Minor in AI

Revising Regularization

From Overfitting to Smart Predictions

April 29, 2025

1 The Exam Score Prediction Challenge

Picture this: you've been diligently logging your study time each day, alternating between focused solo sessions and collaborative group discussions. You believe both contribute to your understanding, but you're not sure how much each hour of effort really pays off in terms of your final exam mark. To make this concrete, you gather data from the last six quizzes:

Individual Study (hrs)	**Group Study (hrs)**	**Score**
1	3	56
2	3	60
3	3	65
4	3	66
5	4	75
6	4	87

At first glance, there's a clear upward trend: more study time generally leads to higher scores. You decide to harness this pattern by fitting a simple linear regression model:

$$\widehat{Y} \;=\; w_1 \times (\text{individual hours}) \;+\; w_2 \times (\text{group hours}) \;+\; b$$

With six data points, the regression engine calculates coefficients that minimize overall error, and when you feed in a new pair—7 hours of solo study and 3 hours of group work—you obtain a prediction of 80.25 marks.

This numeric answer feels impressive, but it raises a critical question: can you trust that prediction to hold true in practice? Real student performance rarely follows a perfectly straight line. Maybe your solo study at hour seven is less effective due to fatigue, or perhaps group sessions beyond a certain length become more social than scholarly. If you lean solely on an unadjusted regression, you risk *over-enthusiasm*—believing your model's high-precision math captures every subtlety of human learning.

Enter **regularization**. By introducing a small penalty on the magnitude of your model weights w_1 and w_2, you intentionally constrain their values. This prevents the model from "over-reacting" to noise in your six sample points and yields a more conservative, robust prediction for that 7+3 study scenario. In the next section, we'll explore how Lasso and Ridge regularization formulas reshape our model's behavior and establish a safeguard against overly confident forecasts.

2 The Science of Smart Predictions

2.1 Linear Regression Basics

Before we regularize, let's recap how a plain *linear regression* models our two study inputs:

$$\hat{y} \;=\; w_1\, x_1 \;+\; w_2\, x_2 \;+\; b \quad \text{where} \quad \begin{cases} x_1 = \text{individual study hours}, \\ x_2 = \text{group study hours}, \\ w_1, w_2 = \text{learned weights}, \\ b = \text{intercept (base score)}. \end{cases} \tag{1}$$

In machine-learning terms, we seek the values of w_1, w_2, and b that minimize the *mean squared error* between our predicted scores $\hat{y}$ and the true scores y.

```
// 1. Load the data into a DataFrame
import pandas as pd
data = {
    'individual_study_hours': [1, 2, 3, 4, 5, 6],
    'group_study_hours':      [3, 3, 3, 3, 4, 4],
    'marks_scored':           [56, 60, 65, 66, 75, 87]
}
df = pd.DataFrame(data)              # Tabular dataset of study hours vs. marks

// 2. Prepare feature matrix X and target vector y
X = df[['individual_study_hours', 'group_study_hours']]  # shape: (6, 2)
y = df['marks_scored']                                   # shape: (6,)

// 3. Fit a linear regression model
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(X, y)  # Finds optimal w1, w2, and b by least squares

// 4. Inspect learned parameters
print(f"Coefficients: {lr.coef_}, Intercept: {lr.intercept_}")
```

Listing 1: Linear Regression Implementation

The model outputs:

$$w_1 = 4.27, \quad w_2 = 6.43, \quad b = 31.77,$$

meaning:

- Each extra hour of *individual* study adds about 4.27 marks.
- Each extra hour of *group* study adds about 6.43 marks.
- Even with zero study, you'd expect a baseline score of 31.77 (e.g. easy "guaranteed" marks).

2.2 The Overfitting Problem

The model fits our six data points nearly perfectly ($R^2 = 0.96$), but its *prediction* for $(x_1 = 7, x_2 = 3)$ is:

$$\hat{y} \;=\; 4.27\times 7 \;+\; 6.43\times 3 \;+\; 31.77 \;=\; 80.25.$$

While mathematically sound, this estimate may be *overconfident*:

- **Noise vs. signal:** Small quirks in our six quiz scores (e.g. an easier quiz or random group chatter) can unduly influence w_1 and w_2.
- **Extrapolation risk:** We only saw up to 6 individual hours. Pushing to 7 may lie outside our "experienced" range.
- **Human factors:** Fatigue, diminishing returns, and social distractions mean the real effect of that seventh hour may differ.

In other words, our model has *overfit* the training data by capturing noise as if it were a true pattern. In the next section, we introduce **regularization**—a controlled way to shrink w_1 and w_2 so that predictions remain robust, even when venturing beyond the original data.

3 Regularization: The Reality Check

In practice, a perfectly fitted line can be too eager—it captures quirks in our six quiz scores and treats them as if they were certainties. Regularization gently reins in this enthusiasm by adding a small "tax" on large weights. We will examine two popular techniques, *Lasso* and *Ridge*, that serve as guardrails against overfitting.

3.1 Two Warriors Against Overfitting

Key Concept

Lasso (L1) applies an absolute-value penalty that can drive unimportant weights exactly to zero, effectively removing features.
Ridge (L2) applies a squared-value penalty that shrinks all weights toward zero but never eliminates them completely.

Both methods modify our original *mean squared error* loss by adding a penalty term:

$$\text{Loss} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{\text{MSE}} + \underbrace{\alpha \sum_{i} |w_i|}_{\text{Lasso penalty}} \quad \text{or} \quad \underbrace{\alpha \sum_{i} w_i^2}_{\text{Ridge penalty}} .$$

Here, α is the *regularization strength*—a dial you turn up to penalize complexity more heavily.

3.2 Regularization in Code

```
// We assume X and y are already defined as before.

// 1. Lasso Regression: L1 penalty tends to zero out small weights.
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=1.0)    # alpha=1 controls the strength of L1
   penalty
lasso.fit(X, y)             # fit model to the same data
print(f"Lasso Coefficients: {lasso.coef_}")
print(f"Lasso Intercept:    {lasso.intercept_}")

// 2. Ridge Regression: L2 penalty shrinks weights but keeps all
   features.
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1.0)    # alpha=1 controls the strength of L2
   penalty
ridge.fit(X, y)             # fit model to the same data
print(f"Ridge Coefficients: {ridge.coef_}")
print(f"Ridge Intercept:    {ridge.intercept_}")
```

Listing 2: Regularization Implementation

In each case the `alpha` parameter balances fit versus simplicity. A higher α forces the model to prefer smaller w_i, even at the cost of slightly higher training error.

3.3 Interpreting the Penalties

Lasso Penalty (L1):

$$\alpha \sum_i |w_i|$$

This term grows in direct proportion to a weight's absolute size, encouraging many weights to collapse to zero when α is large enough—ideal if you suspect some features carry no real signal.

Ridge Penalty (L2):

$$\alpha \sum_i w_i^2$$

This term grows faster for larger weights (quadratically), gently pulling all weights closer to zero but rarely eliminating them outright—useful when you believe every feature contributes something, however small.

3.4 Results Comparison

After fitting each model with $\alpha = 1$, we obtain:

	Linear	**Lasso**	**Ridge**
Individual Weight w_1	4.27	5.58	4.12
Group Weight w_2	6.43	0.00	5.87
Intercept b	31.77	42.33	33.15
Prediction (7,3)	80.25	74.47	77.01

A few observations:

- **Lasso** has driven the group-study weight w_2 to zero, effectively asserting that only solo study matters under its simplicity bias.
- **Ridge** retains both weights but shrinks them slightly, reflecting a belief that both factors count, albeit less aggressively.
- Consequently, the predicted score for (7,3) drops from 80.25 (unregularized) to 74.47 with Lasso and 77.01 with Ridge—more conservative estimates that guard against over-exuberant extrapolation.

By choosing an appropriate α, you tune how much you trust your limited data versus how strongly you favor a simpler, more generalizable model. In the next section, we'll discuss strategies for selecting α and evaluating model performance on unseen data.
Lasso completely ignores group study hours, while Ridge reduces both weights moderately.

4 Why This Matters in the Real World

Regularization is not just an academic exercise—it keeps models honest in high-stakes applications:

Medical Diagnosis In healthcare, we often collect dozens of measurements (blood tests, imaging features, patient history). Without regularization, a model might latch onto spurious correlations (e.g. a lab value that happens to be high in a few sick patients purely by chance). By penalizing large coefficients, Lasso or Ridge forces the model to focus on truly predictive symptoms and discard noise, reducing the risk of misdiagnosis.

Stock Market Prediction Financial markets are notoriously noisy: price movements can reflect sentiment swings, regulatory news, or random fluctuations. A purely unregularized model might "discover" patterns that vanish as soon as the next trading day arrives. Regularization dampens these overfitted signals, producing more stable estimates that generalize better to new market data.

E-commerce Recommendations Retail platforms may track dozens of customer features (browsing time, click rates, past purchases, demographics). A model that overweights every tiny signal will recommend implausible products—think umbrellas to insomniacs or snow boots to beachgoers. By shrinking or eliminating weak predictors, regularization yields more realistic, explainable recommendations.

5 Key Takeaways

Regularization prevents overfitting: it penalizes complex models that memorize noise rather than learn true patterns.
Lasso (L1) encourages sparsity: by driving small weights to zero, it performs implicit feature selection, simplifying the model.
Ridge (L2) ensures stability: by shrinking all weights, it retains every feature but controls their influence to avoid extreme extrapolations.
The hyperparameter α governs balance: larger α increases penalty strength, favoring simpler models, while smaller α allows more flexibility.
Always cross-check with domain knowledge: no amount of math beats a reality check—ensure your predictions make sense in context before deployment.

6 The Art of Balanced Modeling

Regularization teaches our models the virtue of *restraint*. Just as an expert tutor knows when to guide a student and when to let them explore, a well-regularized algorithm learns to distinguish signal from noise. Whether forecasting exam scores, diagnosing diseases, or tuning investment strategies, these techniques help ensure our predictions remain both accurate and trustworthy in the unpredictable complexity of the real world.

Additional Resources

- Notes