Optimization for Statistics and Machine Learning

Eric Chi

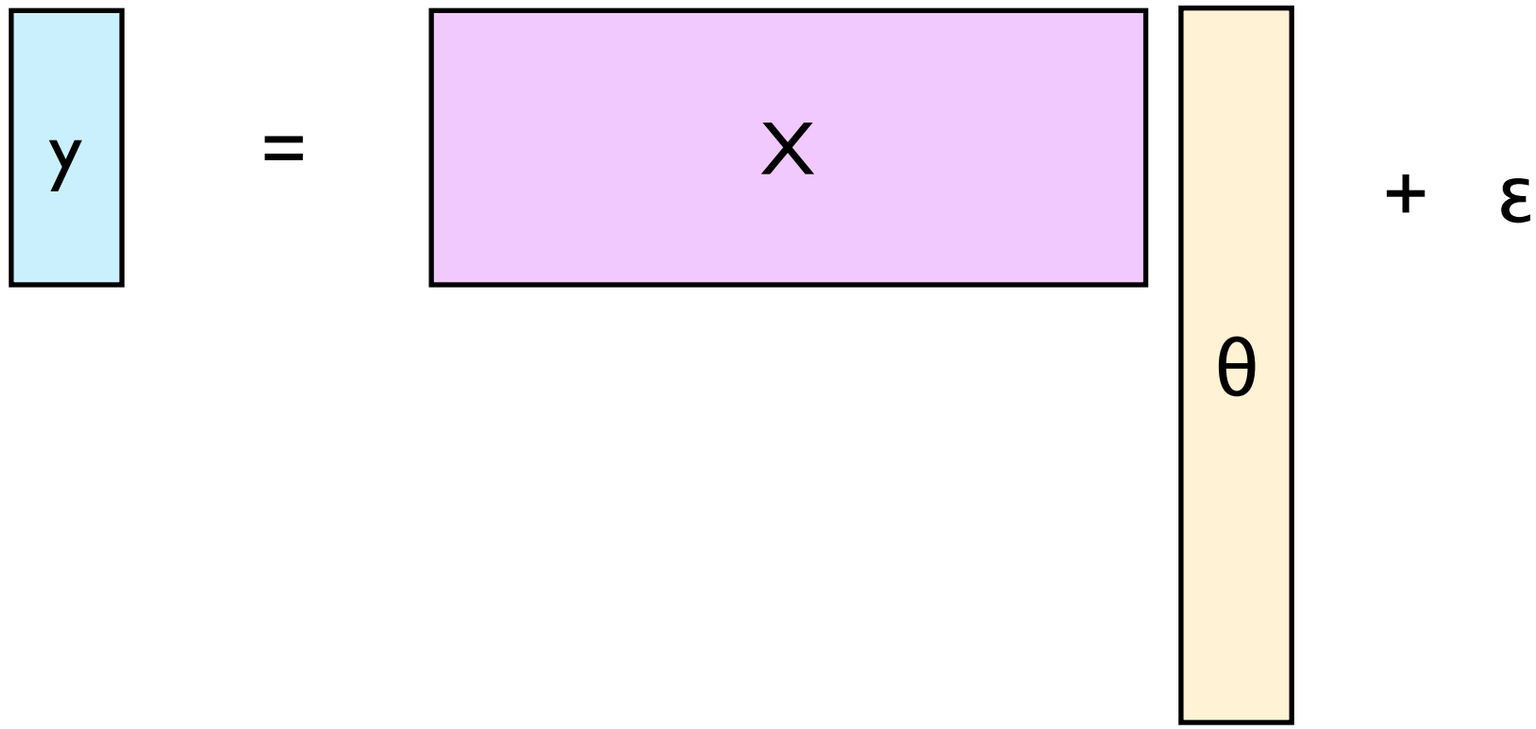
August 11, 2016

Motivation:

$$\min_{\theta \in \Theta} \ell(\theta) + \lambda R(\theta)$$

- Modern data is high-volume ("Big Data")
- Modern data is high-dimensional (Many features per observation)
- Need to impose structure to get better predictions and meaningful models

Two Typical Problems

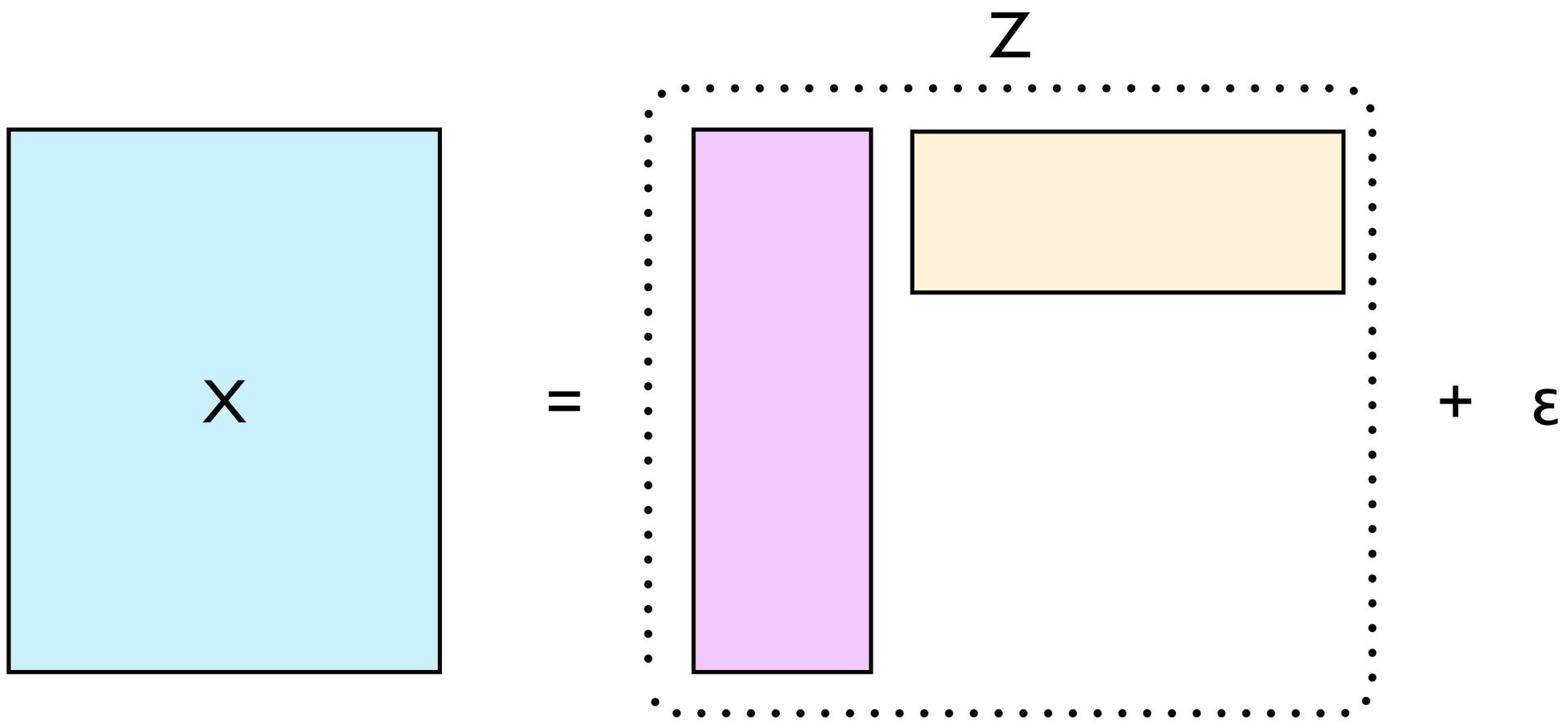


- Regularized estimation to get sparse solutions

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{2}\|y - X\theta\|_2^2 + \lambda\|\theta\|_1$$

- Arises in biomedical problems (high-throughput -omics data) and signal processing

Two Typical Problems

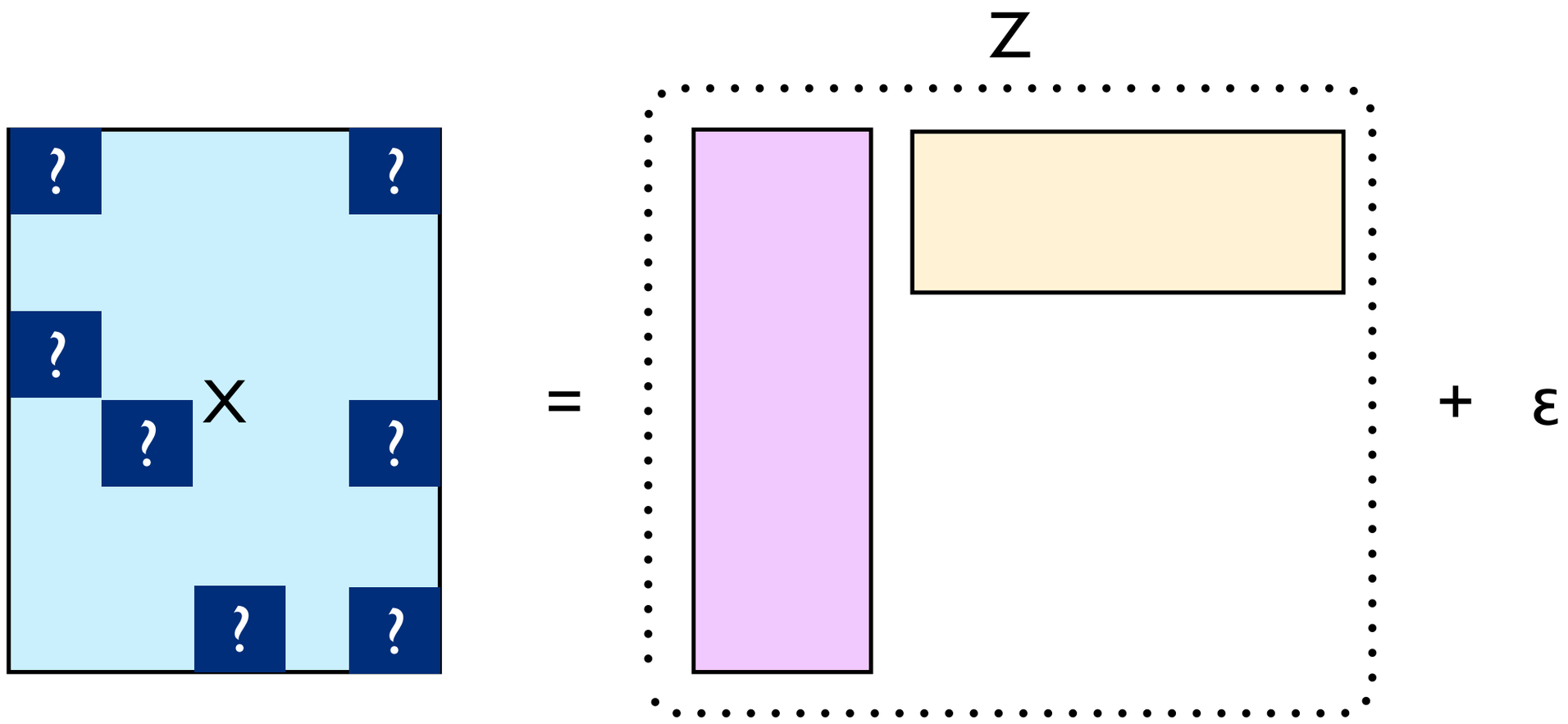


- Regularized estimation to get low-rank solutions

$$\hat{Z} = \arg\min_{Z} \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_2^2 + \lambda\|Z\|_*$$

- Arises in collaborative filtering: Netflix

Two Typical Problems



- Regularized estimation to get low-rank solutions

$$\hat{Z} = \arg\min_{Z} \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_2^2 + \lambda\|Z\|_*$$

- Arises in collaborative filtering: Netflix

The Generic Problem

$$\hat{\theta} = \arg\min_{\theta} \quad \underbrace{L(\theta)}_{\text{Lack of fit}} + \underbrace{J(\theta)}_{\text{Complexity}}$$

Reasons for success:

- Theory: Consistency and convergence rates when $n, p \to \infty$
- Computation: Fast and scalable algorithms for computing θ

Agenda

Day 1

- First half: Smooth optimization

$$\min_{\theta \in \Theta} \ell(\theta)$$

- Second half: Non-smooth optimization

$$\min_{\theta \in \Theta} \ell(\theta) + \lambda R(\theta)$$

Day 2

- KKT conditions, duality theory, applications

Why R?

1. It's the computing language of statistics and data science

But you should know others: Python, Julia, Matlab, etc. . .

2. Hadley Wickham made it **absurdly easy** to make R packages

$$\text{devtools} + \text{Rstudio} = \text{package in} < \text{week!}$$

- $\sim$week to get it onto CRAN. . .

Part I: Smooth Optimization

- Computational Complexity
- Convex sets and functions
- **Gradient Descent**
- Newton's Method, quasi-Newton, accelerated First order methods

Credits

Matieral is borrowed from

- Stephen Boyd (EE364a, EE364b @ Stanford)
- Lieven Vandenberghe (EE236C @ UCLA)
- Lecture notes by Stephen Wright, Wotao Yin, Hua Zhou

What to get out of this lecture

- Main thing is to get familiar with gradient descent.
- Gradient descent is the prototype algorithm
- The big data / high-dimensional versions of gradient descent will be easy to pick up if you understand plain gradient descent.

Computational Complexity Overview

How much work do we have to do to perform classic multiple linear regression?
$y \in \mathbb{R}^n, X \in R^p$, where $n > p$.

$$\hat{\beta} = \arg\min_{\beta} f(\beta) := \frac{1}{2}\|y - X\beta\|^2$$

- How does the work increase if I double the number of observations n?
- How does the work increase if I double the number of covariates p?

Big O notation

Definition: Let f and g be two functions defined on the natural numbers, i.e. $f : \{1, 2, 3, \ldots\} \to R$. We say f is $\mathcal{O}(g)$ if there is a positive constant M and a number n_0 such that

$$|f(n)| \leq M|g(n)|$$

for all $n \geq n_0$.

Example:

$$f(n) = 6n^4 - 2n^3 + 5$$
$$g(n) = n^4$$

Big O properties

- Sum property

Let $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$ then $f_1 + f_2 = \mathcal{O}(|g_1| + |g_2|)$.

- Product property

Let $f_1 = \mathcal{O}(g_1)$ and $f_2 = \mathcal{O}(g_2)$ Then $f_1 f_2 = \mathcal{O}(g_1 g_2)$.

Counting the Costs

Typically only two kinds of costs

- Basic linear algebra operations: dot products, matvecs
- Matrix factorizations: QR, Choleksy, SVD

The latter typically dominate the total algorithmic work

Complexity of vector addition, inner product

Let $x, y \in \mathbb{R}^n$.

- $x + y$ needs n additions, n flops
- $x^T y$ needs n multiplications, $n-1$ additions so: $2n-1$ flops
- Therefore, $x^T y$ is $\mathcal{O}(n)$ flops.

$$x^T y = x_1 y_1 + x_2 y_2 + x_3 y_3$$

Complexity of matrix-vector multiplication (matvec)

Let $A \in R^{m \times n}, x \in \mathbb{R}^n$.

- Ax needs m inner products each of which is $\mathcal{O}(n)$.
- Therefore, Ax needs $\mathcal{O}(mn)$ flops.

$$Ax = \begin{pmatrix} a_1^T x \\ a_2^T x \\ a_3^T x \end{pmatrix}$$

Complexity of matrix-matrix multiplication

Let $A \in R^{m \times n}, B \in R^{n \times p}$.

- AB needs p matvecs, each of which requires $\mathcal{O}(mn)$ flops.
- Therefore, AB needs $\mathcal{O}(mnp)$ flops.

$$AB = \begin{pmatrix} Ab_1 & Ab_2 & Ab_3 \end{pmatrix}$$

Solving a triangular linear system

Let $R \in R^{n\times n}$ be upper triangular, and $b \in \mathbb{R}^n$. How much work is it to solve the system $Rx = b$?

$$\begin{aligned} r_{1,1}x_1 + r_{1,2}x_2 + \cdots + r_{1,n}x_n &= b_1 \\ &\vdots \\ r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n &= b_{n-1} \\ r_{n,n}x_n &= b_n. \end{aligned}$$

Solution Strategy: Back-substitution

Step 1. $x_n = b_n/r_{n,n} \implies$ 1 flop

Step 2. $x_{n-1} = \frac{b_{n-1} - r_{n-1,n}x_n}{r_{n-1,n-1}} \implies$ 3 flops

Step i. $x_{n-i+1} = \ldots \implies (2i-1)$ flops

Total work: $1 + 3 + \cdots + (2n-1) = \sum_{i=1}^{n}(2i-1)$ is $\mathcal{O}(n^2)$.

Multiple Linear Regression

$$\hat{\beta} = \arg\min_{\beta} f(\beta) := \frac{1}{2}\|y - X\beta|^2.$$

Solve for $\nabla f(\beta) = 0$ to get the normal equations.

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

$$\hat{\beta} = (X^T X)^{-1}(X^T y)$$

The numerically stable way to solve this system is to use a QR decomposition of X.

- QR requires $\mathcal{O}(np^2)$ flops.

Multiple Linear Regression

$X = QR$ where $Q \in R^{n \times p}$ and $Q^T Q = I$ and $R \in R^{p \times p}$ is triangular. Let $z = X^t y$.

$$(X^T X)\beta = z$$
$$(R^T Q^T QR)\beta = z$$
$$R^T R\beta = z.$$

Solve $R^T u = z$ using $\mathcal{O}(p^2)$ flops. Then solve $R\beta = u$ using $\mathcal{O}(p^2)$ flops.

Steps to do multiple linear regression:

- Make $z \implies \mathcal{O}(np)$ flops.
- Get QR of $X \implies \mathcal{O}(np^2)$ flops.
- Get $u \implies \mathcal{O}(p^2)$ flops.
- Get $\beta \implies \mathcal{O}(p^2)$ flops.

Total work is $\mathcal{O}(np^2)$ flops.

Multiple Linear Regression

- The most time consuming step is the QR decomposition!

Back to motivating question:

- If I double the number of observations, I double the run time.
- If I double the number of covariates I quadruple the run time.

Convexity Overview

1. Convex optimization problems often admit polynomial time iterative solvers
2. A huge (and important) class of estimation problems can be cast as a convex optimization problem, e.g. variable selection.
3. Even if the problem cannot be cast as a convex one, non-convex optimization problems can sometimes be **relaxed** very effectively to a convex one.

Sometimes the solution to a convex relaxation is a solution to a non-convex problem! e.g. Lasso and basis pursuit

4. Non-convex problems can be approximately solved by solving a sequence of easier to solve convex problems.
5. If you understand the properties of convexity, you'll be able to identify convex problems in the wild and formulate them so that they can be solved efficiently.

Convexity Overview

- Convex sets: definitions, examples, and properties.
- Convex functions.
- Convex sets are fundamental.
- Convex functions can be defined in terms of a convex set.

If you have a convex problem

- There are very good generic solvers: CVX, Gurobi, MOSEK, etc.
- There are scalable, efficient, simple, robust, etc... algorithms that you can handcraft to solve your problem. Literature is immense!

Convex Sets

Definition: A set $C \subset \mathbb{R}^n$ is convex if for all $x, y \in C$ and $\alpha \in [0, 1]$, $z = \alpha x + (1 - \alpha)y \in C$.

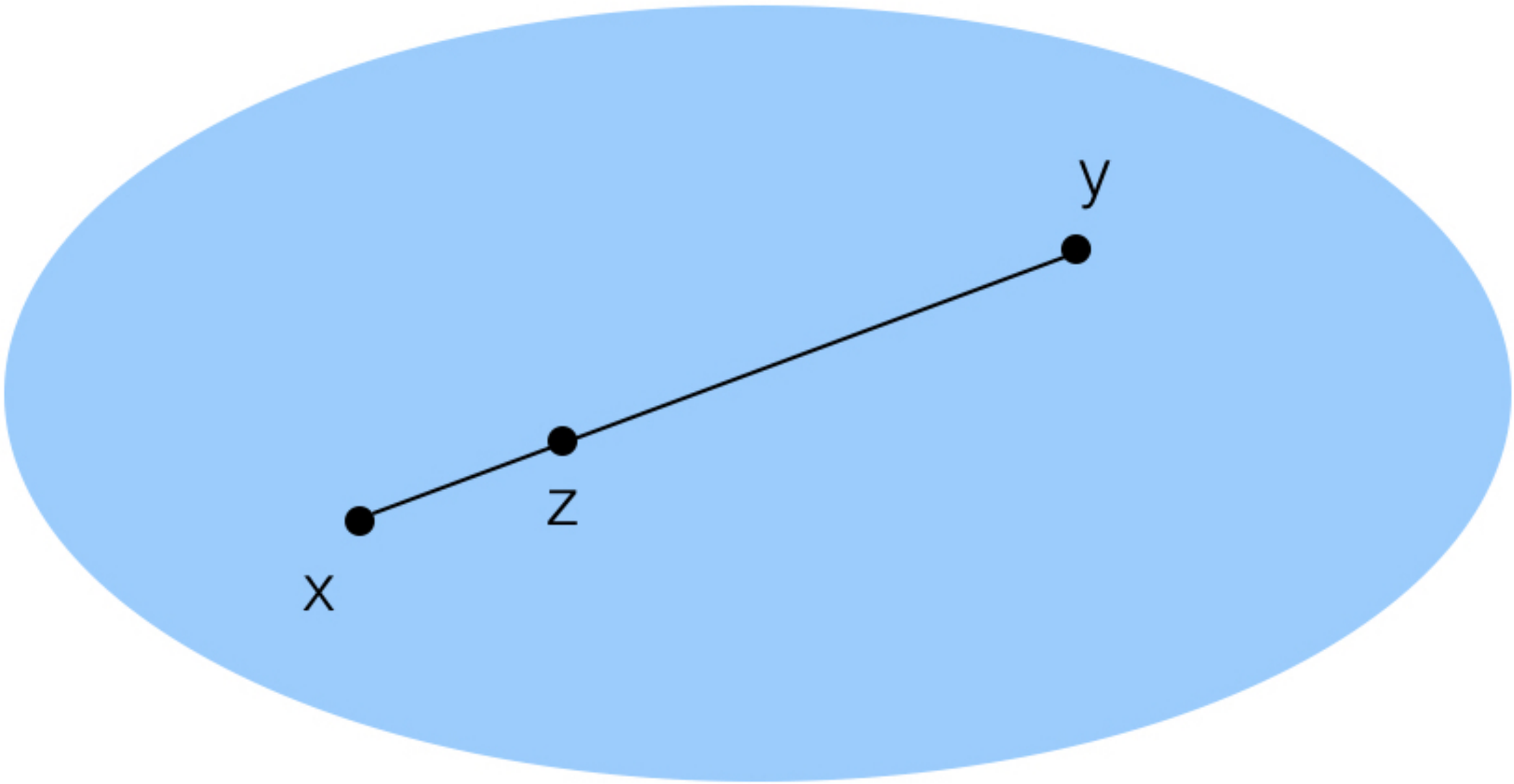


Figure 1: Convex Set

Convex Sets

Definition: A set $C \subset \mathbb{R}^n$ is convex if for all $x, y \in C$ and $\alpha \in [0, 1]$, $z = \alpha x + (1 - \alpha)y \in C$.

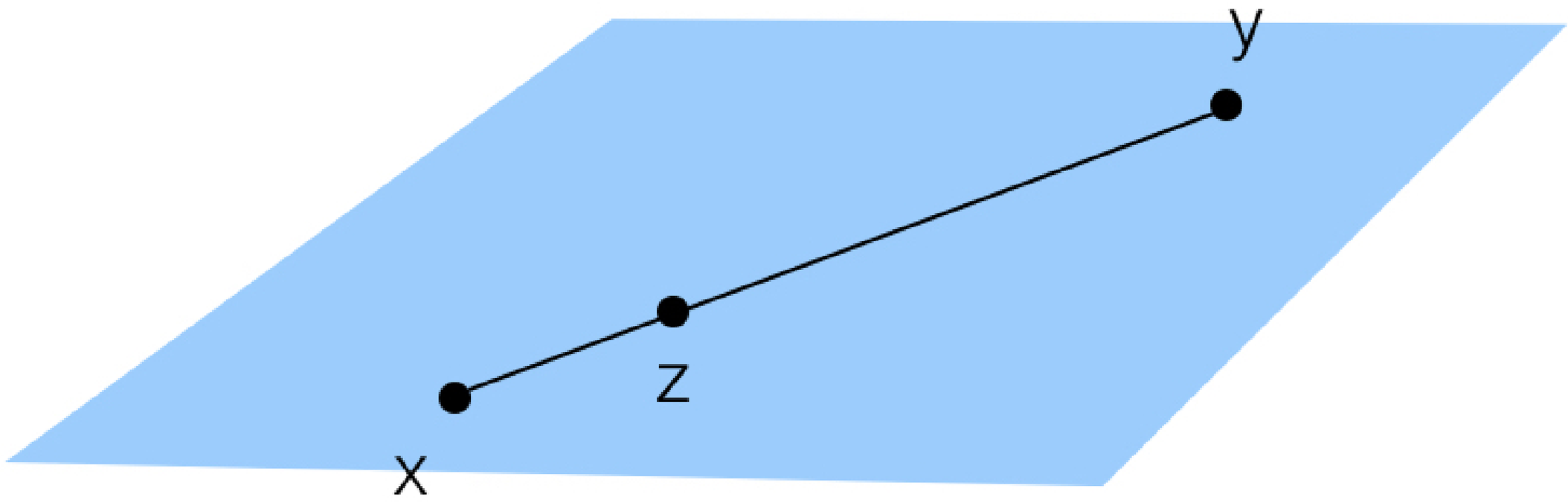


Figure 2: Convex Set

Other Examples

- Unit Euclidean ball: $C = \{x : \|x\|_2 \leq 1\}$.
- Open Halfspace: $C = \{x : x^T v < c\}$
- Set of Positive Semidefinite Matrices:
  $C = \{X \in R^{n \times n} : X = X^T, u^T X u \geq 0 \; \forall u \in \mathbb{R}^n\}$

Non-Examples:

- Set of k-sparse Vectors: $C = \{x \in \mathbb{R}^n : \|x\|_0 \leq k\}$.

Some important classes:

Definition: A set $C \subset \mathbb{R}^n$ is convex cone if for all $x, y \in C$ and $\alpha, \alpha' \geq 0$, $z = \alpha x + \alpha' y \in C$.

Definition: A set $C \subset \mathbb{R}^n$ is affine if for all $x, y \in C$ and $\alpha \in R$, $z = \alpha x + (1 - \alpha) y \in C$.

Properties of Convex Sets

Proposition: The intersection of a collection of convex sets if convex.

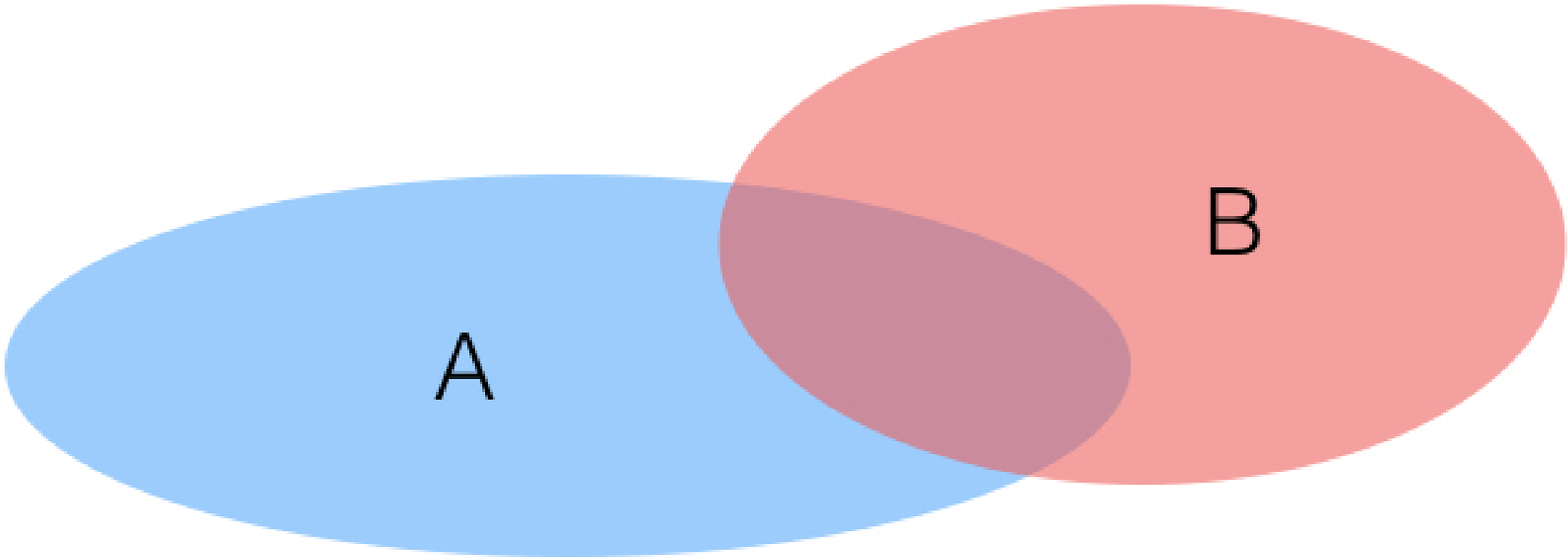


Figure 3: Intersection of A & B is convex

Proposition: Any convex combination $\sum_{i=1}^{m} = \alpha_i x_i$ of points from a convex set C belongs to C. Here each $\alpha_i \geq 0$ and $\sum_{i=1}^{m} \alpha_i = 1$.

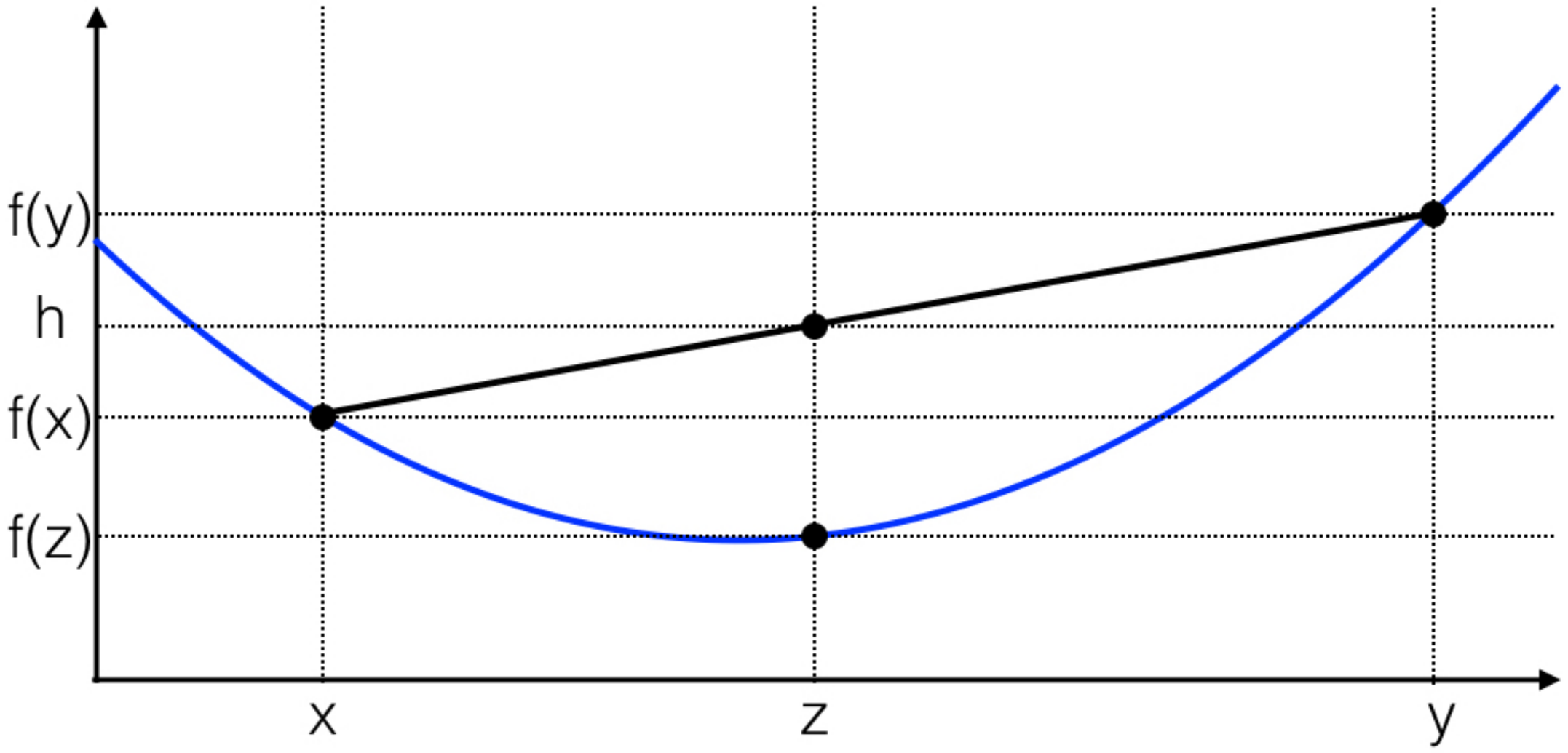
Convex Functions

Definition: Let $C \subset \mathbb{R}^n$ be a convex set. A function $f : \mathbb{R}^n \mapsto R \cup \{\infty\}$ is convex over C for all $x, y \in C$ and $\alpha \in [0, 1]$,

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y).$$

If the inequality is strict whenever $x \neq y$, then f is strictly convex over C.

Definition: A function f is concave if its negative $-f$ is convex.



Jensen's Inequality

Proposition:

$$f\left(\sum_{i=1}^{m} \alpha_i x_i\right) \leq \sum_{i=1}^{m} \alpha_i f(x_i).$$

c.f.

$$f(E[X]) \leq E[f(X)].$$

Differentiable Convex Functions

Proposition: Let f be a differentiable function on the open convex set C. Then f is convex if and only if

$$f(z) \geq f(x) + \nabla f(x)^t (z - x) \quad \forall x, z \in C$$

If the inequality is strict whenever $x \neq z$, then f is strictly convex over C.

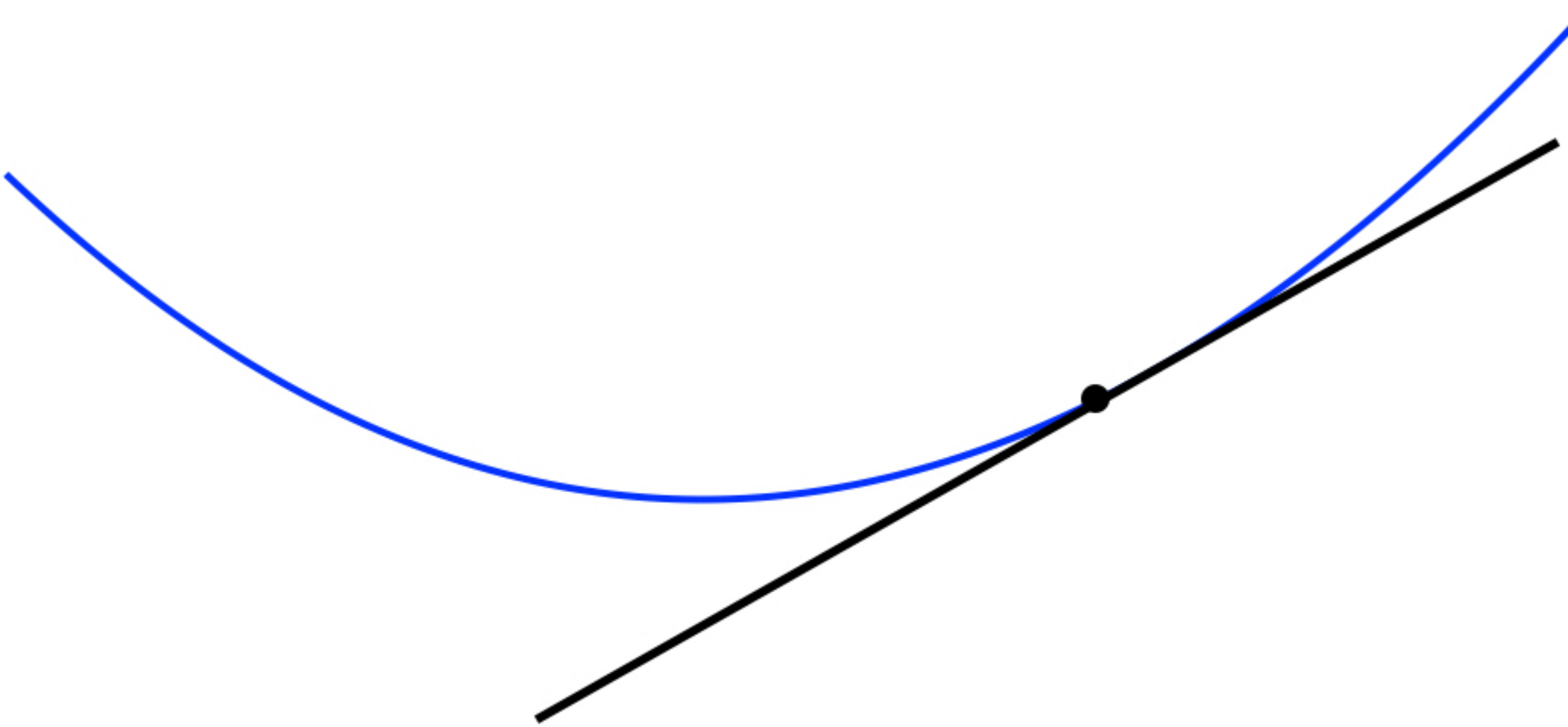


Figure 5: Best linear approximation

Twice Differentiable Convex Functions

Proposition: Let f be a twice differentiable function on the open convex set C. If its Hessian matrix $d^2f(x)$ is positive semidefinite for all x, then f is convex. When $d^2f(x)$ is positive definite for all x, f is strictly convex.

Basic Building Blocks

Convex:

- Affine functions
- Norms
- Exponentials
- Powers

Concave:

- Affine functions
- logarithm

Closure Properties of Convex Functions

Proposition:

1. If $f(x)$ is convex and $g(x)$ is convex and increasing, then $g \circ f(x)$ is convex.
2. If $f(x)$ is convex, then $f(Ax + b)$ is convex.
3. If $f(x)$ and $g(x)$ are convex and $\alpha, \beta \geq 0$, then $\alpha f(x) + (1 - \alpha) g(x)$ is convex.
4. If $f(x)$ and $g(x)$ are convex, then $\max\{f(x), g(x)\}$ is convex.
5. If $f_n(x)$ is a sequence of convex functions, then $\lim_{n \to \infty} f_n(x)$ is convex whenever it exists.
6. If $f(x, y)$ is convex in x for each fixed y, then $g(x) = \sup_y f(x, y)$ is convex provided it is proper.

Least Squares Objective

$$f(\mathbf{b}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{b}\|_2^2$$

- Hessian of $g(z) = \|z\|_2^2$ is positive definite
- Composition of affine and convex is convex: $f(b) = g(y - Xb)$

Least Absolute Deviations Regression Objective

$$f(\mathbf{b}) = \|\mathbf{y} - \mathbf{X}\mathbf{b}\|_1$$

- Norms are convex
- Composition of affine and convex is convex: $f(b) = g(y - Xb)$

Dominant Eigenvalue of a Symmetric Matrix

$$f(\mathbf{X}) = \max_{\|\mathbf{u}\|_2=1} \mathbf{u}^T \mathbf{X} \mathbf{u}$$

- $\mathbf{X} \mapsto \mathbf{u}^T \mathbf{X} \mathbf{u}$ is a linear function and therefore convex
- Pointwise supremum of convex functions is convex

Other examples

- Log-determinant function defined on the space of positive definite matrices, denoted $\mathbf{S}^n_{++}$.

$$f(\mathbf{X}) = \log\det\mathbf{X}$$

- Log-Sum-Exp (Soft-max)

$$f(\mathbf{x}) = \log(e^{x_1} + \cdots + e^{x_n})$$

Minima of Convex Functions

Proposition: Suppose that $f(y)$ is a convex function on the convex set $S \subset R^m$. If x is a local minimum of $f(y)$, then it is a global minimum of $f(y)$, and the set $\{y \in S : f(y) = f(x)\}$ is convex. If $f(y)$ is strictly convex and x is a global minimum, then the set $\{y \in S : f(y) = f(x)\}$ consists of x alone.

Minima of Differentiable Convex Functions

Proposition: Let $C \subset \mathbb{R}^n$ be a non-empty convex set and $f : \mathbb{R}^n \mapsto R$ be convex and differentiable over an open set that contains C. Then a point $x^\star \in C$ minimizes f over C if and only if

$$\nabla f(x^\star)'(x - x^\star) \geq 0, \quad \forall x \in C.$$

Summary

- Computational complexity: Big-O is a (pessimistic / conservative) way to quantify work as a function of problem size
- Convex sets and functions: Become familiar with the calculus
- Why? Convex problems admit iterative algorithms with good properties (Next)

Unconstrained Optimization

$$\min_{\theta} f(\theta)$$

Start by assuming

- f is convex
- f is differentiable
- f has a global minimizer

Gradient Descent

Choose an initial point θ_0 and repeat

$$\theta_{m+1} = \theta_m - \alpha \nabla f(\theta_m)$$

- $\alpha > 0$ is a step-size. More on this later.

Do no harm: If $\theta_0 = \theta^\star$, then $\nabla f(\theta_0) = 0$ and

$$\theta_1 = \theta_0 - \alpha \nabla f(\theta_0) = \theta_0.$$

- Stationary points are fixed points of gradient descent.

Why Gradient Descent?

Good:

- Every iteration is cheap
- Does not require computing and inverting a Hessian

Limitations:

- Objective must be differentiable
- Sensitive to scaling of variables / can be very slow

But fancy versions of gradient descent address these issues

Example: Least Squares Regression

Objective:

$$f(\theta) = \frac{1}{2}\|y - X\theta\|^2$$

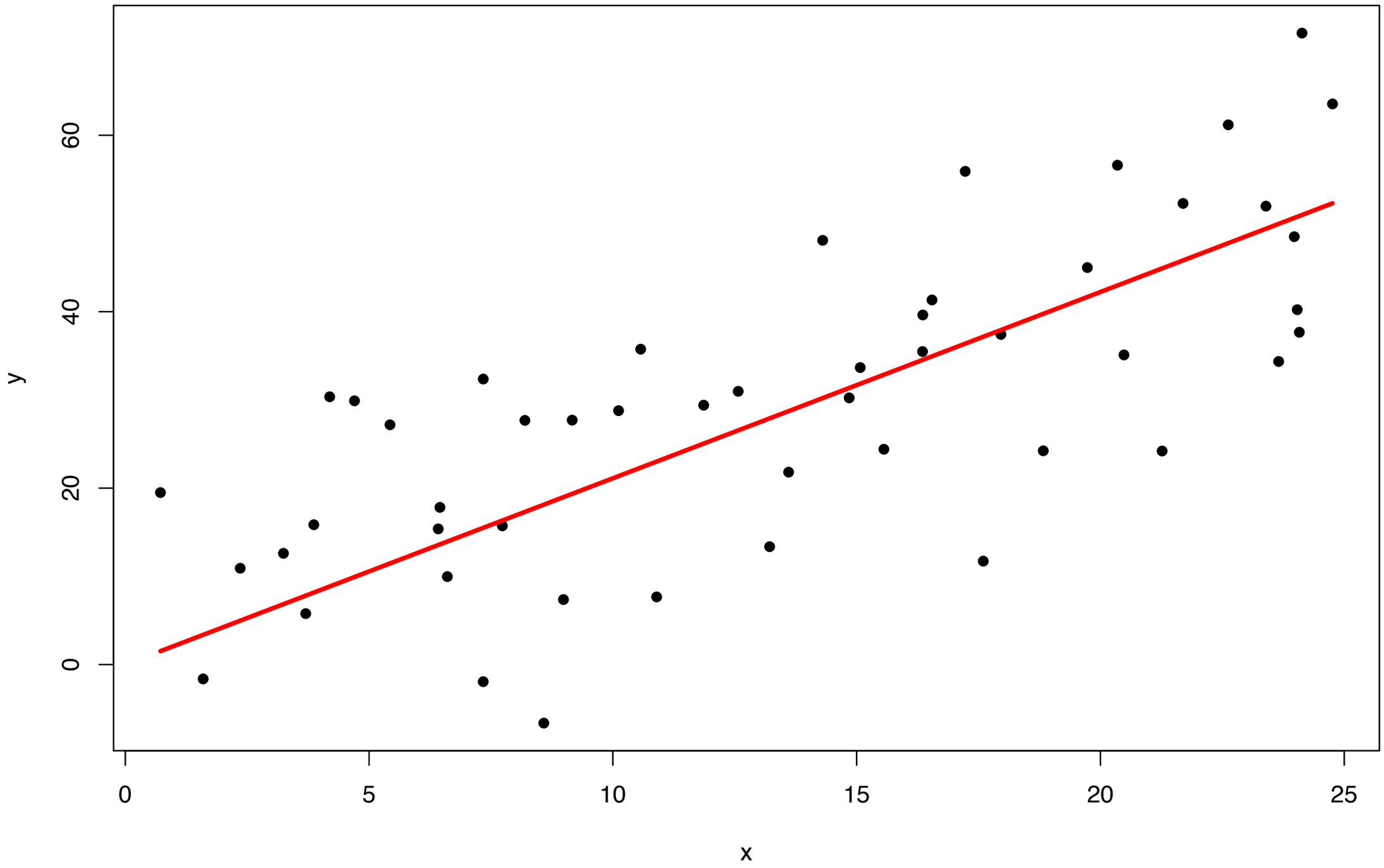
Gradient:

$$\nabla f(\theta) = X^T(X\theta - y)$$

Gradient Step:

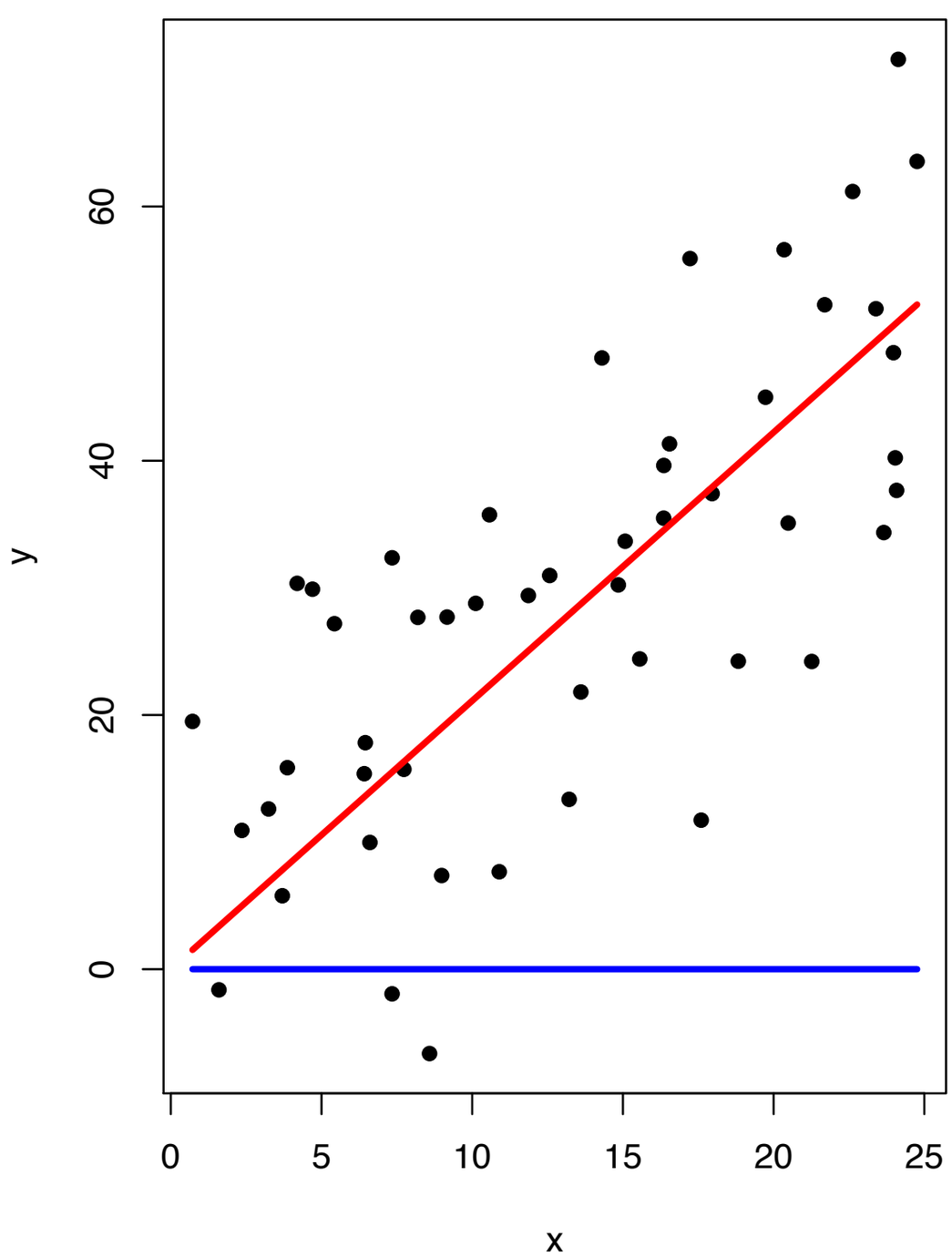
$$\theta_{m+1} = \theta_m - \alpha X^T(X\theta - y)$$

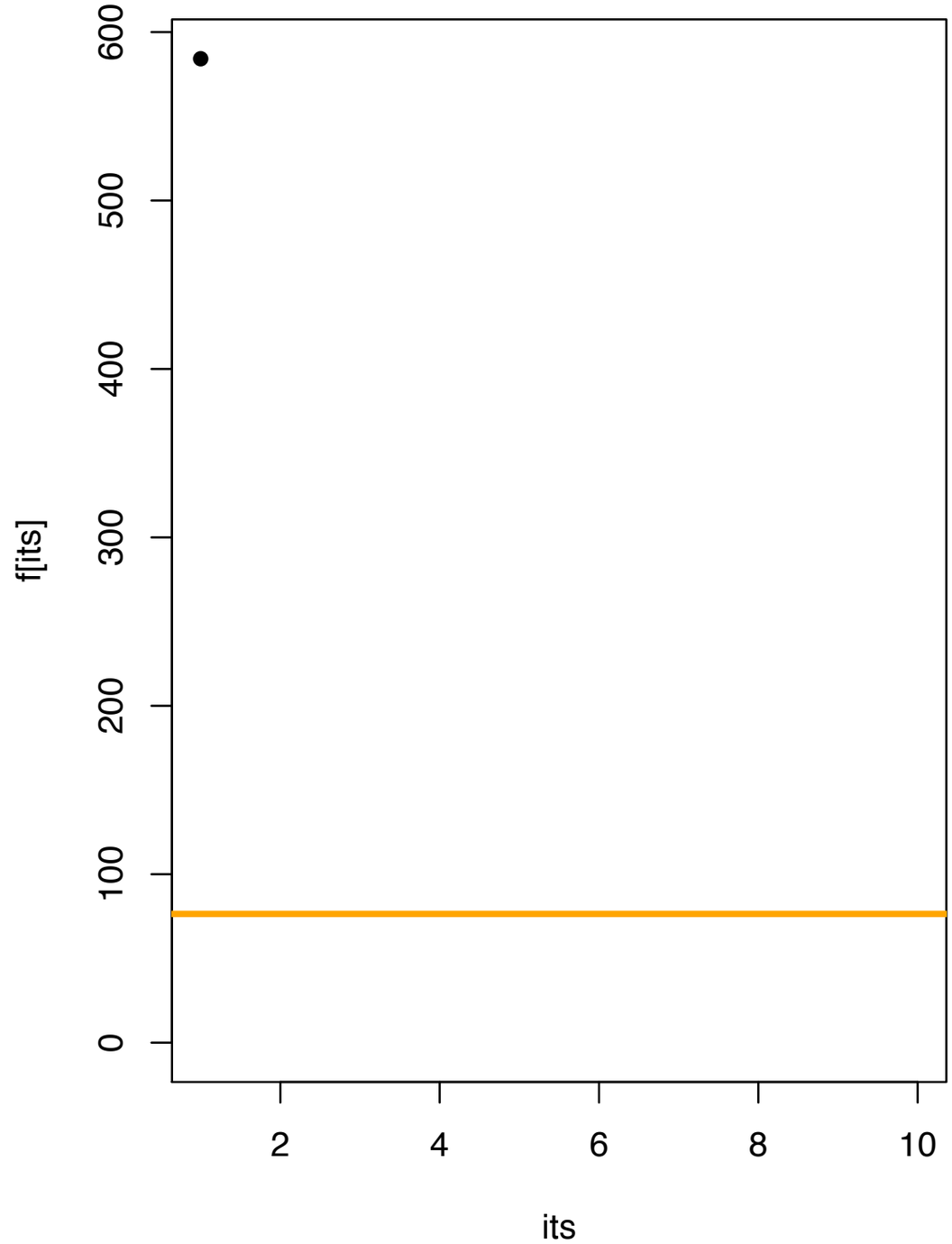
Example: Least Squares Regression



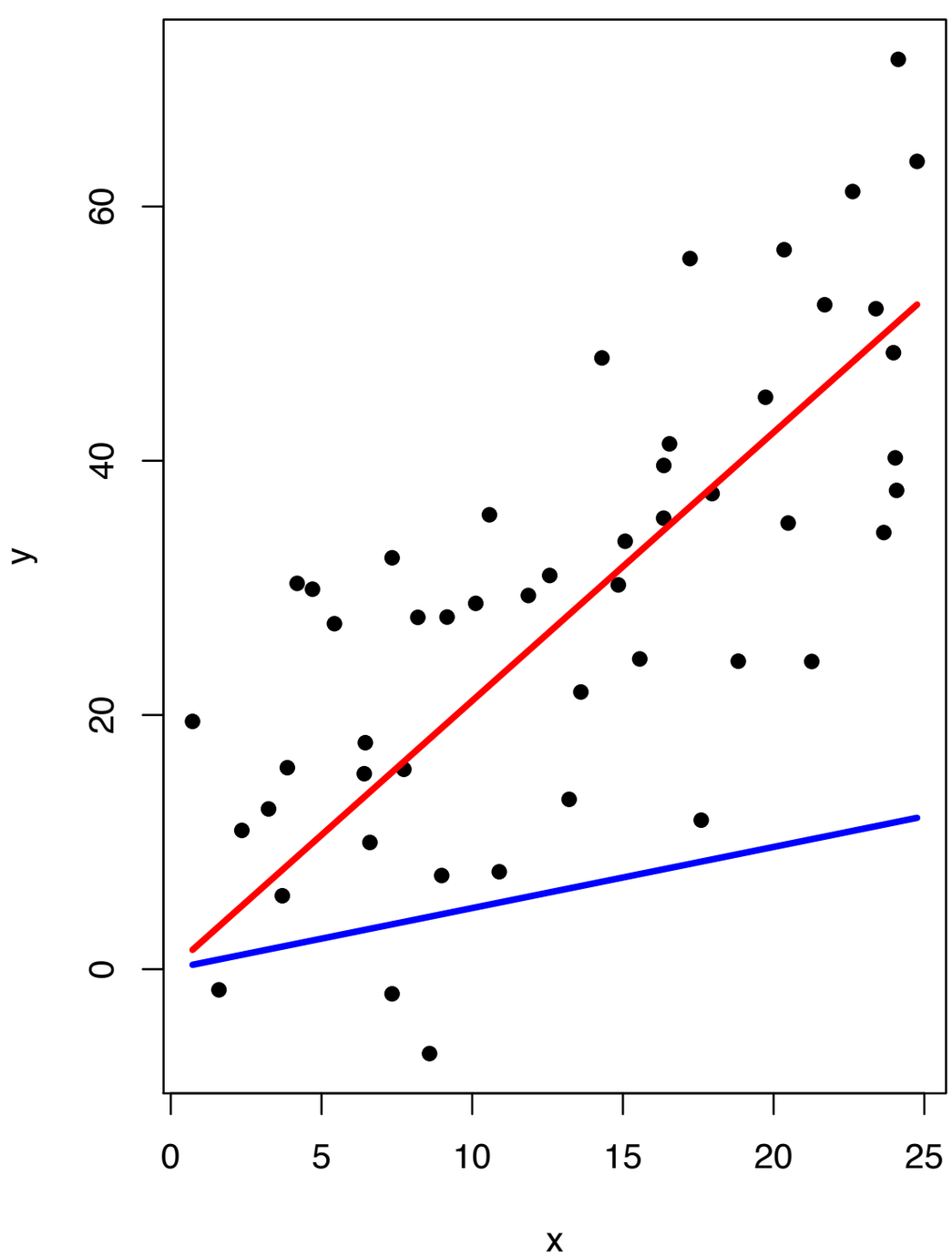
$$\text{RSS} = 76.4365$$

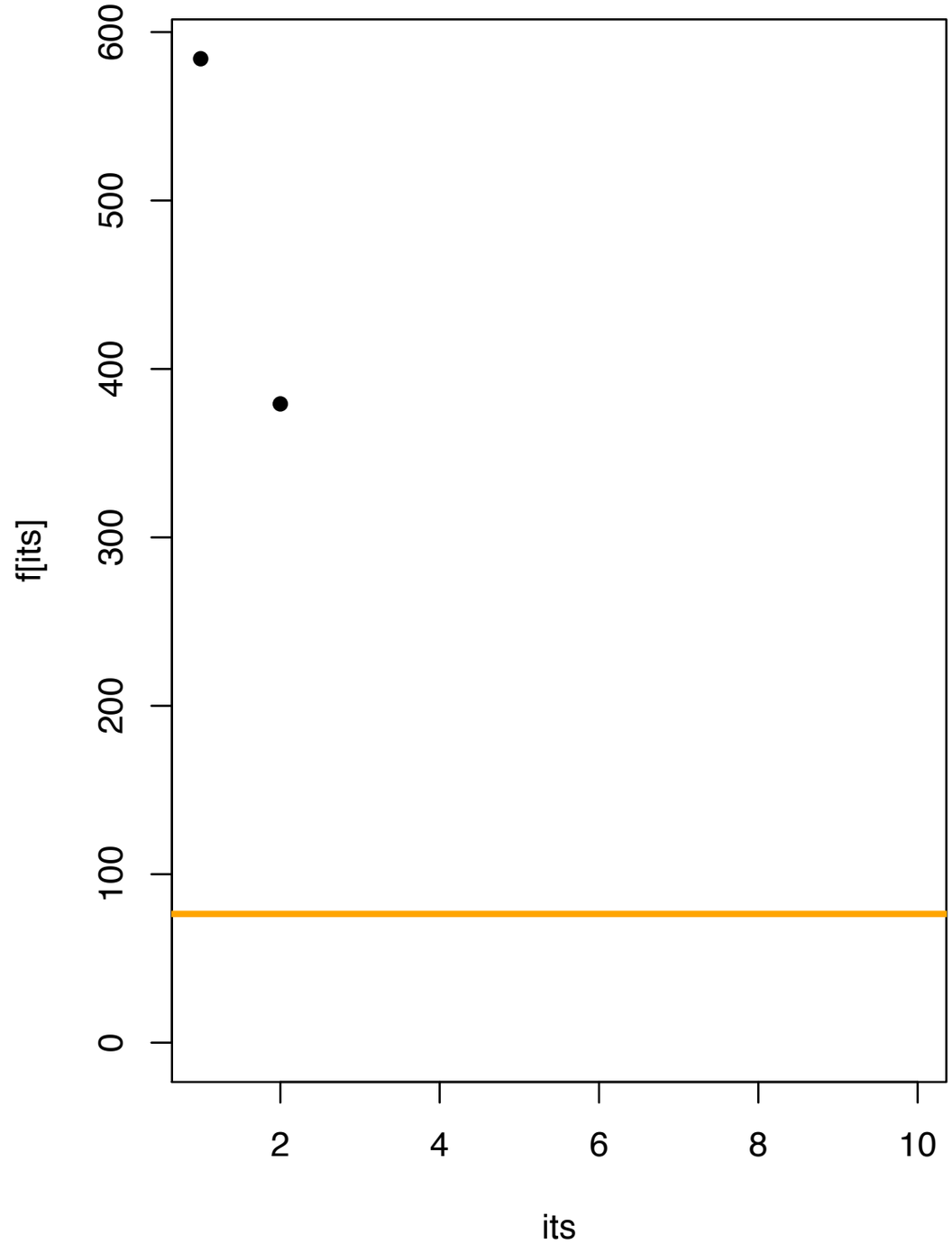
Example: Least Squares Regression



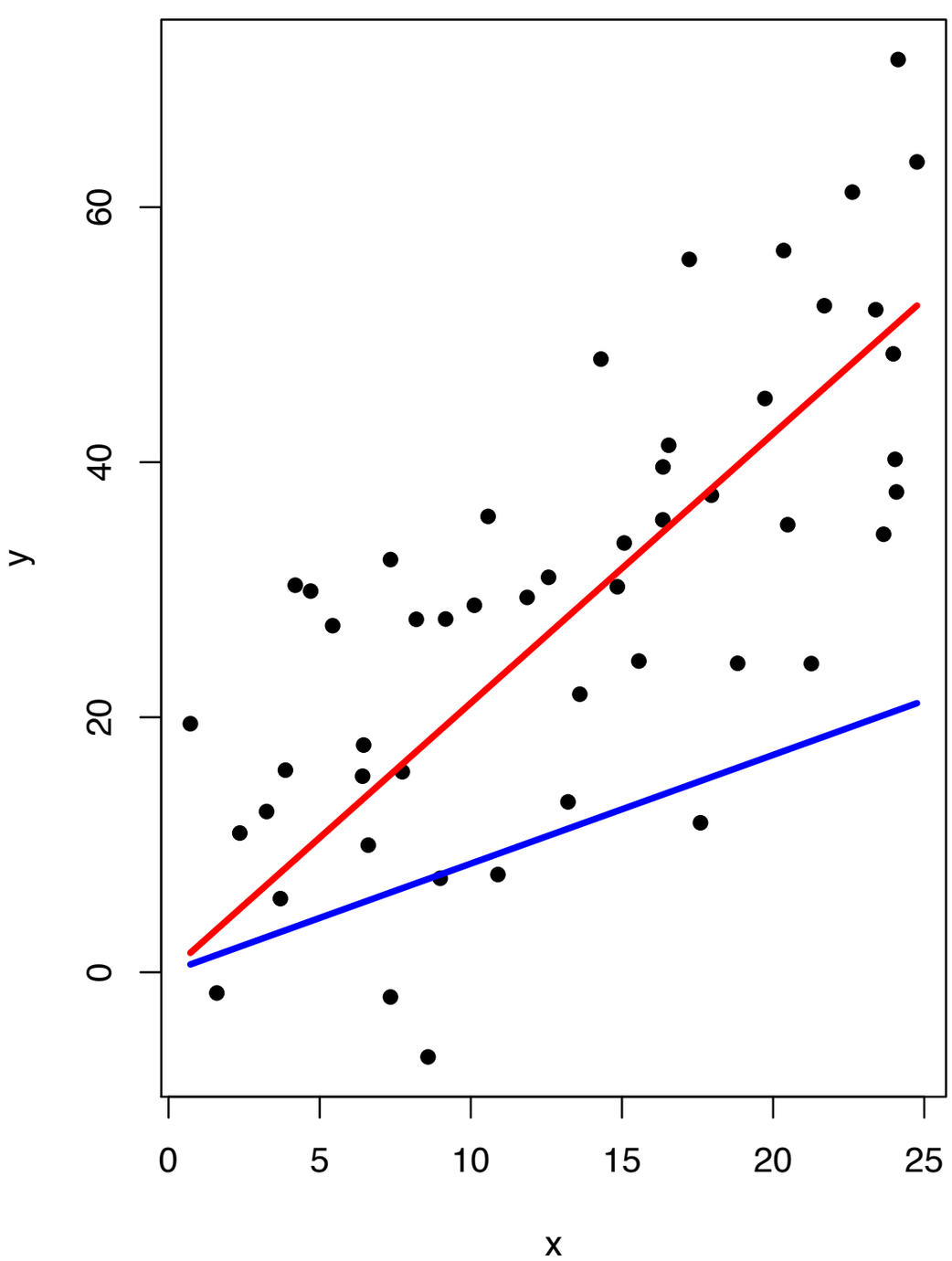


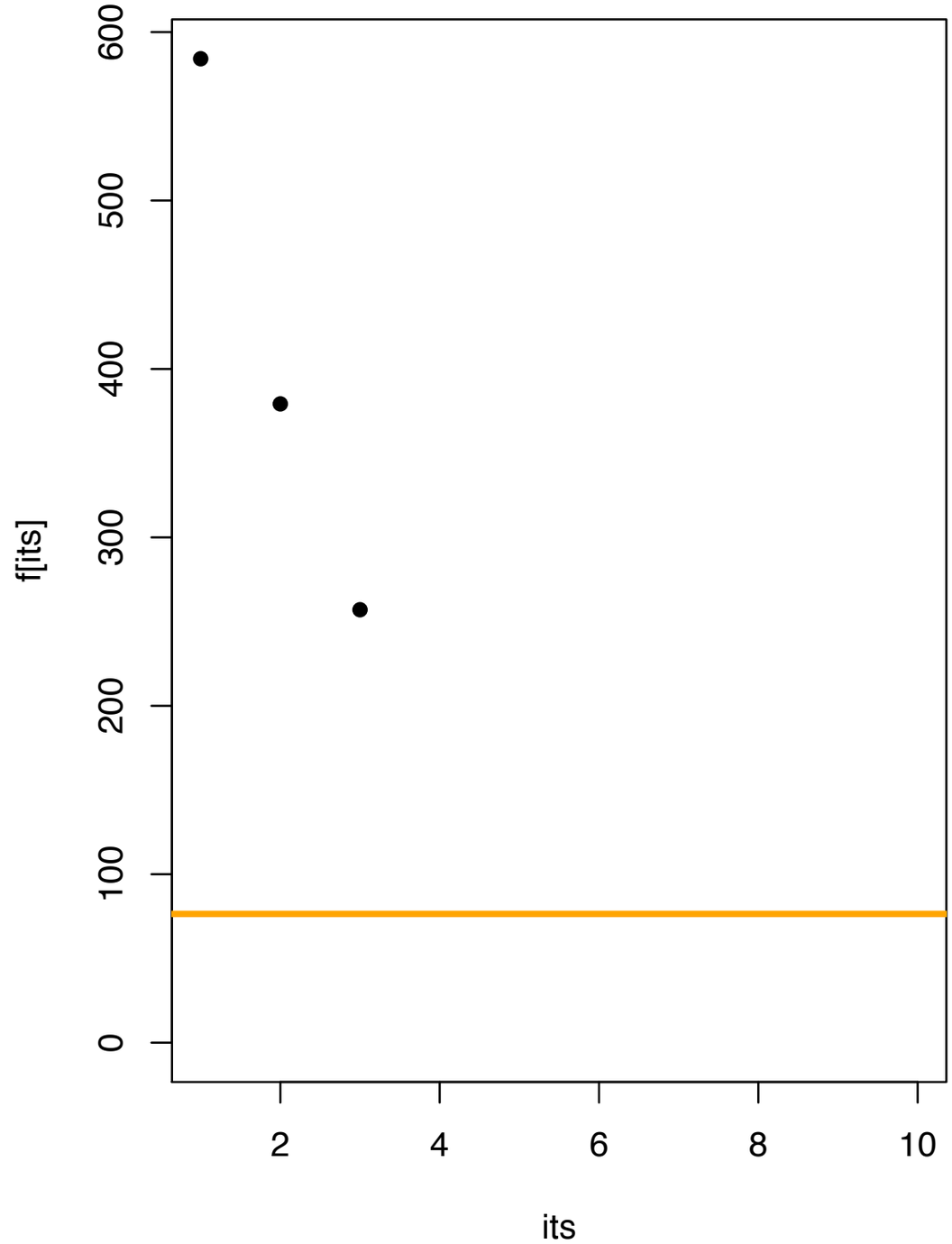
Example: Least Squares Regression



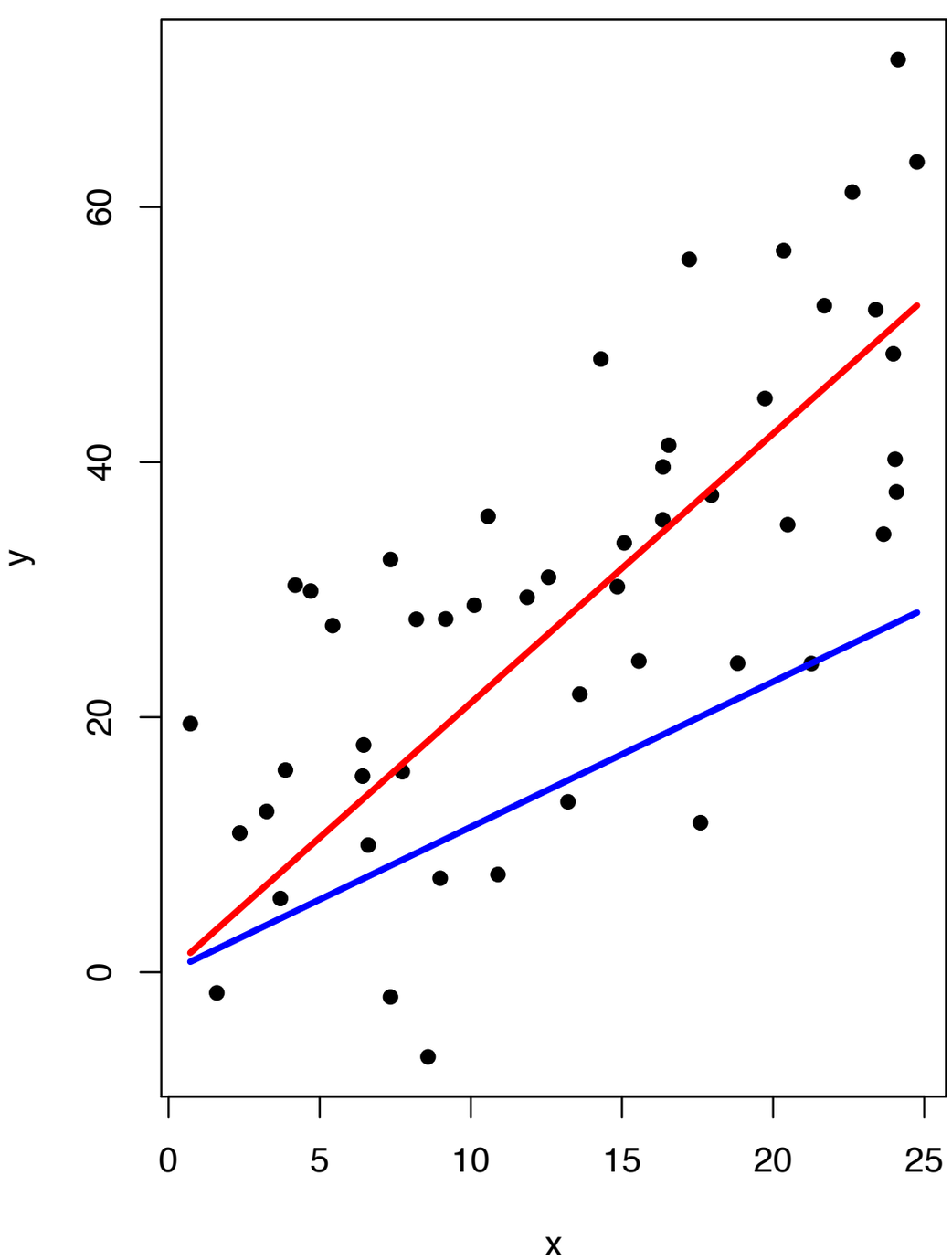


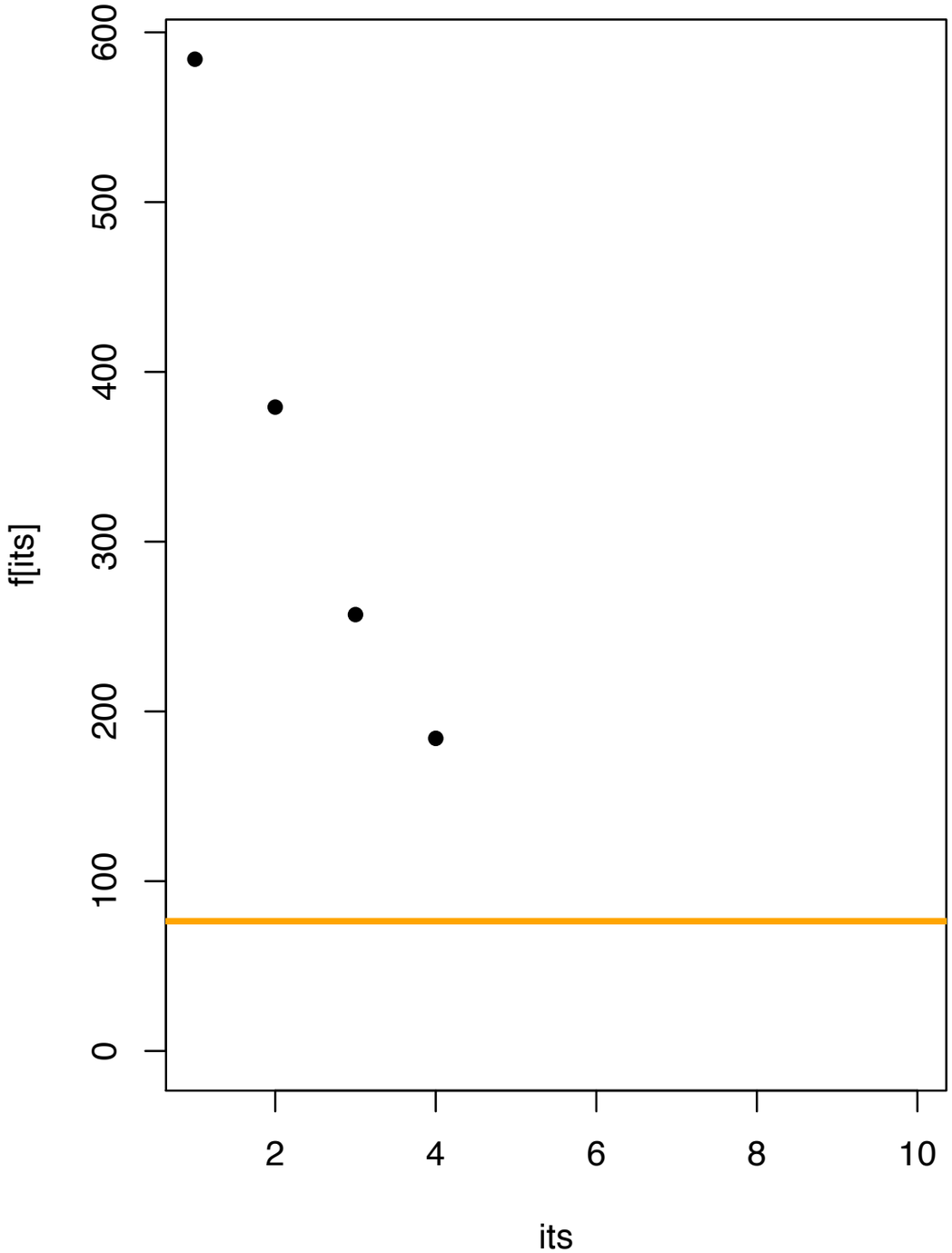
Example: Least Squares Regression



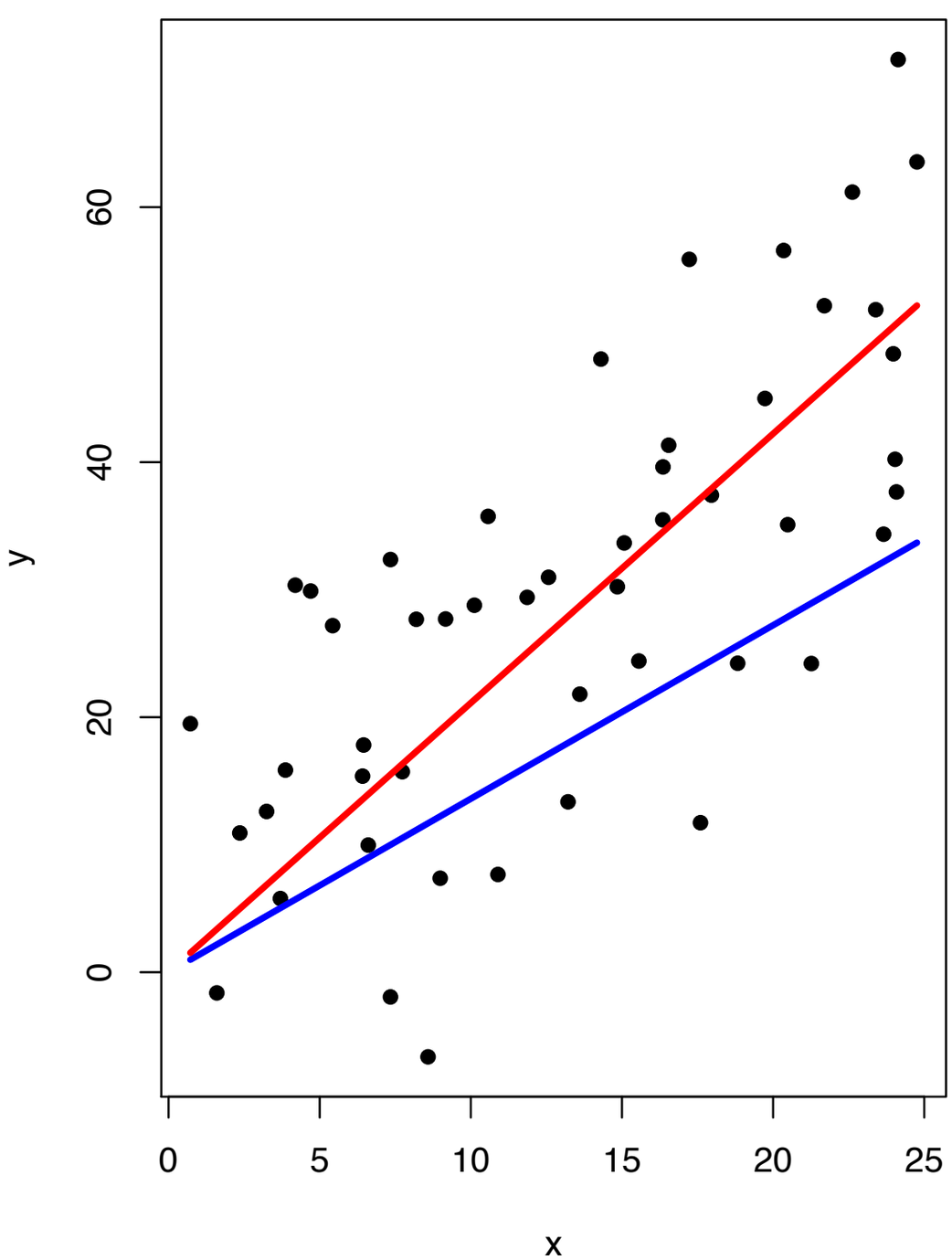


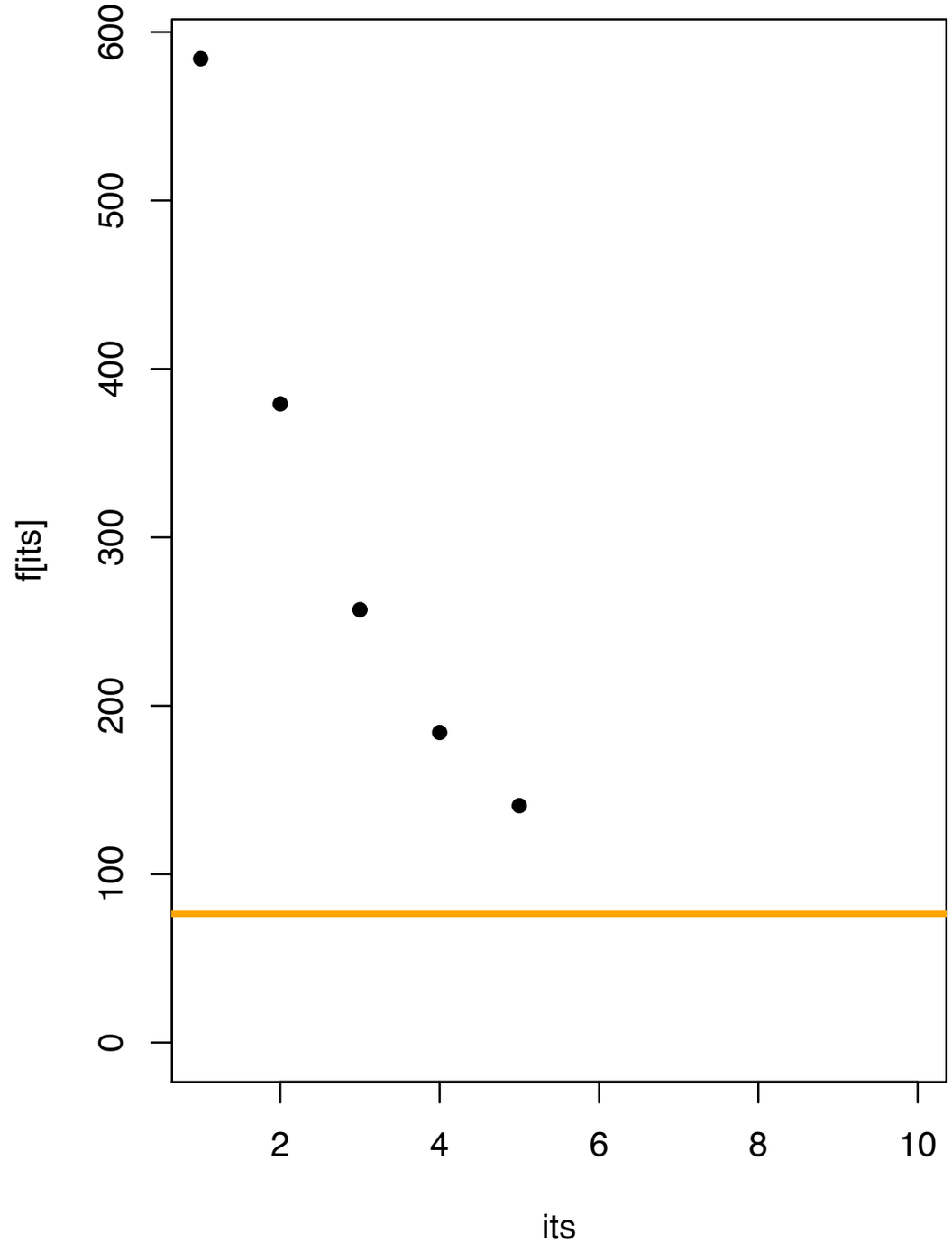
Example: Least Squares Regression



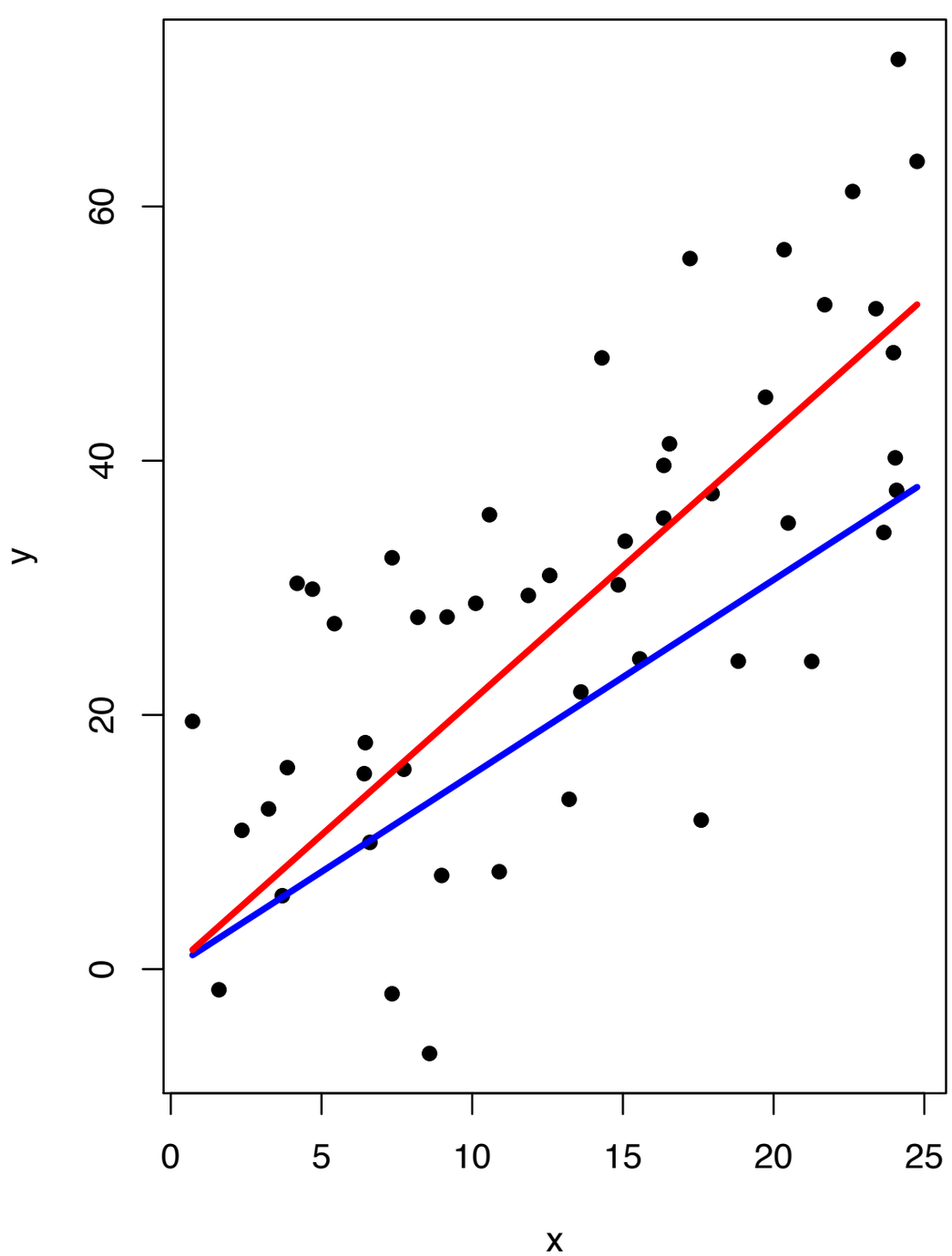


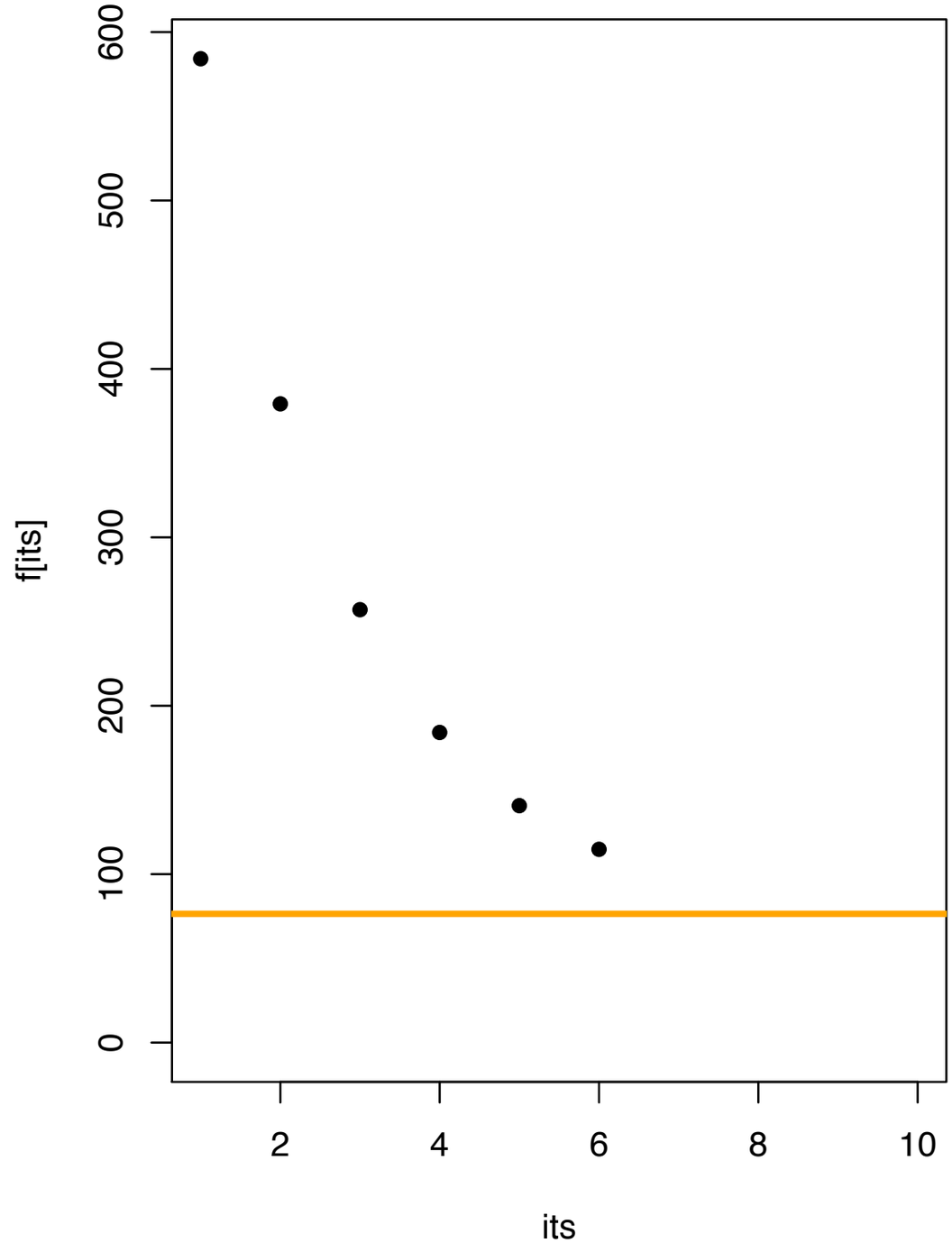
Example: Least Squares Regression



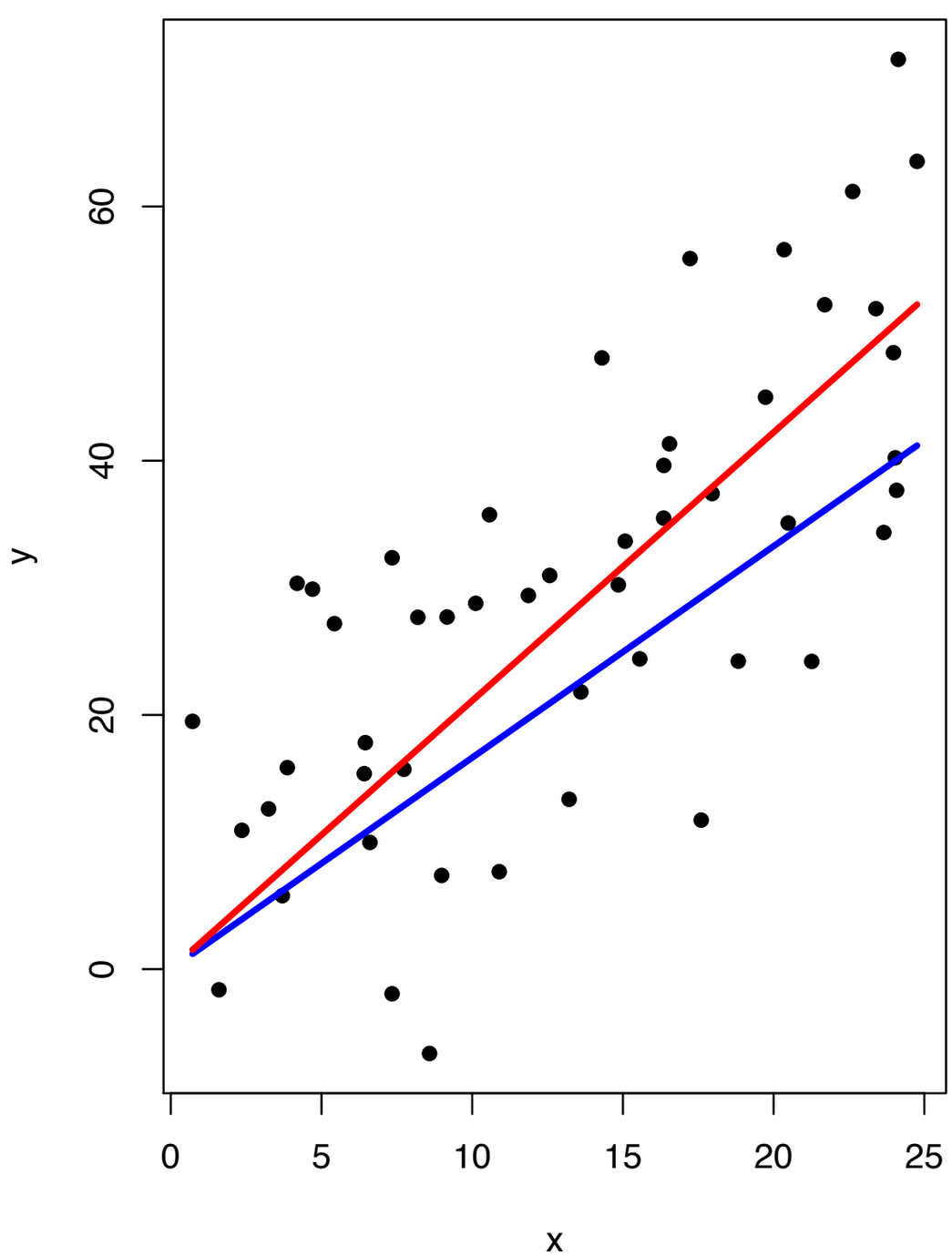


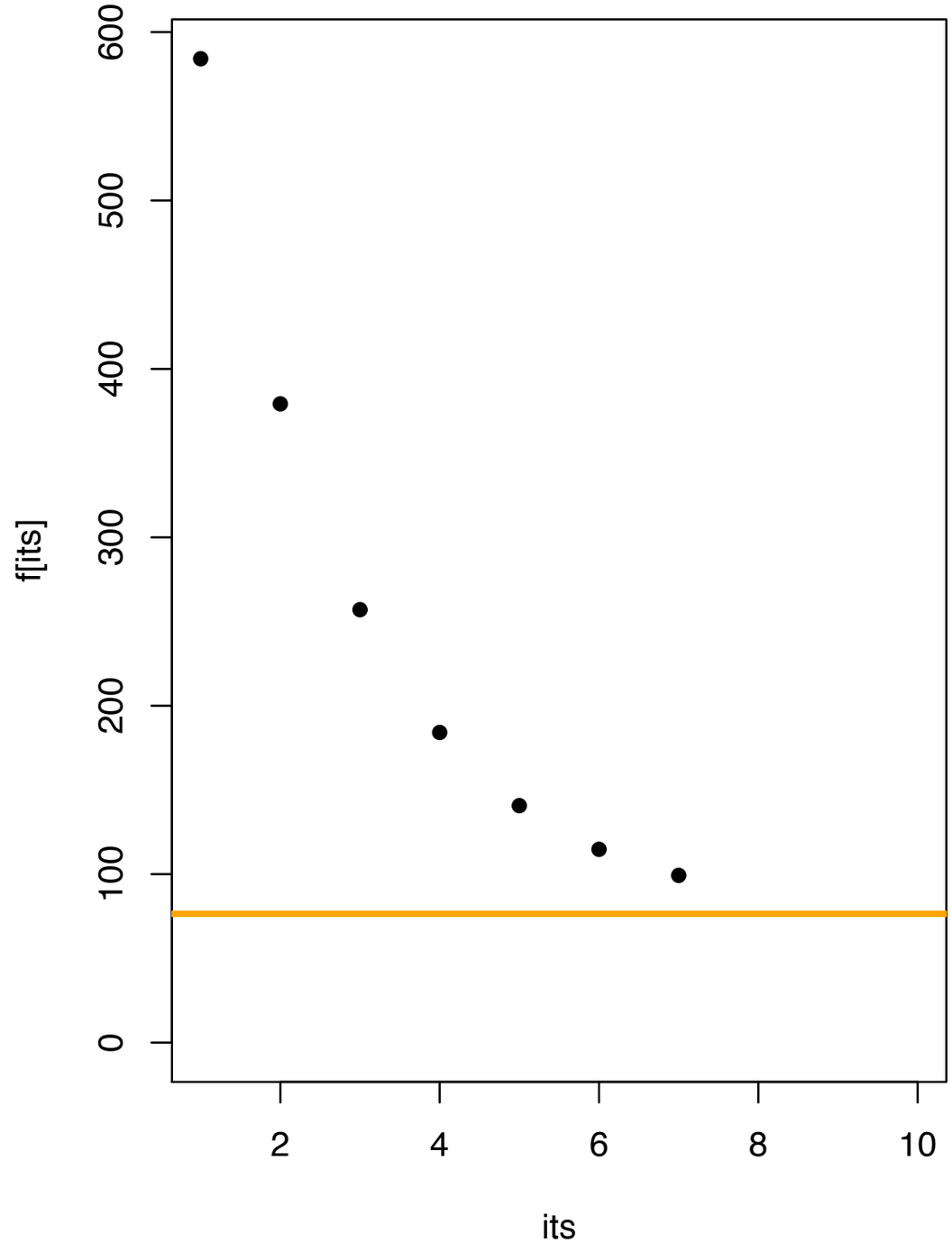
Example: Least Squares Regression



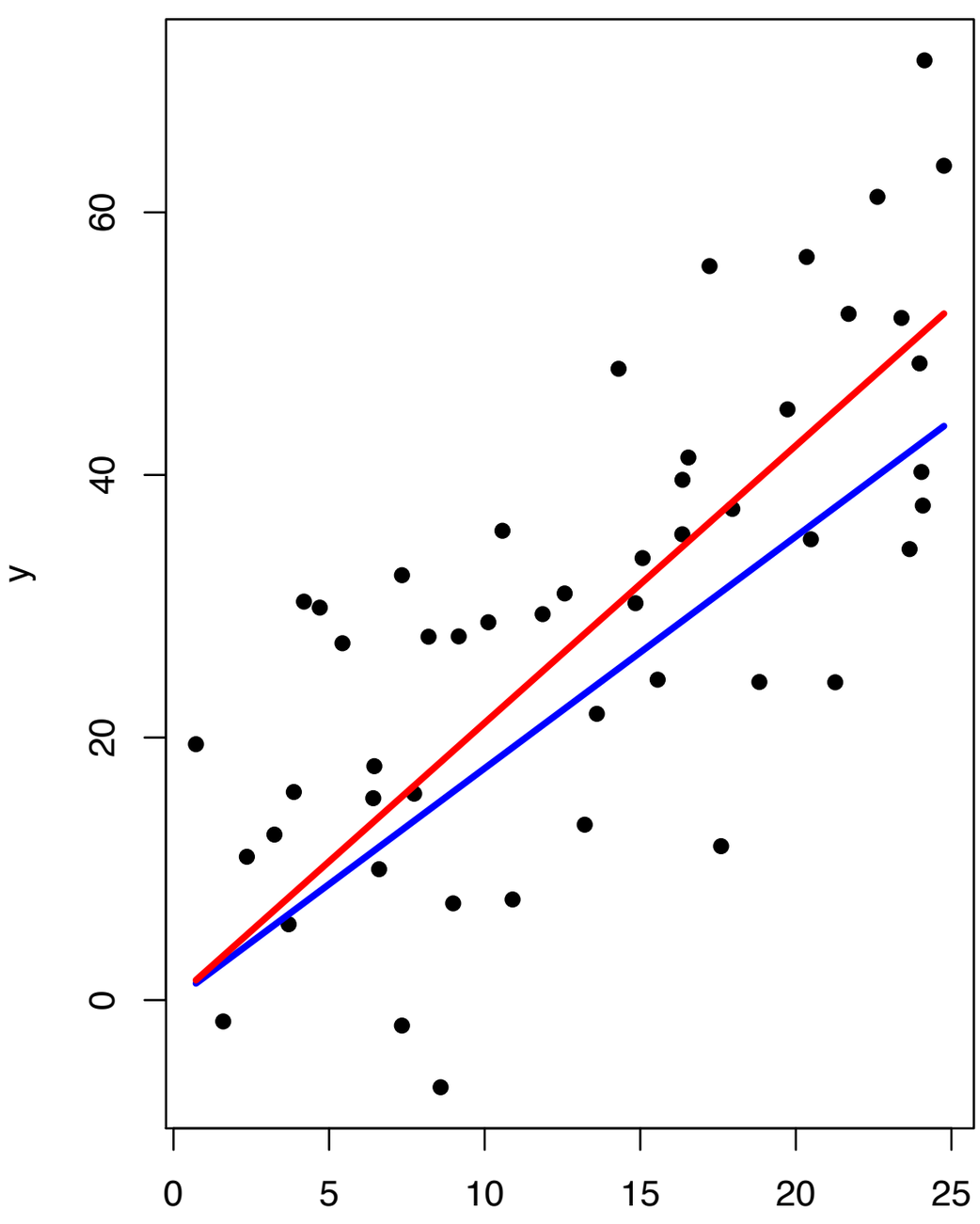


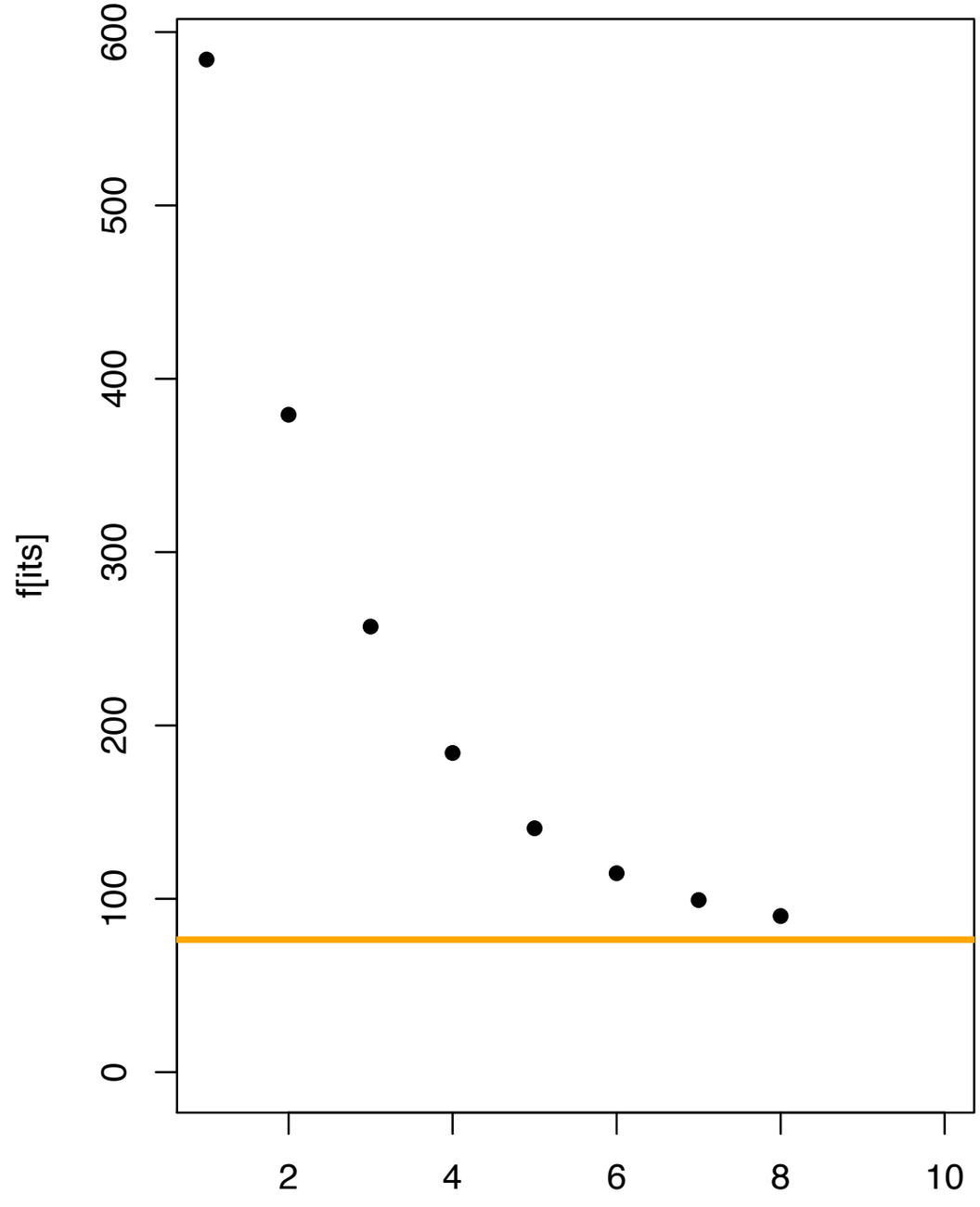
Example: Least Squares Regression



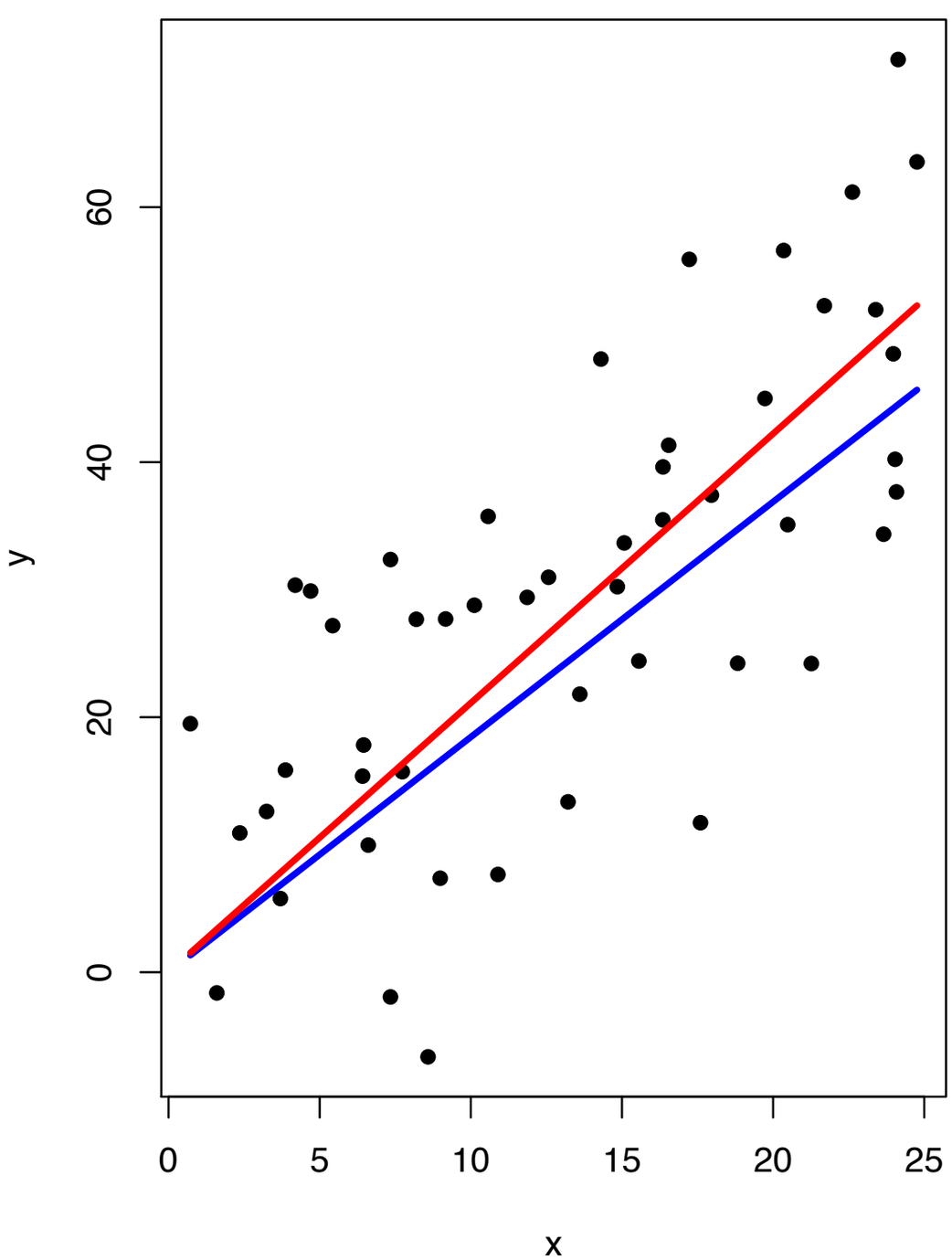


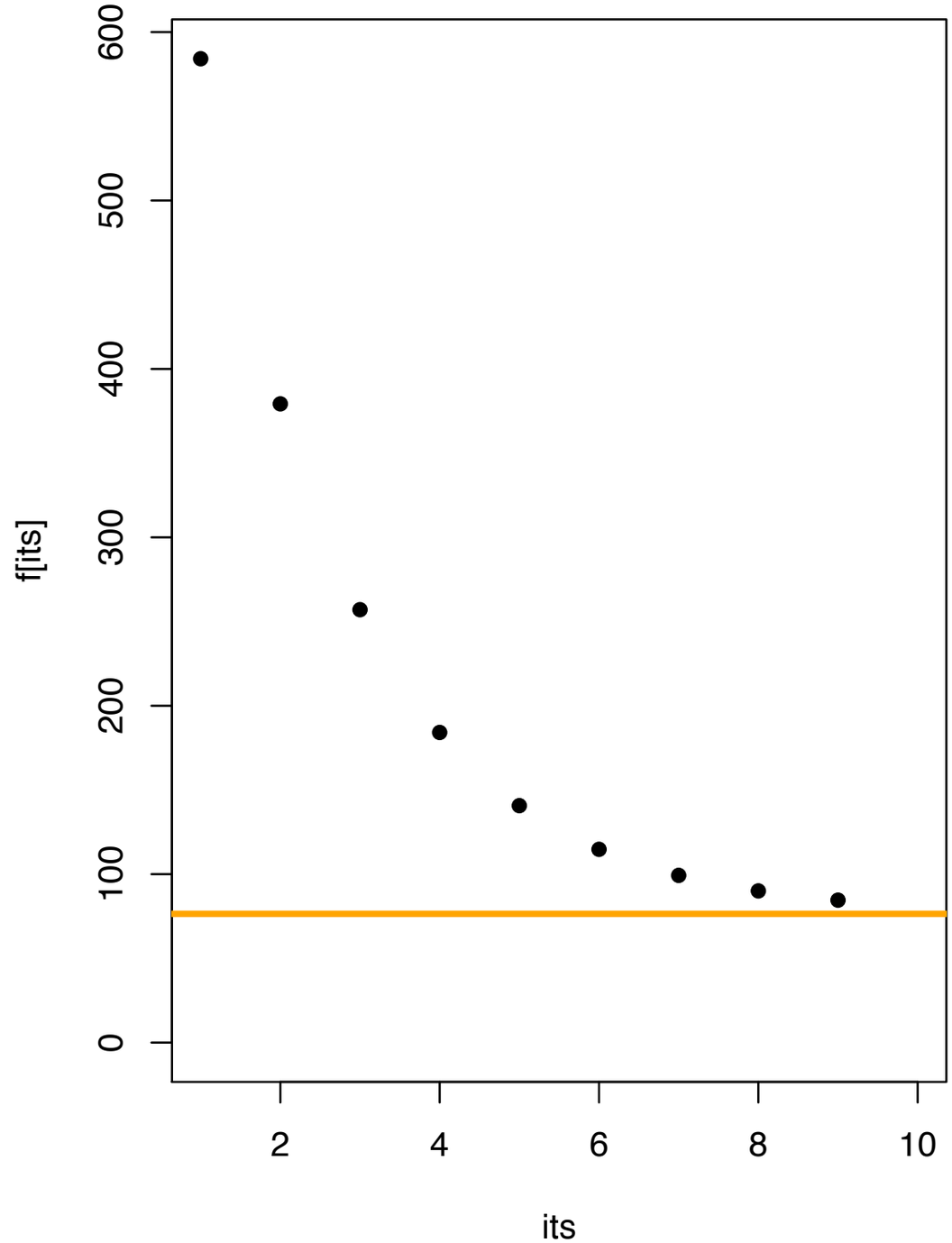
Example: Least Squares Regression



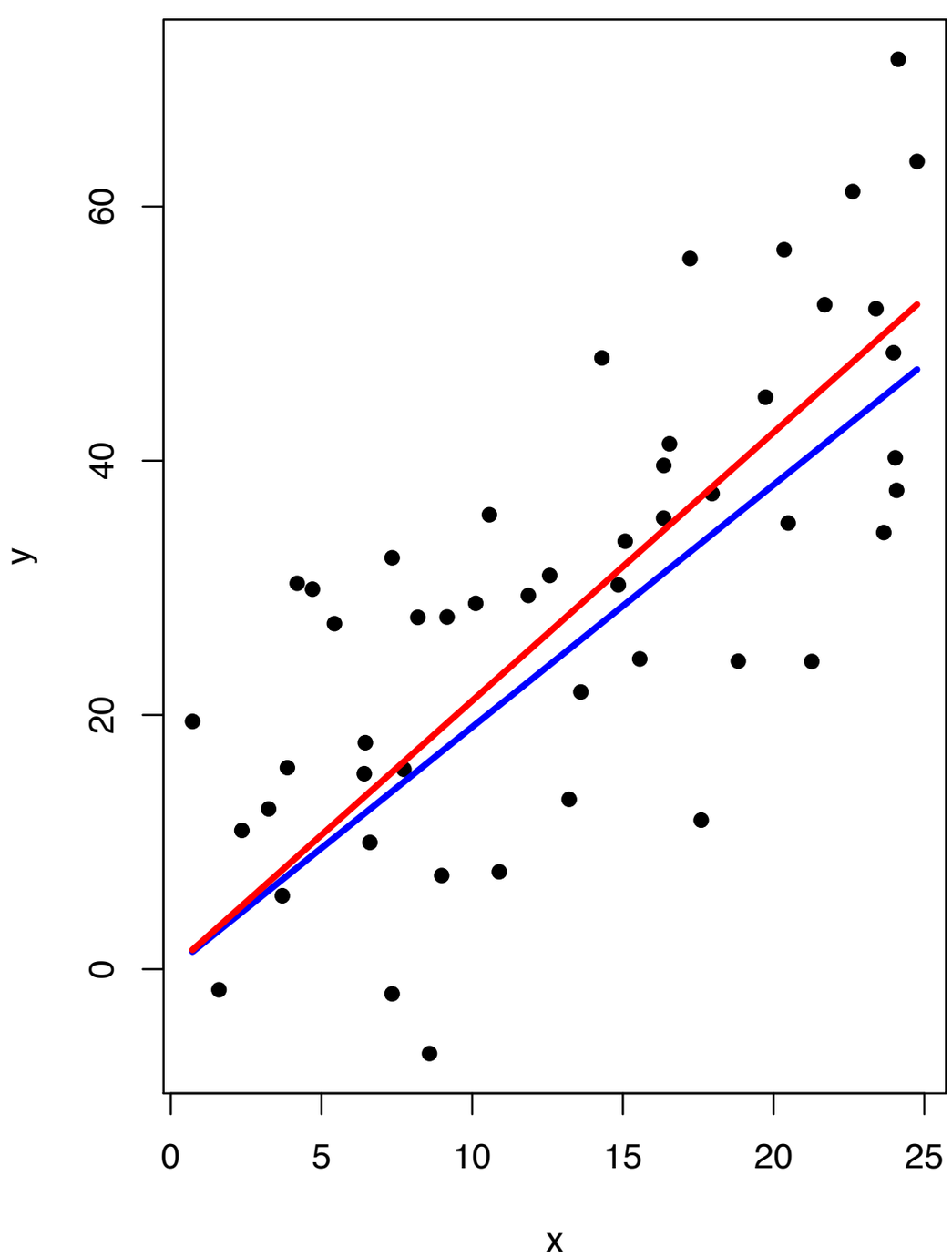


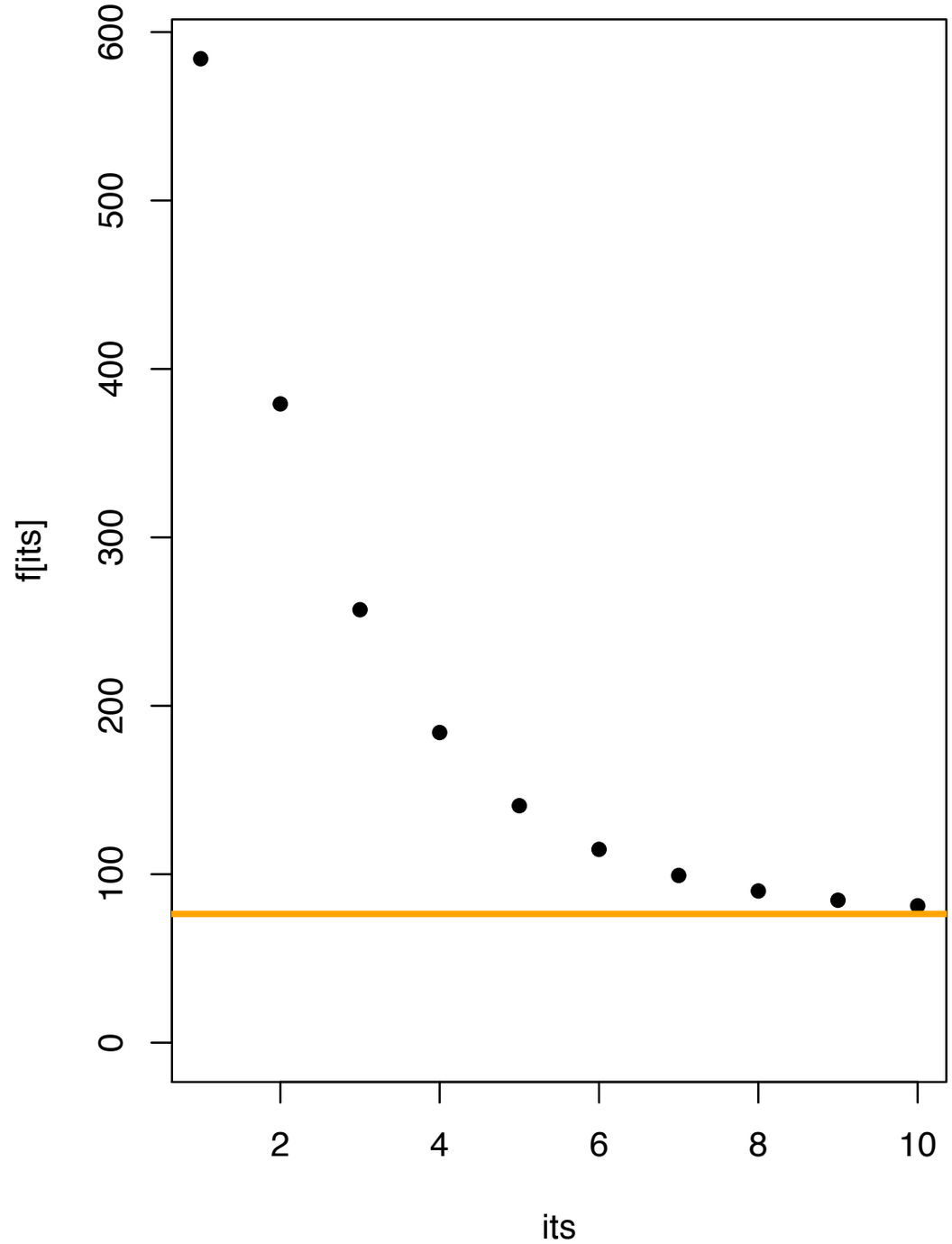
Example: Least Squares Regression





Example: Least Squares Regression





Example: Least Squares Regression

Computational complexity per iteration

Objective:

$$f(\theta) = \frac{1}{2}\|y - X\theta\|^2$$

Gradient:

$$\nabla f(\theta) = X^T(X\theta - y)$$

Gradient Step:

$$\theta_{m+1} = \theta_m - \alpha X^T(X\theta - y)$$

- $X\theta$ is $\mathcal{O}(np)$
- $X\theta - y$ is $\mathcal{O}(n)$
- $X^T(X\theta - y)$ is $\mathcal{O}(np)$
- $\theta_m - \alpha X^T(X\theta - y)$ is $\mathcal{O}(p)$

Total work for k iterations is $\mathcal{O}(knp)$; c.f. $\mathcal{O}(np^2)$ using QR - Practical consequence: Good approximate solution in less time than an exact solution

Deriving Gradient Descent

By Taylor's theorem for θ close to $\tilde{\theta}$,

$$f(\theta) \approx f(\tilde{\theta}) + \nabla f(\tilde{\theta})^T(\theta - \tilde{\theta}) + \frac{1}{2\alpha}\|\theta - \tilde{\theta}\|^2$$

Minimize the quadratic approximation:

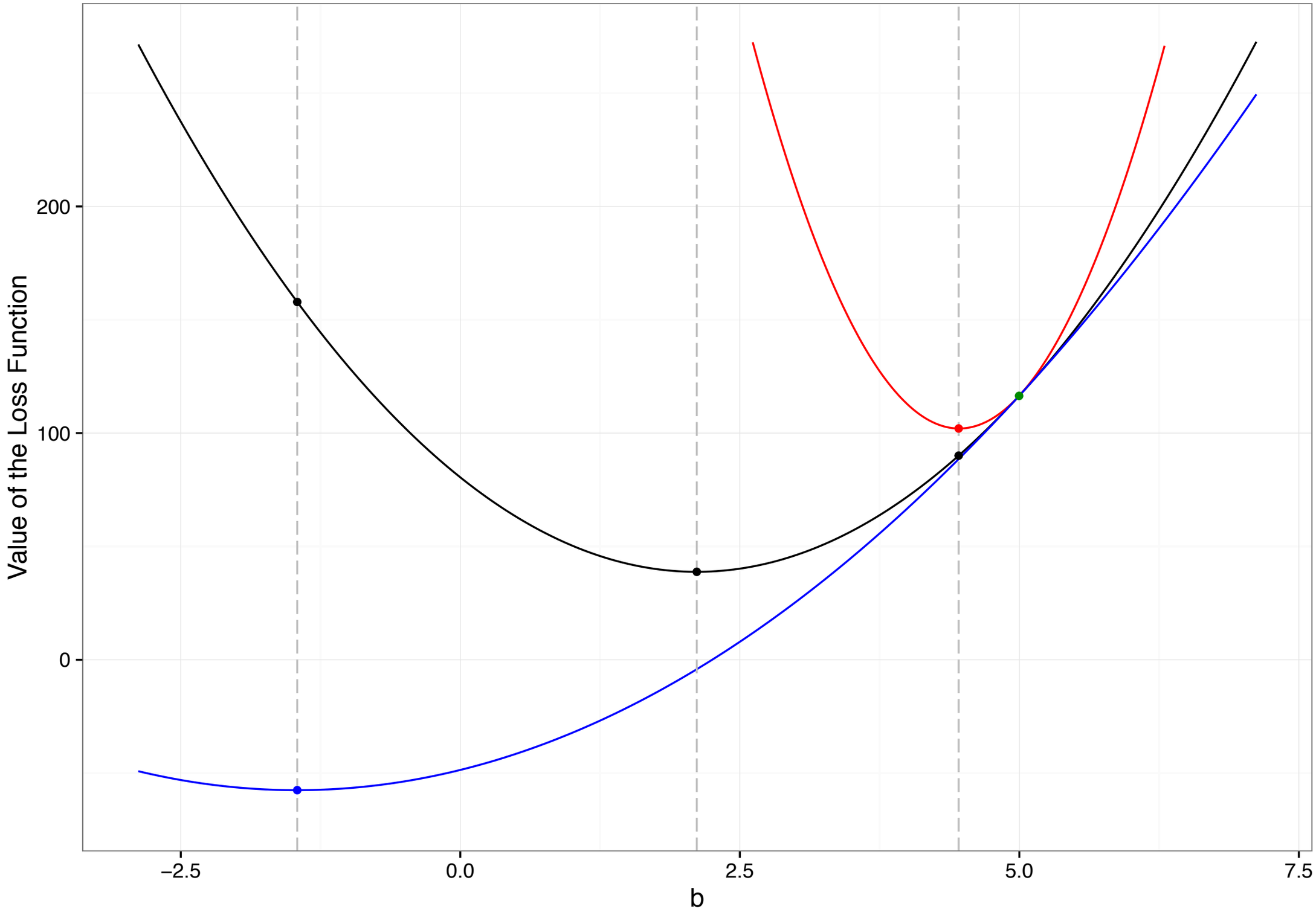
- Take derivative and set to zero

$$\nabla f(\tilde{\theta}) + \frac{1}{\alpha}(\theta - \tilde{\theta}) = 0$$

- Solve for θ

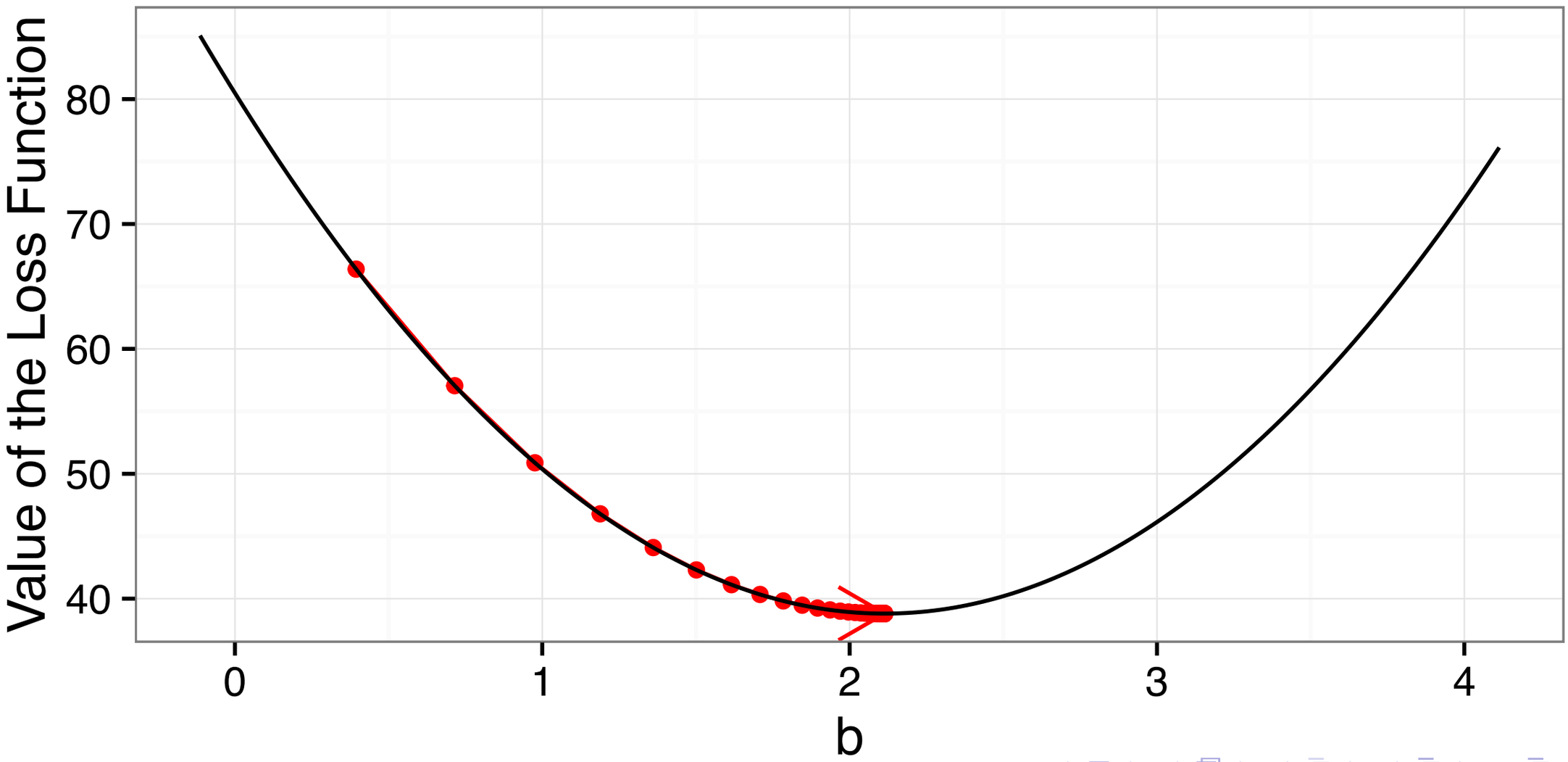
$$\theta = \tilde{\theta} - \alpha \nabla f(\tilde{\theta})$$

Effect of the Step Size



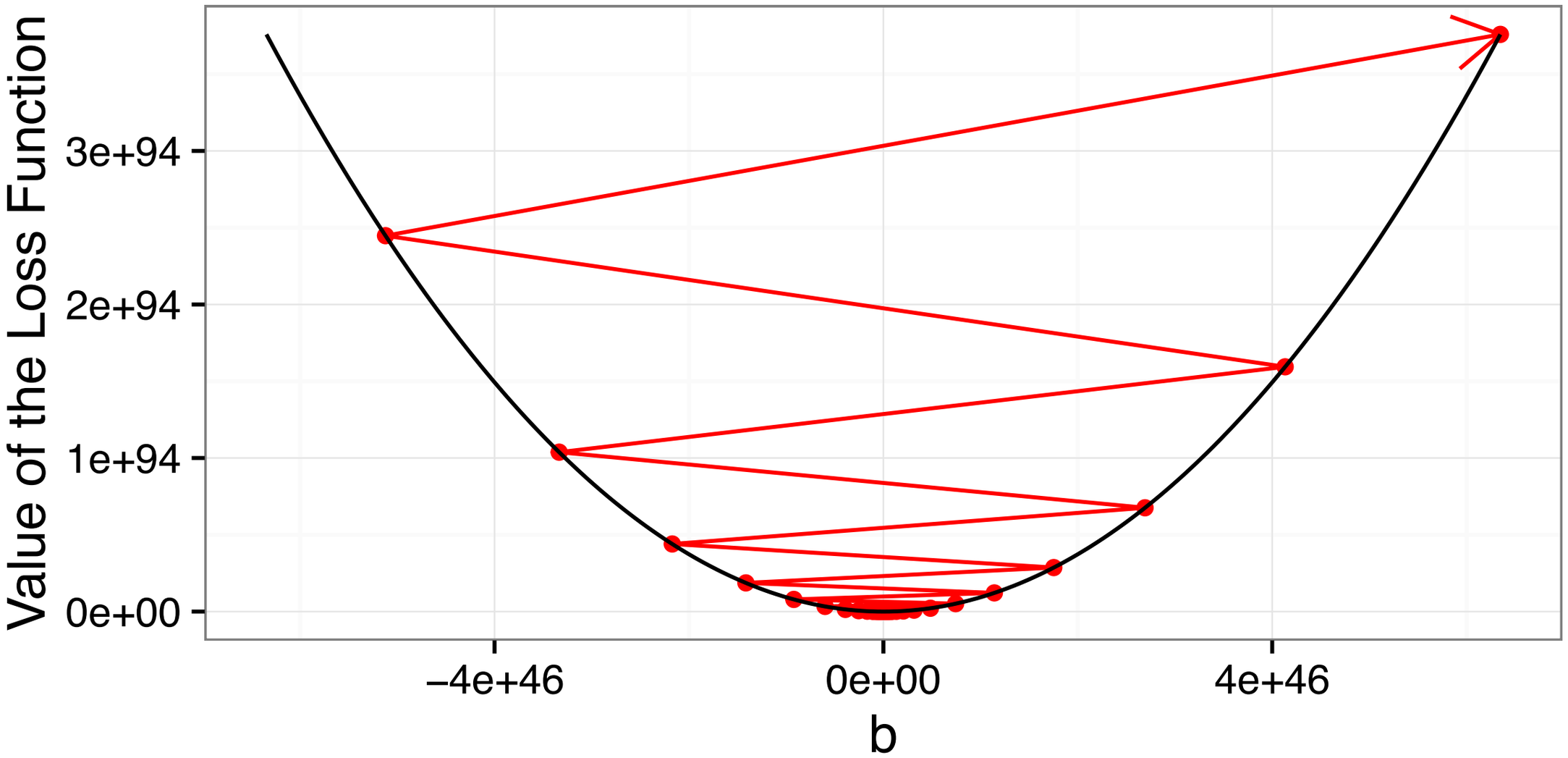
Small Step Size

```
## Minimum function value:
##  38.80753
##
## Coefficient(s):
##  2.113576
```



Big Step Size

```
## Minimum function value not attained. A better result mig
```



Singular Value Decomposition (SVD)

$X \in R^{m \times n}$ with rank k

The SVD for X always exists and is given by

$$X = U \Sigma V^T,$$

- $U \in R^{m \times r}$ is a matrix of left singular vectors
- $V \in R^{n \times r}$ is a matrix of right singular vectors
- $\Sigma \in R^{r \times r}$ is diagonal with positive entries σ_i (singular values)
- $U^T U = I$ and $V^T V = I$

Complexity:
For $m \geq n$, SVD requires $\mathcal{O}(mn^2)$ work to compute.

Operator Norm

$X \in R^{m \times n}$

$$\|X\| = \sup_{\|b\|_2=1} \|Xb\|_2$$

If $X = U\Sigma V^T$, then $\|X\| = \sigma_{\max}$.

Lipschitz Continuity

Definition: A function $g : R^m \to \mathbb{R}^n$ is Lipschitz continuous with parameter L if for all $x, y \in R^m$

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

Example: Gradient in Least Squares Linear Regression

$$f(\theta) = \frac{1}{2}\|y - X\theta\|_2^2$$

$$\nabla f(\theta) = X^T(X\theta - y)$$

$$\|\nabla f(\theta) - \nabla f(\tilde{\theta})\|_2 \leq \|X\|^2\|\theta - \tilde{\theta}\|_2$$

Lipschitz Differentiability & Quadratic Upper Bound

Proposition: Let $f : \mathbb{R}^n \to R$ be L-Lipschitz differentiable, i.e. ∇f is Lipschitz continuous with parameter L. Then for all $x, y \in R$,

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

Lipschitz Differentiability & Choosing a Step Size

$$g(\theta \mid \tilde{\theta}) := f(\tilde{\theta}) + \nabla f(\tilde{\theta})^T(\theta - \tilde{\theta}) + \frac{1}{2\alpha}\|\theta - \tilde{\theta}\|_2^2$$

Gradient update is the solution to

$$\theta^+ = \arg\min_{\theta} g(\theta \mid \tilde{\theta})$$

Observations:

- $f(\tilde{\theta}) = g(\tilde{\theta} \mid \tilde{\theta})$
- $f(\theta) \leq g(\theta \mid \tilde{\theta})$ if $\alpha \leq 1/L$

$$f(\tilde{\theta}) = g(\tilde{\theta} \mid \tilde{\theta}) \geq g(\theta^+ \mid \tilde{\theta}) \geq f(\theta^+)$$

Iteration Complexity of Gradient Descent

$$x_k = x_{k-1} - t_k \nabla f(x_{k-1}), \qquad k = 1, 2, \ldots$$

Assumptions:

1. f is convex and differentiable with dom $f = \mathbb{R}^n$
2. $\nabla f(x)$ is L-Lipschitz continuous
3. f attains its minimum value $f^\star$ at some $x^\star$

Questions:

- Does gradient descent converge? $f(x_k) \to f^\star$?
- If so, how close is $f(x_k)$ to $f^\star$ after k iterations?

Progress in a Single Step (Fixed Step Length)

Let $x^+ = x - t\nabla f(x)$, where $t \in (0, 1/L]$.

$$
\begin{aligned}
f(x^+) &\leq f(x) + \nabla f(x)^\mathsf{T}(x^+ - x) + \frac{L}{2}\|x^+ - x\|_2^2 \\
&= f(x) - t\|\nabla f(x)\|_2^2 + \frac{Lt^2}{2}\|\nabla f(x)\|_2^2 \\
&= f(x) - t(1 - \frac{Lt}{2})\|\nabla f(x)\|_2^2 \\
&\leq f(x) - \frac{t}{2}\|\nabla f(x)\|_2^2 \\
&\leq f^\star + \nabla f(x)(x - x^\star) - \frac{t}{2}\|\nabla f(x)\|_2^2 \\
&= f^\star + \frac{1}{2t}\left(\|x - x^\star\|_2^2 - \|x - x^\star - \nabla f(x)\|_2^2\right) \\
&= f^\star + \frac{1}{2t}(\|x - x^\star\|_2^2 - \|x^+ - x^\star\|_2^2)
\end{aligned}
$$

Progress after k Steps

Let $x = x_{i-1}$ and $x^+ = x_i$.

$$\begin{aligned}\sum_{i=1}^{k}(f(x_i) - f^\star) &\leq \frac{1}{2t}\sum_{i=1}^{k}\left(\|x_{i-1} - x^\star\|_2^2 - \|x_i - x^\star\|_2^2\right) \\ &= \frac{1}{2t}\left(\|x_0 - x^\star\|_2^2 - \|x_k - x^\star\|_2^2\right) \\ &\leq \frac{1}{2t}\|x_0 - x^\star\|_2^2\end{aligned}$$

Since $f(x_i)$ is non-increasing

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}(f(x_i) - f^\star) \leq \frac{1}{2kt}\|x_0 - x^\star\|_2^2$$

Bound on number of iterations so that $f(x_k) - f^\star \leq \epsilon$ is $\mathcal{O}(1/\epsilon)$

Backtracking Line Search

Q: What if I don't know L?
Initialize t_k at $t_0 > 0$. Set $t_k = \beta t_k$ until

$$f(x - t_k \nabla f(x)) < f(x) - \alpha t_k \|\nabla f(x)\|_2^2$$

$\beta \in (0, 1)$ and $\alpha = 1/2$.
If f is L-Lipschitz differentiable, $t_k \geq t_{\min} = \min\{\hat{t}, \beta/L\}$.

Progress in a Single Step (Backtracking)

$$
\begin{aligned}
f(x_i) &\leq f^\star + \frac{1}{2t_i}\left(\|x_{i-1} - x^\star\|_2^2 - \|x_i - x^\star\|_2^2\right) \\
&\leq f^\star + \frac{1}{2t_{\mathsf{min}}}\left(\|x_{i-1} - x^\star\|_2^2 - \|x_i - x^\star\|_2^2\right)
\end{aligned}
$$

Since $f(x_i)$ is non-increasing

$$
f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}(f(x_i) - f^\star) \leq \frac{1}{2kt_{\mathsf{min}}}\|x_0 - x^\star\|_2^2
$$

Bound on number of iterations so that $f(x_k) - f^\star \leq \epsilon$ is $\mathcal{O}(1/\epsilon)$

Strongly Convex Functions

Definition: A function f is strongly convex with parameter $m > 0$ if

$$f(x) - \frac{m}{2}\|x\|_2^2 \quad \text{is convex}$$

Jensen's Inequality

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) - \frac{m}{2}\alpha(1-\alpha)\|x-y\|_2^2$$

First-order condition

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{m}{2}\|x-y\|_2^2$$

Second-order condition

$$\nabla^2 f(x) \succeq mI$$

Gradient descent applied to strongly convex function

Progress in a Single Step (Fixed Step Length)

$$\|x^+ - x^\star\|_2^2 \leq (1 - t\frac{2mL}{m+L})\|x - x^\star\|_2^2$$

Distance to optimum

$$\|x_k - x^\star\|_2^2 \leq c^k\|x_0 - x^\star\|_2^2, \quad c = 1 - t\frac{2mL}{m+L}$$

Therefore,

$$f(x_k) - f^\star \leq \frac{L}{2}\|x_k - x^\star\|_2^2 \leq \frac{c^k L}{2}\|x_0 - x^\star\|_2^2$$

Bound on number of iterations so that $f(x_k) - f^\star \leq \epsilon$ is $\mathcal{O}(\log(1/\epsilon))$

Modern Unsupervised Learning

- Extract complicated features for classification, etc.
- State of the art in image classification

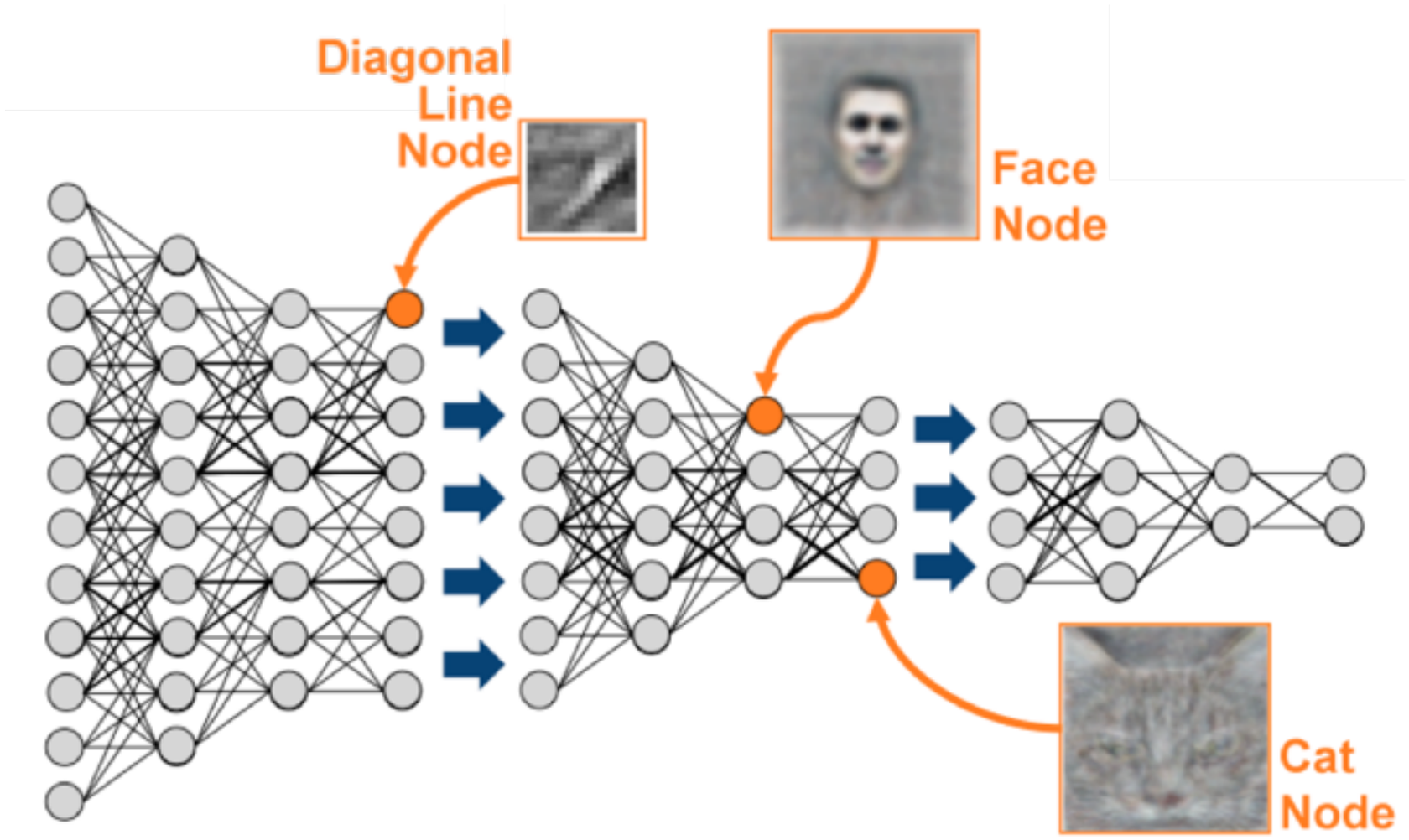
Google DeepMind: Early Example in 2012

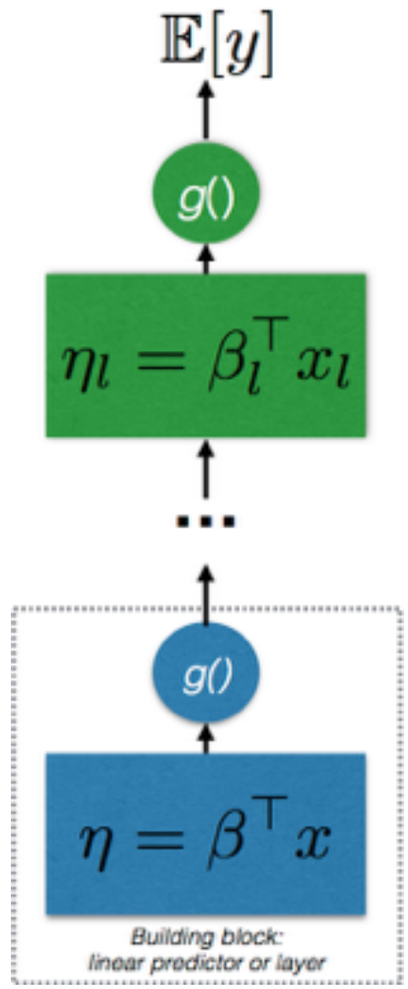
- 10 million digital images found in YouTube videos
- A lot of data!
- 16,000 computer processors

Modern Unsupervised Learning

- Extract complicated features for classification, etc.
- State of the art in image classification

Recursive Generalised Linear Models





See blog post: A Statistical View of Deep Learning

Challenge: Need to fit a very large (non-convex) complicated regression model on a **lot** of data

Reliable Solution: Stochastic Gradient Method + Block Coordinate Descent (Backpropagation)

Stochastic & Incremental Gradient Methods

- Suppose we can't get function values $f(x)$
- At any feasible x, have access only to an unbiased estiamte of an element of the subgradient ∂f

$$f(x) = EF(x, \xi)$$

where ξ is a random vector with distribution P over a set Ξ, e.g.

$$f(x) = \sum_{i=1}^{n} \frac{1}{n} f_i(x)$$

where each f_i is convex and may be nonsmooth

"Classical" Stochastic Approximation

Let $G(x, \xi)$ be a subgradient estimate generated at x. Suppose it is unbiased:

$$EG(x, \xi) \in \partial f(x)$$

Basic SA Scheme: At iteration k, choose ξ_k i.i.d. acccording to distribution P, choose some $\alpha_k > 0$ and set

$$x_{k+1} = x_k - \alpha_k G(x_k, \xi_k)$$

- Note x_{k+1} depends on $\xi_{[k]} = \{\xi_1, \ldots, \xi_k\}$

Research in Stochastic Gradient Methods is very active

- Stochastic Average Gradient (SAG)
- SAGA
- Stochastic Dual Coordinate Ascent (SCDA)
- Proximal SCDA
- Accelerated Proximal SCDA
- Stochastic Variance Reduced Gradient (SVRG)
- Stochastic Quasi-Newton Methods
- ...

Summary

Gradient Descent

- An example of an MM algorithm (with appropriately chosen step size)
- Convergence is guaranteed
- Worst case iteration complexity is “1/k”
- Each gradient step is “cheap” to compute

Root Finding

- $f : \mathbb{R} \mapsto \mathbb{R}$, differentiable
- Find $f(x^\star) = 0$

$$f(y) = f(x) + f'(x)(y - x) + o(|y - x|^2)$$

- If x is close to y, then

$$0 \approx f(x) + f'(x)(y - x)$$

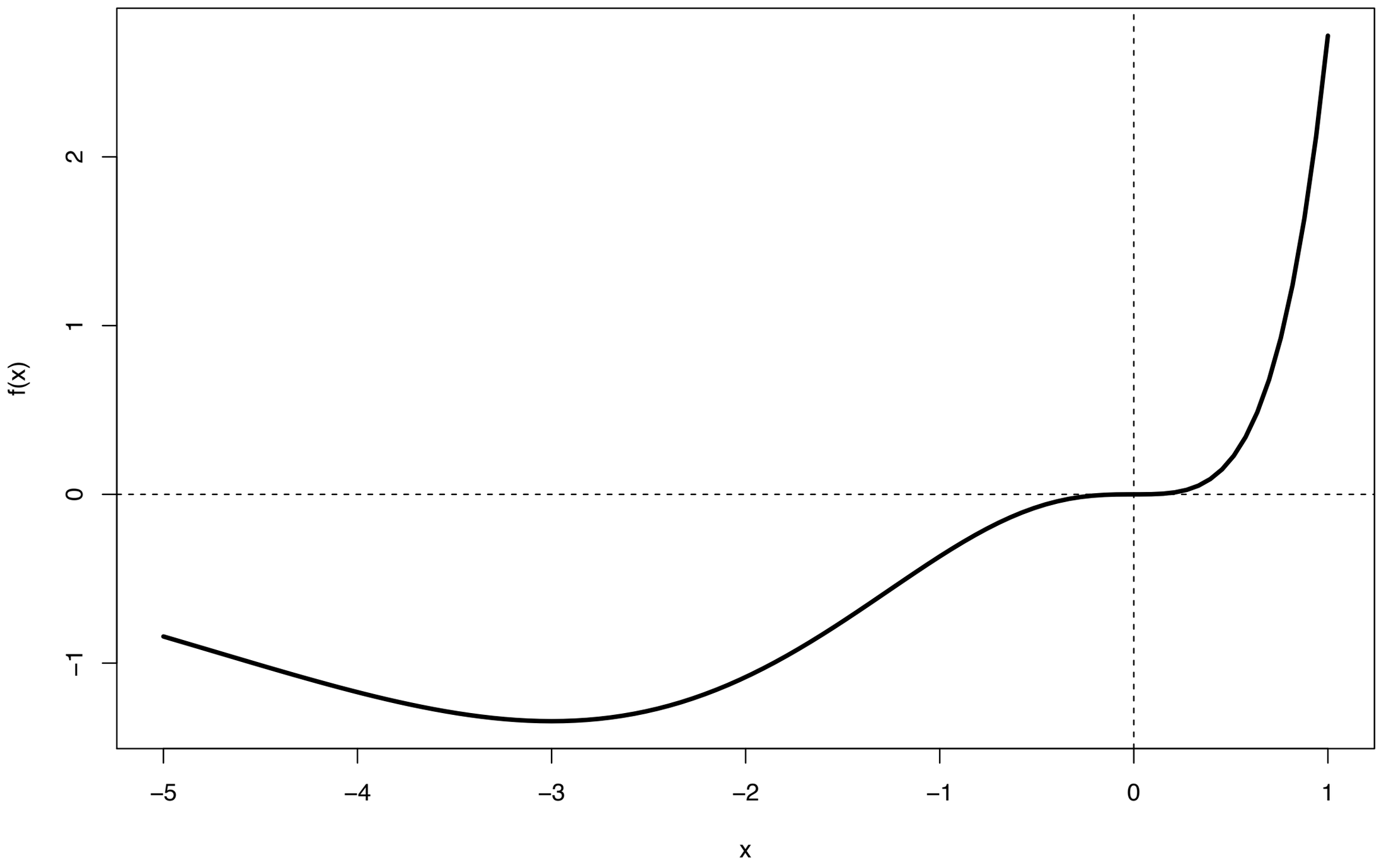
$$y = x - \frac{f(x)}{f'(x)}$$

Newton's method:

$$x^{(m+1)} = x^{(m)} - \frac{f\left(x^{(m)}\right)}{f'\left(x^{(m)}\right)}$$

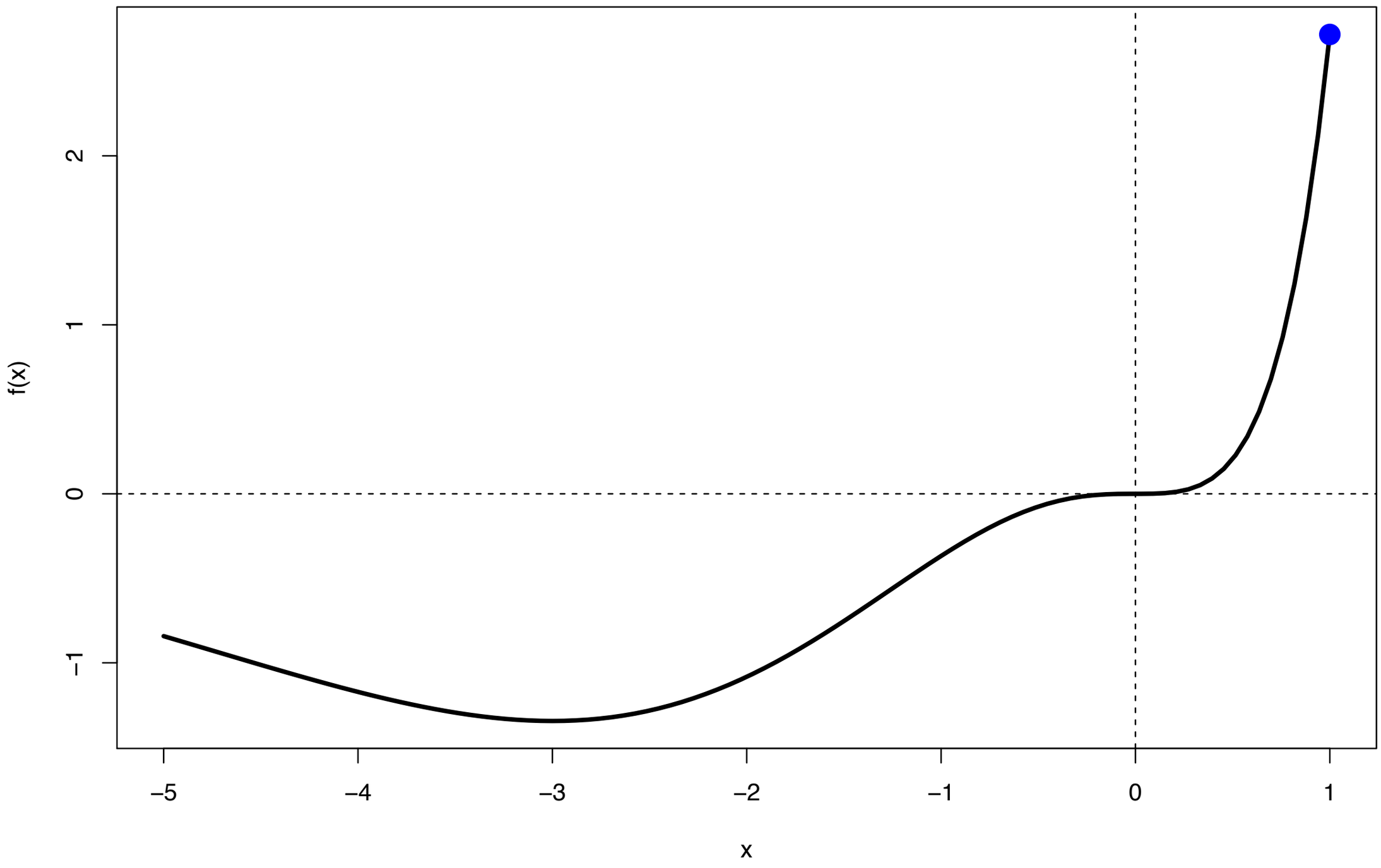
Example 1

$$f(x) = x^3 e^x$$



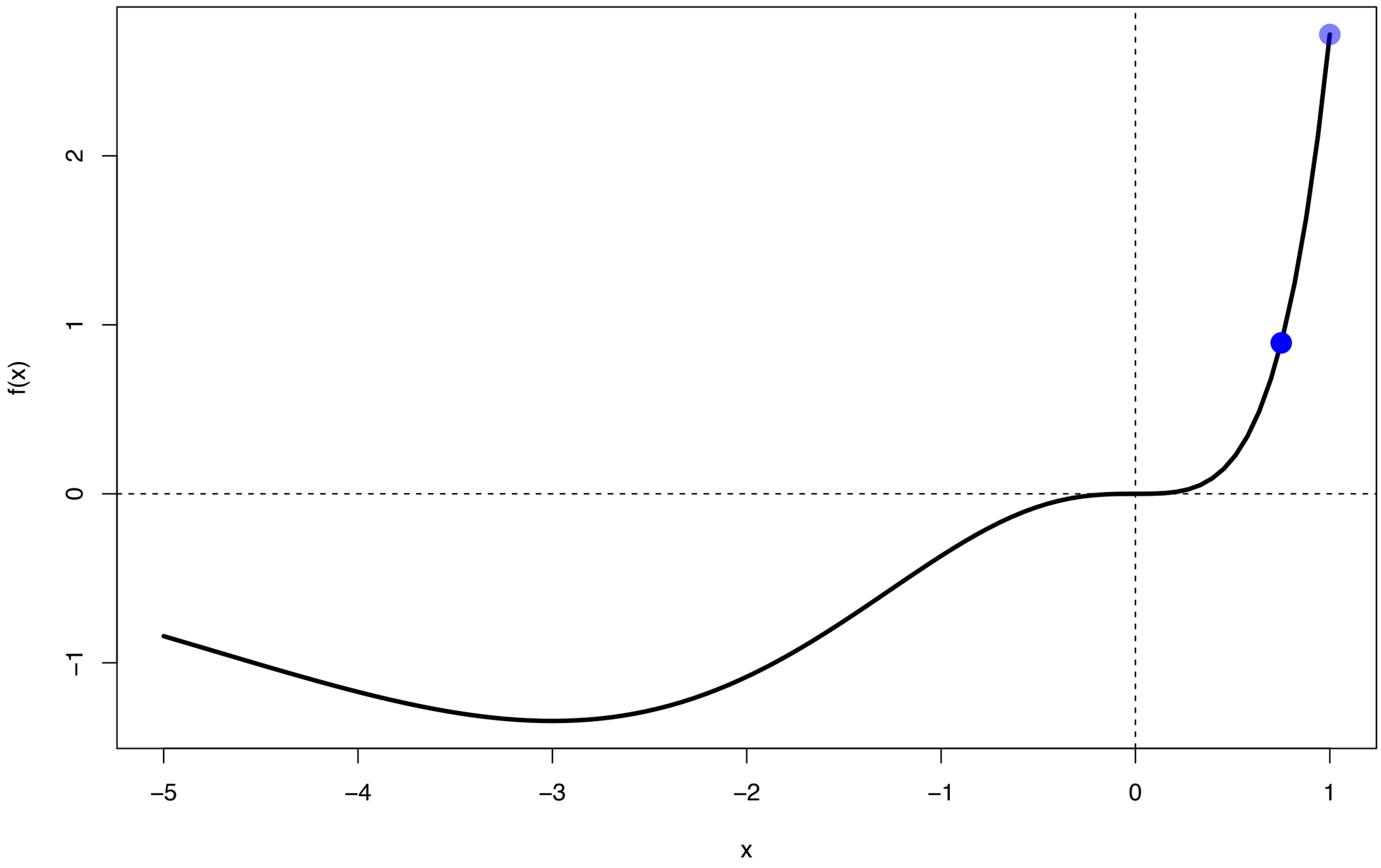
Example 1

$$f(x) = x^3 e^x$$



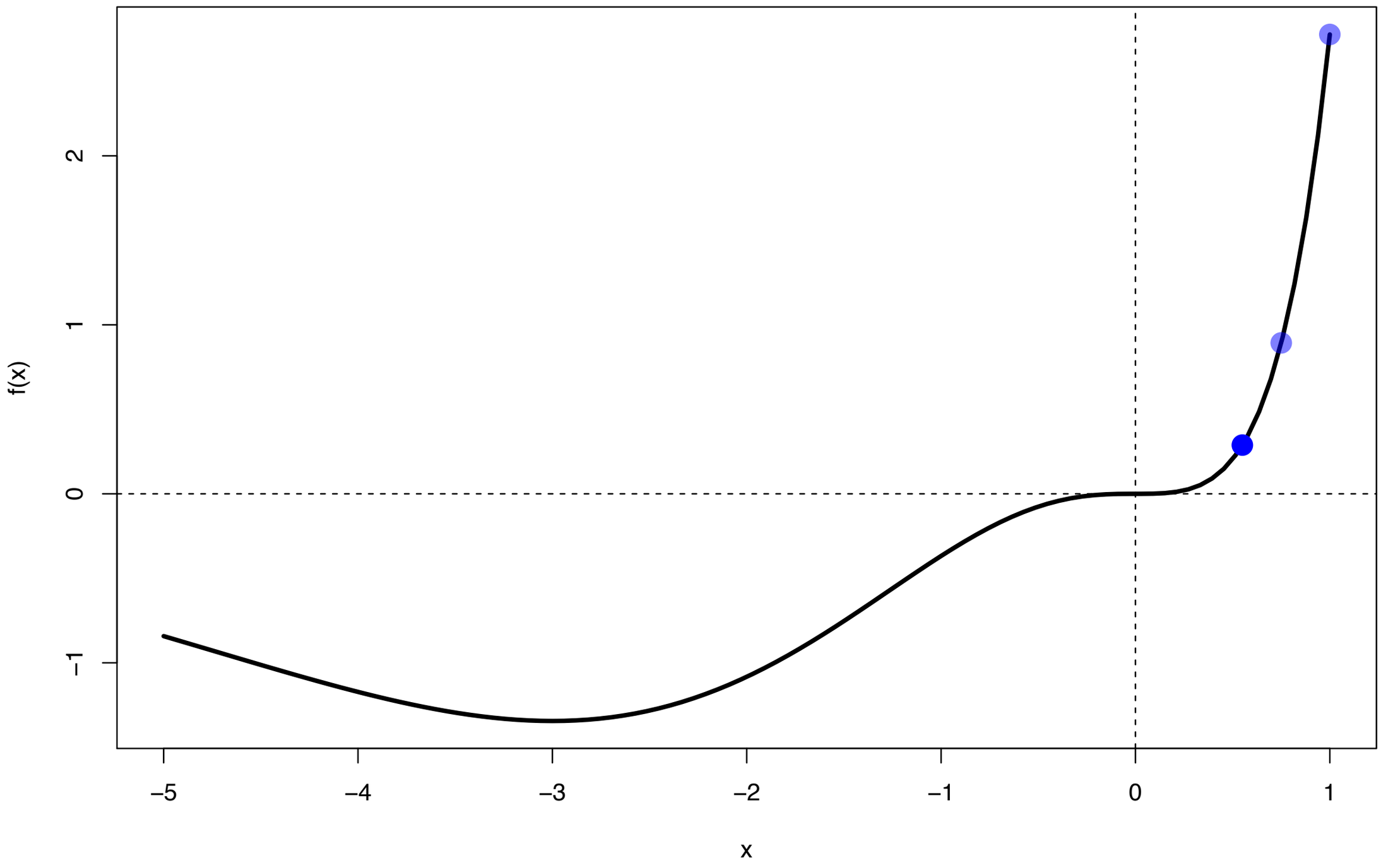
Example 1

$$f(x) = x^3 e^x$$



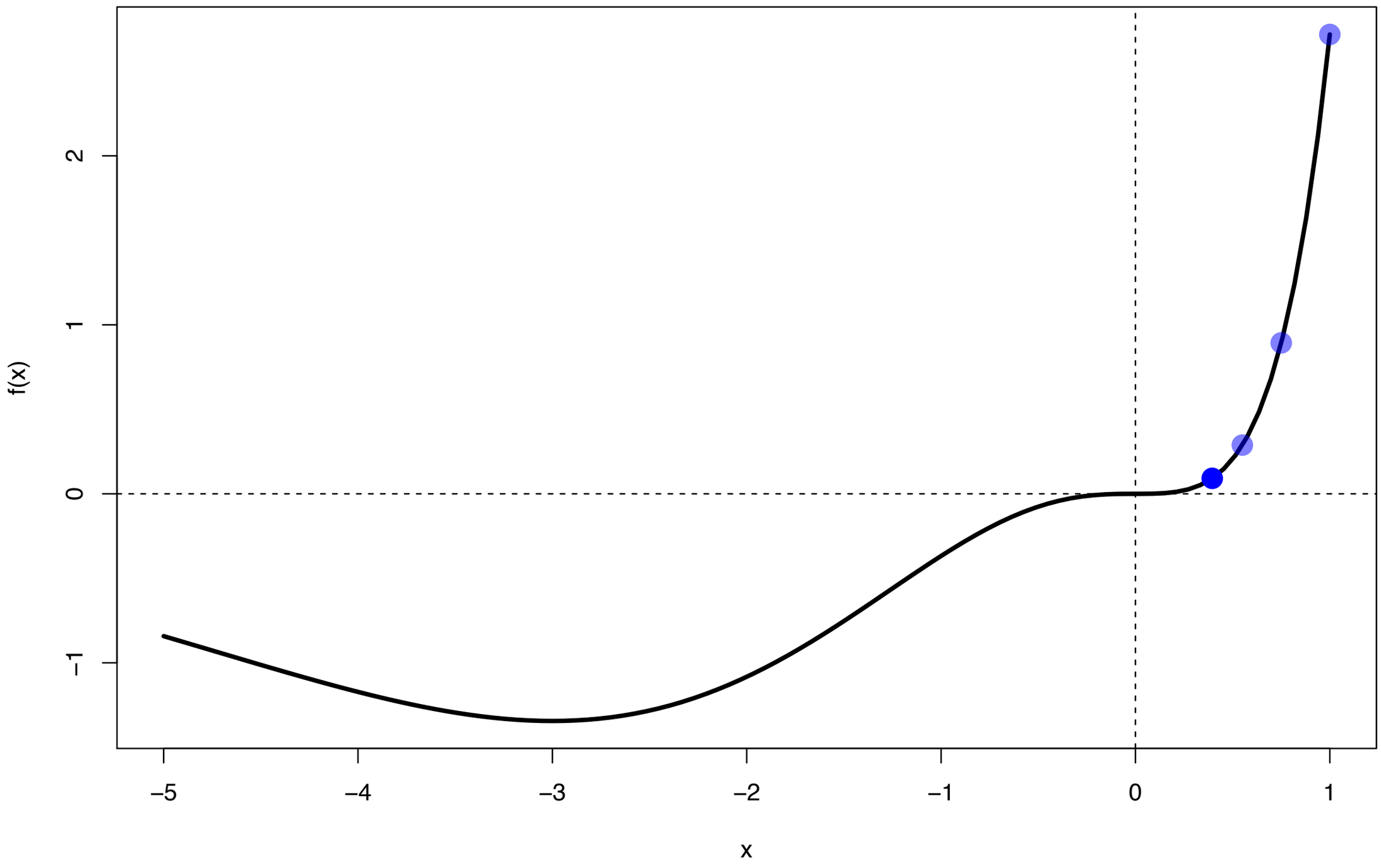
Example 1

$$f(x) = x^3 e^x$$



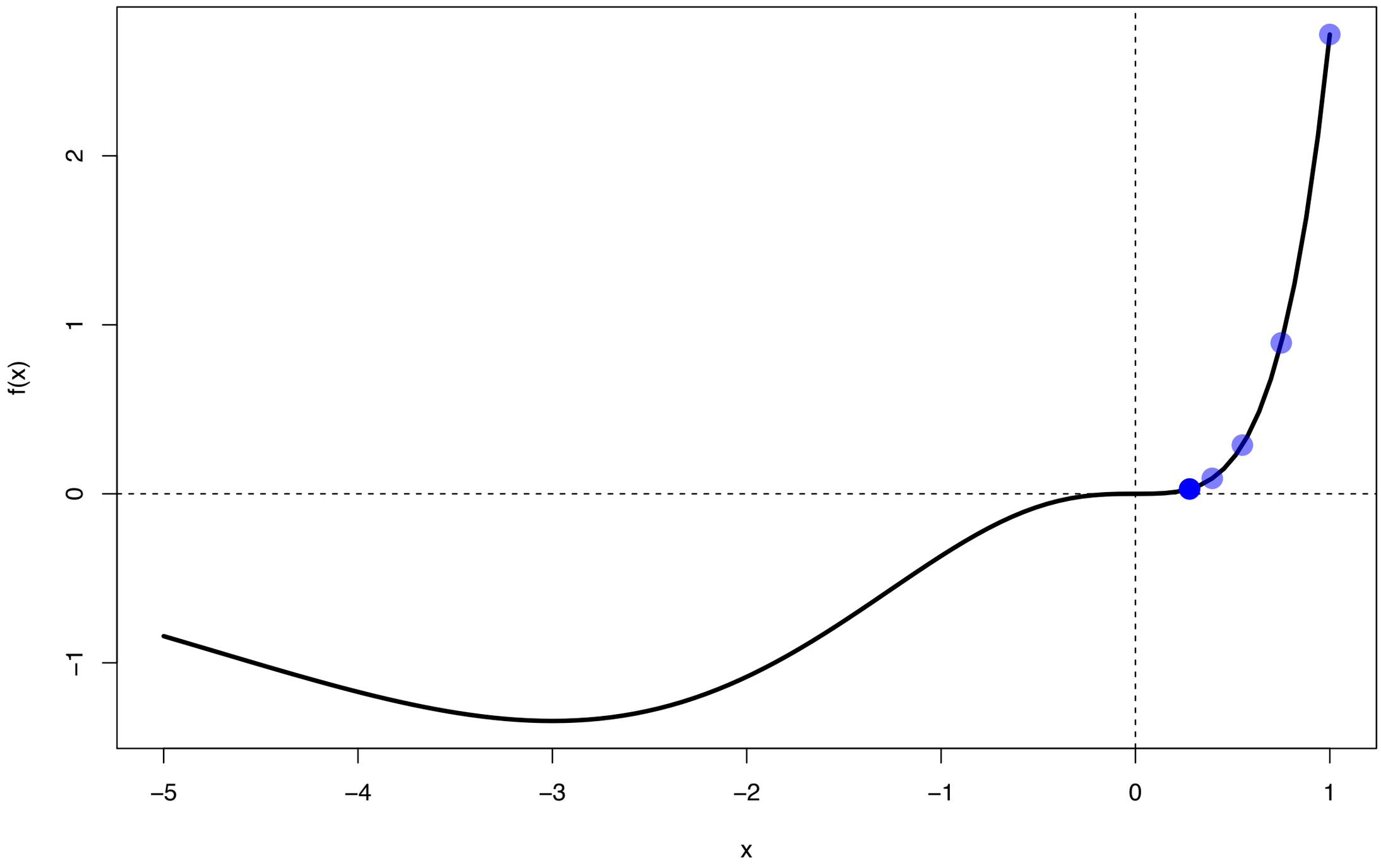
Example 1

$$f(x) = x^3 e^x$$



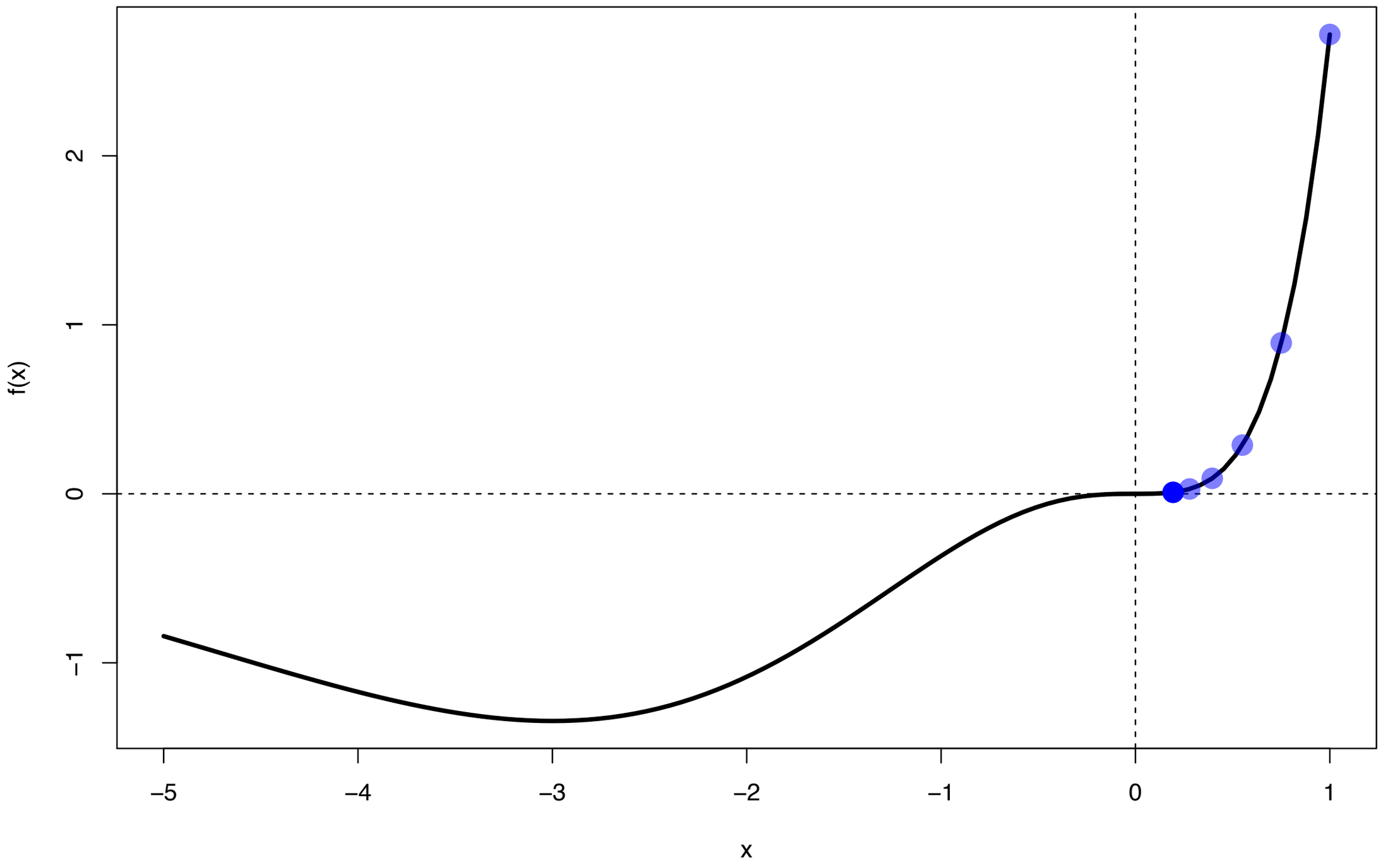
Example 1

$$f(x) = x^3 e^x$$



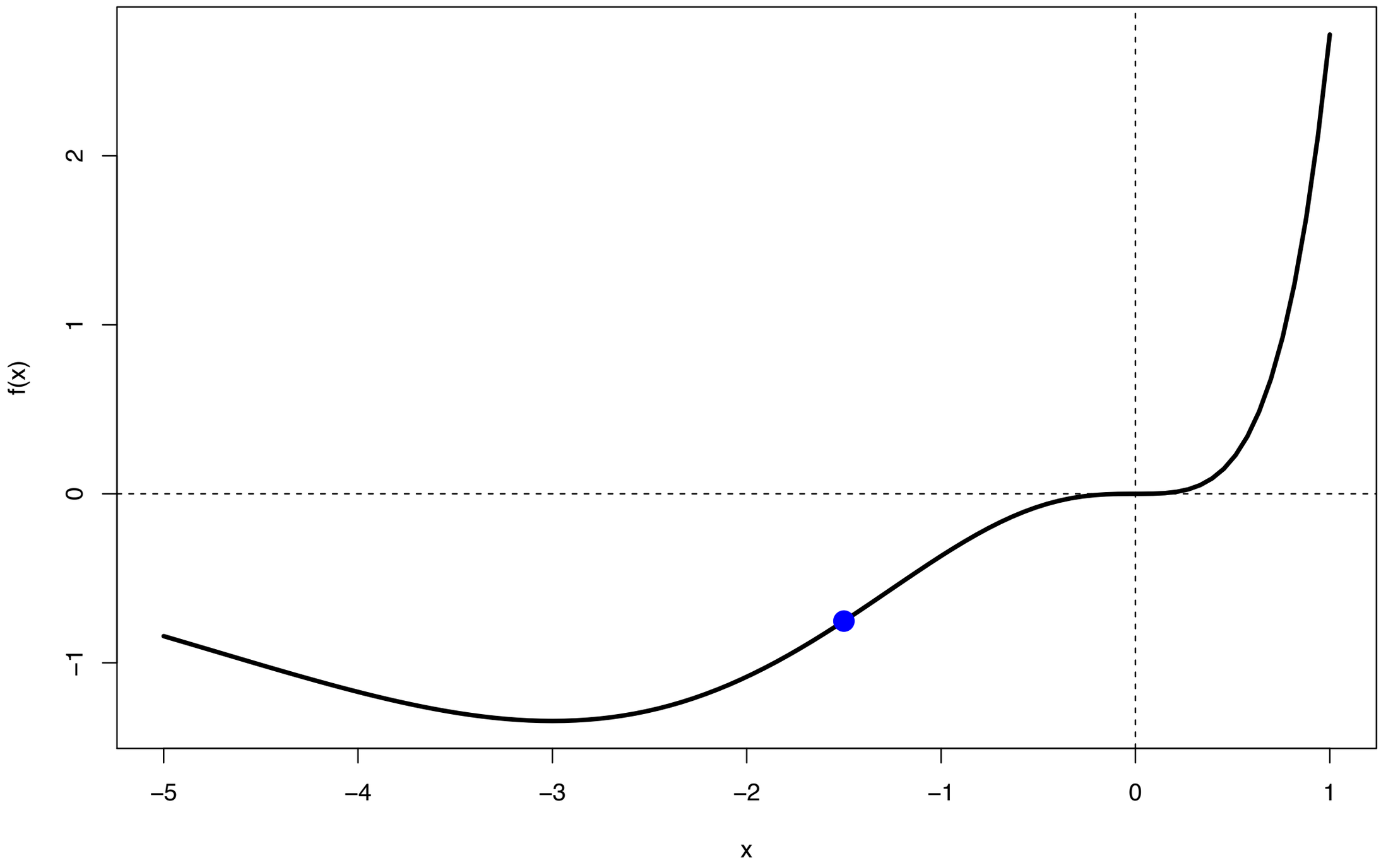
Example 1

$$f(x) = x^3 e^x$$



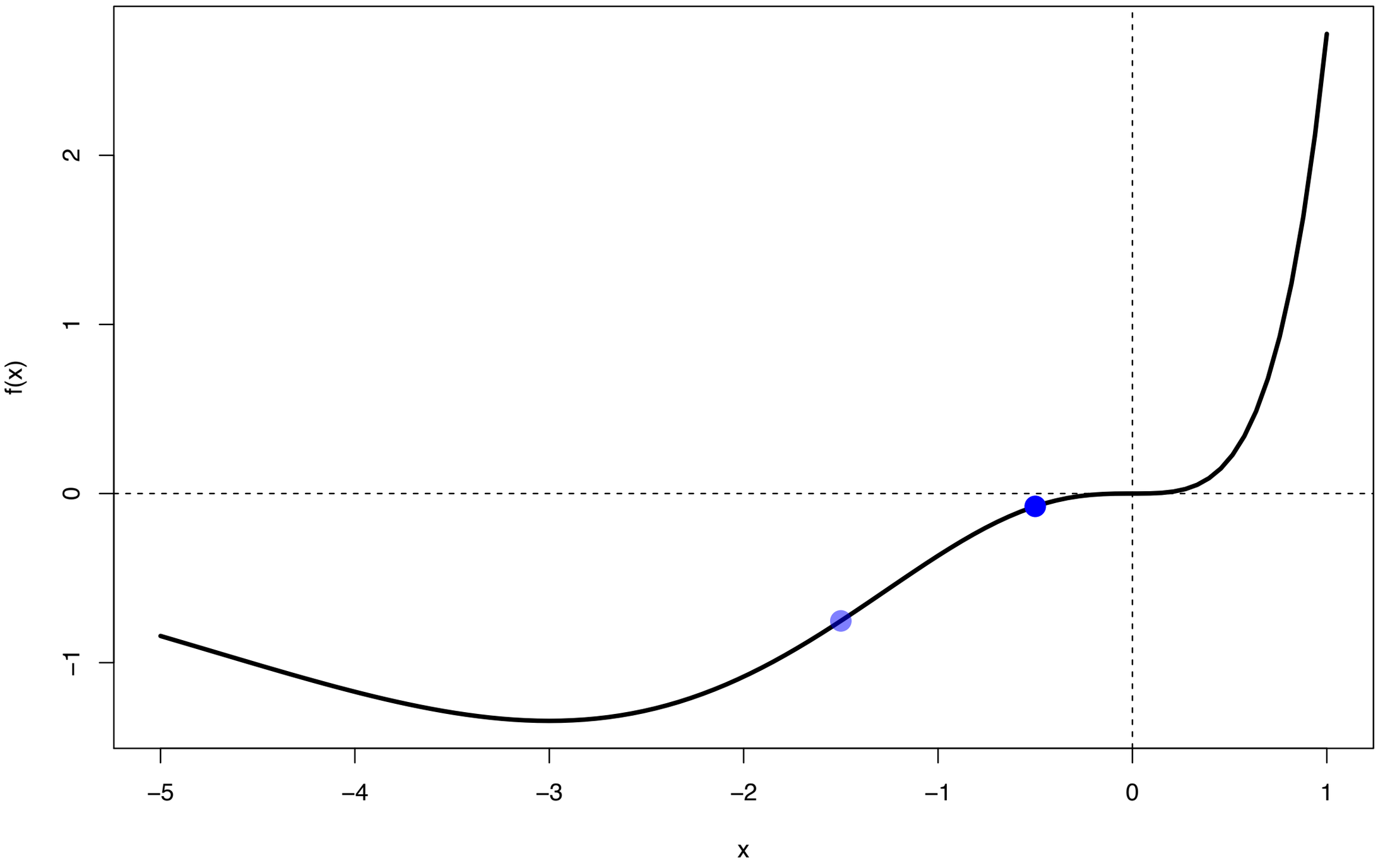
Example 1

$$f(x) = x^3 e^x$$



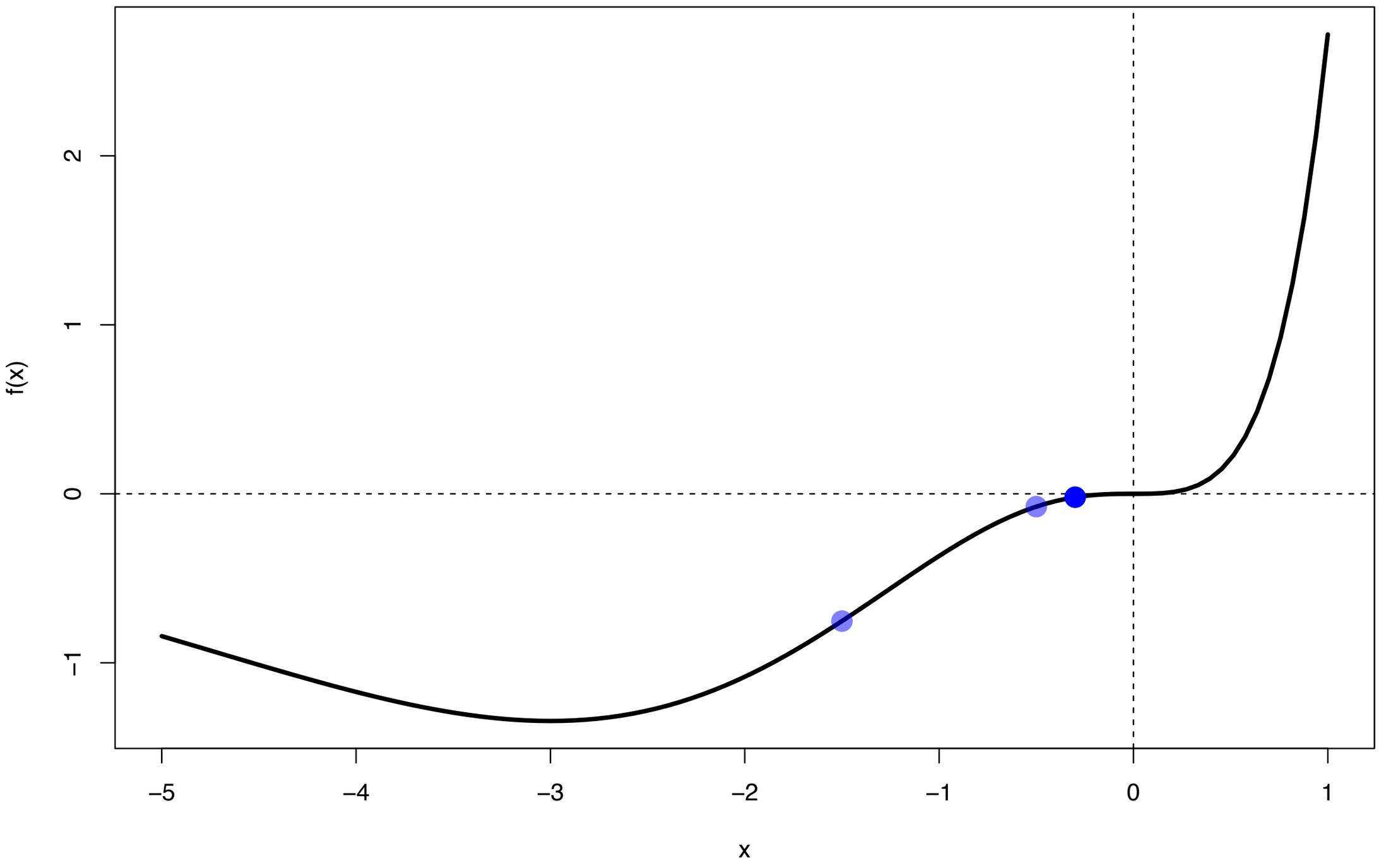
Example 1

$$f(x) = x^3 e^x$$



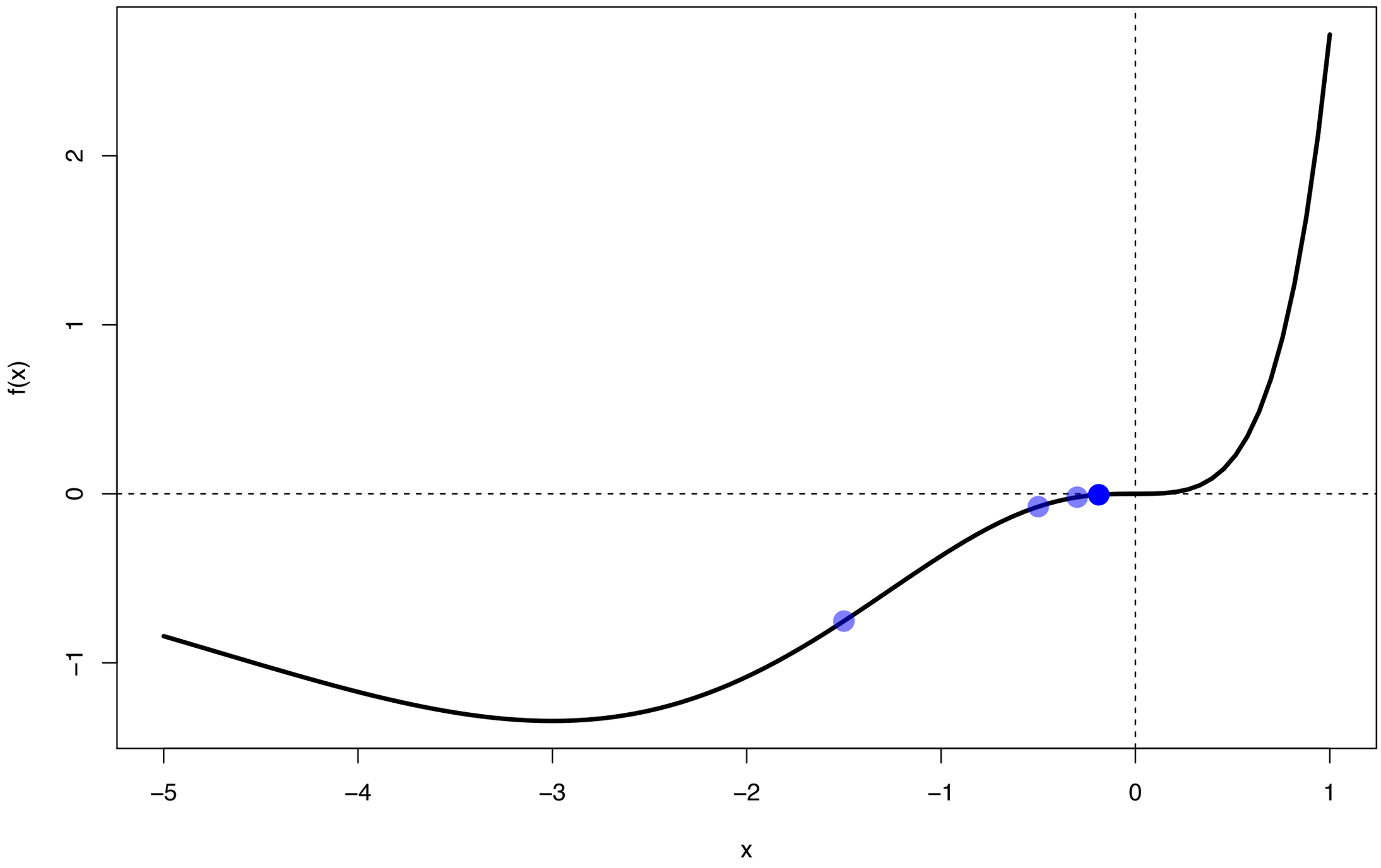
Example 1

$$f(x) = x^3 e^x$$



Example 1

$$f(x) = x^3 e^x$$



Example 2

Babylonian Algorithm for Square Root of a

$$f(x) = x^2 - a$$

Newton Update

$$x^{(m+1)} = \frac{1}{2}\left(x^{(m)} + \frac{a}{x^{(m)}}\right)$$

Example 2

```
sqrt_babylonian <- function(s, x_init=10,max_iter=1e2) {
  x_last <- x_init
  x <- double(max_iter)
  for (iter in 1:max_iter) {
    x_last <- x[iter] <- 0.5*(x_last + s/x_last)
  }
  return(x)
}
```

Iteration	$x^{(m)}$
0	10
1	5.1
2	2.7460784
3	1.7371949
4	1.4442381
5	1.4145257

Relationship between Root Finding and Optimization

- Maximum likelihood estimation

$$\min_{\theta} \ell(\theta)$$

- $\ell(\theta)$ is twice differentiable
- Seek root of $\nabla \ell(\theta)$
- Taylor expansion at a root $\theta^\star$

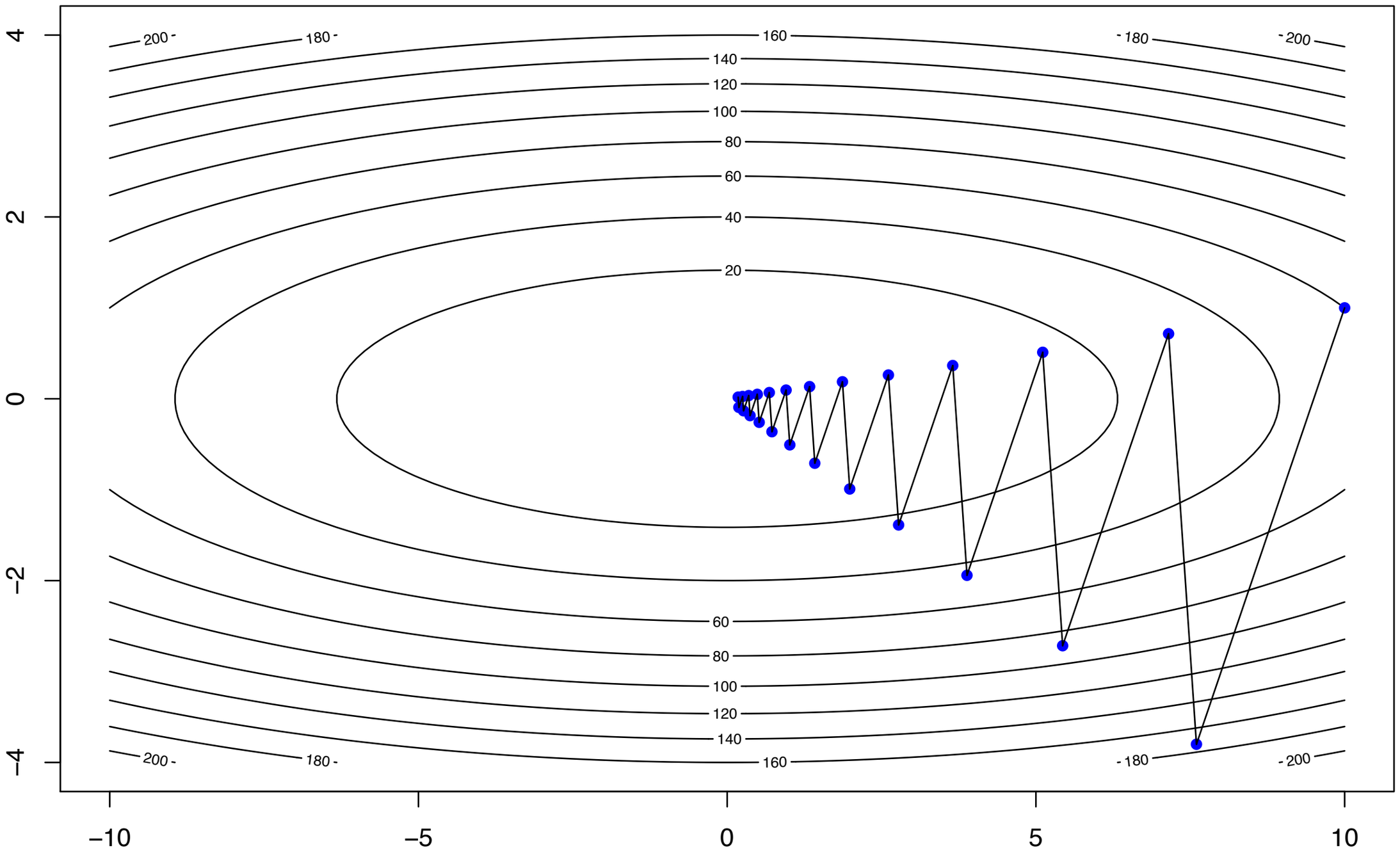
$$\begin{aligned}\nabla \ell(\theta) &= \nabla \ell(\theta^\star) + \nabla^2 \ell(\tilde{\theta})(\theta - \theta^\star) \\ &= \nabla^2 \ell(\tilde{\theta})(\theta - \theta^\star)\end{aligned}$$

for $\tilde{\theta} = \alpha\theta + (1-\alpha)\theta^\star$ for $\alpha \in (0,1)$

$$\theta^\star \approx \theta - \left[\nabla^2 \ell(\theta)\right]^{-1} \nabla \ell(\theta)$$

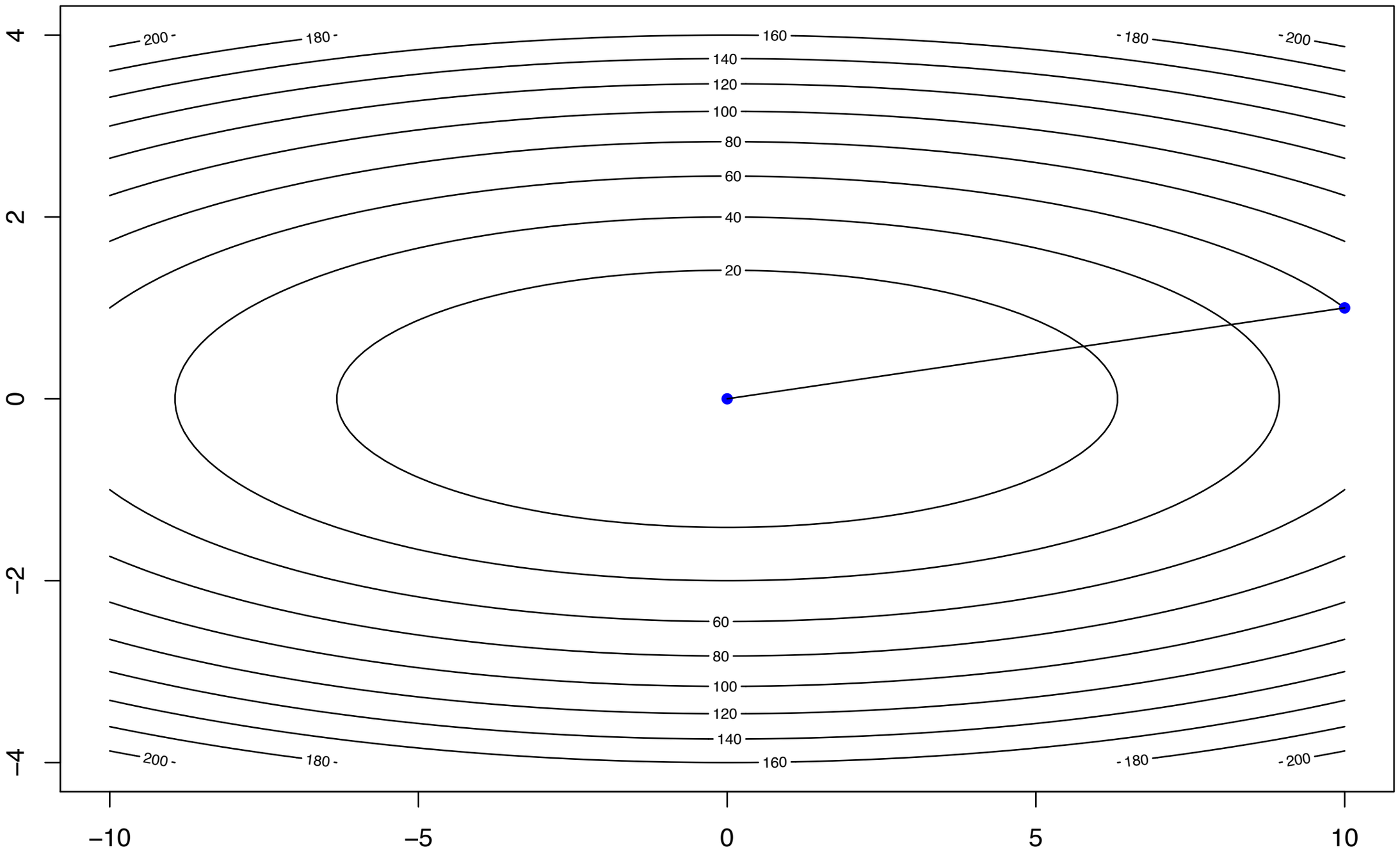
Why use Newton's Method?

$$f(x) = \frac{1}{2}(x_1^2 + 20x_2^2)$$



Why use Newton's Method?

$$f(x) = \frac{1}{2}(x_1^2 + 20x_2^2)$$



Why use Newton's Method?

$$\theta^\star \approx \theta - \left[\nabla^2 \ell(\theta)\right]^{-1} \nabla \ell(\theta)$$

- Accounting for the curvature and rescaling the gradient can reduce the number of iterations.

No free lunch

- Need to compute the Hessian **and** solve a linear system (invert)

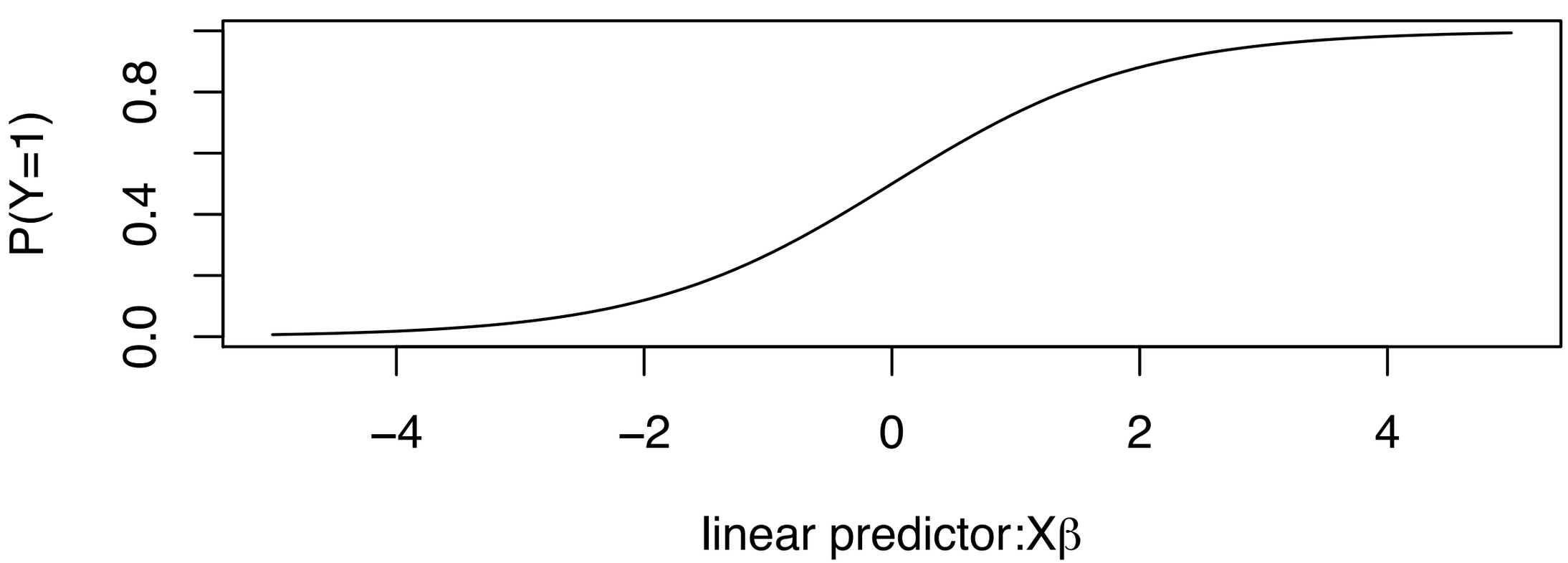
Strategy

- Leverage structure if present
- If no structure, approximate the Hessian with something easier to invert

Logistic Regression

Classification using p-dimensional predictors
$y \in \{0,1\}^n, X \in \mathbb{R}^{n \times p}$, and $n > p$

$$P(y_i = 1 \mid x_i^T \beta) = p_i(\beta) = \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)}$$



Fitting Logistic Regression with Maximum Likelihood Estimation

- Likelihood $\ell(\beta)$ is the probability of observing the data (x_i, y_i) given parameter β (model)

$$\ell(\beta) = \prod_{i=1}^{n} p_i(\beta)^{y_i}(1 - p_i(\beta))^{1-y_i}$$

- Maximum Likelihood Estimation (MLE) seeks the β (model) that makes the data most likely

$$\max_{\beta} \prod_{i=1}^{n} \left(\frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} \right)^{y_i} \left(1 - \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} \right)^{1-y_i}$$

- Typically we minimize the negative log-likelihood

$$f(\beta) = \sum_{i=1}^{n} \left[-y_i x_i^{\mathsf{T}} \beta + \log(1 + \exp(x_i^{\mathsf{T}} \beta)) \right]$$

Newton's Method for Logistic Regression

$y \in \{0,1\}^n, X \in \mathbb{R}^{n \times p}$, and $n > p$

$$f(\beta) = \sum_{i=1}^{n} \left[-y_i x_i^{\mathsf{T}} \beta + \log(1 + \exp(x_i^{\mathsf{T}} \beta)) \right]$$

$$\nabla f(\beta) = X^T (p(\beta) - y)$$

$$\nabla^2 f(\beta) = X^T W(\beta) X,$$

where $W(\beta)$ is diagonal with $w_{ii}(\beta) = p_i(\beta)(1 - p_i(\beta))$
Each Newton step requires

- Computing $X^T W(\beta) X$: $\mathcal{O}(np^2)$ work
- Inverting $X^T W(\beta) X$: $\mathcal{O}(p^3)$
- Total: $\mathcal{O}(np^2)$ when $n > p$

Newton's Method for Ridge Logistic Regression

$y \in \{0,1\}^n, X \in \mathbb{R}^{n \times p}$, and $n < p$

- When $n < p$, $X^T W(\beta) X$ is not invertible
- Remedy: Solve a regularized optimization problem; penalize β that have large magnitude.

$$f(\beta) = \sum_{i=1}^{n} \left[-y_i x_i^{\mathsf{T}} \beta + \log(1 + \exp(x_i^{\mathsf{T}} \beta)) \right] + \frac{\lambda}{2} \|\beta\|_2^2$$

Newton's Method for Ridge Logistic Regression

$$f(\beta) = \sum_{i=1}^{n} \left[-y_i x_i^{\mathsf{T}} \beta + \log(1 + \exp(x_i^{\mathsf{T}} \beta)) \right] + \frac{\lambda}{2} \|\beta\|_2^2$$

$$\nabla f(\beta) = X^T (p(\beta) - y) + \lambda \beta$$

$$\nabla^2 f(\beta) = X^T W(\beta) X + \lambda I$$

where $W(\beta)$ is diagonal with $w_{ii}(\beta) = p_i(\beta)(1 - p_i(\beta))$

Each Newton step requires

- Computing $X^T W(\beta) X + \lambda I$: $\mathcal{O}(np^2)$ work
- Inverting $X^T W(\beta) X + \lambda I$: $\mathcal{O}(p^3)$
- Total: $\mathcal{O}(p^3)$ when $n < p$

Newton's Method for Ridge Logistic Regression

Sherman-Morrison-Woodbury / Matrix Inversion Lemma

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}$$

Inverting $\nabla^2 f(\beta)$

$$(\lambda I + X^T W(\beta) X)^{-1} = \lambda^{-1} I - \lambda^{-2} X^T \left(W(\beta)^{-1} + \lambda^{-1} X X^T\right)^{-1} X$$

- Now we need to invert an $n \times n$ linear system: $\mathcal{O}(n^3)$
- Newton step is now dominated by forming XX^T: $\mathcal{O}(np^2)$

Local Quadratic Convergence

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

- Fixed point $x^\star$ is root of $f(x)$

$$\psi(x^\star) = x^\star - \frac{f(x^\star)}{f'(x^\star)} = x^\star$$

- Assume f is thrice differentiable.
- Assume $f'(x^\star) \neq 0$
- Error at kth step: $e_k = x^{(k)} - x^\star$

Local Quadratic Convergence

$$e_{k+1} = x^{(k+1)} - x^\star = \psi(x^{(k)}) - \psi(x^\star)$$

Taylor's theorem

$$\psi(x^{(k)}) - \psi(x^\star) = \psi'(\tilde{x})(x^{(k)} - x^\star)$$

$$e_{k+1} = \psi'(\tilde{x})e_k$$

where $\tilde{x} = \alpha x^{(k)} + (1-\alpha)x^\star$ for $\alpha \in (0,1)$

Local Quadratic Convergence

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

- Since $f(x^\star) = 0$ and $f'(x^\star) \neq 0$ then $\psi'(x^\star) = 0$
- By continuity of ψ', there is an interval $I_\delta = [x^\star - \delta, x^\star + \delta]$ about $x^\star$ such that if $x \in I_\delta$ then

$$|\psi'(x)| \leq C < 1$$

- Suppose $x^{(0)} \in I_\delta$ then $\tilde{x}^{(0)} \in I_\delta$

$$|e_1| = |\psi'(\tilde{x}^{(0)})||e_0| \leq C\delta < \delta$$

- By induction if $x^{(k-1)} \in I_\delta$, then $x^{(k)} \in I_\delta$ and

$$|e_k| \leq C^k |e_0|$$

- Therefore, $e_k \to 0$

Local Quadratic Convergence

$$\psi(x^{(k)}) - \psi(x^\star) = \frac{1}{2}\psi''(\tilde{x}^{(k)})(x^{(k)} - x^\star)^2$$

$$e_{k+1} = \frac{1}{2}\psi''(\tilde{x}^{(k)})e_k^2$$

$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k} = \frac{1}{2}\psi''(x^\star) = \frac{f''(x^\star)}{2f'(x^\star)}$$

$$e_{k+1} \cong \frac{f''(x^\star)}{2f'(x^\star)} e_k^2$$

local quadratic convergence means accuracy doubles after every iteration

Example 2

$$\sqrt{2} = 1.41421356237$$

Iteration	$x^{(m)}$
0	10
1	5.1
2	2.7460784
3	1.7371949
4	1.444238
5	1.414525
6	1.4142136

Backtracking with Newton's Method

Damped Newton
Initialize t_k at $t_0 > 0$ and $\Delta x_{\text{nt}} = \nabla^2 f(x)^{-1} \nabla f(x)$. Set $t_k = \beta t_k$ until

$$f(x - t_k \Delta x_{\text{nt}}) < f(x) - \alpha t_k \nabla f(x)^{\mathsf{T}} \Delta x_{\text{nt}}$$

- $\beta \in (0, 1)$ and $\alpha = 1/2$
- If f is L-Lipschitz differentiable, $t_k \geq t_{\text{min}} = \min\{\hat{t}, \beta/L\}$

Backtracking with Gradient Descent

Initialize t_k at $t_0 > 0$. Set $t_k = \beta t_k$ until

$$f(x - t_k \nabla f(x)) < f(x) - \alpha t_k \|\nabla f(x)\|_2^2$$

$\beta \in (0, 1)$ and $\alpha = 1/2$.
If f is L-Lipschitz differentiable, $t_k \geq t_{\min} = \min\{\hat{t}, \beta/L\}$.

Backtracking

Basic Idea: Δx_{nt} and $\nabla f(x)$ are descent directions; generically d

$$\nabla f(x)^{\mathsf{T}} d < 0$$

For any d descent direction at x, initialize t_k at $t_0 > 0$. Set $t_k = \beta t_k$ until

$$f(x - t_k d) < f(x) - \alpha t_k \nabla f(x)^{\mathsf{T}} d$$

$\beta \in (0, 1)$ and $\alpha = 1/2$.
If f is L-Lipschitz differentiable, $t_k \geq t_{\text{min}} = \min\{\hat{t}, \beta/L\}$.

Stopping Criterion

Newton decrement

$$\lambda(x) = (\nabla f(x)^{\mathsf{T}} \nabla^2 f(x)^{-1} \nabla f(x))^{1/2}$$

2 ways to interpret

1. Length of Newton step, in quadratic norm defined by the Hessian

$$\lambda(x) = (\Delta x_{\mathsf{nt}}^{\mathsf{T}} \nabla^2 f(x) \Delta x_{\mathsf{nt}})^{1/2}$$

2. Directional derivative of f at x in direciton of Newton steps

$$\nabla f(x)^{\mathsf{T}} \Delta x_{\mathsf{nt}} = -\lambda(x)^2$$

At a fixed point of iteration map (also stationary point of f), $\lambda(x)$ should vanish

Newton's method

Given: Start at some x; set some stop tolerance $\epsilon > 0$
Repeat:

1. Compute Newton step and decrement

$$\Delta x_{\text{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x)$$

$$\lambda^2 := \nabla f(x)^{\mathsf{T}} \nabla^2 f(x)^{-1} \nabla f(x)$$

2. Stop if $\lambda^2/2 \leq \epsilon$
3. Choose step size by t by backtracking line search
4. Update $x := x + t\Delta x_{\text{nt}}$

Computational complexity per iteration?

Quasi-Newton Method

- Once Newton's method is close to a minimizer, it converges rapidly
- Each Newton step can be very expensive however: Compute the Hessian & Invert the Hessian
- Can we use an approximation H for the Hessian?

Three goals

1. H should look like the Hessian
2. H should be easier to compute / store than the Hessian
3. H should be easier to invert than the Hessian

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

$$H_k = H_{k-1} + \frac{yy^\mathsf{T}}{y^\mathsf{T}s} - \frac{H_{k-1}ss^\mathsf{T}H_{k-1}}{s^\mathsf{T}H_{k-1}s}$$

where

$$s = x_k - x_{k-1}$$
$$y = \nabla f(x_k) - \nabla f(x_{k-1})$$

Sherman-Morrison-Woodbury / Matrix Inversion Lemma

$$H_k^{-1} = \left(I - \frac{sy^\mathsf{T}}{y^\mathsf{T}s}\right) H_{k-1}^{-1} \left(I - \frac{sy^\mathsf{T}}{y^\mathsf{T}s}\right) + \frac{ss^\mathsf{T}}{y^\mathsf{T}s}$$

- $y^\mathsf{T}s > 0$ for strictly convex f
- Update / Inverse update is $\mathcal{O}(n^2)$

Properties of BFGS Update

Positive Definiteness

- If $y^{\mathsf{T}}s > 0$, then BFGS update preserves positive definiteness of H_k

Secant Condition

- $H_k s = y$

Convergence

Global Convergence:

- f strongly convex + backtracking line search, BFGS converges for any x_0 and H_0 positive definite

Local Convergence:

- f strongly convex and $\nabla^2 f(x)$ Lipschitz continuous, local convergence is **superlinear**

$$\|x^{(k+1)} - x^\star\|_2 \leq c_k \|x^{(k)} - x^\star\|_2 \to 0$$

where $c_k \to 0$.

- Newton's method is quadratic local convergence
- Gradient descent is linear local convergence

Limited memory quasi-Newton methods

Room for improvement: quasi-Newton method requires storing H_k or H_k^{-1}

limited-memory BFGS (L-BFGS): do not store H_k^{-1} explicitly

- Store m recent values of

$$s_j = x_j - x_{j-1}$$
$$y_j = \nabla f(x_j) - \nabla f(x_{j-1})$$

- Evaluate $\Delta x = H_k^{-1} \nabla f(x_k)$ recursively

$$H_j^{-1} = \left(I - \frac{s_j y_j^{\mathsf{T}}}{y_j^{\mathsf{T}} s_j}\right) H_{j-1}^{-1} \left(I - \frac{s_j y_j^{\mathsf{T}}}{y_j^{\mathsf{T}} s_j}\right) + \frac{s_j s_j^{\mathsf{T}}}{y_j^{\mathsf{T}} s_j}$$

  for $j = k, k-1, \ldots, k-m+1$ with $H_{k-m}^{-1} = I$
- $\mathcal{O}(nm)$ storage and computation per iteration

R package on CRAN

lbfgs: Limited-memory BFGS Optimization

- **Summary:** Don't write your own BFGS!

Motivation for Barzilai-Borwein (BB)

gradient method: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

- choice of α_k: fixed, exact line search, or fixed initial + line search
- **pros:** simple
- **cons:** no use of 2nd order information, "zig-zags" on poorly conditioned problems

Newton's method: $x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k)$

- **pros:** 2nd order information $\implies$ fast convergence near the solution.
- **cons:** forming and inverting $\nabla^2 f(x_k)$ is expensive

A Simple Hessian Approximation

$$\text{minimize } f(x) = \frac{1}{2}x^T Ax - b^T x,$$

where A is positive definite.

- Gradient $\nabla f(x_k) = Ax_k - b$
- Hessian $\nabla^2 f(x_k) = A$
- Newton step: $-A^{-1}\nabla f(x_k)$

Goal: choose α_k so that $-\alpha_k \nabla f(x_k) = -(\alpha_k^{-1} I)^{-1}\nabla f(x_k)$ approximates $-A^{-1}\nabla f(x_k)$.

$$A \approx \frac{1}{\alpha_k} I$$

Q: How can we approximate A without using A explicitly?

Secant Condition

$$\text{Let} \quad s_k := x_k - x_{k-1}, \qquad y_k := \nabla f(x_k) - \nabla f(x_{k-1})$$

Then we have the secant condition.

$$As_k = y_k$$

The Hessian A acts as a derivative operator on the secant s_k.

Seek α_k such that

$$\alpha_k = \arg\min_{\alpha} \|s_k - \alpha y_k\|_2 = \frac{\langle s_k, y_k \rangle}{\|y_k\|_2^2}.$$

Sanity check: For $f = (1/2)x^T Ax + b^T x$ and $0 \leq \mu I \leq A \leq LI$

$$\alpha_k = \frac{s_k^T A s_k}{s_k^T A^2 s_k} \in [L^{-1}, \mu^{-1}]$$

Secant Condition

To justify using the BB step, use Taylor's Theorem

$$\nabla f(x_k) \approx \nabla f(x_{k-1}) + \nabla^2 f(x_{k-1})(x_k - x_{k-1})$$

Let

$$s_k := x_k - x_{k-1}, \qquad y_k := \nabla f(x_k) - \nabla f(x_{k-1})$$

$$\nabla^2 f(x_{k-1}) s_k \approx y_k$$

- Taking the BB step allows for f to increase on some steps.
- Can still guarantee convergence if we ensure sufficient decrease over the last M iterations.

Non-monotone Line Search

Keep track of the worst objective value over the last M iterations.

$$\hat{f}^k = \max\{f^{k-1}, f^{k-2}, \ldots, f^{k-\min\{M,k\}}\}$$

After each gradient step, the following line search condition is checked.

$$f(x_{k+1}) < \hat{f}^k + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{1}{2\alpha_k} \|x_{k+1} - x_k\|_2^2$$

- If condition passes, accept the step x_{k+1}.
- If condition fails, backtrack on α_k, e.g. $\alpha_k \leftarrow 0.5\alpha_k$ and try taking another gradient step.

Many BB Variants

- can use $\alpha_k = s_k^T s_k / s_k^T y_k$ in place of $\alpha_k = s_k^T y_k / y_k^T y_k$
- alternate between these two formulae: **fast adaptive shrinkage/thresholding algorithm (FASTA)**

Many alternatives:

- calculate α_k as above and hold it constant for 2, 3, or 5 successive steps;
- take α_k to be the exact steepest descent step from the **previous** iteration.

Note:

- Nonmonotonicity appears essential to performance.
- Most analyses of BB (including original paper) and related methods are nonstandard,

Summary

Numerous refinements on gradient descent (first order method)

- Newton's Method (2nd order)
- quasi-Newton's Method (Between 1st & 2nd order)
- Barzillai-Borwein
- FISTA (This afternoon)

Further Reading

- Nocedal & Wright's Numerical Optimization
- Lange's Numerical Analysis for Statisticians
- Boyd & Vandenberghe's Convex Optimization
- Stewart's Afternotes on Numerical Analysis