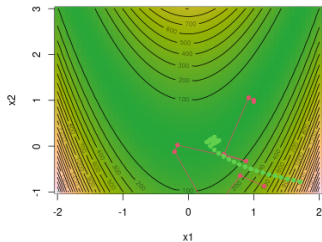


# Optimization in Machine Learning

## Second order methods

### Newton-Raphson



#### Learning goals

- Newton-Raphson
- Limitations

# FROM FIRST TO SECOND ORDER METHODS

- So far: **First order methods**  
⇒ *Gradient* information, i.e., first derivatives
- Now: **Second order methods**  
⇒ *Hessian* information, i.e., second derivatives



# NEWTON-RAPHSON

**Assumption:**  $f \in \mathcal{C}^2$

**Aim:** Find stationary point  $\mathbf{x}^*$ , i.e.,  $\nabla f(\mathbf{x}^*) = \mathbf{0}$

**Idea:** Find root of first order Taylor approximation of  $\nabla f(\mathbf{x})$ :

$$\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}^{[t]}) + \nabla^2 f(\mathbf{x}^{[t]})(\mathbf{x} - \mathbf{x}^{[t]}) = \mathbf{0}$$

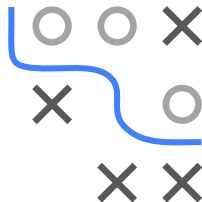
$$\nabla^2 f(\mathbf{x}^{[t]})(\mathbf{x} - \mathbf{x}^{[t]}) = -\nabla f(\mathbf{x}^{[t]})$$

$$\mathbf{x}^{[t+1]} = \mathbf{x}^{[t]} - \left( \nabla^2 f(\mathbf{x}^{[t]}) \right)^{-1} \nabla f(\mathbf{x}^{[t]})$$

**Update scheme:**

$$\mathbf{x}^{[t+1]} = \mathbf{x}^{[t]} + \mathbf{d}^{[t]}$$

with  $\mathbf{d}^{[t]} = - \left( \nabla^2 f(\mathbf{x}^{[t]}) \right)^{-1} \nabla f(\mathbf{x}^{[t]})$



# NEWTON-RAPHSON / 2

**Note:** In practice, we get  $\mathbf{d}^{[t]}$  by solving the linear system

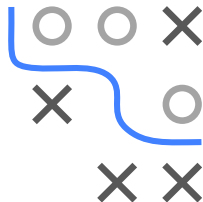
$$\nabla^2 f(\mathbf{x}^{[t]})\mathbf{d}^{[t]} = -\nabla f(\mathbf{x}^{[t]})$$

with direct (matrix decompositions) or iterative methods.

**Relaxed/Damped Newton-Raphson:** Use step size  $\alpha > 0$  with

$$\mathbf{x}^{[t+1]} = \mathbf{x}^{[t]} + \alpha\mathbf{d}^{[t]}$$

to satisfy Wolfe conditions (or just Armijo rule)



# ANALYTICAL EXAMPLE WITH QUADRATIC FORM

$$f(x_1, x_2) = x_1^2 + \frac{x_2^2}{2}$$

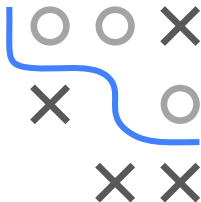
Update direction:  $\mathbf{d}^{[t]} = - \left( \nabla^2 f(x_1^{[t]}, x_2^{[t]}) \right)^{-1} \nabla f(x_1^{[t]}, x_2^{[t]})$

$$\nabla f(x_1, x_2) = \begin{pmatrix} 2x_1 \\ x_2 \end{pmatrix}, \quad \nabla^2 f(x_1, x_2) = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

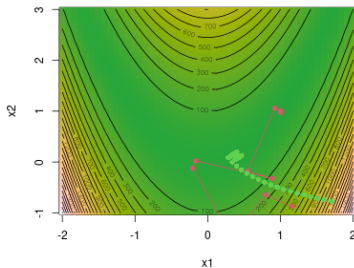
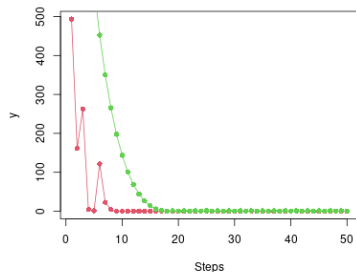
First step:

$$\begin{aligned} \begin{pmatrix} x_1^{[1]} \\ x_2^{[1]} \end{pmatrix} &= \begin{pmatrix} x_1^{[0]} \\ x_2^{[0]} \end{pmatrix} + \mathbf{d}^{[0]} = \begin{pmatrix} x_1^{[0]} \\ x_2^{[0]} \end{pmatrix} - \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2x_1^{[0]} \\ x_2^{[0]} \end{pmatrix} \\ &= \begin{pmatrix} x_1^{[0]} \\ x_2^{[0]} \end{pmatrix} + \begin{pmatrix} -x_1^{[0]} \\ -x_2^{[0]} \end{pmatrix} = \mathbf{0} \end{aligned}$$

**Note:** Newton-Raphson only needs one iteration for quadratic forms



# NEWTON-RAPHSON VS. GD ON BRANIN FUNCTION



Red: Newton-Raphson. Green: Gradient descent.  
Newton-Raphson has much better convergence speed here.

# DISCUSSION

## Advantage:

- For  $f$  sufficiently smooth:

Newton-Raphson converges *locally* quadratically  
(i.e., for starting points close enough to stationary point)



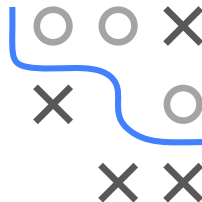
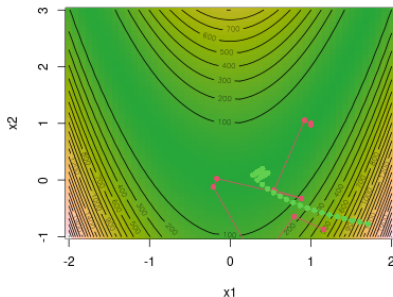
## Disadvantage:

- For “bad” starting points:

Newton-Raphson may diverge

# LIMITATIONS

**Problem 1:** In general,  $\mathbf{d}^{[t]}$  is not a descent direction



**But:** If Hessian is positive definite,  $\mathbf{d}^{[t]}$  is descent direction:

$$\nabla f(\mathbf{x}^{[t]})^\top \mathbf{d}^{[t]} = -\nabla f(\mathbf{x}^{[t]})^\top \left( \nabla^2 f(\mathbf{x}^{[t]}) \right)^{-1} \nabla f(\mathbf{x}^{[t]}) < 0$$

Near minimum, Hessian is positive definite. For initial steps, Hessian is often not positive definite and Newton-Raphson may give non-descending update directions



## LIMITATIONS / 2

**Problem 2:** Hessian can be **computationally expensive** to calculate, since descent direction  $\mathbf{d}^{[t]}$  is the solution of the linear system

$$\nabla^2 f(\mathbf{x}^{[t]})\mathbf{d}^{[t]} = -\nabla f(\mathbf{x}^{[t]}).$$

**Aim:** Find quasi-second order methods not relying on exact Hessians

- Quasi-Newton method
- Gauss-Newton algorithm (for least squares)

