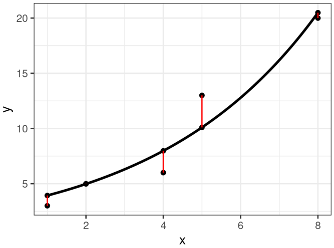
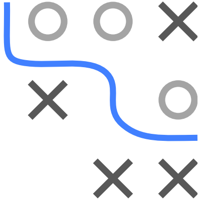
Optimization in Machine Learning

Second order methods
Gauss-Newton





Learning goals

- Least squares
- Gauss-Newton
- Levenberg-Marquardt

LEAST SQUARES PROBLEM

- Consider the problem of minimizing a sum of squares

  $$\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}),$$

  where

  $$g(\boldsymbol{\theta}) = r(\boldsymbol{\theta})^T r(\boldsymbol{\theta}) = \sum_{i=1}^{n} r_i(\boldsymbol{\theta})^2$$

  and

  $$\begin{aligned} r : \mathbb{R}^d &\to \mathbb{R}^n \\ \boldsymbol{\theta} &\mapsto (r_1(\boldsymbol{\theta}), \ldots, r_n(\boldsymbol{\theta}))^T \end{aligned}$$

  maps parameters $\boldsymbol{\theta}$ to residuals $r(\boldsymbol{\theta})$

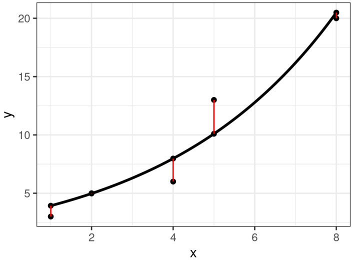
LEAST SQUARES PROBLEM

- **Risk minimization with squared loss** $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
- **Least squares regression:**

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right) = \sum_{i=1}^{n} \underbrace{\left(y^{(i)} - f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)^2}_{r_i(\boldsymbol{\theta})^2}$$

- $f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)$ might be a function that is **nonlinear in $\boldsymbol{\theta}$**
- Residuals: $r_i = y^{(i)} - f(\mathbf{x}^{(i)} \mid \boldsymbol{\theta})$
- **Example:**

$$\mathcal{D} = \left(\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right)_{i=1,\ldots,5}$$
$$= ((1,3),(2,5),(4,6),(5,13),(8,20))$$



LEAST SQUARES PROBLEM

- Suppose, we suspect an *exponential* relationship between $x \in \mathbb{R}$ and y

$$f(x \mid \boldsymbol{\theta}) = \theta_1 \cdot \exp(\theta_2 \cdot x), \quad \theta_1, \theta_2 \in \mathbb{R}$$

- **Residuals:**

$$r(\boldsymbol{\theta}) = \begin{pmatrix} \theta_1 \exp(\theta_2 x^{(1)}) - y^{(1)} \\ \theta_1 \exp(\theta_2 x^{(2)}) - y^{(2)} \\ \theta_1 \exp(\theta_2 x^{(3)}) - y^{(3)} \\ \theta_1 \exp(\theta_2 x^{(4)}) - y^{(4)} \\ \theta_1 \exp(\theta_2 x^{(5)}) - y^{(5)} \end{pmatrix} = \begin{pmatrix} \theta_1 \exp(1\theta_2) - 3 \\ \theta_1 \exp(2\theta_2) - 5 \\ \theta_1 \exp(4\theta_2) - 6 \\ \theta_1 \exp(5\theta_2) - 13 \\ \theta_1 \exp(8\theta_2) - 20 \end{pmatrix}$$

- **Least squares problem:**

$$\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{i=1}^{5} \left(y^{(i)} - \theta_1 \exp\left(\theta_2 x^{(i)} \right) \right)^2$$

NEWTON-RAPHSON IDEA

- **Approach:** Calculate Newton-Raphson update direction by solving:

$$\nabla^2 g(\boldsymbol{\theta}^{[t]})\mathbf{d}^{[t]} = -\nabla g(\boldsymbol{\theta}^{[t]}).$$

- Gradient is calculated via chain rule

$$\nabla g(\boldsymbol{\theta}) = \nabla(r(\boldsymbol{\theta})^T r(\boldsymbol{\theta})) = 2 \cdot J_r(\boldsymbol{\theta})^T r(\boldsymbol{\theta}),$$

where $J_r(\boldsymbol{\theta})$ is Jacobian of $r(\boldsymbol{\theta})$.

- In our example:

$$J_r(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial r_1(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial r_1(\boldsymbol{\theta})}{\partial \theta_2} \\ \frac{\partial r_2(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial r_2(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots & \vdots \\ \frac{\partial r_5(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial r_5(\boldsymbol{\theta})}{\partial \theta_2} \end{pmatrix} = \begin{pmatrix} \exp(\theta_2 x^{(1)}) & x^{(1)}\theta_1 \exp(\theta_2 x^{(1)}) \\ \exp(\theta_2 x^{(2)}) & x^{(2)}\theta_1 \exp(\theta_2 x^{(2)}) \\ \exp(\theta_2 x^{(3)}) & x^{(3)}\theta_1 \exp(\theta_2 x^{(3)}) \\ \exp(\theta_2 x^{(4)}) & x^{(4)}\theta_1 \exp(\theta_2 x^{(4)}) \\ \exp(\theta_2 x^{(5)}) & x^{(5)}\theta_1 \exp(\theta_2 x^{(5)}) \end{pmatrix}$$

NEWTON-RAPHSON IDEA

- Hessian of g, $\mathbf{H}_g = (H_{jk})_{jk}$, is obtained via product rule:

$$H_{jk} = 2 \sum_{i=1}^{n} \left(\frac{\partial r_i}{\partial \theta_j} \frac{\partial r_i}{\partial \theta_k} + r_i \frac{\partial^2 r_i}{\partial \theta_j \partial \theta_k} \right)$$

- **But:**

  Main problem with Newton-Raphson: Second derivatives can be computationally expensive.

GAUSS-NEWTON FOR LEAST SQUARES

- Gauss-Newton approximates $\mathbf{H}_g$ by dropping its second order part:

$$
\begin{aligned}
H_{jk} &= 2\sum_{i=1}^{n}\left(\frac{\partial r_i}{\partial \theta_j}\frac{\partial r_i}{\partial \theta_k} + r_i \frac{\partial^2 r_i}{\partial \theta_j \partial \theta_k}\right) \\
&\approx 2\sum_{i=1}^{n}\frac{\partial r_i}{\partial \theta_j}\frac{\partial r_i}{\partial \theta_k} \\
&= 2 J_r(\boldsymbol{\theta})^T J_r(\boldsymbol{\theta})
\end{aligned}
$$

- **Note**: We assume that

$$
\left|\frac{\partial r_i}{\partial \theta_j}\frac{\partial r_i}{\partial \theta_k}\right| \gg \left| r_i \frac{\partial^2 r_i}{\partial \theta_j \partial \theta_k}\right|.
$$

- This assumption may be valid if: Residuals r_i are small in magnitude **or** functions are only “mildly” nonlinear s.t. $\frac{\partial^2 r_i}{\partial \theta_j \partial \theta_k}$ is small

GAUSS-NEWTON FOR LEAST SQUARES

- If $J_r(\boldsymbol{\theta})^T J_r(\boldsymbol{\theta})$ is invertible, Gauss-Newton update direction is

$$\begin{aligned}
\mathbf{d}^{[t]} &= -\left[\nabla^2 g(\boldsymbol{\theta}^{[t]})\right]^{-1} \nabla g(\boldsymbol{\theta}^{[t]}) \\
&\approx -\left[J_r(\boldsymbol{\theta}^{[t]})^T J_r(\boldsymbol{\theta}^{[t]})\right]^{-1} J_r(\boldsymbol{\theta}^{[t]})^T r(\boldsymbol{\theta}) \\
&= -(J_r^T J_r)^{-1} J_r^T r(\boldsymbol{\theta})
\end{aligned}$$

- **Advantage**:

  Reduced computational complexity since no Hessian necessary.

- **Note:** Gauss-Newton can also be derived by starting with

$$r(\boldsymbol{\theta}) \approx r(\boldsymbol{\theta}^{[t]}) + J_r(\boldsymbol{\theta}^{[t]})^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}) = \tilde{r}(\boldsymbol{\theta})$$

  and $\tilde{g}(\boldsymbol{\theta}) = \tilde{r}(\boldsymbol{\theta})^T \tilde{r}(\boldsymbol{\theta})$. Then, set $\nabla \tilde{g}(\boldsymbol{\theta})$ to zero

LEVENBERG-MARQUARDT ALGORITHM

- **Problem:** Gauss-Newton may not decrease g in every iteration but may diverge, especially if starting point is far from minimum
- **Solution:** Choose step size $\alpha > 0$ s.t.

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

  decreases g (e.g., by satisfying Wolfe conditions)
- However, if α gets too small, an **alternative** method is the

Levenberg-Marquardt algorithm

$$(J_r^T J_r + \lambda D)\mathbf{d}^{[t]} = -J_r^T r(\boldsymbol{\theta})$$

- D is a positive diagonal matrix
- $\lambda = \lambda^{[t]} > 0$ is the *Marquardt parameter* and chosen at each step

LEVENBERG-MARQUARDT ALGORITHM

- **Interpretation:** Levenberg-Marquardt *rotates* Gauss-Newton update directions towards direction of *steepest descent*
- Let $D = I$ for simplicity. Then:

$$\begin{aligned}\lambda \mathbf{d}^{[t]} &= \lambda(J_r^T J_r + \lambda I)^{-1}(-J_r^T r(\boldsymbol{\theta})) \\ &= (I - J_r^T J_r/\lambda + (J_r^T J_r)^2/\lambda^2 \mp \cdots)(-J_r^T r(\boldsymbol{\theta})) \\ &\to -J_r^T r(\boldsymbol{\theta}) = -\nabla g(\boldsymbol{\theta})/2\end{aligned}$$

  for $\lambda \to \infty$
- **Note:** $(\mathbf{A} + \mathbf{B})^{-1} = \sum_{k=0}^{\infty}(-\mathbf{A}^{-1}\mathbf{B})^k \mathbf{A}^{-1}$ if $\|\mathbf{A}^{-1}\mathbf{B}\| < 1$
- Therefore: $\mathbf{d}^{[t]}$ approaches direction of negative gradient of g
- Often: $D = \text{diag}(J_r^T J_r)$ to get scale invariance (**Recall:** $J_r^T J_r$ is positive semi-definite $\Rightarrow$ non-negative diagonal)