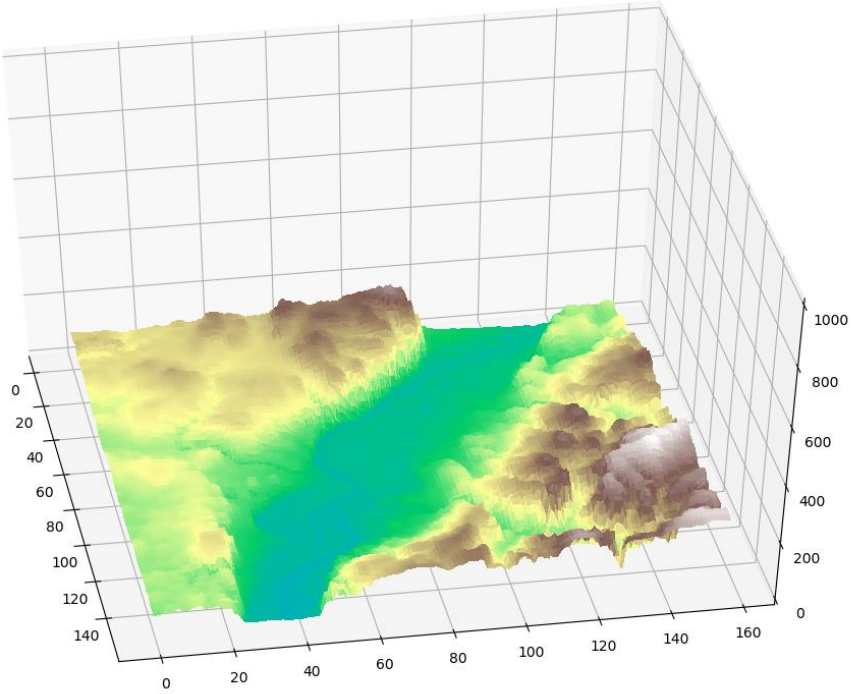
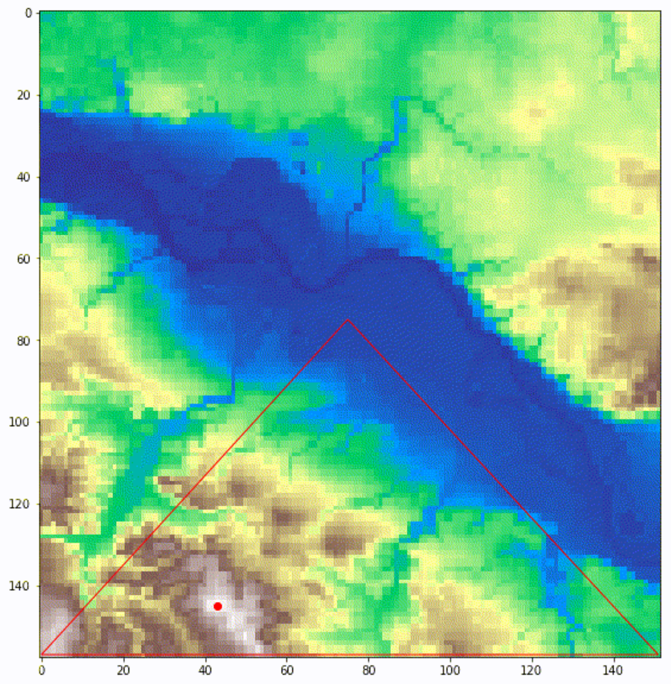
Recap: Optimization Techniques





Regression

Programming for Data Science

Warm Up

Dresden Database Systems Group

Answer the following statements! Give reason for your answers.

1. Why is matrix multiplication a good way for solving regression?
2. Why is matrix multiplication fast in R/python?
3. Why does Lasso/Elastic Net regression prefer some parameters to be 0?
4. How do you implement an optimization problem?
5. How do you model an constant factor (i.e. n) in matrix multiplication?

https://amcs.website

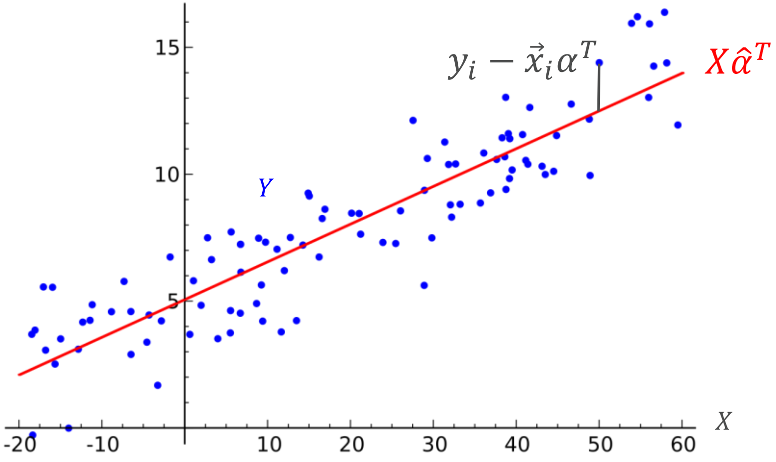
Ordinary Least Squares (OLS)

Dresden Database Systems Group

Estimator for linear regression based on minimizing the sum of squared errors (SSE) for a set of observations $X = \{\vec{x}_1, \dots, \vec{x}_n\}$ with $\vec{x}_i = (x_{i1}, \dots, x_{id})$ and values $Y = \{y_1, \dots, y_n\}$

- X: independent variable, Y: dependent variable, α: regression coefficients
- Linear model: $y_i = \alpha_1 \mathrm{x}_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_d x_{id}$ or $y_i = \vec{x}_i \alpha^T$ with $\alpha = (\alpha_1, \dots, \alpha_d)$

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \vec{x}_i \alpha^T)^2$$



Ordinary Least Squares (OLS)

Dresden Database Systems Group

Given $X = \{\vec{x}_1, \dots, \vec{x}_n\}$ with $\vec{x}_i = (x_{i1}, \dots, x_{id})$ and $Y = \{y_1, \dots, y_n\}$

- X: independent variable, Y: dependent variable, α: regression coefficients, $\hat{\alpha}$: coefficient estimator
- SSE: Sum of squared errors
- Minimization problem has unique solution (if all d features are linear independent):

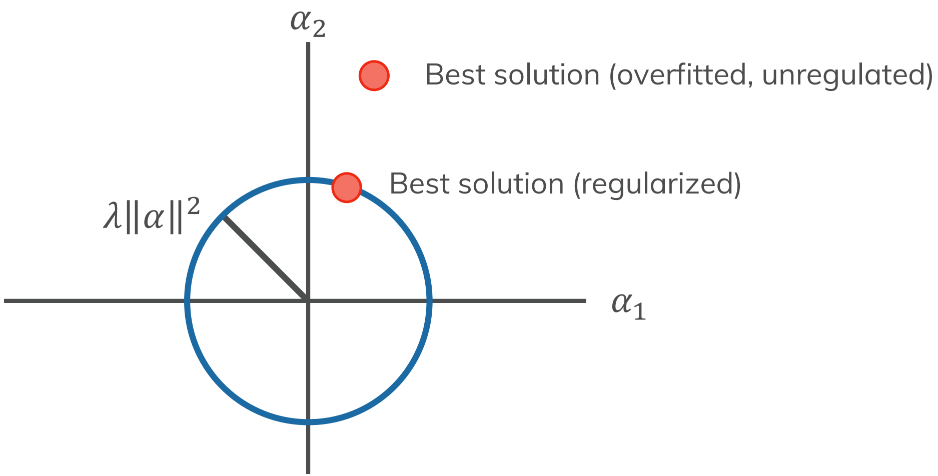
$$\begin{aligned}\hat{\alpha} &= \underset{\alpha}{\operatorname{argmin}}\, SSE \\ &= \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \vec{x}_i \alpha^T)^2 \\ &= (X^T X)^{-1} X^T Y^T\end{aligned}$$

Ridge Regression

Introduces regularization for regression models

- OLS is prone to overfitting and underfitting
- Idea: Prefer a certain solution for α with limited variation
- $\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \vec{x}_i \alpha^T)^2 + \lambda \sum_{j=1}^{d} \alpha_j^2$

Example for $j \in \{1,2\}$

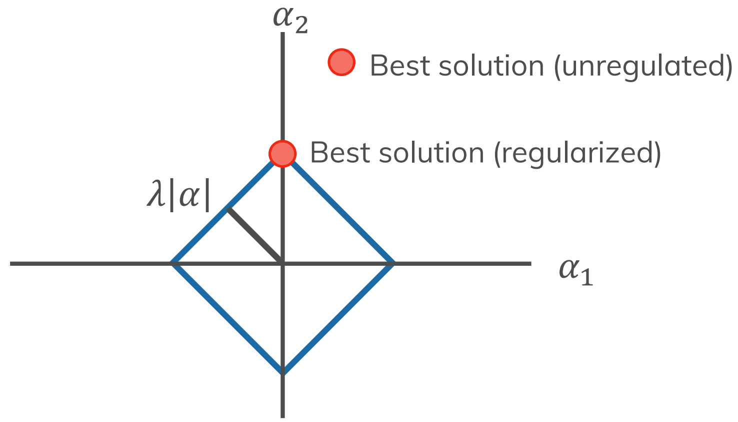


More Regularization

Dresden Database Systems Group

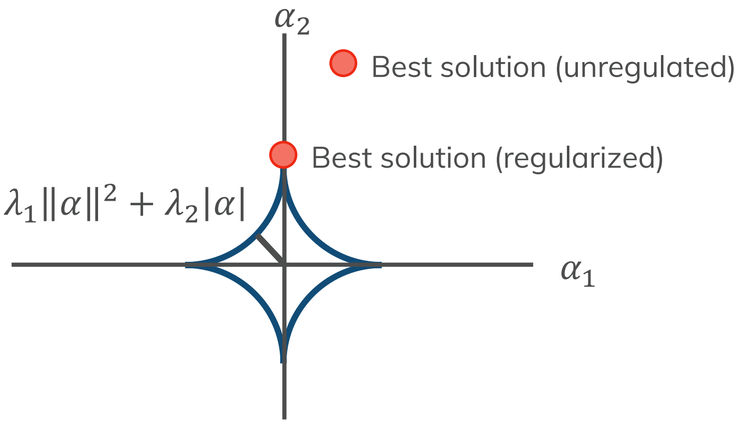
Lasso Regression

- $\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n}(y_i - \vec{x}_i\alpha^T)^2 + \lambda \sum_{j=1}^{d}|\alpha_j|$
- advantage: some α_i can be 0 (feature reduction)



Elastic Net

- $\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n}(y_i - \vec{x}_i\alpha^T)^2 + \lambda_1 \sum_{j=1}^{d} \alpha_j^2 + \lambda_2 \sum_{j=1}^{d}|\alpha_j|$
- advantage: even more α_i can be 0



Task

Step 0

- You will get a csv file from us. Load it in your language/environment.
- Explore the data in it.

Step 1

- Implement a function* for OLS using optim/minimize.
- Find $\hat{\alpha} = (\mathrm{m}, \mathrm{n}), \mathrm{y_i} = \mathrm{mx_i} + \mathrm{n}$

Step 2

- Implement a function* for OLS using the matrix solution.
- Find $\hat{\alpha}$ and compare its run time to your other OLS function

Step 3

- Implement Ridge, Lasso, and Elastic Net regression* using optim/minimize.
- Compare their resulting $\hat{\alpha}$ to each other and to OLS. Use different values for λ, λ_1, and λ_2.

*use your own implementation

Package suggestions

R

- microbenchmark

python3

- numpy
- scipy
- timeit
- (matplotlib.pyplot)

Exercise Appointment

Dresden Database Systems Group

We compare and discuss the results

- Tuesday, 24.11.2019,
- Consultation: Please use the forum in Opal.
- Please prepare your solutions! Send us your code!

If you have questions, please mail us:

claudio.hartmann@tu-dresden.de Orga + Code + R
lucas.woltmann@tu-dresden.de Tasks + Python