> **Subject**: ML slides pack - regression by Claudio Sartori
>
> **Course:** Artificial Intelligence - LM
>
> **Department**: DISI (Department of Computer Science and Engineering) - University of Bologna, Italy
>
> **Author**: Roberto Zanolli
>
> **TIPS**
>
> - w.r.t means "with respect to"

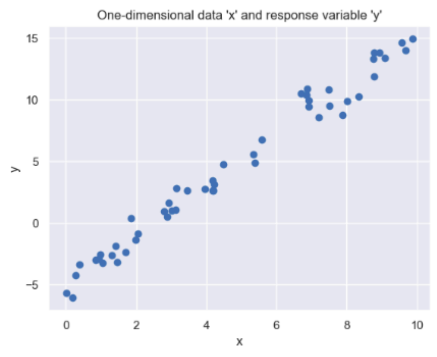
Regression Forecasting continuous values

Regression is a supervised learning task where the target variable is numeric. The objective is to minimize the prediction error with respect to the true target values.

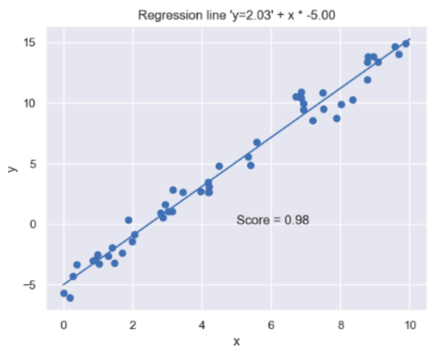
Linear regression

> Given a dataset X with N rows and D columns, where x_i is a D-dimensional data element, and a response vector y with N values y_i, we model the relationship as:
>
> $$y_i \approx w^T \cdot x_i \quad \forall i \in [1 \ldots N]$$
>
> where w is a D-dimensional vector of coefficients to be learned.

This classical statistical method dates back to 1805.



One–dimensional data and response variable



Regression and score - Score range $(-\inf : 1)$

Objective function and minimization

> The objective function minimizes the **sum of squared errors** (SSE):
>
> $$O = \sum_{i=1}^{N} (w^T \cdot x_i - y_i)^2 = ||Xw^T - y||^2 = (Xw^T - y)^T \cdot (Xw^T - y)$$
>
> Taking the gradient with respect to w:
>
> $$\nabla_w O = 2X^T(Xw^T - y)$$

Setting the gradient to zero gives the optimization condition:

$$X^T X w^T = X^T y$$

If the symmetric matrix $X^T X$ is invertible, the solution is:

$$w = (X^T X)^{-1} X^T y$$

The forecast is then:

$$y_f = X \cdot w^T$$

Matrix calculus considerations

When $X^T X$ is not invertible, alternatives include:

- Moore-Penrose pseudoinverse
- Tikonov regularization (ridge regression)
- Lasso regularization

Quality of fitting - R^2

To evaluate regression quality:

- **Mean of observed data:** $y_{avg} = \frac{1}{N} \sum_i y_i$
- **Sum of squared residuals:** $SS_{res} = \sum_i (y_i - y_{f_i})^2$
- **Total sum of squares:** $SS_{tot} = \sum_i (y_i - y_{avg})^2$
- **Coefficient of determination:** $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

The R^2 score ranges from $(-\infty, 1]$, where values closer to 1 indicate better fit.

Intuition about R^2

The R^2 coefficient **compares the fit of the chosen regression model with that of a simple horizontal straight line** (the mean model).

- With **perfect fitting**, the sum of squared residuals (SS_{res}) becomes zero, making the second term vanish and giving $R^2 = 1$
- If the **model fails to capture** the data trend, SS_{res} can reach or exceed the total sum of squares (SS_{tot}), resulting in negative R^2 values
- Despite its name, R^2 isn't actually the square of anything - it's a ratio comparing explained variance to total variance

R^2 and Mean Squared Error

Both metrics evaluate prediction errors but serve different purposes:

R^2 **characteristics**:

- Standardized index ranging from $(-\infty, 1]$
- Measures how well predictor variables explain the variation in the response variable
- Not meaningful for non-linear or non-algebraic regression models

RMSE (Root Mean Squared Error) characteristics:

- Measures the mean prediction error in absolute terms
- Influenced by the order of magnitude of the data
- More interpretable as it's in the same units as the response variable

Practical considerations:

- For comparing accuracy among different linear regression models, RMSE is often a better choice than R^2
- R^2 is preferred when you want to understand the proportion of variance explained by the model
- RMSE gives a more direct sense of prediction error magnitude

Multiple regression

When the response variable depends on two or more features, we use multiple regression. The regression technique is conceptually similar to simple linear regression but operates in higher dimensions:

- The model becomes: $y_i \approx w_1 x_{i1} + w_2 x_{i2} + \cdots + w_D x_{iD}$
- The matrix formulation and solution method remain essentially the same
- In scikit-learn, the same `LinearRegression` estimator handles both simple and multiple regression cases
- Interpretation becomes more complex as we need to consider the combined effect of multiple predictors

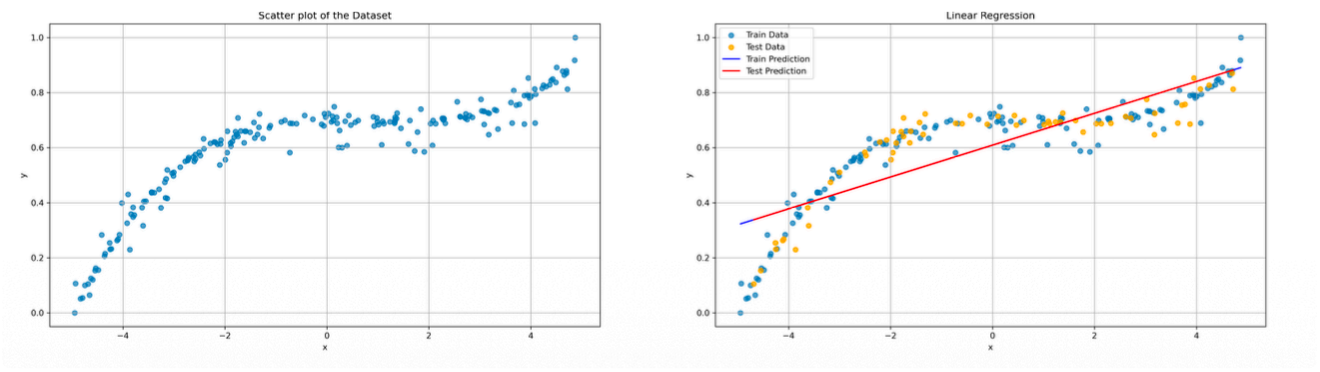
Overfitting and regularization

When working with a high number of features, overfitting becomes a significant risk. This occurs when the model performs **well on training data** but **poorly on test data** due to capturing noise rather than underlying patterns. (see decision trees notes)

Regularization addresses this by:

- Reducing the influence of less important attributes
- Penalizing complex models to prevent overfitting
- Improving generalization to unseen data

Polynomial regression (univariate)



What if the relationship between the independent variable and target isn't linear? Polynomial regression extends linear regression to model nonlinear relationships by fitting polynomial curves to the data.

Univariate polynomial regression

This approach models the relationship between input x and output $y(x)$ as an n-degree polynomial:

$$y(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon$$

Where:

- $\beta_0, \beta_1, \ldots, \beta_n$ are the model parameters to be learned
- ϵ represents the error term
- The model can capture nonlinear patterns while remaining linear in parameters

Steps in polynomial regression

Step 1: Generate polynomial features

Transform the original input variable x into higher-order polynomial terms. For degree $n = 2$, the feature matrix becomes:

$$X = [1, x, x^2]$$

This transformation extends to higher degrees ($n = 3, n = 4$, etc.), creating additional polynomial features.

Step 2: Fit a linear regression model

Despite the polynomial relationship, the problem remains linear in terms of $\beta_0, \beta_1, \ldots, \beta_n$.
We use **least squares estimation** to **minimize the sum of squared residuals**:

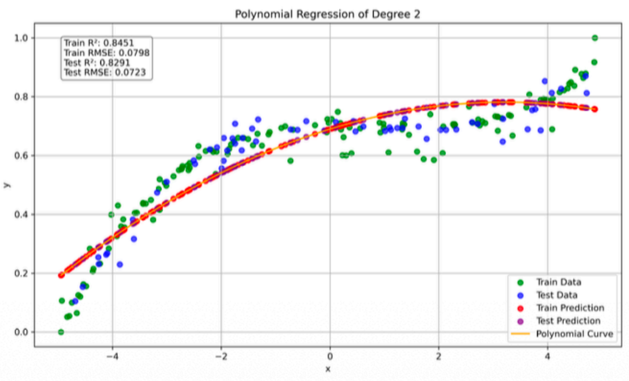
$$\min_{\beta} \sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_n x_i^n))^2$$

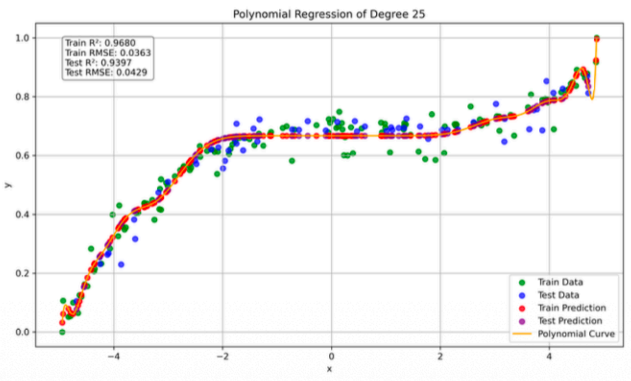
Step 3: Evaluate the model

- Use standard regression metrics like Root Mean Squared Error (RMSE)
- Control overfitting through cross-validation to determine the optimal polynomial degree
- Higher degrees can fit training data better but may generalize poorly to new data

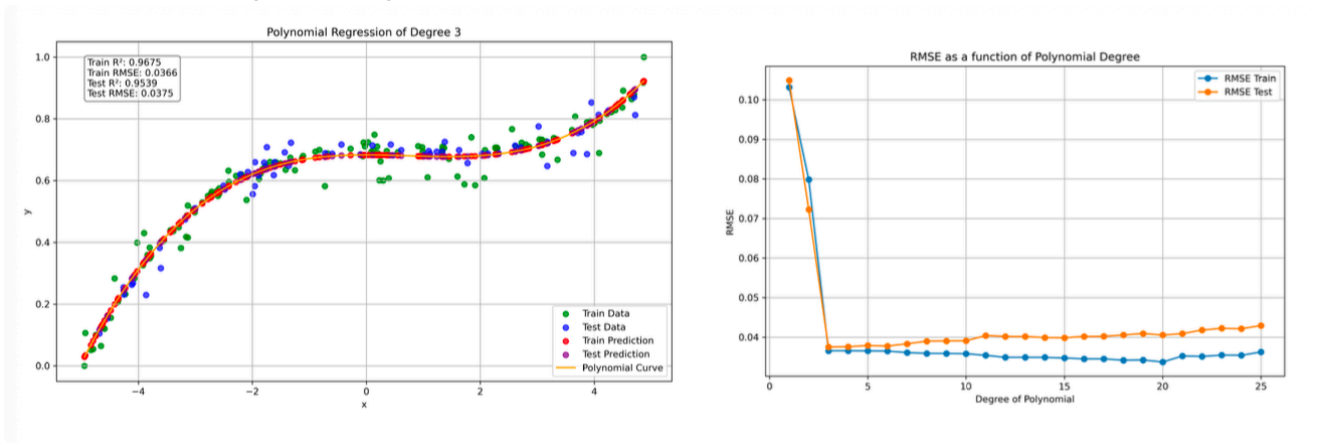
Examples

We can see a case of underfitting (left) where a degree too low fails to capture correctly the pattern and a casa of overfitting (right) where a degree too high captures all patterns including noise and uninteresting ones.





Here we can see a good fitting example



Regularized regression

Standard multivariate linear regression lacks hyperparameters to control fitting quality and ensure good performance on test data. A general approach to prevent overfitting is to simplify the model.

For linear multivariate models, simplification typically involves constraining the coefficient values to prevent them from becoming too large or complex.

Loss functions in regression

> The loss function quantifies the error between model predictions and actual values. Common regression loss functions include:
>
> **Mean Squared Error (MSE):**
>
> $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
>
> **Root Mean Squared Error (RMSE):** $\sqrt{MSE}$
>
> **Mean Absolute Error (MAE):** $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Ordinary Least Squares (OLS)

OLS is the standard, unregularized linear regression.
OLS regression **minimizes the sum of squared errors**:

$$L(w) = \sum_{i=1}^{N}(y_i - w^T x_i)^2$$

This approach finds the coefficient vector w that minimizes prediction error on the training set.

The need for regularization

While OLS minimizes training error, it risks **overfitting**, especially with:

- Many predictor variables
- Noisy data
- Multicollinearity (correlated predictors)

Regularization addresses this by penalizing model complexity. The core idea is to reduce coefficient values in various ways to find a good trade-off between:

- Accuracy on training data
- Model simplicity and generalization ability

By constraining coefficient magnitudes, **regularization helps prevent overfitting and improves performance on unseen test data**.

Lasso regression

Lasso (Least Absolute Shrinkage and Selection Operator) is a linear regression method that adds **L1-regularization** to the cost function. This approach encourages sparse models by shrinking some coefficients to exactly zero, making it particularly useful for feature selection and regularization in high-dimensional datasets.

Cost function

The Lasso regression cost function combines prediction error with an L1 penalty term:

$$L(w) = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{D} x_{ij} w_j \right)^2 + \alpha \sum_{j=1}^{D} |w_j|$$

Components:

- **Residual sum of squares**: Measures prediction error on training data
- **L1-norm penalization**: $\sum_{j=1}^{D} |w_j| = ||w||_1$ penalizes the sum of absolute coefficient values
- **Regularization parameter** α: Controls the strength of the penalty

L1 regularization mechanism

The Lasso penalty $\alpha \sum_{j=1}^{D} |w_j|$ grows linearly with coefficient magnitudes and creates a strong incentive for coefficients to become exactly zero due to:

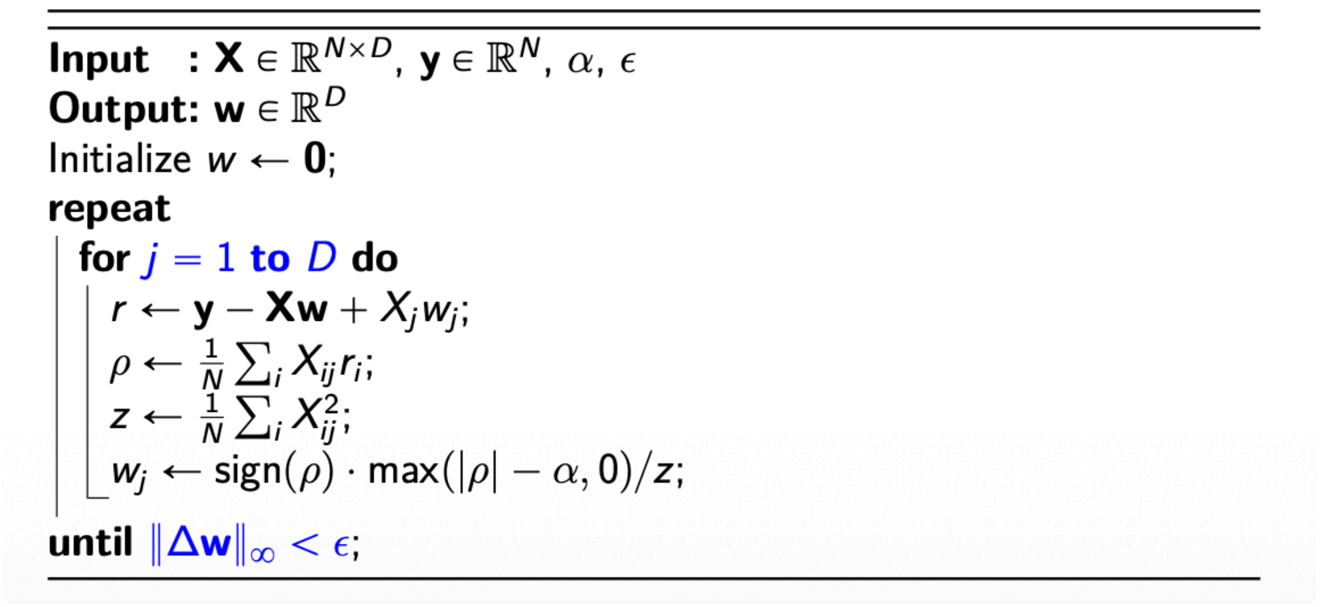
Equal penalty contribution:

- Small changes in coefficient magnitude contribute equally to the penalty regardless of whether coefficients are large or small
- This creates a "corner" in the optimization landscape that pushes coefficients to exactly zero

Efficient penalty reduction:

- When coefficients are near zero, shrinking them to zero entirely results in significant penalty reduction
- The cost to the residual sum of squares (RSS) is minimal when eliminating small coefficients

This property makes Lasso particularly effective for **feature selection**.
It automatically identifies and excludes irrelevant features by setting their coefficients to zero, creating sparse models that are more interpretable and generalize better to new data.



```
Input  : $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathbf{y} \in \mathbb{R}^{N}$, $\alpha$, $\epsilon$
Output: $\mathbf{w} \in \mathbb{R}^{D}$
Initialize $w \leftarrow \mathbf{0}$;
repeat
    for $j = 1$ to $D$ do
        $r \leftarrow \mathbf{y} - \mathbf{X}\mathbf{w} + X_j w_j$;
        $\rho \leftarrow \frac{1}{N} \sum_i X_{ij} r_i$;
        $z \leftarrow \frac{1}{N} \sum_i X_{ij}^2$;
        $w_j \leftarrow \text{sign}(\rho) \cdot \max(|\rho| - \alpha, 0)/z$;
until $\|\Delta \mathbf{w}\|_\infty < \epsilon$;
```

Soft-thresholding function in Lasso

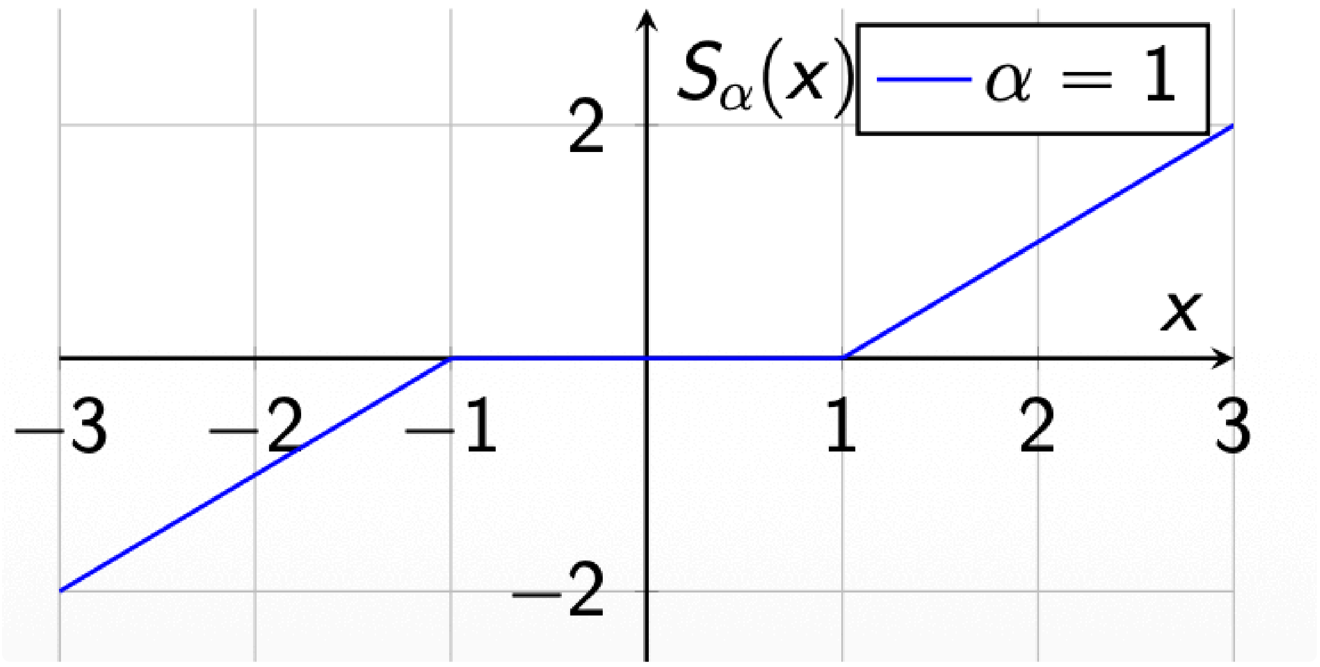
Lasso updates coefficients using the **soft-thresholding function**:

$$S_\alpha(x) = \text{sign}(x) \cdot \max(|x| - \alpha, 0)$$

This function promotes sparsity by shrinking small coefficient values to exactly zero. The mechanism works as follows:

- **For** $|x| > \alpha$: The coefficient is reduced by α in magnitude, maintaining its original sign
- **For** $|x| \leq \alpha$: The coefficient is set to exactly zero

This thresholding behavior explains why Lasso produces sparse solutions. → It removes small coefficients (the ones under treshold)



Computational complexity of Lasso

Training complexity

The computational cost depends on:

- Number of features (D)
- Number of samples (N)
- Number of iterations (T)

For **coordinate descent** optimization:

- Complexity: $O(TND)$
- For large datasets, this scales linearly with both N and D

Convergence behavior

- **Faster convergence** when many coefficients are sparse (many zeros)
- **Slower convergence** for high-dimensional dense datasets

Prediction complexity

- Linear in $\bar{D}$ (the number of nonzero coefficients)
- Since Lasso produces sparse solutions, prediction becomes very efficient

The coefficient vector w in Lasso

The vector w represents the coefficients (weights) of the linear regression model:

Structure: $w = [w_0, w_1, \ldots, w_D]$

- w_0: The intercept term (bias)
- w_j: The weight for the j-th feature, where $j = 1, \ldots, D$

Prediction: For a sample x_i, the predicted value is:

$$\hat{y}_i = w_0 + \sum_{j=1}^{D} w_j x_{ij}$$

Role of w in Lasso regression

The optimization process adjusts w to achieve two objectives:

1. **Minimize residual error**:

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

2. **Penalize large coefficients** using L1-norm regularization:

$$\alpha \sum_{j=1}^{D} |w_j|$$

Key aspects:

- α is a hyperparameter that controls the regularization strength
- Multiple α values should be tried with cross-validation

- L1-regularization encourages **sparsity** in w (many coefficients become exactly zero)
- w embodies feature importance while ensuring model simplicity and robustness

Ridge regression

Ridge regression is a **regularized linear regression** technique that adds an **L2 penalty term** to the cost function to prevent overfitting. The name derives from the matrix representation of the solution, where the regularization parameter α adds a "ridge" to the main diagonal of the $X^T X$ matrix.

Key features:

- Reduces model complexity by shrinking coefficient magnitudes
- Improves generalization performance on unseen data
- Particularly effective when dealing with multicollinearity (correlated predictors)

Ridge regression cost function

Ridge regression modifies ordinary least squares (OLS) by adding an L2 penalty:

$$L(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \alpha \|w\|_2^2$$

Where:

- α: Regularization parameter controlling penalty strength ($\alpha \geq 0$)
- $\|w\|_2^2$: Squared L2 norm of the weight vector = $\sum_{j=1}^{D} w_j^2$

Coordinate descent update for ridge regression:

$$\beta_j \leftarrow \frac{1}{1+\alpha} \cdot \left(\frac{1}{N} \sum_{i=1}^{N} X_{ij} \left(y_i - \sum_{k \neq j} X_{ik} \beta_k \right) \right)$$

This update rule reveals how ridge regression shrinks coefficients:

- The term $\frac{1}{1+\alpha}$ acts as a shrinkage factor
- When $\alpha = 0$, we get standard **OLS** update: $\beta_j \leftarrow \frac{1}{N} \sum_{i=1}^{N} X_{ij} (y_i - \sum_{k \neq j} X_{ik} \beta_k)$
- As α increases, coefficients are progressively shrunk toward zero
- The shrinkage is proportional to $\frac{1}{1+\alpha}$, so larger α values result in stronger regularization

Effects of regularization

High α values:

- Strong penalty on coefficient magnitudes
- Leads to smaller coefficient values (shrinkage toward zero)
- Reduces variance but increases bias (bias-variance tradeoff)
- More regularization, simpler model

Low α values:

- Weak penalty, resembles standard OLS regression
- Retains model variance but may overfit training data

- Less regularization, more complex model

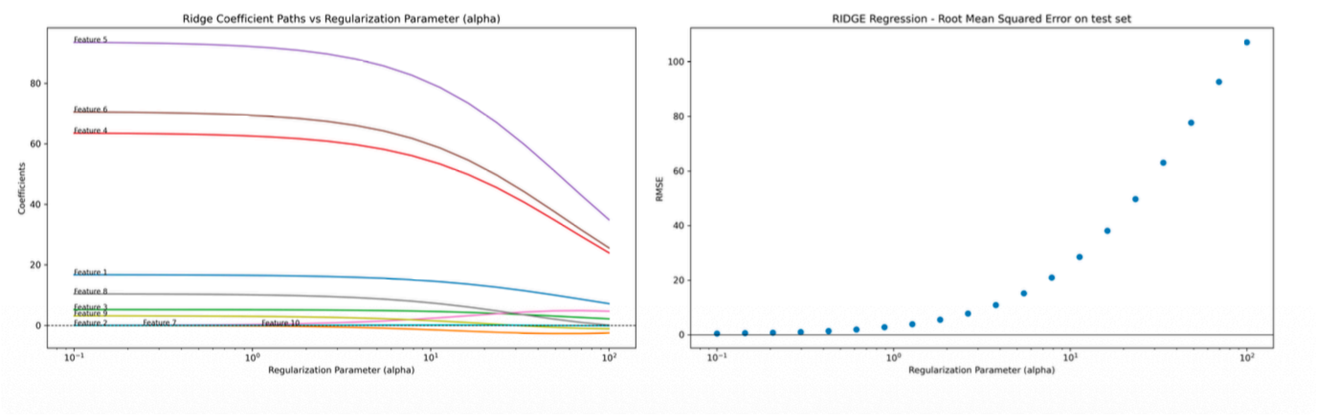
Choosing α:

- Typically selected via cross-validation
- Balances between underfitting (high α) and overfitting (low α)
- Optimal α minimizes prediction error on validation data

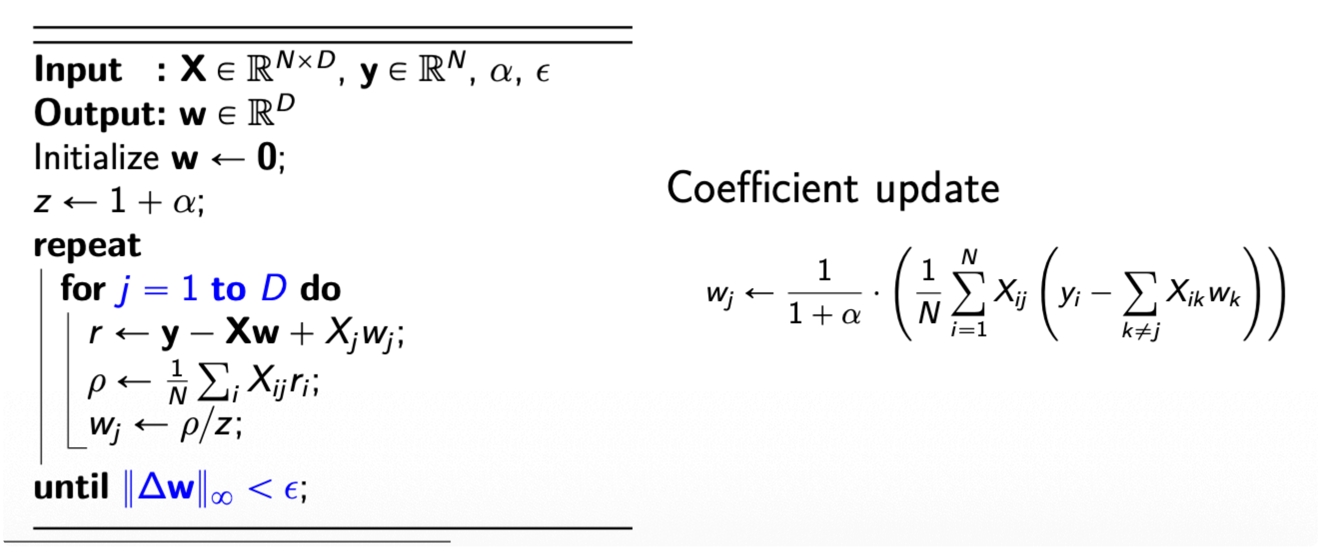
Ridge effect and RMSE

The regularization parameter α directly influences both model complexity and prediction accuracy (measured by RMSE).
As α increases:

- Coefficient magnitudes decrease (shrinkage effect)
- Model becomes less sensitive to individual data points
- RMSE on training data typically increases
- RMSE on test/validation data may initially decrease then increase (U-shaped curve)



Coordinate descent algorithm in detail (for ridge regression):



```
Input  : $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathbf{y} \in \mathbb{R}^{N}$, $\alpha$, $\epsilon$
Output: $\mathbf{w} \in \mathbb{R}^{D}$
Initialize $\mathbf{w} \leftarrow \mathbf{0}$;
$z \leftarrow 1 + \alpha$;
repeat
    for $j = 1$ to $D$ do
        $r \leftarrow \mathbf{y} - \mathbf{Xw} + X_j w_j$;
        $\rho \leftarrow \frac{1}{N} \sum_i X_{ij} r_i$;
        $w_j \leftarrow \rho / z$;
until $\|\Delta \mathbf{w}\|_\infty < \epsilon$;
```

Coefficient update

$$w_j \leftarrow \frac{1}{1+\alpha} \cdot \left(\frac{1}{N} \sum_{i=1}^{N} X_{ij} \left(y_i - \sum_{k \neq j} X_{ik} w_k \right) \right)$$

Elastic net regression

Elastic net regression is a **linear regression** method that combines penalties from both ridge regression and lasso regression. This hybrid approach addresses limitations of using either method alone.

Why elastic net?

Elastic net was developed to overcome specific limitations of ridge and lasso regression:

- **Ridge limitation**: Cannot perform feature selection (coefficients are shrunk but never reach exactly zero)
- **Lasso limitation**: Struggles when features are highly correlated (tends to select only one feature from a correlated group)

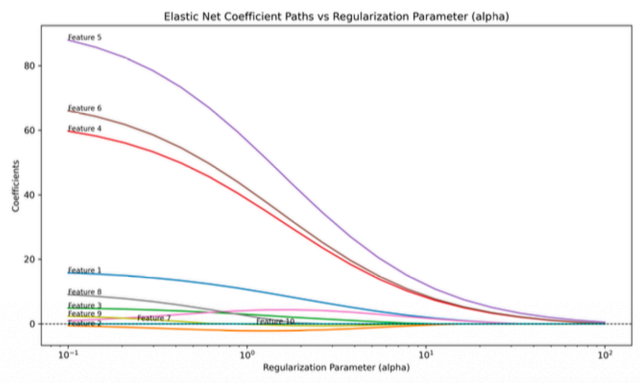
By combining both penalties, elastic net offers a balanced approach that:

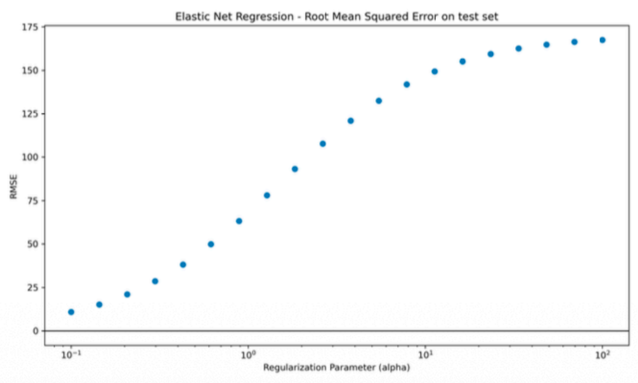
- Performs feature selection like lasso
- Handles correlated features better than lasso
- Provides the stability of ridge regression

The elastic net cost function

> The method starts with the ordinary least squares (OLS) cost function:
>
> $$L(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$
>
> Elastic net modifies OLS by adding both L1 and L2 penalty terms:
>
> $$L(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \alpha_1 \|w\|_1 + \alpha_2 \|w\|_2^2$$
>
> Where:
>
> - α_1: Controls the strength of the Lasso penalty (L1 norm)
> - α_2: Controls the strength of the Ridge penalty (L2 norm)
> - $\|w\|_1 = \sum_{j=1}^{D} |w_j|$ is the L1 norm (sum of absolute coefficients)
> - $\|w\|_2^2 = \sum_{j=1}^{D} w_j^2$ is the squared L2 norm (sum of squared coefficients)

This combined penalty structure allows elastic net to leverage the benefits of both regularization approaches simultaneously, providing a flexible framework for handling various data scenarios while maintaining good predictive performance and model interpretability.





Coordinate descent algorithm in detail (for elastic net):

```
Input  : $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathbf{y} \in \mathbb{R}^N$, $\alpha$, $\eta$, $\epsilon$
Output: $w \in \mathbb{R}^D$
Initialize $w \leftarrow \mathbf{0}$;
repeat
    for $j = 1$ to $D$ do
        $r \leftarrow \mathbf{y} - \mathbf{X}w + X_j w_j$;
        $\rho \leftarrow \frac{1}{N} \sum_i X_{ij} r_i$;
        $z \leftarrow \frac{1}{N} \sum_i X_{ij}^2 + \alpha(1 - \eta)$;
        $w_j \leftarrow \mathrm{sign}(\rho) \cdot \max(|\rho| - \alpha\eta, 0)/z$;
until $\|\Delta w\|_\infty < \epsilon$;
```

Elastic net: pros

Properties:

- Encourages sparse coefficients (like Lasso)
- Groups correlated features (like Ridge)

Advantages:

- Handles multicollinear data effectively
- Balances feature selection with model stability
- Useful for high-dimensional datasets

Comparison of regularized regression techniques

Lasso, Ridge, and Elastic Net are regularization methods that address overfitting and multicollinearity by adding penalties to the cost function.

Lasso regression

Strengths:

- Performs feature selection by setting coefficients to zero
- Useful for high-dimensional datasets with many irrelevant features
  Limitations:
- Struggles with highly correlated predictors

Use cases:

- Genomics: Identifying relevant genes for diseases
- Text processing: Selecting keywords in sentiment analysis
- Sensor networks: Identifying critical sensors in IoT

Ridge regression

Strengths:

- Handles multicollinearity by shrinking coefficients
- Retains all predictors
  Limitations:
- No feature selection capability

Use cases:

- Finance: Predicting stock prices with correlated indicators
- Marketing: Modeling customer demand
- Medical imaging: Analyzing high-dimensional MRI/CT data

Elastic Net regression

Strengths:

- Combines Lasso and Ridge benefits
- Selects groups of correlated features
  Limitations:
- Requires tuning two parameters (α_1 and α_2)

Use cases:

- Genomics: Selecting gene groups
- Healthcare: Modeling patient outcomes
- Climate science: Modeling correlated climate variables

Comparison table

Feature	Lasso	Ridge	Elastic Net
Feature Selection	Yes	No	Yes
Groups Correlated Features	No	Yes	Yes
Handles Multicollinearity	Weak	Strong	Strong
Model Interpretability	High (sparse coefficients)	Moderate	Moderate (sparse, grouped features)
Dataset Characteristics	High-dimensional, sparse predictors	Correlated predictors	Sparse and correlated predictors

Explanation

Sparsity: Elastic Net sets some coefficients to zero like Lasso, removing irrelevant predictors for simpler, more interpretable models.

Groups features: When predictors are correlated, Elastic Net selects them together rather than arbitrarily choosing one, shrinking coefficients toward each other using Ridge-like penalties.

This combines L1 penalty (Lasso) for sparsity with L2 penalty (Ridge) for multicollinearity handling.