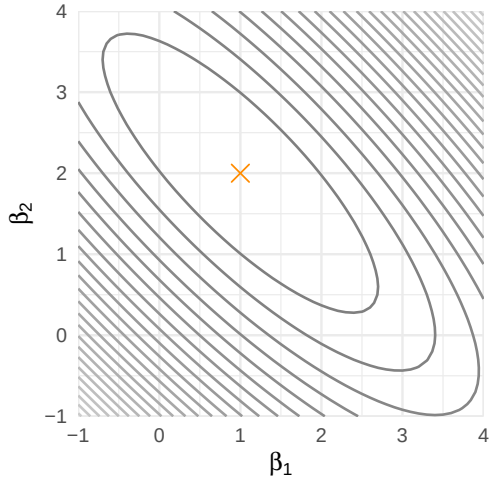
Running Example: OLS

We are going to use a simple example of OLS as a running example.

We will let $p = 2$, and generate some data from a multivariate normal distribution:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix})$$


f
B1
B2


4
3
2
1
0
−1
−1
0
1
2
3
4
β2
β1

Analytical solution, up to machine precision.

Example: OLS

Solve

$$X^T X\beta = X^T y$$

for $\beta \in \mathbb{R}^p$.

How do you solve this in practice? In R you call

```
solve(crossprod(X), crossprod(X, y))
```

but what happens under the hood?

Iterative Methods

Update a current best guess iteratively until it is "good enough".

Taxonomy by Order

Can be categorized by order, in terms of the information used

- **0-order**: only use function value ($f(x)$)
- **first-order**: also use first derivative ($f'(x)$)
- **second-order**: also use second derivative ($f''(x)$)

Zero-Order Methods

Use only the function value.

Examples

- Hill climbing
- Bisection
- Grid search (cross validation)

First-Order Methods

Use the gradient of the function (**first** derivative).

Examples

- **Gradient descent**
- Conjugate gradient

Second-Order Methods

Use the Hessian of the function (**second** derivative).

Examples

- **Newton's Method**
- Trust-region methods

The quintessential first-order method!

Algorithm 1: Gradient descent

Data: Step size $t > 0$

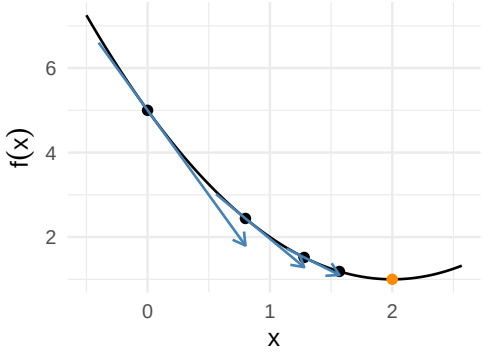
repeat

$\quad x \leftarrow x - t\nabla f(x)$;

until *convergence*;

return x

Converges to a local minimum if f is convex and differentiable and t small enough.



Step Size

But how do we choose t?

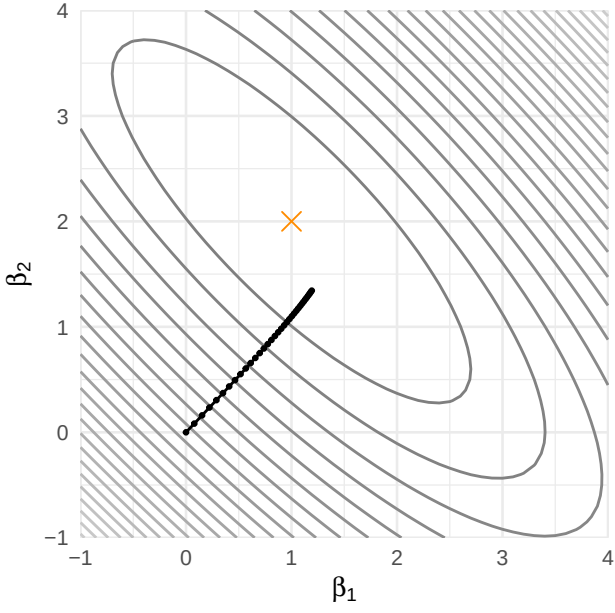


Figure 4: The step size is too small!

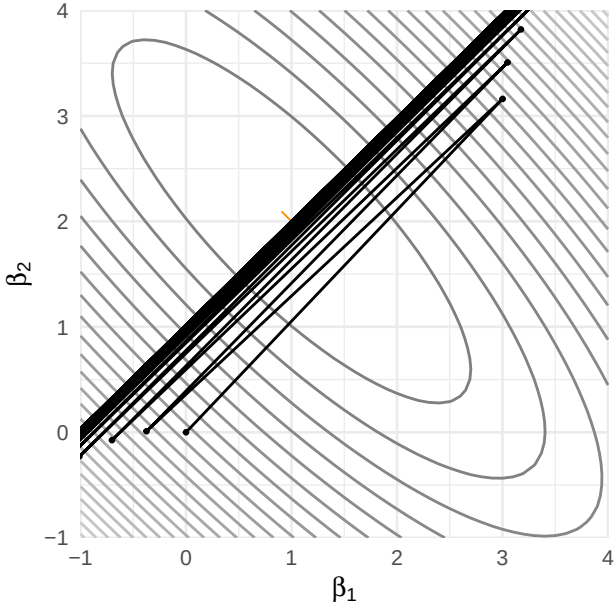


Figure 5: The step size is too large!

Derivation of Update

Take the second-order Taylor expansion of f around x:

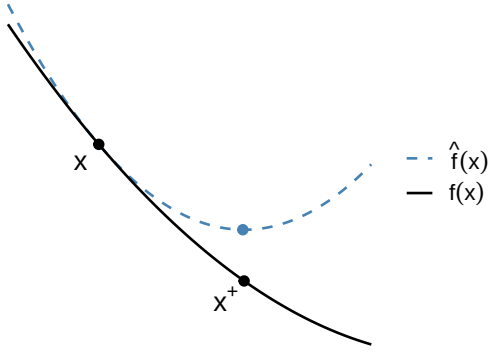
$$f(y) \approx \hat{f}(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(x)(y-x).$$

Replace $\nabla^2 f(x)$ with $\frac{1}{t}I$ and minimize w.r.t y to get

$$x^+ = x - t\nabla f(x).$$

Step Size

But we still need to choose t!



The Descent Lemma (from Taylor's Theorem)

Taylor's Theorem

For any x, y in the domain of f, there exists a c on the line segment between x and y such that

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(c)(y - x).$$

Descent Lemma

Suppose we know that the Hessian is bounded above by LI, i.e.,

$$\nabla^2 f(x) \preceq LI \quad \text{for all } x.$$

Then

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}\|y - x\|_2^2.$$

Lipschitz Constant

The bound that we used in the descent lemma, L, is known as the **Lipschitz constant** of ∇f.

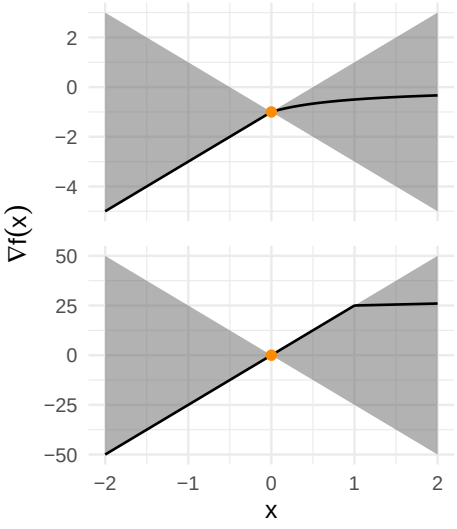
It is defined as follows.

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2.$$

It measures how fast the gradient of f can change, relative to the change in x.

Twice-Differentiable Case
If f is twice-differentiable, then the Lipschitz constant is the smallest L such that

$$\nabla^2 f(x) \preceq LI \quad \text{for all } x.$$



The Connection to Gradient Descent

If we apply the descent lemma with $y = x - t\nabla f(x)$, we get

$$f(x - t\nabla f(x)) \leq f(x) - t\|\nabla f(x)\|_2^2 + \frac{Lt^2}{2}\|\nabla f(x)\|_2^2.$$

This means that if $t \leq 1/L$, then

$$f(x - t\nabla f(x)) \leq f(x) - \frac{t}{2}\|\nabla f(x)\|_2^2.$$

In other words, the function value decreases at each step.

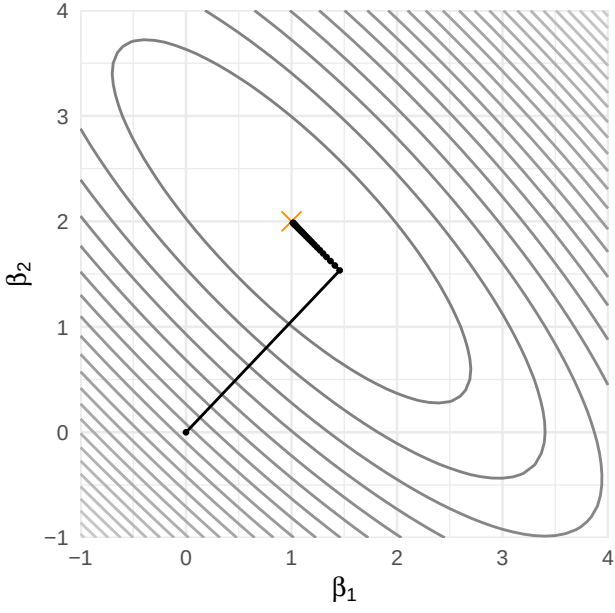


Figure 6: The step size is just right ($t = 1/L$)!

Line Searching

In practice, we may not know L or it may be difficult to compute.

Exact Line Search
It may be tempting to try to solve

$$t^* = \arg_t \min f(x - t\nabla f(x)).$$

But this is typically too expensive. Instead, we typically use an inexact line search, like **backtracking line search**.

Backtracking Line Search

Algorithm 2: Backtracking line search

Data: Initial step size $t_0 > 0$, parameters $\alpha \in (0, 1/2]$ and $\gamma \in (0, 1)$

$t \leftarrow t_0$;

while $f(x - t\nabla f(x)) > f(x) - \alpha t \|\nabla f(x)\|_2^2$ **do**

  $t \leftarrow t\gamma$;

return t

Guarantees that the step size is not too large, and that the function value decreases sufficiently.

Adapts to local curvature of the function.

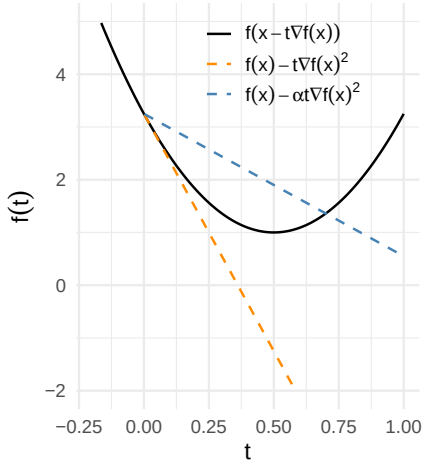


Figure 7: An illustration of backtracking line search.

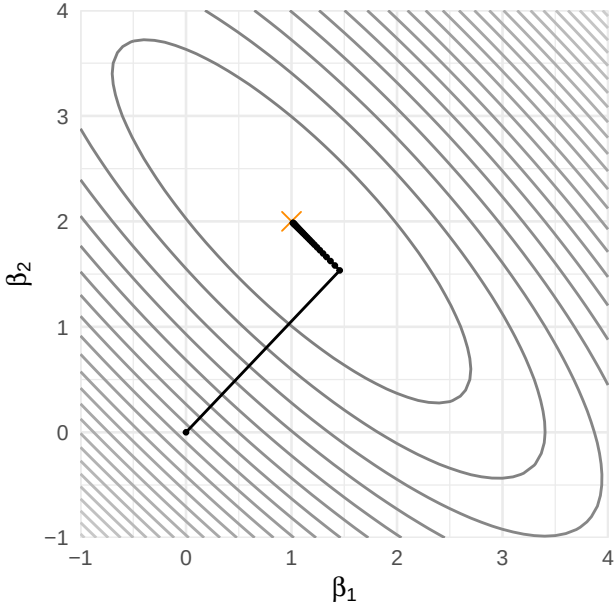


Figure 8: Backtracking line search.

Conditioning

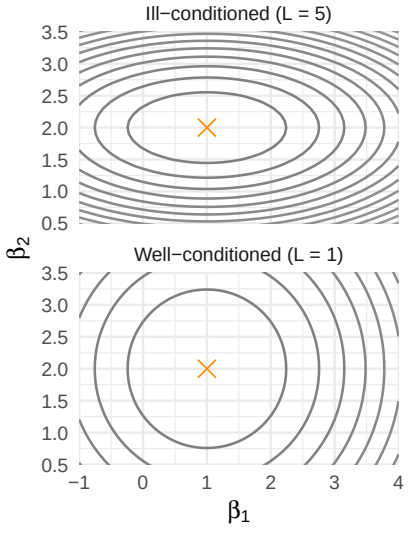
So, L decides how large step size we can take.

But L is based on the largest eigenvalue of $\nabla^2 f(x)$, which means that it may be pessimistic in some directions.

Condition Number
The **condition number** of f is defined as

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

If κ is large, we say that f is **ill-conditioned**. If κ is small, we say that f is **well-conditioned**.



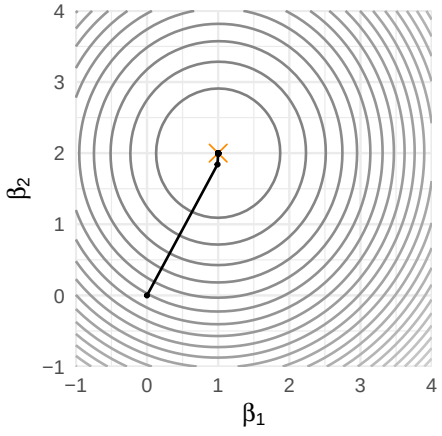


Figure 9: Gradient descent on a well-conditioned problem.

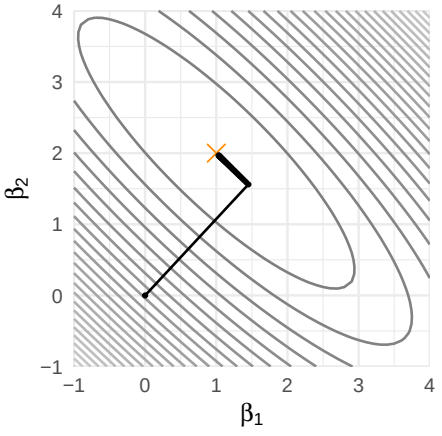


Figure 10: Gradient descent on an ill-conditioned problem.

The problem with gradient descent is that it is sensitive to this kind of conditioning.

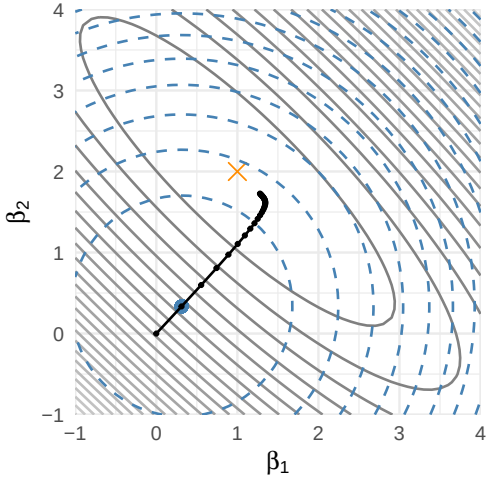


Figure 11: The gradient descent approximation.

Newton's Method

Recall second-order expansion of f around x:

$$f(y) \approx \hat{f}(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(x)(y-x).$$

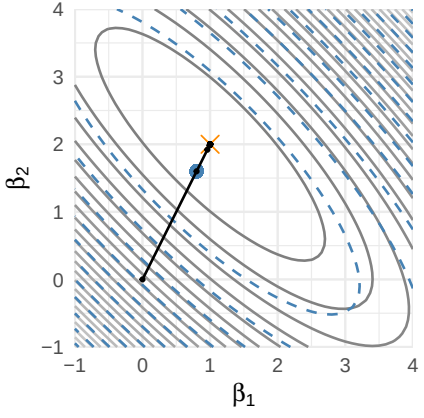
Gradient Descent Update

Replace $\nabla^2 f(x)$ with $\frac{1}{t}I$ and minimize.

Newton Update

Instead, minimize $\hat{f}(y)$ directly:

$$x^+ = x - \left(\nabla^2 f(x)\right)^{-1}\nabla f(x).$$



Damped Newton

This might actually not converge, so instead we use **damped** Newton:

$$x^+ = x - t\nabla^2 f(x)^{-1}\nabla f(x),$$

with $t > 0$ chosen by backtracking line search.

Phases

- As long as $t < 1$, we are in the **damped** phase.
- When $t = 1$, we enter the **pure** phase and t will remain 1.
- In pure phase, we have quadratic convergence.

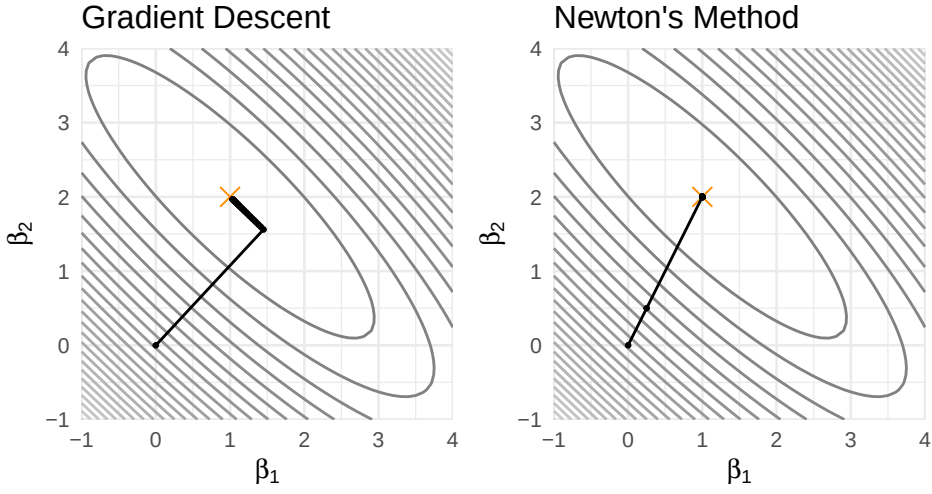
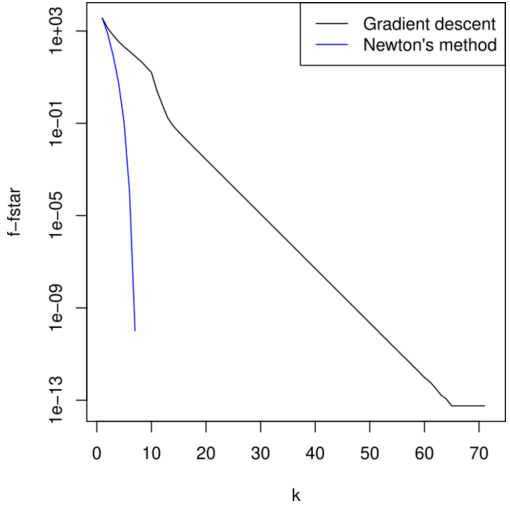
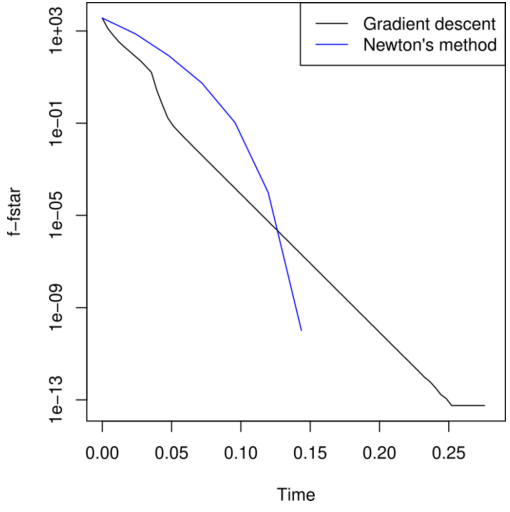


Figure 12: Gradient descent vs Newton's method on the ill-conditioned problem.

Convergence in Time and Iterations





In practice, results depend on the problem.

Gradient Descent vs Newton's Method

Gradient Descent

- Cheap iterations: $O(p)$
- Sensitive to conditioning
- Linear convergence ($O(1/k)$)
- Few assumptions on f

Newton's Method

- Expensive iterations: $O(p^3)$
- Insensitive to conditioning
- Quadratic convergence ($O(\log\log(1/k))$)
- Requires f to be twice-differentiable and Hessian to be positive definite

Replace $\nabla^2 f(x)$ with an approximation B_k.

BFGS
Complexity: $O(p^2)$, still same memory cost.

L-BFGS
Same complexity but lower memory cost.

Stopping Criteria

Common (ad-hoc) choices

- Small gradient:

$$\|\nabla f(x)\|_r \leq \epsilon$$

- Small relative change in x:

$$\frac{\|x^{(k+1)} - x^{(k)}\|_2}{\|x^{(k)} + 10^{-q}\|_2} \leq \epsilon$$

- Small relative change in f:

$$\frac{f(x^{(k+1)}) - f(x^{(k)})}{f(x^k)} \leq \epsilon$$

Duality Gap

If f is **strongly** convex and x optimal, then the duality gap is zero at the optimum.

Most principled stopping criterion, but not always available.

Out of scope for this course!

Summary

Today

- Optimization problems
- Algorithms to solve optimization problems
  - Gradient descent
  - The Newton method

Things We Did Not Cover

- Accelerated gradient methods
- Conjugate gradient
- Constrained optimization
- Nonsmooth objectives

Exercise

We will attempt to minimize the following function:

$$f(x) = \log(1 + e^x) + \frac{1}{2}(x - 2)^2.$$

Step 1

Implement a gradient descent method for solving this problem. Plot convergence in terms of iterations and time. Implement a stopping criterion.

Step 2

Test and benchmark your implementation against R's built-in `optimize()`.

Step 3

Implement Newton's method for solving this problem and repeat the analysis.

Next Time

Testing

We will cover how to systematically test your code using **testthat**.

Debugging

We will cover how fix bugs in your code using debugging tools in R.

This part will mostly be a hands-on sessions, so bring your laptop!