Machine Learning

Gradient Descent

Sebastian Engelke

Master in Business Analytics

UNIVERSITÉ
DE GENÈVE

Numerical optimization: gradient descent

$$\widehat{\beta}_0 = \text{argmin}_{\beta_0 \in \mathbb{R}} \text{RSS}(\beta_0) = \text{argmin}_{\beta_0 \in \mathbb{R}} J(\beta_0)$$

For simplicity: fix β_1 to the true value β_1^* and only estimate β_0 by least squares.

Gradient descent

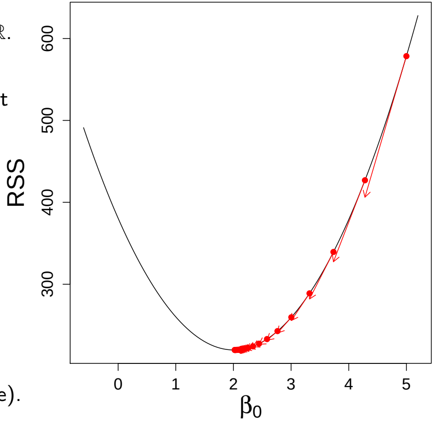
- Start with an arbitrary initial value $\beta_0^{(0)} \in \mathbb{R}$.
- In the *i*th step:
- Compute the gradient/derivative of J at the position $\beta_0^{(i-1)}$:

$$\nabla J(\beta_0^{(i-1)}) = \frac{\partial J(\beta_0)}{\partial \beta_0}\bigg|_{\beta_0 = \beta_0^{(i-1)}}$$

- Update your estimate to

$$\beta_0^{(i)} = \beta_0^{(i-1)} - \alpha \nabla J(\beta_0^{(i-1)}),$$

  where α is a tuning parameter.
- Stop when converged ($\beta_0^{(i)}$ changes no more).



Gradient descent: higher dimensions

$$(\widehat{\beta}_0, \widehat{\beta}_1) = \text{argmin}_{(\beta_0,\beta_1)\in\mathbb{R}^2} \text{RSS}(\beta_0, \beta_1) = \text{argmin}_{\theta\in\mathbb{R}^d} J(\theta)$$

Gradient descent

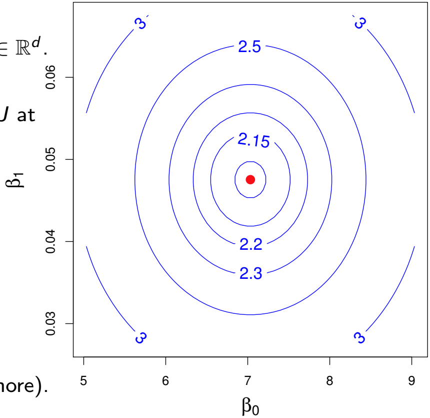
- Start with an arbitrary initial value $\theta^{(0)} \in \mathbb{R}^d$.
- In the ith step:
- Compute the gradient/derivative of J at the position $\theta^{(i-1)}$:

$$\nabla J(\theta^{(i-1)}) = \frac{\partial J(\theta)}{\partial \theta}\bigg|_{\theta=\theta^{(i-1)}} \in \mathbb{R}^d$$

- Update your estimate to

$$\theta^{(i)} = \theta^{(i-1)} - \alpha \nabla J(\theta^{(i-1)}) \in \mathbb{R}^d,$$

  where α is a tuning parameter.
- Stop when converged ($\theta^{(i)}$ changes no more).



Gradient descent: remarks

- Gradient descent can be used to solve numerically a minimization problem
$$\operatorname{argmin}_{\theta \in \mathbb{R}^d} J(\theta).$$
- The ingredients are the `objective function` J and its derivatives in all directions, namely the `gradient` ∇J.
- Recall that the gradient $\nabla J(\theta)$ points in the direction of the `steepest increase` of J at the point θ.
- Important restriction: J needs to be differentiable!
- Gradient descent is a generic method that typically finds a `local minimum`.
- Only if the problem is convex, convergence to the `global minimum` can be guaranteed.
- The tuning parameter, also called `learning rate`, has to be adapted to the problem (many techniques exist).
- In complex models, gradient descent is the only way to find a (local) minimum.
- Depending on the model, different implementations/approximations exist, e.g., `stochastic gradient descent`, `backpropagation`.