Week 5: Machine Learning & Regularization for Portfolio Optimization

MSc Banking and Finance – FinTech Course

October 7, 2025

Abstract

This document provides comprehensive coverage of Ridge and Lasso regression techniques applied to portfolio optimization. Building on Week 4's ARIMA and GARCH forecasts, we introduce regularization methods to prevent overfitting, perform feature selection, and optimize portfolio weights. Topics include L1/L2 penalties, cross-validation, prompt-assisted feature engineering, and practical applications in cryptocurrency portfolio management.

Contents

1 Introduction & Motivation

1.1 Why Do We Need Regularization?

In Week 4, we built **ARIMA** and **GARCH** models to forecast:

- **Returns** (hard to predict – markets are efficient!)
- **Volatility** (more predictable – clustering patterns exist)

Week 5 Question: Can we use these forecasts as **features** in a machine learning model to optimize portfolio weights?

Problem: With many features, models can **overfit** – they memorize noise instead of learning patterns.

Solution: Regularization – penalize complex models to prevent overfitting.

2 The Overfitting Problem

2.1 What is Overfitting?

Imagine you have 50 features but only 100 data points:

- Model fits training data **perfectly** (100% accuracy)
- Model fails on new data **miserably** (poor generalization)

Why? The model memorized noise, not true patterns.

2.2 Bias-Variance Tradeoff

The total prediction error can be decomposed as:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error} \tag{1}$$

- **High Bias:** Model too simple (underfitting)
- **High Variance:** Model too complex (overfitting)
- **Goal:** Find the sweet spot!

Key Insight

Regularization increases bias slightly to dramatically reduce variance, leading to better generalization on unseen data.

3 Ridge Regression (L2 Regularization)

3.1 Standard Linear Regression

Standard OLS minimizes mean squared error:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

Problem: With many features, coefficients can become **very large**.

3.2 Ridge Solution

Ridge regression adds an L2 penalty:

$$\min_w \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} w_j^2 \right\} \tag{3}$$

where:

- α (alpha): Regularization strength
  - $\alpha = 0 \rightarrow$ Standard regression
  - $\alpha = \infty \rightarrow$ All coefficients $\rightarrow 0$
- Penalty: Sum of **squared** coefficients (L2 norm: $||w||_2^2$)

3.3 What Ridge Does

1. **Shrinks coefficients** toward zero (but never exactly zero)
2. **Keeps all features** (no feature selection)
3. **Reduces variance** at the cost of small bias
4. **Works well** when all features are somewhat useful

3.4 Mathematical Intuition

Ridge adds a "cost" to having large coefficients:

- Large $w_j \rightarrow$ High penalty $\rightarrow$ Model prefers smaller coefficients
- Prevents any single feature from dominating predictions

3.5 Choosing α (Hyperparameter Tuning)

Use **cross-validation** to find optimal α:

```
alphas = [0.001, 0.01, 0.1, 1, 10, 100]
for alpha in alphas:
    model = Ridge(alpha=alpha)
    cv_score = cross_val_score(model, X, y, cv=5)
    # Pick alpha with best CV score
```

4 Lasso Regression (L1 Regularization)

4.1 Lasso Solution

Lasso (Least Absolute Shrinkage and Selection Operator) uses an L1 penalty:

$$\min_w \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} |w_j| \right\} \tag{4}$$

- **Penalty:** Sum of **absolute** coefficients (L1 norm: $||w||_1$)
- **Key difference:** Uses absolute value, not squared

4.2 What Lasso Does

1. **Shrinks coefficients** toward zero
2. **Sets some coefficients exactly to zero** $\star$
3. **Automatic feature selection** (sparse solution)
4. **Works well** when only a few features are truly important

4.3 Why Lasso Creates Sparsity

The L1 penalty creates "corners" at zero:

- Optimization often lands **exactly on zero**
- Ridge's L2 penalty is "smooth" – rarely hits zero exactly

Geometric Intuition:

- L1 constraint: Diamond shape (has corners)
- L2 constraint: Circle shape (smooth, no corners)
- Solution where constraint meets error contours $\rightarrow$ Lasso hits corners (zeros)!

4.4 Lasso for Feature Selection

After fitting Lasso with $\alpha = 0.1$:

$$\begin{aligned} &\text{Feature 1:} \quad w = 0.50 \quad \checkmark \text{ Important} \\ &\text{Feature 2:} \quad w = 0.00 \quad \times \text{ Removed} \\ &\text{Feature 3:} \quad w = 0.23 \quad \checkmark \text{ Important} \\ &\text{Feature 4:} \quad w = 0.00 \quad \times \text{ Removed} \end{aligned}$$

Only features with non-zero coefficients are kept!

5 Ridge vs Lasso: Comparison

Aspect	**Ridge (L2)**	**Lasso (L1)**
Penalty	$\alpha \sum w_j^2$	$\alpha \sum \lvert w_j \rvert$
Coefficients	Shrunk, never exactly 0	Some exactly 0 (sparse)
Feature Selection	No	Yes (automatic)
Best When	All features useful	Many irrelevant features
Multicollinearity	Handles well	Picks one, zeros others
Interpretation	Harder	Easier (fewer features)

Table 1: Ridge vs Lasso Comparison

5.1 ElasticNet: Best of Both Worlds

ElasticNet combines L1 + L2 penalties:

$$\min_w \left\{ \text{MSE} + \alpha_1 \sum |w_j| + \alpha_2 \sum w_j^2 \right\} \tag{5}$$

- Mix of Lasso's sparsity + Ridge's stability
- Two hyperparameters to tune: α_1 and α_2

6 Feature Engineering with Prompt-Assisted AI

6.1 What is Feature Engineering?

Transforming raw data into meaningful features that help models learn better.

6.2 Week 5 Features (Building on Week 4)

6.2.1 From Week 4 Forecasts:

1. `BTC_return_forecast` – ARIMA prediction
2. `BTC_volatility_forecast` – GARCH prediction
3. `ETH_return_forecast`, `ETH_volatility_forecast`
4. `DOGE_return_forecast`, `DOGE_volatility_forecast`

6.2.2 Engineered Features:

1. Risk-Adjusted Returns (Sharpe-like)

$$\text{risk\_adj\_return} = \frac{\text{return\_forecast}}{\text{volatility\_forecast}} \tag{6}$$

Why? High return with low volatility = better investment!

2. Momentum Indicators

$$\text{momentum} = \text{MA}_7(\text{return\_forecast}) \tag{7}$$

Why? Captures recent trends (momentum trading strategy of the previous 7-days)

3. Volatility Regime

$$\text{high\_vol} = 1\{\text{volatility} > \text{median}(\text{volatility})\} \tag{8}$$

Why? Different strategies for calm vs volatile markets

4. Cross-Asset Features

$$\text{BTC\_ETH\_spread} = R_{\text{BTC}} - R_{\text{ETH}} \tag{9}$$

Why? Relative performance matters for diversification

6.3 Prompt Engineering for Features

Example Prompt to AI

"Given crypto return and volatility forecasts, suggest 5 features that capture:

1. *Risk-adjusted performance*
2. *Market momentum*
3. *Cross-asset relationships*
4. *Volatility regimes*
5. *Mean reversion signals"*

AI might suggest:

- Sharpe ratio proxies
- Correlation breakdowns
- Volatility z-scores
- Price/moving average ratios

Then test these features in your model!

7 Cross-Validation (CV)

7.1 Why Cross-Validation?

Problem: Training score is overly optimistic (model has "seen" the data)
Solution: Test on **unseen** data using K-Fold CV

7.2 K-Fold Cross-Validation Process

Data split into $K = 5$ folds:

```
[Fold1][Fold2][Fold3][Fold4][Fold5]

Iteration 1: [Test ][Train][Train][Train][Train] → Score
Iteration 2: [Train][Test ][Train][Train][Train] → Score
Iteration 3: [Train][Train][Test ][Train][Train] → Score
Iteration 4: [Train][Train][Train][Test ][Train] → Score
Iteration 5: [Train][Train][Train][Train][Test ] → Score

Final Score = Average(Score, Score, ..., Score)
```

7.3 Implementation

```python
from sklearn.model_selection import KFold, cross_val_score

kfold = KFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(model, X, y, cv=kfold,
                         scoring='neg_mean_squared_error')

avg_mse = -scores.mean()
```

8 Portfolio Application

8.1 Pipeline: Week 4 → Week 5

```
Week 4 Output:
 ARIMA forecasts (returns)
 GARCH forecasts (volatility)
        ↓
Week 5 Feature Engineering:
 Risk-adjusted returns
 Momentum indicators
 Volatility regimes
 Cross-asset features
        ↓
Week 5 ML Models:
 Ridge: Uses ALL features
 Lasso: Selects IMPORTANT features
        ↓
Portfolio Optimization:
 Optimal weights (BTC, ETH, DOGE)
```

8.2 Objective Function

We want to predict portfolio return:

$$R_{\text{portfolio}} = w_1 R_1 + w_2 R_2 + w_3 R_3 \tag{10}$$

Where:

- R_i = Return of asset i
- w_i = Weight of asset i
- Constraint: $\sum w_i = 1$ and $w_i \geq 0$ (no short selling)

8.3 Using Lasso Predictions for Weights

1. **Train Lasso** to predict portfolio return using features
2. **Extract important features** (non-zero coefficients)
3. **Calculate risk-adjusted returns** from forecasts
4. **Optimize weights** using softmax:

$$w_i = \frac{\exp(\text{RiskAdjReturn}_i)}{\sum_j \exp(\text{RiskAdjReturn}_j)} \tag{11}$$

This ensures:

✓ All weights sum to 1

✓ All weights ≥ 0

✓ Higher risk-adjusted returns → higher weights

9 Practical Examples

9.1 Example 1: Ridge vs No Regularization

Scenario: 20 features, 50 samples

```
# No regularization
lr = LinearRegression()
lr.fit(X_train, y_train)
train_score = lr.score(X_train, y_train)  # 0.95 (great!)
test_score = lr.score(X_test, y_test)     # 0.45 (terrible!)
#     Overfitting!

# Ridge regularization
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
train_score = ridge.score(X_train, y_train)  # 0.82 (lower)
test_score = ridge.score(X_test, y_test)     # 0.78 (much better!)
#     Good generalization!
```

9.2 Example 2: Lasso Feature Selection

Scenario: 50 features, only 5 are useful

```
lasso = Lasso(alpha=0.1)
lasso.fit(X, y)

# Check coefficients
n_selected = np.sum(lasso.coef_ != 0)
print(f"Features selected: {n_selected}/50")  # Output: 7/50
```

10 Key Takeaways

1. **Regularization prevents overfitting** by penalizing model complexity
2. **Ridge (L2):** Shrinks all coefficients, keeps all features
3. **Lasso (L1):** Automatic feature selection (sparse solution)
4. **Feature engineering is crucial** – AI prompts can help discover patterns
5. **Cross-validation is mandatory** for honest performance evaluation
6. **Portfolio optimization:** Use ML predictions + constraints for optimal weights

11 Further Reading

11.1 Books

- James et al. (2023), *An Introduction to Statistical Learning* (Python ISLP) – Chapter 6
- Hastie et al. (2009), *The Elements of Statistical Learning* – Section 3.4

11.2 Papers

- Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," *JRSS-B*
- Zou, H., & Hastie, T. (2005), "Regularization and Variable Selection via the Elastic Net"

11.3 Applications

- DeMiguel et al. (2009), "Optimal Versus Naive Diversification"
- Feng et al. (2020), "Taming the Factor Zoo," *Journal of Finance*

12 Connection to Previous Weeks

Week	Topic	Connection to Week 5
Week 1	Data Collection	Raw price data → Returns
Week 2	Data Cleaning	Handle missing values for features
Week 3	Econometrics	Understand factor models, regression
Week 4	Time Series	ARIMA/GARCH forecasts = Week 5 features
Week 5	ML Regularization	Ridge/Lasso for portfolio optimization
Week 6	Tree Ensembles	Non-linear extensions (next week!)

Table 2: Course Integration

13 Pro Tips for Students

1. **Always standardize features** before Ridge/Lasso (scale matters!)
2. **Start with Lasso** for feature selection, then try Ridge
3. **Use log scale** when searching for alpha (try 0.001, 0.01, 0.1, 1, 10...)
4. **Domain knowledge > Algorithms** – good features beat complex models
5. **Validate, validate, validate** – never trust training scores alone!

Next Week: Tree Ensembles (Random Forest, XGBoost) for non-linear patterns!