Improving Model Performance

Subset Selection

Subset selection is a technique used in machine learning and statistics to choose a smaller set of predictors (features) from a larger set, in order to build a simpler and more interpretable model.

The main idea is:

✔️ You might have many features, but not all of them are helpful or necessary.

✔️ Subset selection helps identify which features are most useful for making predictions.

✔️ By selecting only the important features, you can improve the model's performance, reduce overfitting, and make the model easier to understand.

✅ Why Subset Selection is Used

- **Avoid overfitting** – Using too many features can make the model fit noise instead of the true pattern.
- **Improve interpretability** – A smaller number of features is easier to explain.
- **Reduce computational cost** – Fewer features mean faster training and prediction.
- **Handle irrelevant or redundant features** – Some features might not provide new information.

✅ Types of Subset Selection Methods

1. **Best Subset Selection**
    - Tries all possible combinations of features and selects the best one according to some criterion (like lowest error).
    - Computationally expensive for large datasets.

2. **Forward Selection**
    - Starts with no features.
    - Adds one feature at a time that improves the model the most.
    - Stopping when adding more features no longer improves the model.
3. **Backward Elimination**
    - Starts with all features.
    - Removes one feature at a time that contributes the least to the model's performance.
    - Stopping when removing more features would worsen the model.
4. **Stepwise Selection**
    - A combination of forward and backward methods.
    - Features are added and/or removed at each step depending on which action improves the model the most.

✅ Example

Suppose you have 5 features: Age, Income, Education, Experience, and Marital Status.

- Subset selection might find that only **Age**, **Income**, and **Experience** are important for predicting salary.
- The model will then use these 3 features instead of all 5.

✅ Relation to Regularization

Subset selection is one way to simplify the model. Another popular method is **regularization**, which doesn't remove features but **shrinks their coefficients** to prevent overfitting.

Examples:

- **Lasso regression** → Can shrink some coefficients to zero → effectively performing subset selection.

- **Ridge regression** → Shrinks coefficients but doesn't reduce them to zero → keeps all features but reduces their impact.

✅ Takeaway

✔️ Subset selection helps create simpler, more generalizable models.

✔️ It's especially useful when you have many features but not all are necessary.

✔️ It balances model complexity and performance.

✔️ It's often combined with other techniques like regularization for better results.

Additional Metrics used in Subset Selection

✅ 1. Cp or Akaike Information Criterion (AIC)

$$C_p = \frac{1}{n}\left(RSS + 2d\hat{\sigma}^2\right)$$

- **RSS** = Residual Sum of Squares (how well the model fits the data).
- **d** = number of predictors.
- $\hat{\sigma}^2$ = estimate of error variance.

🔑 **Purpose**: Penalizes the model for adding more predictors. It balances goodness of fit with model complexity.

A lower C_p suggests a better model — one that fits well but isn't too complex.

✅ 2. Bayesian Information Criterion (BIC)

$$BIC = \frac{1}{n}\left(RSS + \log(n) \cdot d\hat{\sigma}^2\right)$$

- Similar to C_p, but the penalty for extra predictors is larger because of the $\log(n)$ term.

🔑 **Purpose**: Encourages simpler models even more strongly than C_p, especially when nn (sample size) is large.

✅ 3. Adjusted R²

D

- Adjusts the regular R^2 (which always increases when adding predictors) to account for the number of predictors.
- A higher Adjusted R^2 indicates a better model, but only if the new predictor truly improves the fit.

🔑 **Purpose**: Avoids overfitting by penalizing models that don't add real value.

✅ 4. Cross Validation (CV) Error

- Uses resampling (like K-fold CV) to estimate how the model will perform on unseen data.
- Helps ensure that the model isn't just fitting noise in the training data.

🔑 **Purpose**: Choose models that generalize better to new data.

✅ Takeaways

✔️ Subset selection is not just about fitting the training data but also about generalizing well.

✔️ C_p, BIC, Adjusted R^2, and CV error are metrics that help choose the right subset by balancing fit and complexity.

✔️ The goal is to find a "sweet spot" where the model is accurate without being unnecessarily complicated.

Regularization (Shrinkage)

Regularization, also called **shrinkage**, is a technique used to prevent a model from overfitting the training data by controlling or limiting the complexity of the

model. It does this by adding a penalty to the model's loss function that discourages large or unnecessary coefficients (weights).

✅ Why Regularization is Needed

- A model with too many parameters or large weights can fit the training data very well but fail to generalize to new, unseen data → **overfitting**.
- Regularization helps by "shrinking" the weights toward zero or reducing their magnitude, making the model simpler and more robust.

✅ How It Works

Regularization adds a penalty term to the original cost (loss) function. The model tries to both fit the data and keep the weights small.

For example, the **linear regression loss function** is:

$$\text{Loss} = RSS = \sum (y_i - \hat{y}_i)^2$$

With regularization, this becomes:

$$\text{Loss} = RSS + \lambda \cdot \text{Penalty}$$

Where:

- **λ** controls how strong the penalty is.
- **Penalty** depends on the type of regularization used.

✅ Types of Regularization (Shrinkage)

1️⃣ Ridge Regression (L2 Regularization)

$$\text{Loss} = RSS + \lambda \sum w_j^2$$

- Penalizes the **square of the weights**.
- Encourages smaller but non-zero weights.

- Useful when all features are important but you want to reduce their impact.

2️⃣ Lasso Regression (L1 Regularization)

$$\text{Loss} = RSS + \lambda \sum |w_j|$$

- Penalizes the **absolute values of the weights**.
- Can shrink some weights to **exactly zero**, effectively selecting a subset of features.
- Useful when you believe only a few features are important.

3️⃣ Elastic Net (Combination of L1 and L2)

$$\text{Loss} = RSS + \lambda_1 \sum |w_j| + \lambda_2 \sum w_j^2$$

- Combines both Ridge and Lasso penalties.
- Balances between feature selection and weight shrinkage.

✅ Key Insights

✔️ Regularization controls overfitting by discouraging large weights.

✔️ The parameter λ determines how much penalty to apply:

- Small λ: close to ordinary regression.
- Large λ: pushes weights closer to zero.

✔️ Lasso can completely remove some features, while Ridge shrinks them without eliminating.

✔️ Regularization improves generalization and model interpretability.

✅ When to Use Regularization

- You have many features, and some may be irrelevant or noisy.
- Your model performs well on training data but poorly on test data.

- You want to make the model more interpretable and stable.

✅ Example Analogy

Think of regularization as adding **brakes** to a car:

- Without brakes → the car speeds recklessly → overfitting.
- With brakes → the car stays controlled → better handling on new roads → better generalization.

Polynomial Regression, Constraints, and Regularization

✅ 1. Polynomial Regression – Transformed Problem

Problem setup

- You are given data points:

  $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$

- Goal: Fit a polynomial model of degree Q, i.e.,

  $y = w_1 x + w_2 x^2 + ... + w_Q x^Q$

Feature transformation

- Transform each xix_i into a higher-dimensional vector:

  $z_i = [x_i, x_i^2, x_i^3, ..., x_i^Q]$

- Now the problem becomes linear in terms of z_i.

Objective function

- Minimize the **in-sample error**:

  $E_{in}(w) = \frac{1}{N} \sum_{i=1}^{N} (w^T z_i - y_i)^2$

- In matrix form:

$$E_{in}(w) = \frac{1}{N}(Zw - y)^T(Zw - y)$$

where Z is the matrix of all transformed vectors z_i.

Solution (normal equation)

$$w_{\text{lin}} = (Z^T Z)^{-1} Z^T y$$

- This is analogous to solving linear regression problems.

✅ 2. Constraints on Weights

Why apply constraints?

- To avoid overfitting by limiting model complexity.
- Prevents the polynomial from becoming too flexible and fitting noise.

Hard Constraint

- Force certain weights to be exactly zero:

  $$w_q = 0 \quad \text{for } q > 2$$

- Example: Fit a second-order polynomial and ignore higher powers.

Soft Constraint

- Allow weights to vary but limit their overall magnitude:

  $$\sum_{q=1}^{Q} w_q^2 \leq C$$

- Here, C controls how large the weights can be.
- Acts as a **regularization mechanism** that penalizes complexity.

✅ 3. Constrained Optimization

Concept

- Real-world problems often involve optimizing a function under restrictions:

- Individuals maximizing utility under a budget constraint.
- Firms maximize output under cost constraints.

General Form

- Objective function:

  $E_{in}(w)$

- Constraint:

  $u(w) \leq C$

  where u(w) is some function of the weights.

Purpose

- Helps find solutions that are feasible and prevent overfitting or unrealistic solutions.

✅ 4. Regularization (Shrinkage)

From constrained to unconstrained

- Instead of explicitly imposing a constraint, add a penalty term to the objective function.

Formulation

$\text{Minimize } E_{in}(w) + \lambda w^T w$

- E_{in}: Mean squared error.
- λ: Regularization parameter controlling the strength of penalization.
- $w^T w$: Squared norm of weights (sum of squares).

Interpretation

- Larger λ → more penalty → smaller weights → simpler model.
- Smaller λ → less penalty → larger weights → more flexible model.

Graphical intuition

- The solution space is restricted to an area where $w^T w \leq C$.
- Increasing C allows larger weights; decreasing it forces smaller weights.

Effect of λ

- **High λ** → models with lower weights → less prone to overfitting.
- **Low λ** → models with higher weights → more expressive but more likely to overfit.

✅ 5. Key Takeaways

1. **Polynomial regression** helps model non-linear relationships by transforming features into higher dimensions.
2. **Constraints on weights** (hard or soft) prevent overfitting by controlling model complexity.
3. **Constrained optimization** is a common way to formalize problems where trade-offs must be made.
4. **Regularization** is a soft constraint technique that adds a penalty term to control the size of the weights.
5. The regularization parameter λ\lambda balances the trade-off between bias and variance.

✅ Extra Notes

- Regularization techniques include:
  - **Ridge regression (L2)**: Penalizes the squared magnitude of weights.
  - **Lasso regression (L1)**: Penalizes the absolute magnitude of weights.
- The transformed polynomial regression can be seen as applying linear regression in a new feature space.
- Regularization leads to better generalization and prevents extreme weight values that can fit noise.

Regularization Variants – Weighted Penalization

✅ 1. Standard Regularization vs Weighted Regularization

Standard regularization

- The regularization term is typically:

$$\lambda \sum_{q=0}^{Q} w_q^2$$

  where:

  - λ is a single global parameter.
  - All weights w_q are penalized equally.

Weighted regularization (variant)

- Instead of using a single λ, assign a **different penalty** to each weight:

$$\sum_{q=0}^{Q} \gamma_q w_q^2$$

  where:

  - γ_q is a weight-specific parameter.
  - Each weight w_q is penalized differently.

✅ 2. Why Use γ_q?

- Allows **fine-grained control** over how much each weight is regularized.

- Helps prevent overfitting more effectively, especially when some features should contribute less.

✅ 3. Example Variants

Example 1: Exponential weighting

$$\gamma_q = 2^q$$

- Higher-order polynomial terms (larger q) are penalized more.
- Encourages the model to focus on simpler, lower-degree features.
- Reduces the effect of higher powers like x^3, x^4, ..., which are prone to overfitting.

Example 2: Exponential decay

$$\gamma_q = 2^{-q}$$

- Higher-order terms are penalized less.
- This is generally not desirable because it may allow higher-degree terms to dominate.
- Example: x^8 would be multiplied by $\frac{1}{2^8}$, greatly reducing its impact.

Example 3: Linear weighting

γ_q = q

- Higher-order terms are penalized linearly more than lower ones.
- Example:

$$\gamma_1 = 1, \quad \gamma_2 = 2, \quad \gamma_3 = 3$$

- Less aggressive than exponential penalties.

✅ 4. Intuition Behind Penalizing Higher-Order Terms

- Higher powers of x (like x^3, x^4) can cause models to:
  - Overfit the training data.
  - Be too sensitive to small variations or noise.
- Penalizing them helps:
  - Improve generalization.
  - Avoid unrealistic fits.
  - Ensure simpler models when possible.

✅ 5. Practical Takeaway

- Using γ_q instead of a global λ gives you the flexibility to:
  - Encourage smoother fits.
  - Control complexity at different polynomial degrees.
- Choosing appropriate γ_q depends on the problem and domain knowledge.

✅ Final Notes

- Weighted regularization is a powerful extension to standard ridge regression.
- Exponential and linear penalties are common choices.
- Helps balance bias and variance in polynomial models.
- Especially useful when certain features or polynomial terms are expected to be less reliable or more noisy.

What is Ridge Regression?

Ridge Regression is a regularized version of **Linear Regression** where a penalty is added to the loss function to prevent overfitting. It is particularly useful when:

- The dataset has **multicollinearity** (i.e., features are correlated).

- The model has many features and might overfit to noise in the data.

The core idea is to **shrink the coefficients** by adding a penalty term proportional to the square of the coefficients.

✅ Why Do We Need Ridge Regression?

- In **Ordinary Least Squares (OLS)** regression, the objective is to minimize the squared error between the predicted and actual values:

$$E_{OLS} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

  However, when features are highly correlated or when the number of features is large compared to the number of observations, this leads to large coefficient estimates that fit the training data too well but fail to generalize.

- Ridge regression solves this by adding a penalty on the size of the coefficients, encouraging smaller, more robust weights.

✅ Mathematical Formulation

1. Loss Function

The Ridge Regression objective is:

$$E_{ridge} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p} w_j^2$$

Where:

- N = number of samples
- p = number of features
- w_j = coefficient for feature j
- λ = regularization parameter controlling the strength of the penalty

- $y_i, \hat{y}_i$ = actual and predicted values

The first term is the usual Mean Squared Error (MSE), and the second term penalizes large weights.

---

2. **Closed-Form Solution**

The optimal coefficients can be calculated analytically using:

$$\mathbf{w}_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

Where:

- X = matrix of feature values
- y = vector of target values
- I = identity matrix of size p×p
- λ controls the penalty strength

Why this works:

- $X^T X$ may be singular or poorly conditioned in OLS.
- Adding λI ensures the matrix is invertible and stable.

✅ Effect of λ

- λ **= 0:** Ridge regression is equivalent to OLS.
- $\lambda \to \infty$**:** Coefficients shrink toward zero, but never exactly zero.
- As λ increases:
  - The bias of the model increases
  - The variance decreases
  - The model generalizes better but may underfit the data

✅ Geometric Interpretation

- Ridge regression adds a constraint on the magnitude of the coefficients:

$$\sum_{j=1}^{p} w_j^2 \leq C$$

  where C is a constant determined by λ.

- The solution space is constrained within a circle (or sphere in higher dimensions), forcing the coefficients to stay small.

✅ When to Use Ridge Regression

✔️ When you have **many features** and want to control overfitting

✔️ When features are **correlated**

✔️ When you don't want to eliminate any features but reduce their influence

✔️ In high-dimensional problems where inversion of $X^T X$ is unstable

---

✅ Advantages

✔️ Reduces model complexity without eliminating features

✔️ Handles multicollinearity well

✔️ Improves generalization performance

✔️ Has a closed-form solution — computationally efficient

✅ Limitations

❌ Coefficients are shrunk but never exactly zero → not ideal for feature selection

❌ Sensitive to the choice of λ → requires tuning

❌ Assumes linear relationships between features and target

✅ Example

Let’s say you have a dataset with three features and some correlation among them.

- Without regularization, OLS might assign large weights to compensate for noise or redundancy.
- Ridge regression applies the constraint:

  $$\lambda(w_1^2 + w_2^2 + w_3^2)$$

  penalizing large weights and forcing the model to find a compromise solution that performs well on both training and unseen data.

By tuning λ, you can balance between fitting the data and controlling overfitting.

✅ Choosing λ

- Usually done with **cross-validation**.
- Grid search or random search over a set of possible values.
- Evaluation metrics like **MSE** or **R^2** are used to pick the best λ.

✅ Key Takeaways

- Ridge regression improves model robustness by shrinking coefficients.
- The penalty term is quadratic, which discourages large weights but keeps all features.
- It's mathematically solvable and stable for multicollinear datasets.
- The strength of regularization is controlled by λ\lambda.
- Best suited when interpretability isn't as critical as prediction accuracy.

What is Lasso Regression?

Lasso Regression is a form of **linear regression** that includes a penalty term to prevent overfitting, similar to Ridge Regression. However, unlike Ridge, it uses an **L1 penalty** that encourages sparsity in the coefficients — meaning it can shrink some coefficients **exactly to zero**. This makes it a great tool for **feature selection**.

✅ Why Do We Need Lasso Regression?

- In datasets with many features, not all of them may be important.
- Some features might be redundant or correlated with others.
- Ordinary Least Squares (OLS) may overfit the data and assign non-zero weights to irrelevant features.
- Lasso helps by **automatically selecting** a smaller set of meaningful features.

✅ Mathematical Formulation

1. Loss Function

The objective of Lasso Regression is to minimize:

$$E_{lasso} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p}|w_j|$$

Where:

- N = number of samples
- p = number of features
- w_j = coefficient for feature jj
- λ = regularization parameter controlling the penalty
- $y_i, \hat{y}_i$ = actual and predicted values

The first term is the usual error term (MSE), and the second is the **L1 penalty**, which is the sum of the absolute values of the coefficients.

✅ Why Does L1 Penalty Lead to Sparsity?

- The L1 norm has sharp corners at zero, which means the optimization process often hits these points.
- Coefficients corresponding to irrelevant or less important features are shrunk to exactly zero.

- The optimization favors simpler models by eliminating unnecessary complexity.

✅ Geometric Interpretation

- The constraint is like a diamond (in 2D) rather than a circle (as in Ridge).
- The solution space intersects the corners of the diamond, forcing some coefficients to zero.
- This leads to a sparse representation where only key features have non-zero weights.

✅ Effect of λ

- **$\lambda = 0$:** Equivalent to OLS → all features are used.
- $\lambda \to \infty$**:** Coefficients are forced toward zero → fewer features remain.
- As λ increases, more coefficients shrink to zero → simpler models.

✅ When to Use Lasso Regression

✔️ When you want automatic **feature selection**

✔️ When dealing with **high-dimensional data**

✔️ When you suspect many features are irrelevant

✔️ When you want interpretability and sparsity in the model

✅ Closed-form Solution?

Unlike Ridge, Lasso doesn't have a closed-form solution due to the non-differentiable L1 term at zero. It requires specialized optimization methods such as:

- **Coordinate descent**
- **Least Angle Regression (LARS)**

These iterative methods efficiently solve the optimization problem.

✅ Example

Imagine you have a dataset with 100 features but only 10 are actually useful.

- OLS would assign non-zero weights to all features → noisy predictions.
- Ridge would shrink the coefficients but still keep most features.
- Lasso would shrink many coefficients to zero → only a handful of important features remain, making the model easier to interpret and more robust.

✅ Advantages

✔️ Performs feature selection and regularization simultaneously

✔️ Helps interpret models by removing irrelevant features

✔️ Reduces overfitting and improves generalization

✔️ Useful in sparse datasets where most features are noise

✅ Limitations

❌ Can struggle if features are highly correlated → it arbitrarily picks one

❌ Requires careful tuning of λ\lambda

❌ The optimization is more computationally intensive than Ridge

---

✅ Choosing λ

- Done using cross-validation techniques.
- A grid of λ values is tested and the one that minimizes validation error is chosen.
- Tools like **LassoCV** in Python can automatically find the best λ.

✅ Key Takeaways

- Lasso adds an L1 penalty, encouraging sparsity in coefficients.
- It's a great tool for feature selection in datasets with many irrelevant features.

- The choice of λ\lambda is crucial and affects how many features are retained.
- While it improves interpretability, correlated features may pose challenges.

Ridge vs Lasso – Key Differences

Feature / Aspect	Ridge Regression (L2)	Lasso Regression (L1)
Penalty	Sum of squares of coefficients: $\lambda \sum w_j^2$	Sum of absolute values of coefficients: $\lambda \sum \lvert w_j \rvert$
Effect on Coefficients	Shrinks coefficients but never sets them exactly to zero	Shrinks coefficients and can set some exactly to zero
Feature Selection	❌ Doesn't remove features	✅ Automatically selects important features
Interpretability	Lower → includes all features	Higher → simpler model with fewer features
Geometry of Constraint	Circular (smooth)	Diamond-shaped (sharp edges at axes)
Optimization	Closed-form solution exists	Requires iterative algorithms (e.g., coordinate descent)
When to Use	Multicollinearity, all features relevant	Many irrelevant features, sparse solution desired
Correlation Issues	Works well with correlated features	May arbitrarily choose among correlated features
Prediction vs Sparsity	Better prediction accuracy if features matter	Better for sparsity and interpretability

✅ When to Use Ridge Regression

Use Ridge when:

✔️ You suspect that **most or all features are important,** but want to reduce overfitting

✔️ Features are **highly correlated**, and you don't want to arbitrarily ignore some

✔️ You need to improve model generalization without necessarily removing features

✔️ You prefer a stable solution with all variables included

Example:

- A dataset with medical measurements where most features contribute to the prediction.

✅ When to Use Lasso Regression

Use Lasso when:

✔️ You expect **many irrelevant features** and want to remove them

✔️ You want a **simpler, interpretable model** with fewer predictors

✔️ You want to **perform feature selection** during model training

✔️ You are dealing with **high-dimensional data**, where the number of features is large compared to the number of samples

Example:

- A dataset with hundreds of possible customer features, where only a few are relevant.

✅ Choosing Between Ridge and Lasso

1. **If prediction is the goal** → Ridge may perform better when you want to use all features without discarding any.
2. **If interpretability is the goal** → Lasso is better for selecting a few meaningful features.
3. **If features are correlated** → Ridge is generally safer; Lasso may ignore some useful features.
4. **If the dataset is sparse or high-dimensional** → Lasso is preferable to automatically drop irrelevant features.
5. **If you want the benefits of both** → Use **Elastic Net**, which combines L1 and L2 penalties.

✅ Elastic Net – Hybrid Solution

- Combines both Ridge and Lasso:

$$E = \frac{1}{N}\sum(y_i - \hat{y}_i)^2 + \lambda_1 \sum |w_j| + \lambda_2 \sum w_j^2$$

- It balances sparsity and stability → useful when dealing with correlated features.

✅ Final Thoughts

✔️ Ridge is like “shrink but keep everything”

✔️ Lasso is like “shrink and throw out the unnecessary”

✔️ Your choice depends on the dataset, your goals (accuracy vs interpretability), and whether features are correlated.