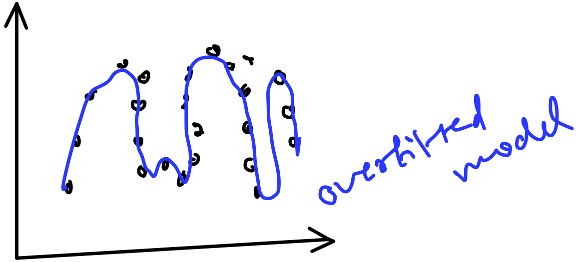
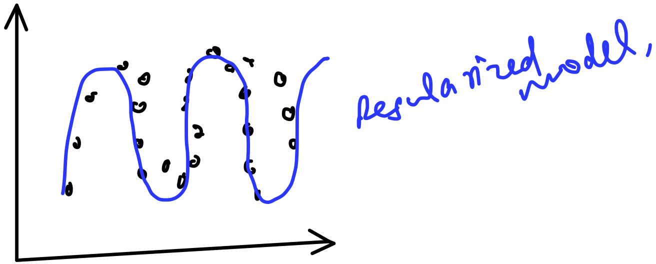
# Norms and Regularization

Machine Learning

# What is Regularization

- A set of techniques to discourage overly complex models.
- as an overly complex model might mean that model is overfitting.




Overfitting: When a model exhibits very low error (or high accuracy) on the training data but performs significantly worse when presented with new, unseen data (i.e., the validation or test set).

# Steps in a Machine Learning Problem

Total Loss = Data Loss + Penalty.

Problem Definition

Data Collection and Wrangling

Data Preprocessing and Feature Engineering

Modeling and Algorithm Theory

Evaluation and Improvement

Regularization

# How do we solve this Problem

Generally we use the concept of Norms

In ML, we measure size of vectors using a function called a norm.

Formally, Lp Norm

It is represented by:

$$\boxed{||x||_p = (\Sigma_i |x_i|^p)^{1/p}}$$

L1 Norm is known as Frobenius Norm or Manhattan Norm M

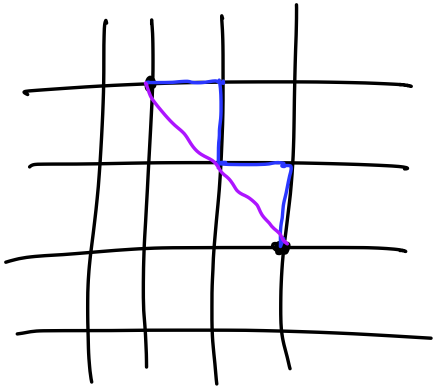
L2 Norm is known as Euclidean Norm E

L1 Norm → $p = 1$.

→ $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

L2 Norm → $p = 2$

$\|x\|_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}$



# Quick History Lesson

**Lp Norm:**

French mathematician Henri Léon Lebesgue. His work on the Lebesgue integral (early 1900s) was crucial for defining these spaces.

# Quick History Lesson

**L2 Regularization (Ridge Regression):** → Density W Matrix.

Arthur E. Hoerl and Robert W. Kennard (1970) introduced Ridge Regression in their paper: Ridge Regression: Biased Estimation for Nonorthogonal Problems.

“Adding a small amount of bias (the L2 penalty) to the least squares objective could significantly reduce the variance of the coefficient estimates, solving the problem of multicollinearity.”

**L1 Regularization (Lasso Regression):** → Sparsity W Matrix.

Tibshirani (1996) introduced the Lasso (Least Absolute Shrinkage and Selection Operator) in his paper: Regression Shrinkage and Selection via the Lasso.

“Using the L1 norm penalty had a unique and highly desirable side effect: it would shrink some coefficients exactly to zero, effectively performing automatic feature selection.” This sparsity property distinguished it from Ridge Regression.

# L2 Norm

$$\|x\|_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} = (x_1^2 + x_2^2 + \dots x_n^2)^{\frac{1}{2}}$$

Total loss = D.L. + Penalty.

$$\text{Penalty} = \underbrace{\lambda}_{\text{strength}} \cdot \underbrace{(\text{Lp Norm})}_{p=1,2}$$

$$\boxed{P = \lambda \cdot \left( \cancel{\sum_{i=1}^{n}} (x_i)^2 \right)^{\frac{1}{2}}}$$

$w \leftarrow x_i$

Causes Dens |W|.

$$\frac{\partial P}{\partial w} = 2\lambda w.$$

$$\boxed{\partial P \propto w}$$

$w \uparrow \rightarrow$ Penalty $\uparrow$

$w \downarrow \rightarrow$ Penalty $\downarrow$

# L1 Norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| = |x_1| + |x_2| - - - |x_n|.$$

$$\text{Total Loss} = D.L + \underbrace{\lambda . \sum_{i=1}^{n} |x_i|}_{P}.$$

$$\frac{\partial P}{\partial w} = \lambda . (\text{sign}(w)).$$

$$\boxed{\partial P \propto \text{constant}}$$

Sparsity → w Matrix,

$w \uparrow \rightarrow P = K.$

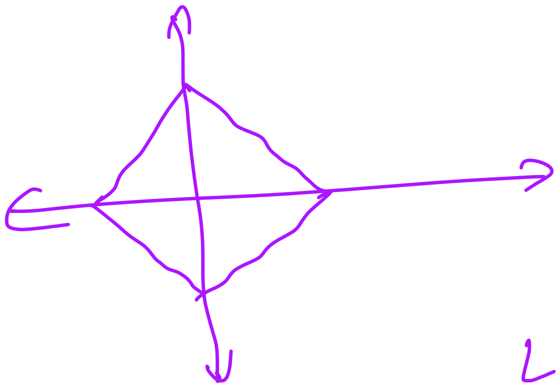
$w \downarrow \rightarrow P = K.$

## Plotting L1 and L2 Norm

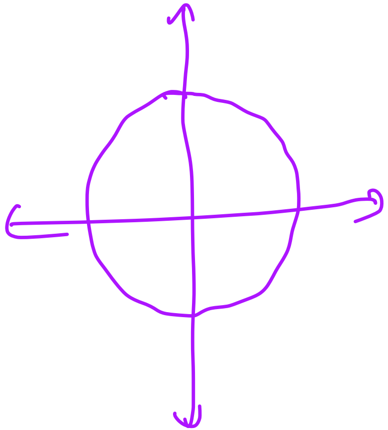
$w_1$ & $w_2$.

$$L_1 = |w_1| + |w_2|$$

$$L_2 = |w_1|^2 + |w_2|^2$$



$L_1$



$L_2$

# Weights Updation

$L_1$ Reg. $\rightarrow$ $w_i^{new} = w_i^{old} - L.R. \left( \frac{\partial loss}{\partial w} + \lambda . sign(w_i^{old}) \right)$.

$w_i^{old} \rightarrow$ large (+). $\rightarrow$ $w_i^{new} \downarrow$

$\rightarrow$ small (+) $\rightarrow$ $w_i^{new} \downarrow$ effect >>>

$\rightarrow$ near (0). $\rightarrow$ $w_i^{new} \simeq 0$.

Penalty. = const.

$L_2$ Reg. $\rightarrow$ $w_i^{new} = w_i^{old} - LR \left( \frac{\partial loss}{\partial w} + \underline{2 \lambda w_i^{old}} \right)$.

$w_i^{old} \rightarrow$ large (+) $\rightarrow$ $w_i^{new} \downarrow$

$\rightarrow$ large (-) $\rightarrow$ $w_i^{new} \rightarrow 0$.

$\rightarrow$ small (+) $\rightarrow$ slight push.

Penalty $\propto$ w.

# Code - Python

```python
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.datasets import load_diabetes

from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

diabetes = load_diabetes()

X, y = diabetes.data, diabetes.target
feature_names = diabetes.feature_names

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

```

```python

model_linear = LinearRegression()

# L2 Regularization (Ridge)
alpha_ridge = 1.0  # Common starting point for L2
model_ridge = Ridge(alpha=alpha_ridge)


# L1 Regularization (Lasso)
alpha_lasso = 0.1  # A value to promote sparsity in this dataset
model_lasso = Lasso(alpha=alpha_lasso, max_iter=10000)

models = {
    "Linear Regression (No Reg)": model_linear,
    f"Ridge Regression (L2, α={alpha_ridge})": model_ridge,
    f"Lasso Regression (L1, α={alpha_lasso})": model_lasso
}

results = []
coef_df = pd.DataFrame(index=feature_names)

```

# Code - Python

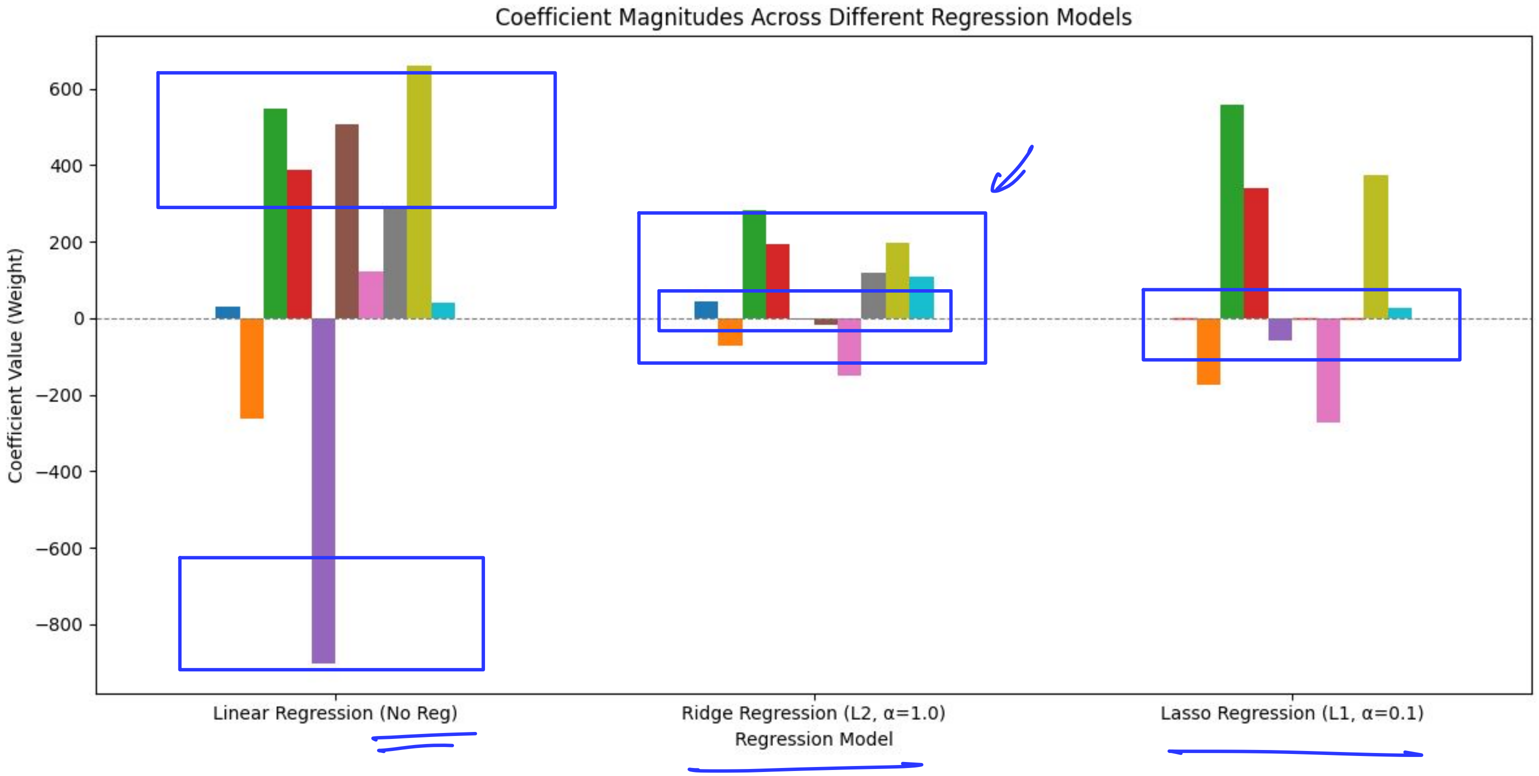
Train

```python

print("--- Training and Evaluation ---")
for name, model in models.items():
    # 1. Train the model
    model.fit(X_train, y_train)

    # 2. Predict and Evaluate
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    train_r2 = r2_score(y_train, y_train_pred)
    test_r2 = r2_score(y_test, y_test_pred)

    # 3. Store Coefficients
    # LinearRegression and Ridge/Lasso models have a 'coef_' attribute
    coef_df[name] = model.coef_

    # 4. Store Metrics
    results.append({
        'Model': name,
        'Train R2': train_r2,
        'Test R2': test_r2,
        'Generalization Gap (Train - Test)': train_r2 - test_r2
    })

    print(f"\n{name}:")
    print(f"  R2 Train/Test: {train_r2:.3f} / {test_r2:.3f}")
    print(f"  Generalization Gap: {train_r2 - test_r2:.3f}")


print("\n" + "="*50)
print("                    Model Performance Summary")
print("="*50)
results_df = pd.DataFrame(results).set_index('Model')
print(results_df)

print("\n" + "="*50)
print("     Model Coefficient Comparison (Weight Magnitude)")
print("="*50)

# Round the coefficients for cleaner output
coef_df_rounded = coef_df.T.round(1)
print(coef_df_rounded)

```

Plotting.

```python

fig, ax = plt.subplots(figsize=(12, 6))

# Plot the coefficients side-by-side
coef_df.T.plot(kind='bar', ax=ax, legend=False)

# Add line for zero coefficient
ax.axhline(0, color='grey', linestyle='--', linewidth=0.8)

ax.set_title("Coefficient Magnitudes Across Different Regression Models")
ax.set_xlabel("Regression Model")
ax.set_ylabel("Coefficient Value (Weight)")
ax.set_xticklabels(coef_df.columns, rotation=0)

# Highlight Lasso's zero coefficients
for rect in ax.patches:
    if rect.get_x() > 1.5 and abs(rect.get_height()) < 0.1: # Check for Lasso bars near zero
        rect.set_color('red')

plt.tight_layout()
plt.show()
```

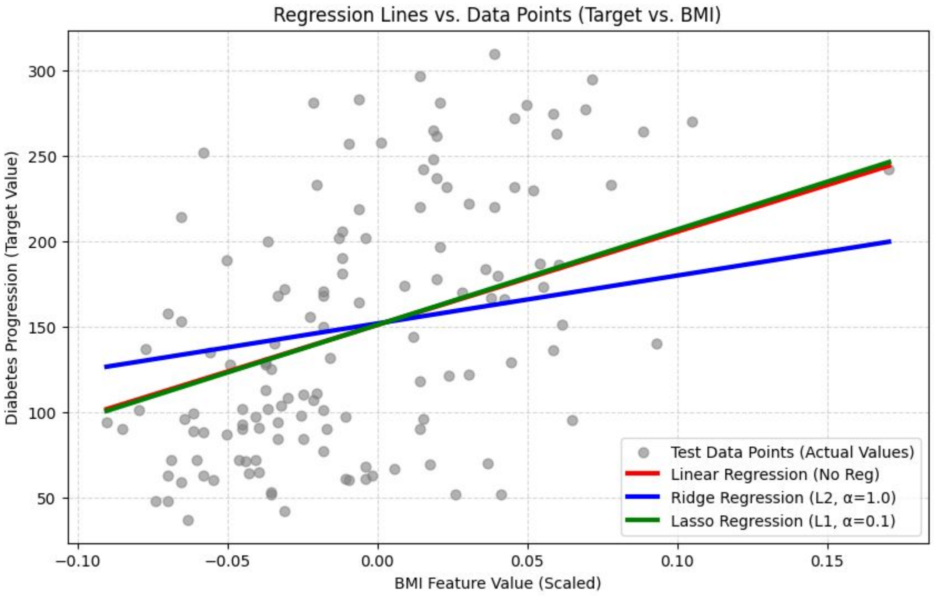
# Results



Coefficient Magnitudes Across Different Regression Models
Coefficient Value (Weight)
600
400
200
0
−200
−400
−600
−800
Linear Regression (No Reg)
Ridge Regression (L2, α=1.0)
Lasso Regression (L1, α=0.1)
Regression Model

# Training and Evaluation Results

```
==================================================
            Model Performance Summary
==================================================
                              Train R2   Test R2  \
Model
Linear Regression (No Reg)    0.524412  0.477290
Ridge Regression (L2, α=1.0)  0.428318  0.423344
Lasso Regression (L1, α=0.1)  0.513414  0.485919

                              Generalization Gap (Train - Test)
Model
Linear Regression (No Reg)                             0.047123
Ridge Regression (L2, α=1.0)                           0.004974
Lasso Regression (L1, α=0.1)                           0.027495

==================================================
     Model Coefficient Comparison (Weight Magnitude)
==================================================
                               age    sex    bmi     bp     s1     s2     s3  \
Linear Regression (No Reg)    29.3 -261.7  546.3  388.4 -902.0  506.8  121.2
Ridge Regression (L2, α=1.0)  45.1  -71.9  280.7  195.2   -2.2  -17.5 -148.7
Lasso Regression (L1, α=0.1)   0.0 -173.3  558.9  339.4  -58.7   -0.0 -274.1

                                 s4     s5     s6
Linear Regression (No Reg)    288.0  659.3   41.4
Ridge Regression (L2, α=1.0)  120.5  198.6  106.9
Lasso Regression (L1, α=0.1)    0.0  372.8   25.6
```



# References


1. Ridge Regression: https://homepages.math.uic.edu/~lreyzin/papers/ridge.pdf
2. Lasso Regression: https://academic.oup.com/jrsssb/article/58/1/267/7027929
3. Notebook with Code: https://colab.research.google.com/drive/1vQw3tayW6sKwG1mdiRFgx9gF8gaGCXp_?usp=sharing
4. Bishop - Pattern Recognition and Machine Learning 2006: https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
5. Deep Learning - Ian Goodfellow, Yoshua Bengio and Aaron Courville: https://www.deeplearningbook.org/