OLS AND ITS REGULARIZATIONS

Ridge, Lasso, and Elastic Net

WHEN OLS FAILS

In this context *fails* means that OLS is unable to provide a unique solution, such as a unique set of coefficients for the input variables.

Plain *vanilla* multiple linear regression (OLS) fails if the number of observations is smaller than the number of features.

Example: If the dependent variable is the Sales Price, we cannot uniquely determine the weights for the features if we have only 4 observations.

Dist. to School	Prop. Area	Housing Area	Value	Prop. Tax	Bathrooms	Sales Price
7	0.4	1800	234	9.8	2	267.5
2.3	0.8	1980	244	10.5	2.5	278.2
4.3	1.1	2120	252	16.2	3	284.5
3.8	0.6	2500	280	18.4	3.5	310.4

So, what's wrong with this dataset?

WHEN OLS FAILS - NOT ENOUGH DATA

Suppose we want to predict the "Sales Price" by using a linear combination of the feature variables, such as "Distance to School", "Property Area", etc.

$$M \cdot \vec{\beta} = \beta_1 \cdot col_1(M) + \beta_2 \cdot col_2(M) + ... + \beta_p \cdot col_p(M)$$

The Ordinary Least Squares (OLS) method aims at finding the coefficients β such that the the sum or squared errors is minimal, i.e.

$$\underset{\beta}{\operatorname{argmin}} |\text{Sales Price} - M \cdot \vec{\beta}|^2$$

Important Question: Why would OLS fail in this case? Hint: the problem to solve is ill-posed in the sense that it allows too many perfect solutions.

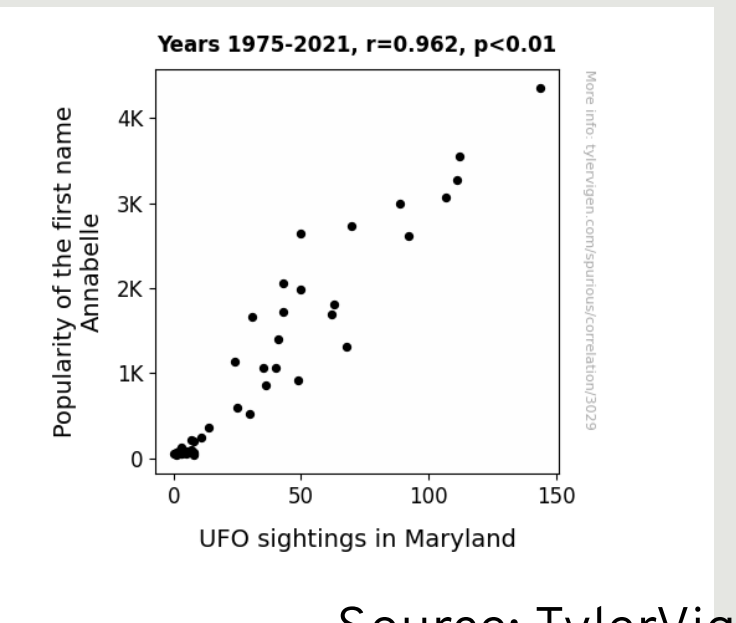
WHAT HAPPENS IF WE HAVE TOO MANY FEATURES?

https://www.tylervigen.com/spurious/correlation/3029_popularity-of-the-first-name-annabelle_correlates-with_ufo-sightings-in-maryland

WHAT HAPPENS IF WE HAVE TOO MANY FEATURES?

GenAI made-up explanation:

"As the number of Annabelles grew, so did the collective power of their positive energy, inadvertently attracting curious extraterrestrial beings to the skies above Maryland. It seems that the universe just couldn't resist the charm and magnetism that this particular name exuded, leading to a surge in close encounters of the Annabelle kind!"



Source: TylerVigen.com

OLS FAILS - NOT ENOUGH DATA

The assumption for multiple linear regression is

$$Y = X\beta + \sigma\epsilon$$

where σ is the standard deviation of the noise, further, we assume that the "noise" ϵ is independent and identically distributed with a zero mean.

We believe that the output is a linear combination of the input features.

Thus, if we would like to solve for the "weights" β we may consider

$$X^tY = X^tX\beta + \sigma X^t\epsilon$$

And if the matrix X^tX is invertible then we can solve for expected value of β:

$$\mathbb{E}(\beta) = (X^tX)^{-1}X^tY$$

We can show by using *Linear Algebra* that the OLS solution obtained form minimizing the sum of the square residuals is equivalent.

IMPORTANT: When the matrix X^tX is not invertible we cannot apply this method to get $\mathbb{E}(\beta)$. In this case if we minimize the sum of the squared errors the algorithm cannot find just *one* best solution.

WHEN OLS FAILS - WHAT'S THE SOLUTION?

Theory of the problem: if $X^T X$ is not invertible, consider

$$X^T X + \lambda I$$

for a small $\lambda > 0$.

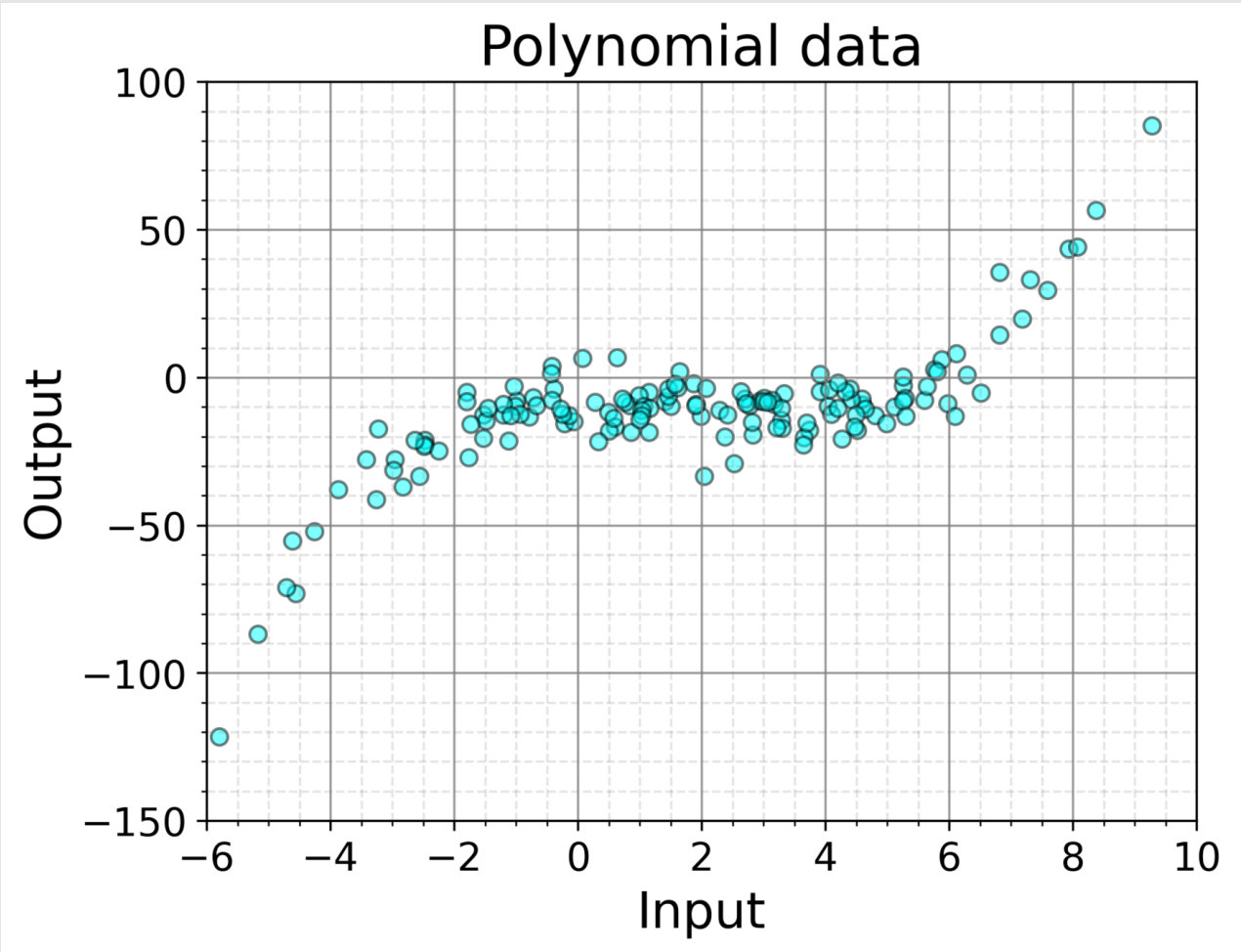
Theory of the solution: you want to solve your regression problem, so you need to find the "best" set of weights. Therefore, you should minimize the sum of squared errors by imposing a constraint on the weights.

The two "theories" meet together when the constraint is placed on the sum of squared weights. This *trick* is called Ridge regression.

$$\sum w_j^2 < c$$

WHY CARE ABOUT MANY FEATURES?

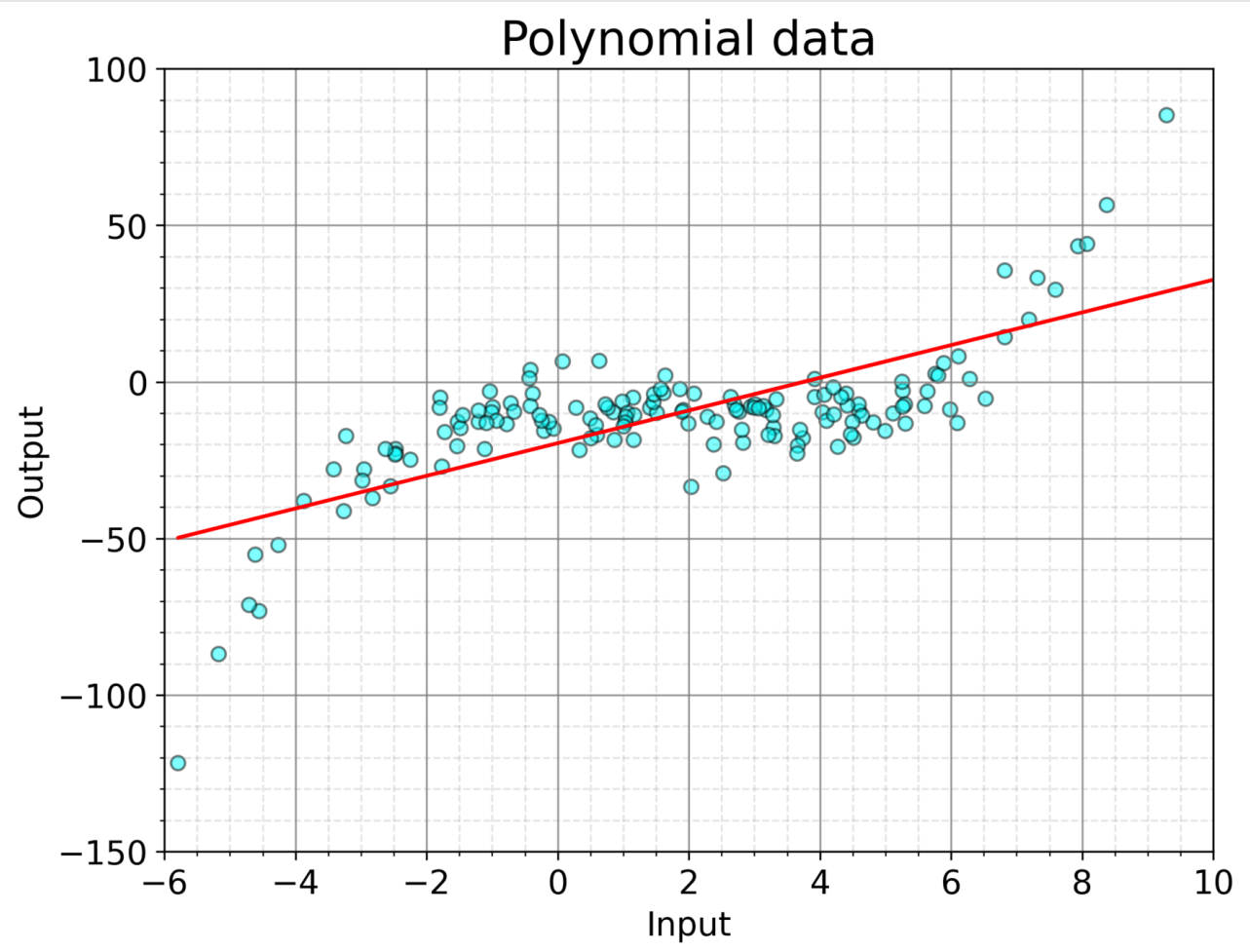
FEATURE ENGINEERING



What kind of features do we need for describing $\mathbb{E}[Y|X=x]$?

WHY CARE ABOUT MANY FEATURES?

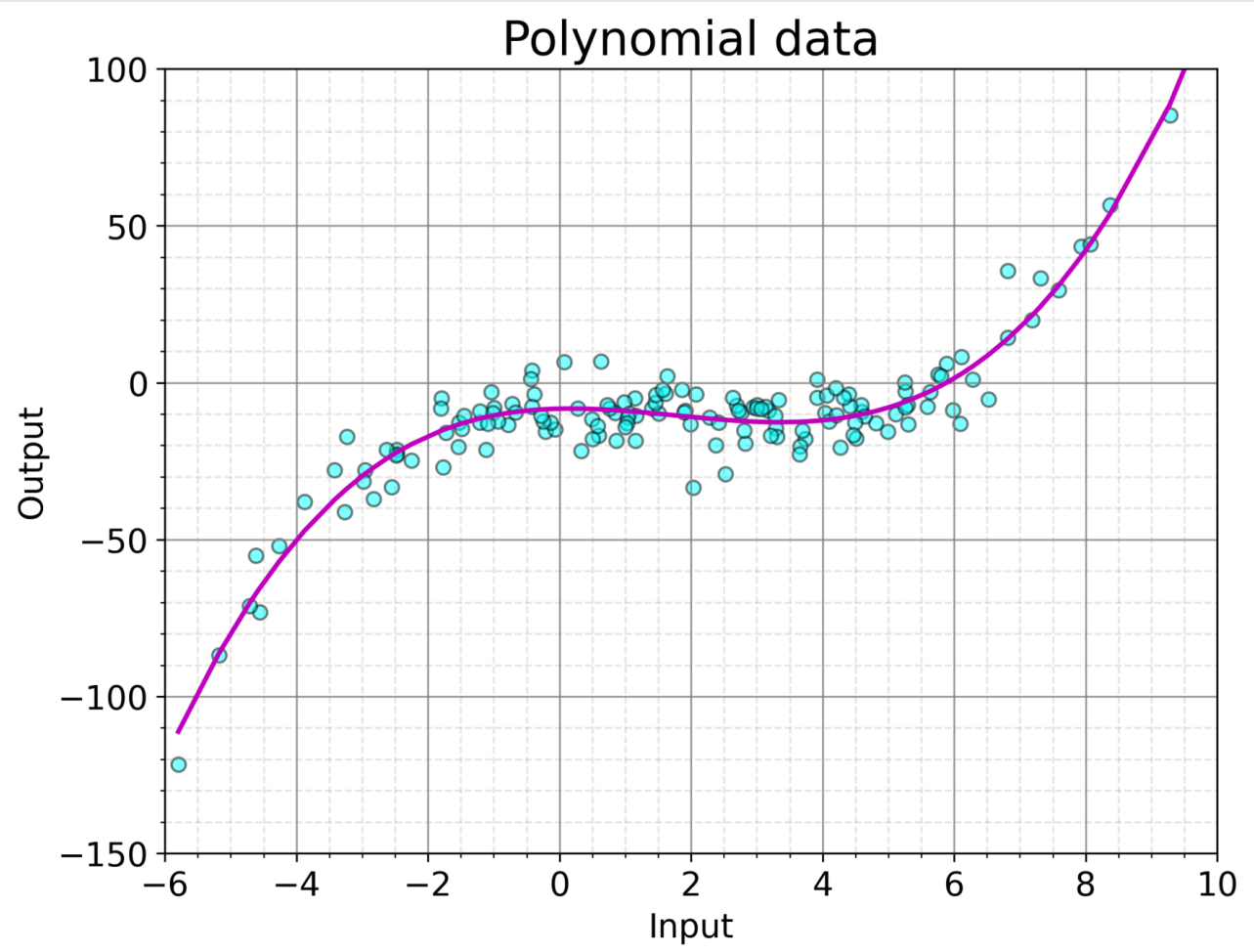
FEATURE ENGINEERING



Is this a good choice for describing $\mathbb{E}[Y|X=x]$?

WHY CARE ABOUT MANY FEATURES?

FEATURE ENGINEERING



Is this a better choice for describing $\mathbb{E}[Y|X = x]$?

WHY CARE ABOUT MANY FEATURES?

FEATURE ENGINEERING

Main idea: Linear combination of different powers of the feature values.

$$P(x) := \beta_p x^p + \beta_{p-1} x^{p-1} + ... + \beta_1 x + \beta_0$$

What we hope to achieve:

$$\mathbb{E}(Y|X = x) \approx \beta_p x^p + \beta_{p-1} x^{p-1} + ... + \beta_1 x + \beta_0$$

IMPORTANT: $P(x)$ is nonlinear in x. However if x is fixed (x is your data) and β is the input we have

$$L(\beta) := \beta_p x^p + \beta_{p-1} x^{p-1} + ... + \beta_1 x + \beta_0$$

is linear in $\beta = (\beta_0, \beta_1, \beta_2, ...\beta_p)$.

$$L(\beta + \gamma) = L(\beta) + L(\gamma)$$

and

$$L(c \cdot \beta) = c \cdot L(\beta)$$

for any two vectors β and γ, and any scalar (real number) c.

L2 (RIDGE) REGULARIZATION

Main Idea: minimize the sum of the square residuals plus a constraint on the vector of weights

The L2 norm is

$$\|\beta\|_2 := \left(\sum_{j=1}^{p} \beta_j^2 \right)^{1/2}$$

The Ridge model (also known as the *Tikhonov regularization*) consists of *learning* the weights by the following optimization:

$$\text{minimize} \frac{1}{n} \sum_{i=1}^{n} (\text{Error}_i)^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

where α is a constant that can be adjusted based on a feedback loop so it is a hyperparameter ("tunning" parameter).

This optimization is equivalent to minimizing the sum of the square errors with a constraint on the sum of the squared weights.

L1 (LASSO) REGULARIZATION

Main Idea: minimize the sum of the square residuals plus a constraint on the vector of weights

The L1 norm is

$$\|\beta\|_1 := \sum_{j=1}^{p} |\beta_j|$$

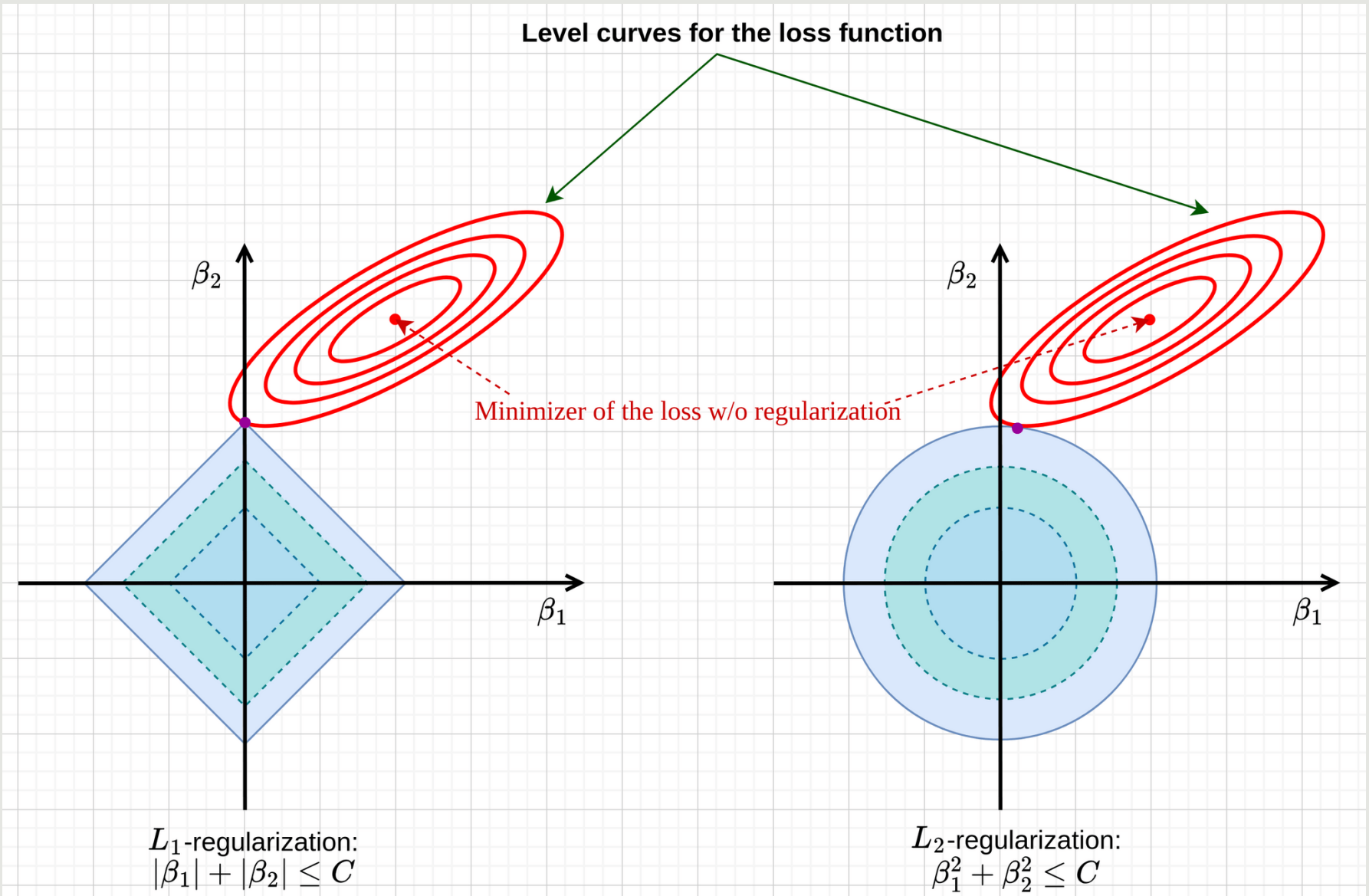
The lasso model (R. Tibshirani) consists of *learning* the weights by the following optimization:

$$\text{minimize} \frac{1}{n} \sum_{i=1}^{n} (\text{Error}_i)^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

where α is a constant that can be adjusted based on a feedback loop so it is a hyperparameter ("tunning" parameter).

This optimization is equivalent to minimizing the sum of the square errors with a constraint on the sum of the absolute value of the weights.

L1 (LASSO) VS L2 (RIDGE) - WHAT'S THE DIFFERENCE?



COMBINING L1 AND L2 - ELASTIC NET

Tha main idea is to combine the L2 and L1 regularizations in a *weighted* way, such as:

$$\lambda \cdot \sum_{j=1}^{p} |\beta_j| + 0.5 \cdot (1 - \lambda) \cdot \sum_{j=1}^{p} \beta_j^2$$

Here $0 \leq \lambda \leq 1$ is called the L1_ratio.

The Elstic Net regularization consists of *learning* the weights by solving the following optimization problem:

$$\text{minimize} \frac{1}{n} \sum_{i=1}^{n} (\text{Error}_i)^2 + \alpha \left(\lambda \cdot \sum_{j=1}^{p} |\beta_j| + 0.5 \cdot (1 - \lambda) \cdot \sum_{j=1}^{p} \beta_j^2 \right)$$

So with this rgularization approach we have two hyperparameters that we need to decide on.

APPLYING ELASTIC NET

The objective function is

$$f(\beta) := \frac{1}{n}\sum_{i=1}^{n}(\text{Error}_i)^2 + \alpha\left(\lambda\cdot\sum_{j=1}^{p}|\beta_j| + 0.5\cdot(1-\lambda)\cdot\sum_{j=1}^{p}\beta_j^2\right)$$

This objective function is U-shaped (convex), and its gradient will help find the optimal weights.

This is matrix-vector product.

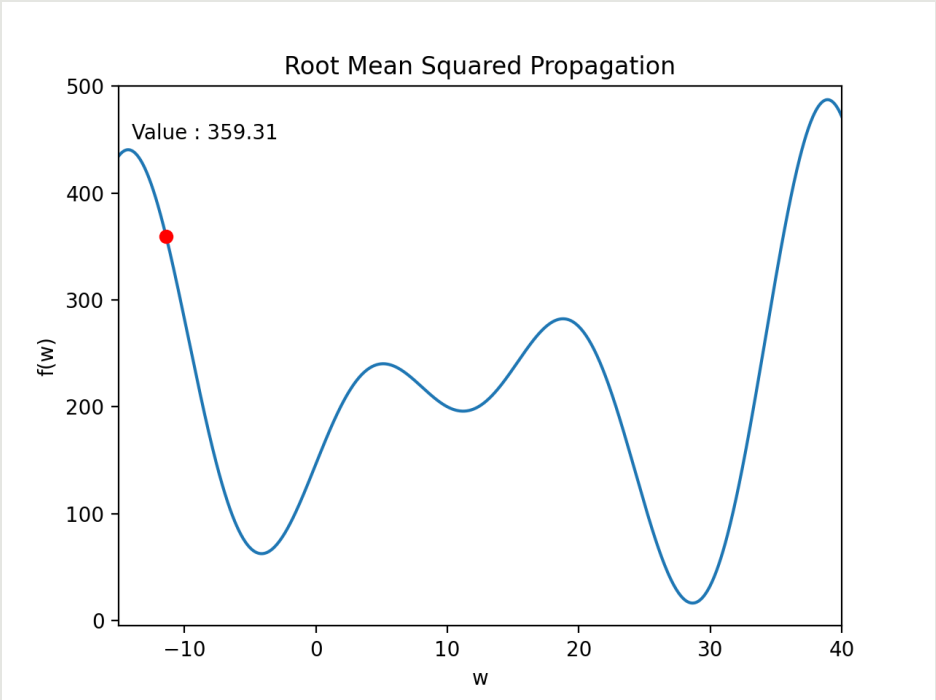
$$\nabla f(\beta) := \frac{2}{n}(-X^T)\cdot(\text{Errors}) + \alpha\left(\lambda\cdot\text{sign}(\beta_j) + (1-\lambda)\cdot\beta_j\right)$$

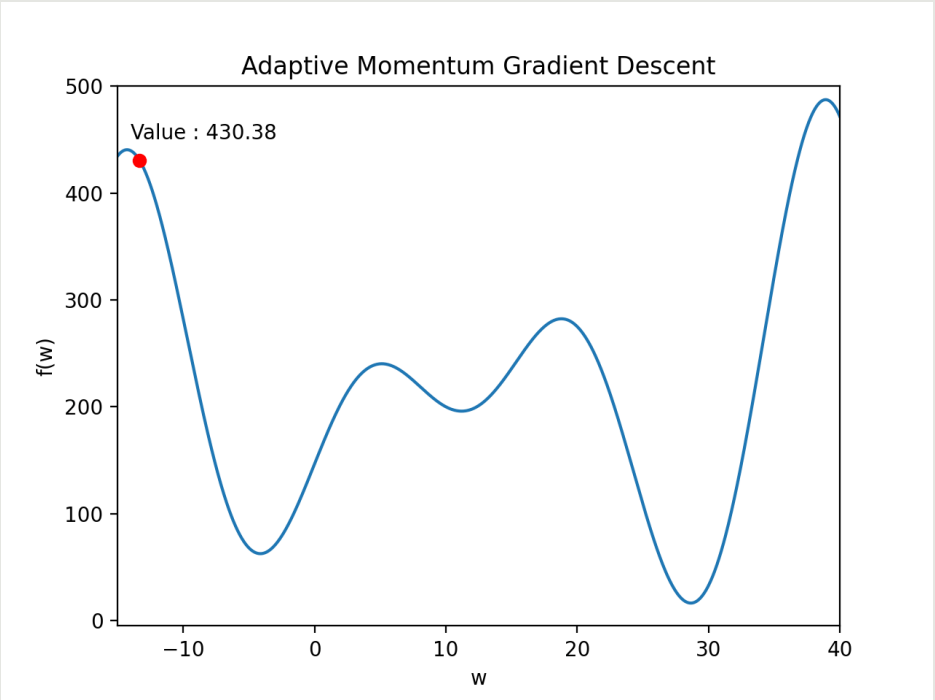
$$\begin{cases} +1 \text{ if } \beta_j > 0 \\ 0 \text{ if } \beta_j = 0 \\ -1 \text{ if } \beta_j < 0 \end{cases}$$

IMPORTANT: Once you have the gradient in closed form, you can apply a flavor of gradient descent, such as mini-batch Root Mean Squared Propagation or Adaptive Momentum Gradient Descent.

RMSPROP VS ADAM

To clarify, this situation is difficult because the objective function is not U-shaped; it is rather multimodal.



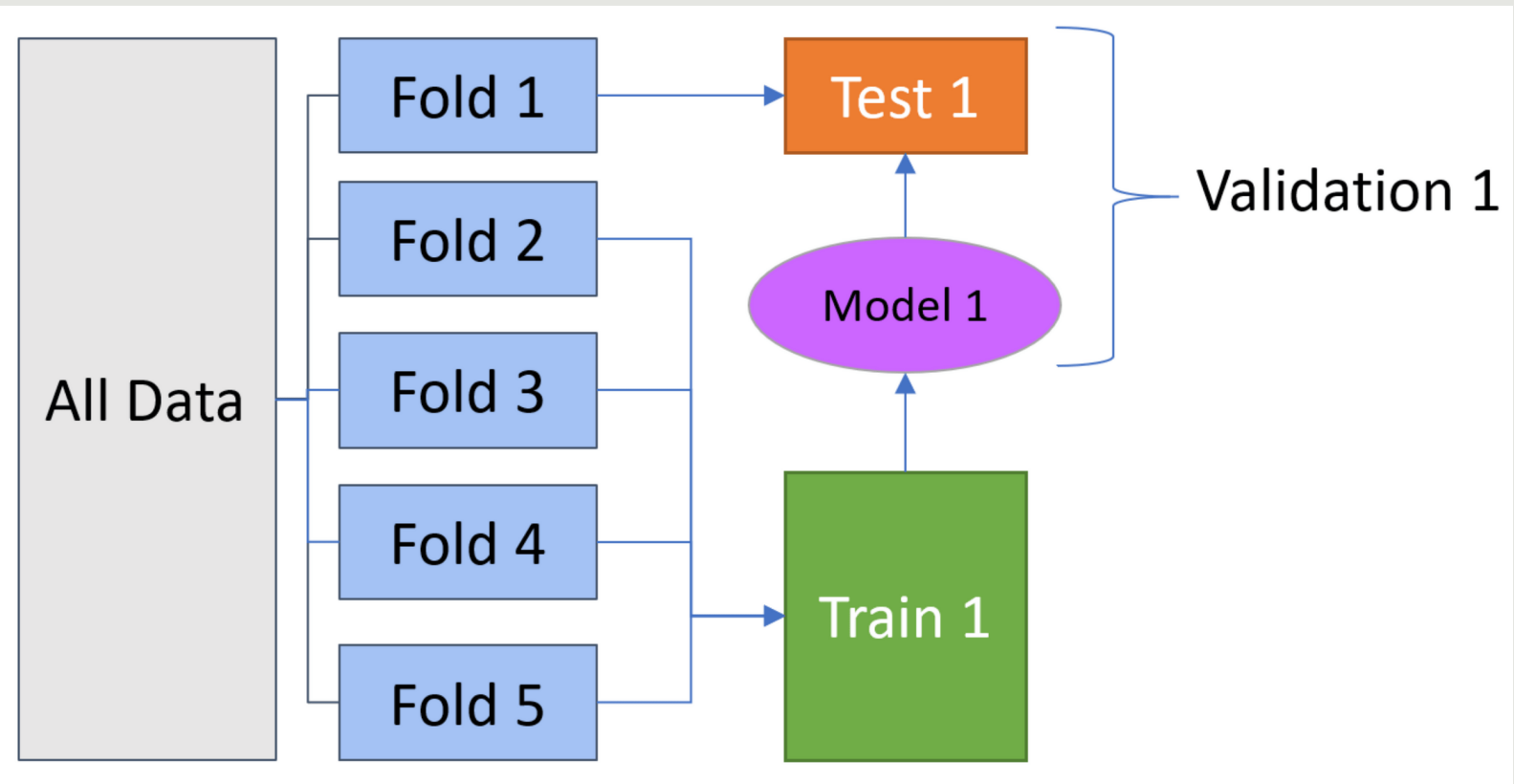


Do you see the difference?

HOW TO SELECT HYPERPARAMETERS?

The theory of the solution suggests K-Fold cross validations.

The following is a schematic for 5-Fold cross-validations



This is an iteration with 5 steps.

Now, it's time for coding applications!