# L2 Regularization

Shah Md. Arshad Rahman Ziban

## Regularization

Regularization is a technique used to prevent overfitting in machine learning models. It discourages the model from becoming too complex, helping it generalize better to new data by adding a penalty to the model's complexity.

### L2 Regularization (Ridge) $\rightarrow$ "Weight Shrinker"

L2 regularization adds a penalty based on the square of the model's weights. The penalty is proportional to the sum of the squares of the coefficients.

#### Effect

- Unlike L1 regularization, L2 regularization doesn't make the coefficients exactly zero. Instead, it reduces their magnitude.
- This results in all features being retained, but their coefficients shrink toward zero, effectively preventing any one feature from dominating the model.
- L2 regularization is used when you want to reduce the influence of less important features without completely eliminating them.

#### Example

Imagine you're building a house and want to make sure all the components (e.g., walls, doors, windows) are balanced in size. If you don't shrink the size of some components, they could dominate the overall look of the house. L2 regularization is like gently shrinking the sizes of all components to maintain balance, without removing any of them.

### Use L2 when

- You don't want to eliminate features but wish to reduce the impact of less important ones.
- Your model is overfitting due to too many large weights.

## Steps for L2 Regularization (Ridge Regression)

Let's go through the steps to implement **Ridge Regression** (L2 regularization) using Python:

### Step 1: Generate the dataset

We generate random values for $X$ and $y$, just as we did with Lasso regression. Here, we use `numpy` to create the dataset.

```
import numpy as np

# Generate random values for X and y
X = np.random.randint(1, 100, size=20)
y = np.random.randint(1, 100, size=20)
```

### Step 2: Fit the Ridge regression model

We use the `Ridge` class from the `sklearn.linear_model` module to apply L2 regularization and fit the model.

```
from sklearn.linear_model import Ridge

# Reshape X to a 2D array
X = X.reshape(-1, 1)

# Initialize the Ridge regression model
ridge = Ridge(alpha=1.0)

# Fit the model to the data
ridge.fit(X, y)
```

#### Why Reshape?

Similar to Lasso regression, we reshape $X$ to be a 2D array because scikit-learn models, including Ridge regression, require $X$ to be in this format. Even though $X$ starts as a 1D array, we reshape it to a 2D array with 20 rows and 1 column (`X.reshape(-1, 1)`).

### Step 3: Predict the value for the given input

We now use the fitted Ridge model to predict the value for an input, for example, 103.

```
# Predict the value for X = 103
prediction = ridge.predict([[103]])

# Print the prediction
print("The␣predicted␣value␣for␣103␣is:", prediction[0])
```

## Final Note

- The actual predicted value will vary each time you run the code, as the dataset is randomly generated.
- The prediction will be less sensitive to fluctuations in the dataset compared to Lasso because L2 regularization doesn't zero out any features.

## Ridge Regression (L2 Regularization)

Ridge regression is a type of linear regression that incorporates an L2 regularization term to penalize large coefficients. This helps to reduce the model complexity, prevent overfitting, and keep all features in the model, but with smaller coefficients. It's ideal when you want to maintain all the features while preventing overfitting by controlling the magnitude of the coefficients.