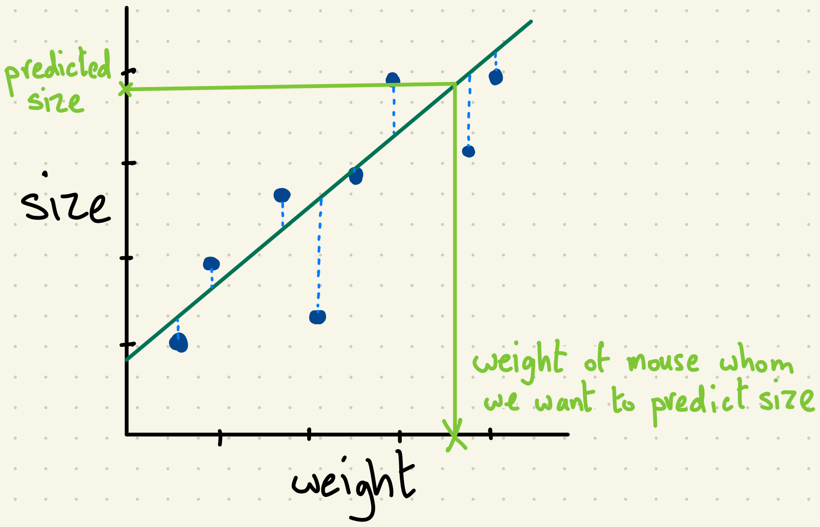
Notes on **Regularization** in machine learning

- Ridge regression (L2 regularization)
- Lasso regression (L1 regularization)
- ElasticNet regression

- Let's start with data that represents weights and sizes of mice



Since these data look relatively linear, we will use Linear Regression, AKA least squares, to model the relationship between weight and size.

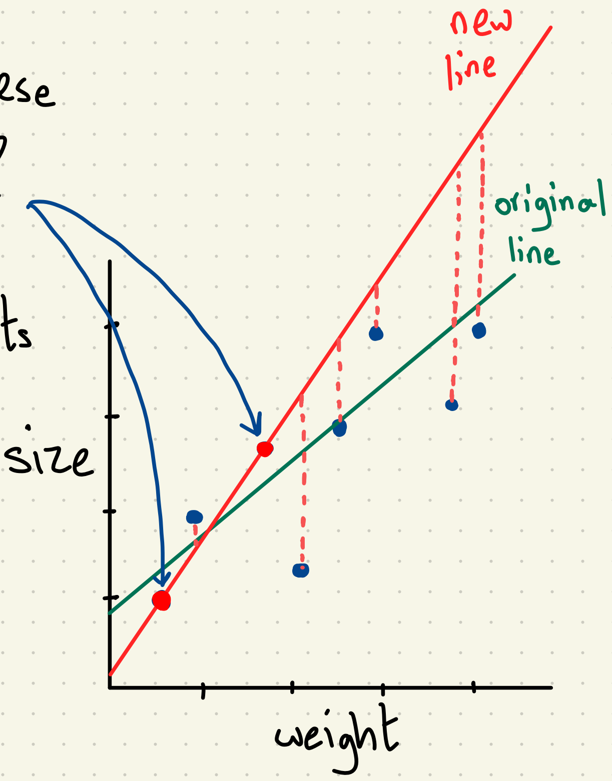
- Ultimately, we end up with this equation for the line:

$$\text{size} = \underbrace{0.9}_{\text{intercept}} + \underbrace{0.75}_{\text{slope}} * \text{weight}$$

When we have a lot of measurements, we can be fairly confident that the least squares line accurately reflects the relationship between size and weight

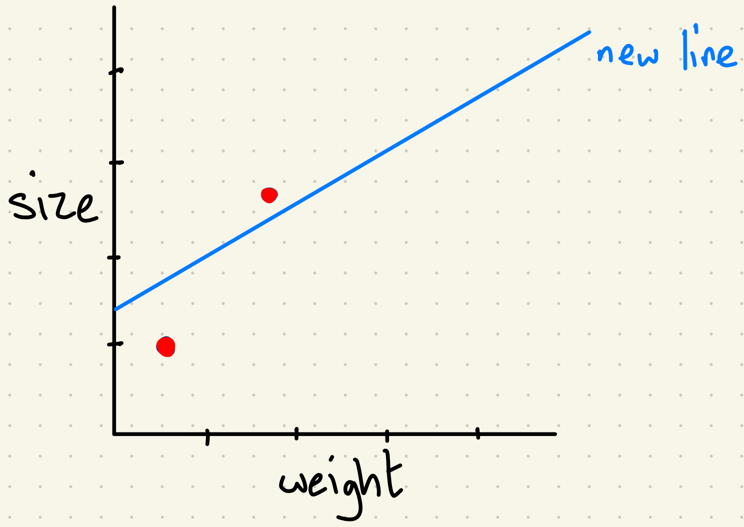
- But what if we had these two measurements only?
- We get a new line that overlaps the two data points (SSR = 0)

  size = 0.4 + 1.3 * weight
- Let's call red dots the training data, and blue dots the testing data



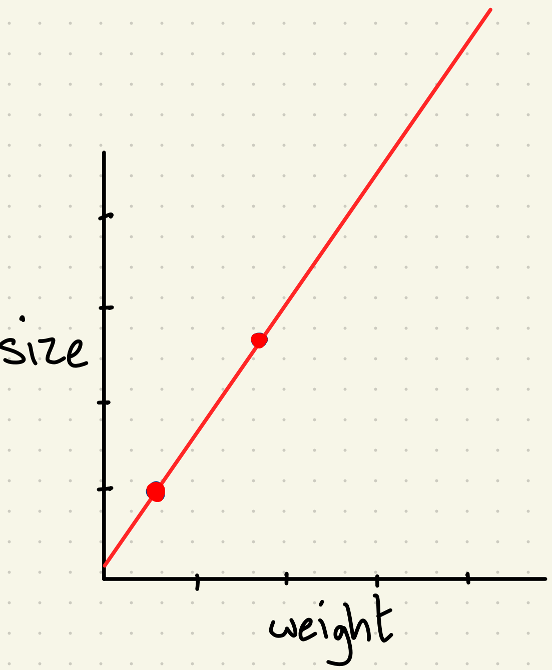
- SSR for training data is 0
- SSR for testing data is high
- It means that the new line has high variance
- New line is over fit to the training data

The main idea behind ridge regression is to find a new line that doesn't fit the training data as well, in other words, we introduce a small amount of bias into how the new line is fit



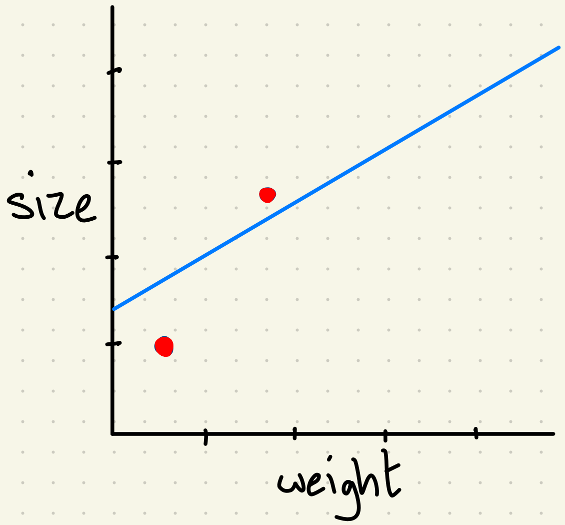
but in return, for that small amount of bias, we get a significant drop in variance. In other words, by starting with a slightly worse fit, ridge regression can provide better long term predictions.

How does ridge regression work ?



least squares

minimizes SSR



ridge regression

$$\text{minimizes } SSR + \underset{\text{how severe the penalty is}}{\lambda} . \underbrace{\text{slope}^2}_{\text{penalty}}$$

- Let's calculate loss : (with $\lambda = 1$)

least squares line :

$\text{size} = 0.4 + 1.3 \times \text{weight} + 1 . \text{weight}^2 \Rightarrow \text{loss} = 0 + 1.3^2 = 1.69$

ridge regression line :

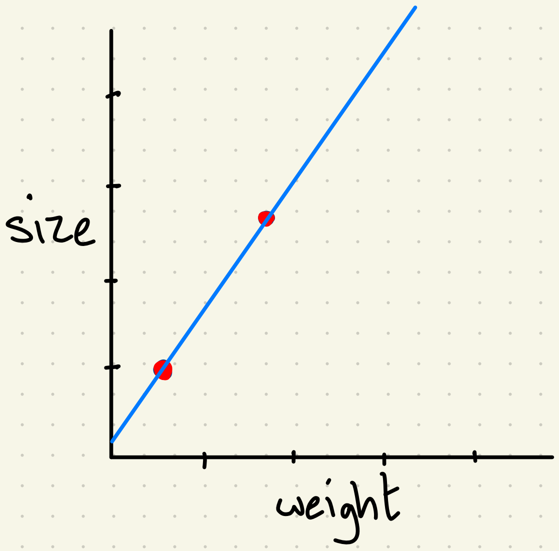
$\text{size} = 0.9 + 0.8 \times \text{weight} + 1 . \text{weight}^2 \Rightarrow \text{loss} = 0.3^2 + 0.1^2 + 0.8^2$

$= 0.74 < 1.69$

Thus, if we wanted to minimize the sum of the squared residuals + the ridge regression penalty, we would choose the ridge regression line over the least squares line

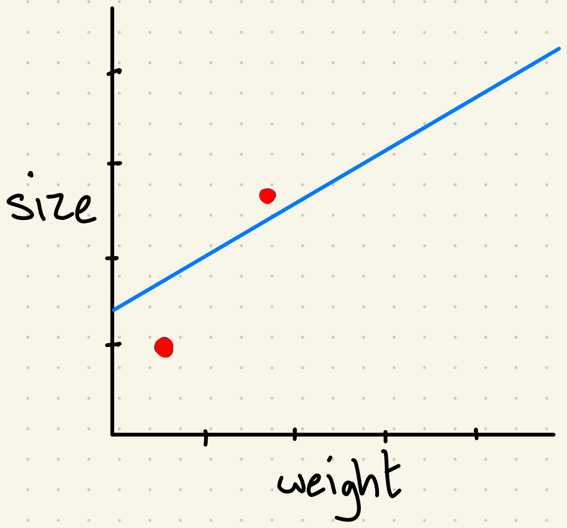
Lambda (λ):

When the slope of the line is steep, then the prediction for size is very sensitive to relatively small changes in weight. And vice versa, when the slope is small, predictions for size are much less sensitive to changes in weight.

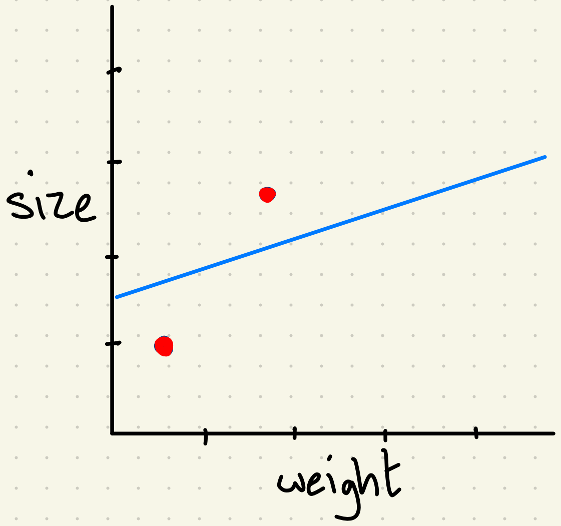


$\lambda = 0$

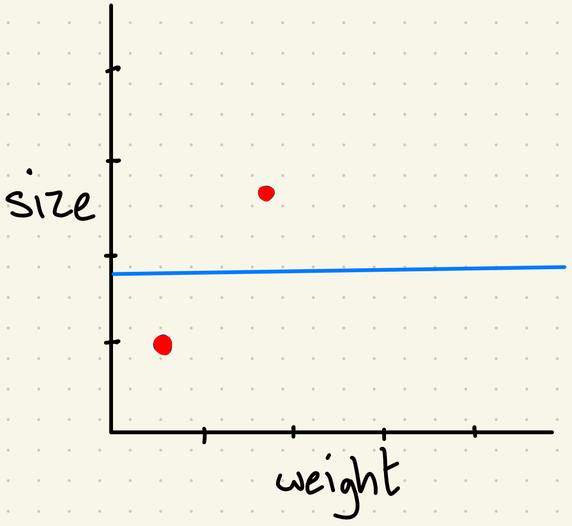
(equivalent to least squares)



$\lambda = 1$



$\lambda = 3$



$\lambda = 1000$

So the larger λ gets, our predictions for size become less and less sensitive to weight.

and the larger we make λ, the slope gets asymptotically close to 0

How do we decide the value of λ?

We just try a bunch of values for λ and use **cross validation**, typically **10-fold C.V.**, to determine which one results in the lowest **variance**.

$\lambda \in [0; +\infty)$

In the previous example, we've seen how ridge regression would work when predicting a **continuous variable**, using a **continuous variable**. But it also works when using a **discrete variable**

Ridge regression can also be used for **logistic regression**

We optimize the sum of the likelihoods $+ \lambda . \text{slope}^2$

So far, we've seen examples of how ridge regression helps reduce variance by shrinking parameters and making predictions less sensitive to them

---

We can also apply ridge regression when we have more than 1 parameter

Example:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$

We minimize:

$$\text{loss} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

Minimizing $\quad RSS + \lambda\theta^T\theta$

$$(y - X\theta)^T(y - X\theta) + \lambda\theta^T\theta$$

$$y^Ty - 2\theta^TX^Ty + \theta^TX^TX\theta + \lambda\theta^T\theta$$

because $(X\theta)^T = \theta^TX^T$

$$\frac{\partial\, y^Ty - 2\theta^TX^Ty + \theta^TX^TX\theta + \lambda\theta^T\theta}{\partial\theta} = -2X^Ty + 2X^TX\theta + 2\lambda\theta$$

set it equal to 0, solve for θ

$$0 = -2X^Ty + 2X^TX\theta + 2\lambda\theta$$

$$0 = -X^Ty + X^TX\theta + \lambda\theta$$

$$X^Ty = X^TX\theta + \lambda\theta$$

$$X^Ty = (X^TX + \lambda I)\theta$$

$$(X^TX + \lambda I)^{-1}X^Ty = \theta$$

Lasso regression:

Difference with ridge regression:

ridge regression: minimize SSR + $\underline{\lambda \times \text{slope}^2}$

lasso regression: minimize SSR + $\underline{\lambda \times |\text{slope}|}$

They look similar, and they can be used in similar situations

NOTE: Just like with ridge regression, in lasso regression, $\lambda \in [0;\infty)$ and can be determined using cross validation. And it also produces a line with a little bit of bias but less variance than least squares

- With multiple parameters:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$

$$\text{loss} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\theta_j|$$

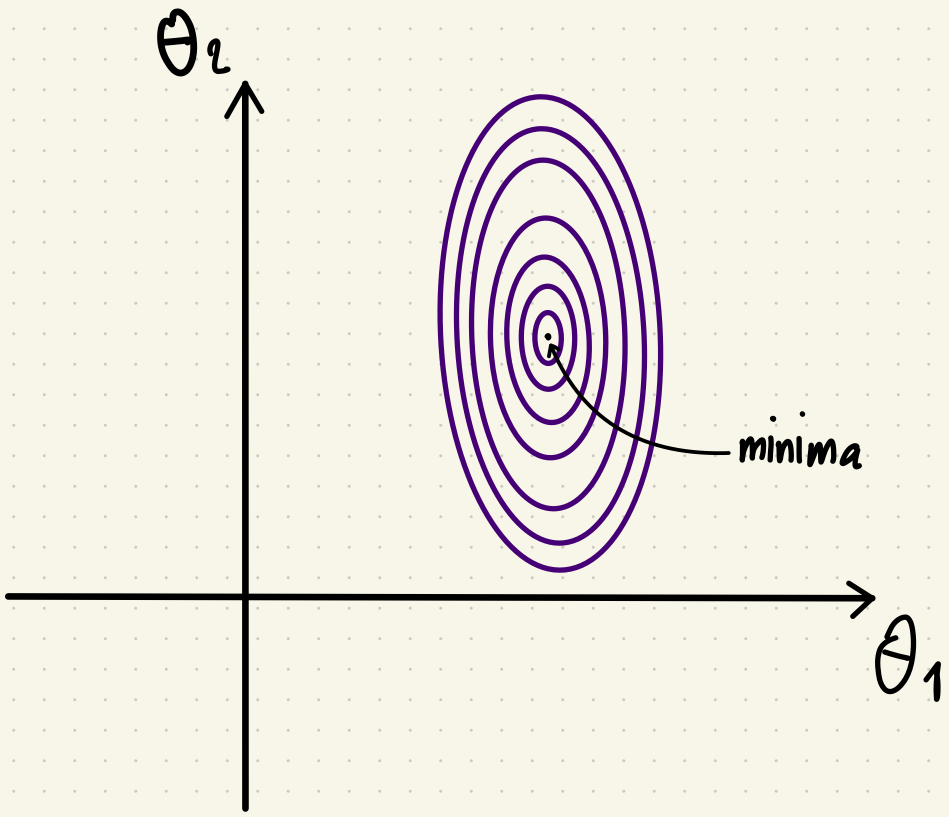
So what's the difference?

Ridge regression can only shrink the slope asymptotically close to 0, while lasso regression can shrink the slope all the way to 0

- So, lasso regression can exclude useless variables from equations, it is a little bit better than ridge regression at reducing the variance in models that contain a lot of useless variables.
- In contrast, ridge regression tends to do a little better when most variables are useful.

Explanation with a contour plot:

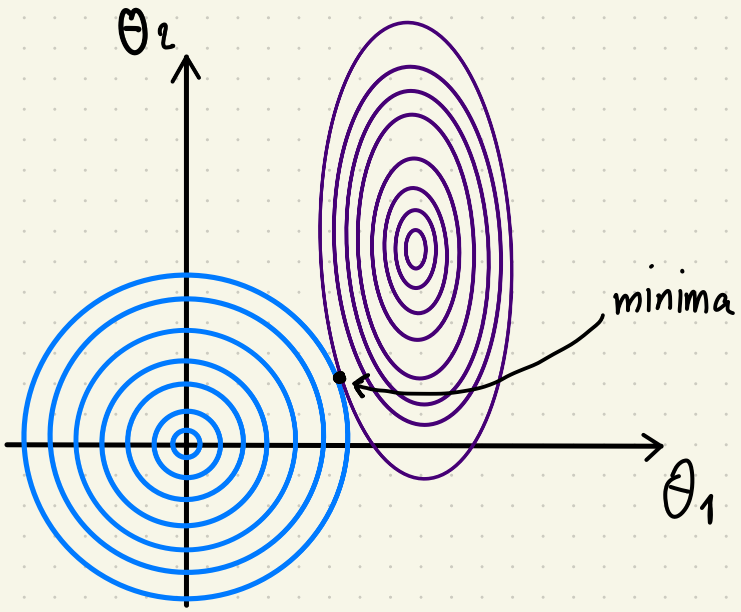
Without regularization, the goal is to minimize $\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$ only, the minima will be inside the circles



But with ridge regression, there will be another term, that will "attract" parameters (here θ_1, θ_2) to zero. So here our algorithm will try to minimize the sum of two terms:

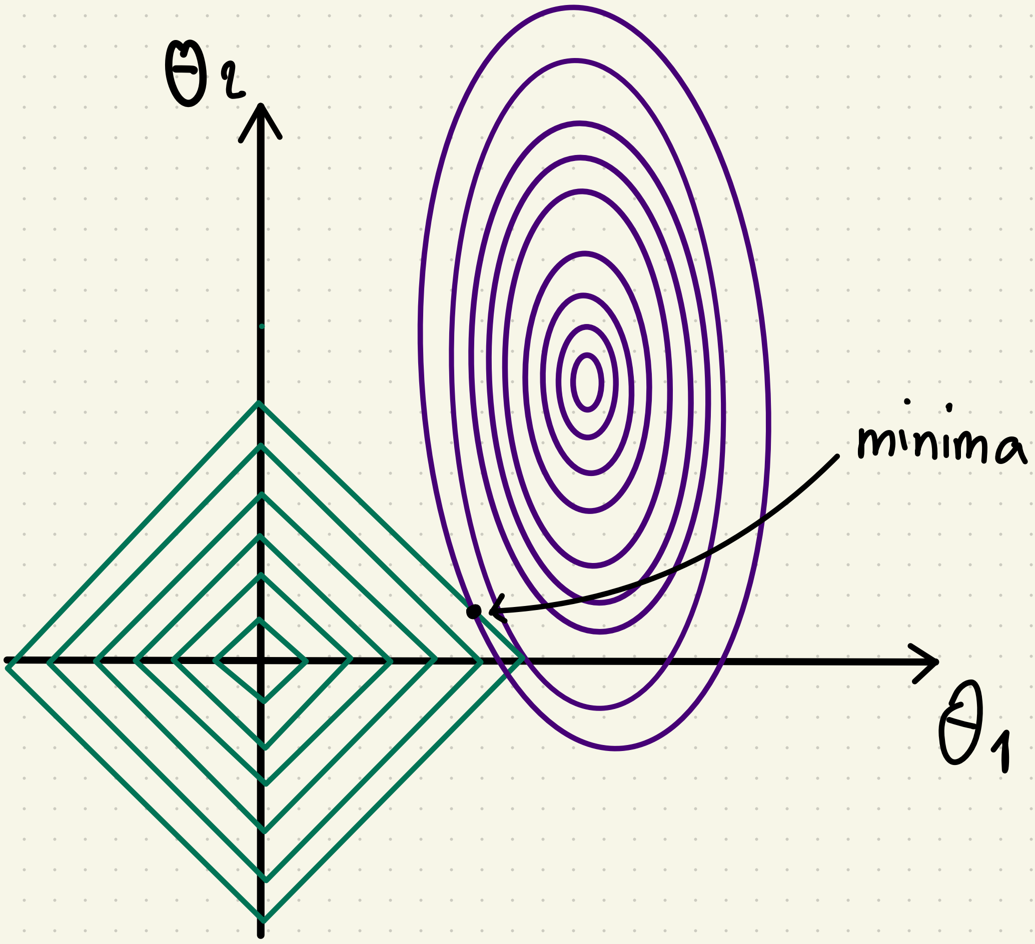
$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$, which attracts our vector to the center of purple circles, and $\lambda \sum_{j=1}^{n} \theta_j^2$, which attracts our vector to $\vec{0}$.

So our minima will be somewhere in-between



And the higher λ is, the more "powerful" the second term will be, our vector get closer to $\vec{0}$.

Same logic with lasso regression, the second term, $\lambda \sum_{j=1}^{n} |\theta_j|$, will attract our vector to $\vec{0}$



ElasticNet regression:

- We said that lasso regression works best when your model contains a lot of useless variables
- We also said that ridge regression works best when most of the variables are useful
- So if we know a lot about all of the parameters in our model, it's easy to choose

But what if we don't?

We use ElasticNet regression

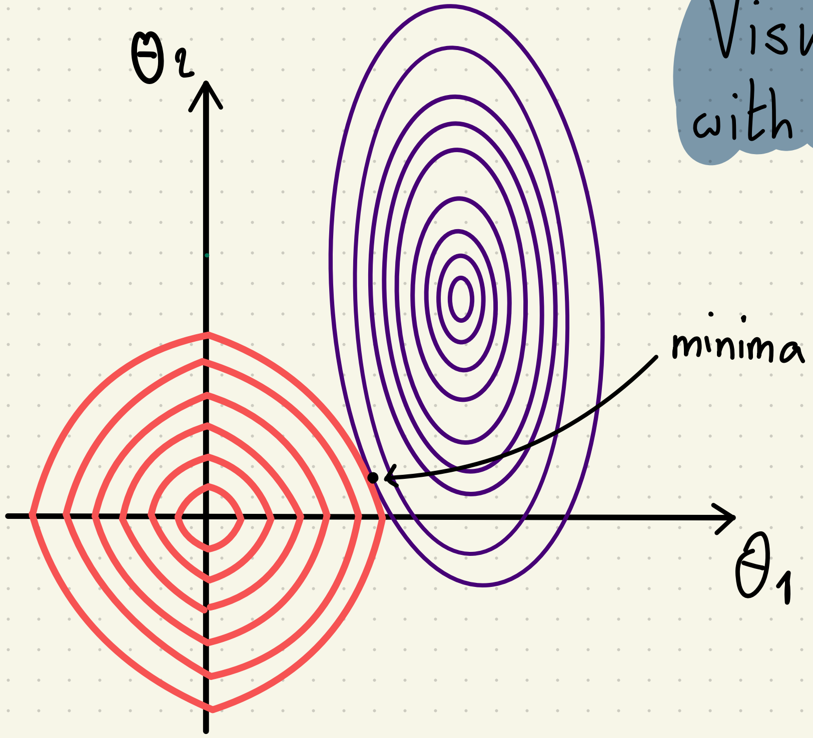
It combines both lasso regression and ridge regression, and takes advantage of both of their benefits

With ElasticNet regression, we want to minimize :

$$\sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{n} |\theta_j| + \lambda_2 \sum_{j=1}^{n} \theta_j^2$$

Note that λ_1 and λ_2 don't have to be equal, and their values can be found with **cross validation**

Visualization with contour plot :



θ_2
θ_1
minima