MACHINE LEARNING AND NEURAL NETWORKS ASSIGNEMENT

By Sonia Suvarna Dokala

GitHub Repository link: https://github.com/Dokalasoniasuvarna

L1 and L2 Regularization in Logistic Regression: A Comprehensive Guide

Summary

Regularization stands as one of the most powerful techniques for improving machine learning model generalization. When models become overly complex relative to the training data, they tend to memorize noise and patterns specific to the training set, resulting in poor performance on unseen data. L1 and L2 regularization provide complementary approaches to addressing this fundamental challenge. L2 regularization (Ridge) applies to a quadratic penalty on feature weights, encouraging smooth, distributed shrinkage across all features while maintaining non-zero weights. L1 regularization (Lasso) applies an absolute value penalty that drives many feature weights to exactly zero, providing automatic feature selection alongside regularization. This tutorial provides a comprehensive exploration of both techniques, comparing their mathematical formulations, geometric interpretations, empirical performance, and practical applications. Using logistic regression as the foundational model, we demonstrate how L1 and L2 regularization affect decision boundaries, weight distributions, convergence properties, and classification performance through detailed analysis and visualization. The work establishes theoretical understanding while emphasizing practical insights for algorithm selection and hyperparameter tuning in real-world machine learning applications.

1. Introduction to Regularization in Machine Learning

Machine learning models aim to discover general patterns in training data that extend to new, unseen data. However, models can succeed too well at matching training data, memorizing specific noise and idiosyncratic patterns that do not generalize. This phenomenon, called overfitting, plagues practitioners across domains. A model that achieves perfect training accuracy may perform poorly on test data.

Regularization techniques combat overfitting by constraining model complexity. Rather than optimizing purely to fit training data, regularization adds a penalty term that increases with model complexity. This penalty-augmented objective balances fitting training data with maintaining simplicity, encouraging models to discover fundamental patterns rather than noise.

The fundamental regularized objective takes the form:

$$\text{Loss} = \text{Data Fit} + \lambda \cdot \text{Penalty}$$

where λ (lambda) is a hyperparameter controlling the regularization strength. A larger λ emphasizes simplicity over fit, while $\lambda = 0$ recovers the unregularized objective. The challenge lies in selecting appropriate λ to optimize the bias-variance tradeoff.

The two dominant regularization approaches differ in how they define the penalty:

- **L2 Regularization (Ridge)**: Penalty proportional to the sum of squared weights
- **L1 Regularization (Lasso)**: Penalty proportional to the sum of absolute values of weights

While conceptually similar, these approaches have dramatically different effects on weight distributions, feature selection, and practical performance.

2. L2 Regularization (Ridge Regression/Logistic Regression)

L2 regularization, also known as Ridge regression in the context of linear models, applies a quadratic penalty on feature weights.

2.1 Mathematical Formulation

For logistic regression with L2 regularization, the objective function becomes:

$$J = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}i)] + \frac{\lambda}{2} \sum{j=1}^{p} w_j^2$$

The first term is the logistic loss measuring fit to training data. The second term, $\frac{\lambda}{2} \sum_{j=1}^{p} w_j^2$, is the L2 penalty. This penalty increases quadratically with weight magnitude, heavily penalizing large weights while moderately penalizing small weights.

The gradient of the L2 penalty with respect to each weight is:

$$\frac{\partial}{\partial w_j} \left(\frac{\lambda}{2} w_j^2 \right) = \lambda w_j$$

This linear gradient means that the penalty term is proportional to the weight itself. During gradient descent optimization, each weight experiences a regularizing force proportional to its current magnitude, pulling weights toward zero.

2.2 Effect on Weight Distribution

The quadratic penalty has a crucial consequence: weights are shrunk proportionally, but never driven to exactly zero. Even under strong regularization, weights remain non-zero, though small.

Key insight: L2 regularization performs smooth, distributed shrinkage. If a feature has large initial weight, it experiences large regularizing force, shrinking rapidly. If a feature has small initial weight, it experiences small regularizing force, shrinking slowly. All features remain in the model.

2.3 Geometric Interpretation

L2 regularization can be visualized geometrically as a constraint in weight space. The penalty term \sum_{j=1}^{p} w_j^2 defines a circle (or sphere in higher dimensions) centered at the origin. Minimizing the regularized objective is equivalent to minimizing training loss subject to the constraint that weights lie within this circle.

The optimization solution is found where the training loss contours are tangent to this constraint circle. The radius of the constraint circle is inversely related to \lambda ”stronger regularization (larger \lambda) enforces a smaller circle, pushing weights closer to zero.

2.4 Decision Boundaries

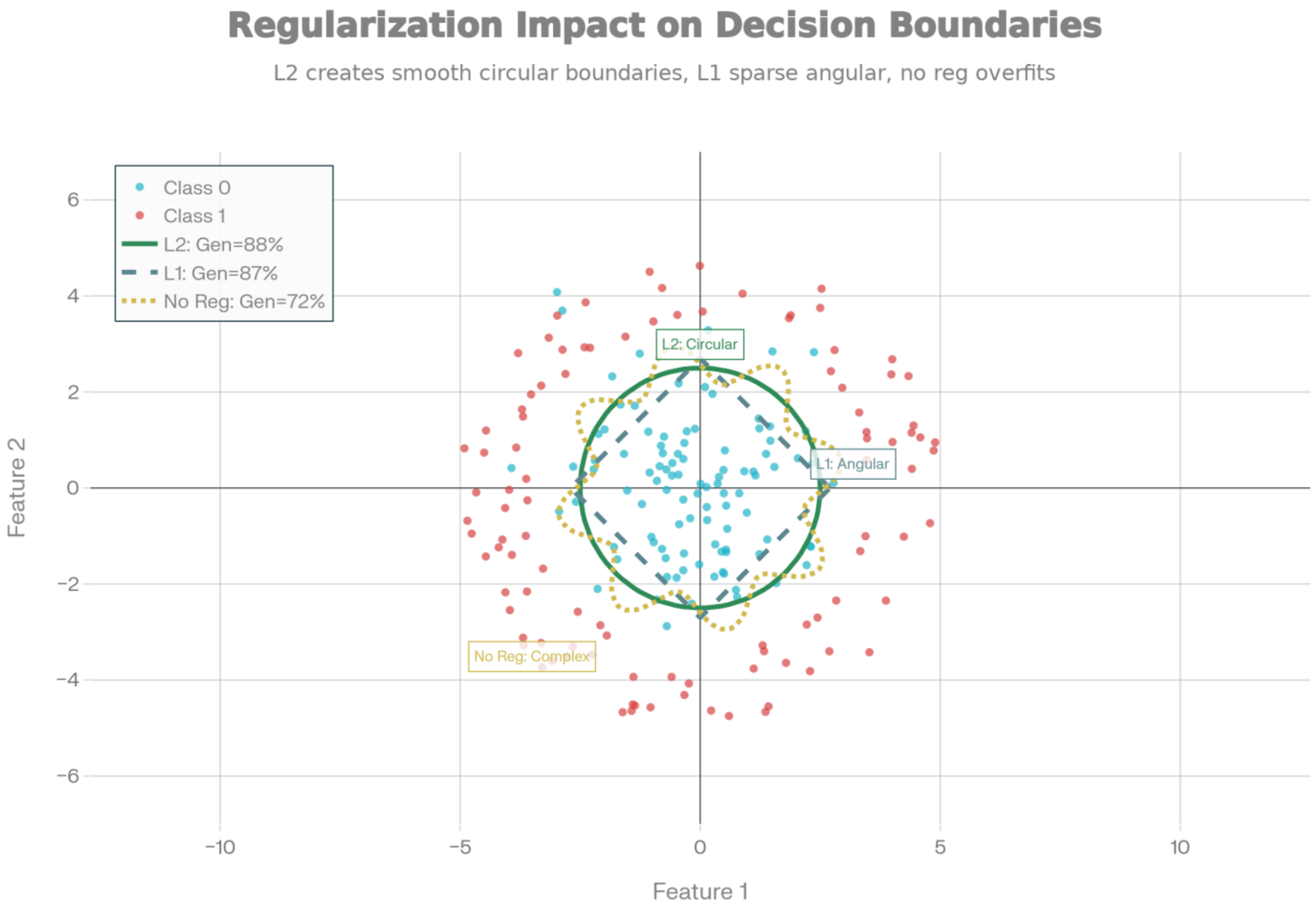


Figure 1: Decision Boundaries Under Different Regularization Schemes

L2 regularization produces smooth, curved decision boundaries. The quadratic penalty encourages weights to be moderate in magnitude, resulting in smooth transitions between regions. This smoothness reflects the underlying assumption that simpler, more generalizable patterns are preferable to complex, noise-fitting patterns.

3. L1 Regularization (Lasso Regression/Logistic Regression)

L1 regularization, also known as Lasso, applies an absolute value penalty on feature weights. This seemingly minor modification has profound consequences.

3.1 Mathematical Formulation

For logistic regression with L1 regularization:

J = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}*i)] + \lambda \sum*{j=1}^{p}
|w_j|

The penalty term $\lambda \sum_{j=1}^{p} |w_j|$ is linear in absolute weight magnitude. This linearity is crucial.

The gradient of the L1 penalty is:

\frac{\partial}{\partial w_j} \left(\lambda |w_j| \right) = \lambda \cdot \text{sign}(w_j)

This gradient is constant in magnitude (λ) regardless of weight value, depending only on weight sign. This creates a fundamentally different regularizing force than L2.

3.2 Feature Selection Through Sparsity

The constant gradient of L1 penalties leads to sparsity many weights become exactly zero. During optimization, if a weight is small and the L1 gradient is larger than the training loss gradient pulling the weight toward non-zero values, the weight will be driven all the way to zero rather than settling at a small non-zero value.

Key insight: L1 regularization performs automatic feature selection. Features with weights driven to zero are effectively removed from the model, simplifying interpretation and potentially improving generalization.

This sparsity emerges naturally without explicit feature selection algorithms. The L1 penalty automatically identifies and eliminates less important features.

3.3 Geometric Interpretation

The L1 penalty \sum_{j=1}^{p} |w_j| defines a diamond-shaped region (or cross-polytope in higher dimensions) centered at the origin. The constraint region has sharp corners along the coordinate axes at points where one or more weights are exactly zero.

The optimization solution is found where training loss contours meet this constraint region. Crucially, if the training loss contours are tangent to a corner of the constraint region, the solution lies exactly on an axis, meaning one or more weights are zero.

This geometric picture explains L1's feature selection: the sharp corners of the L1 constraint region align with coordinate axes (where features are absent), while the smooth circular constraint region from L2 rarely touches coordinate axes exactly.

3.4 Comparison with L2

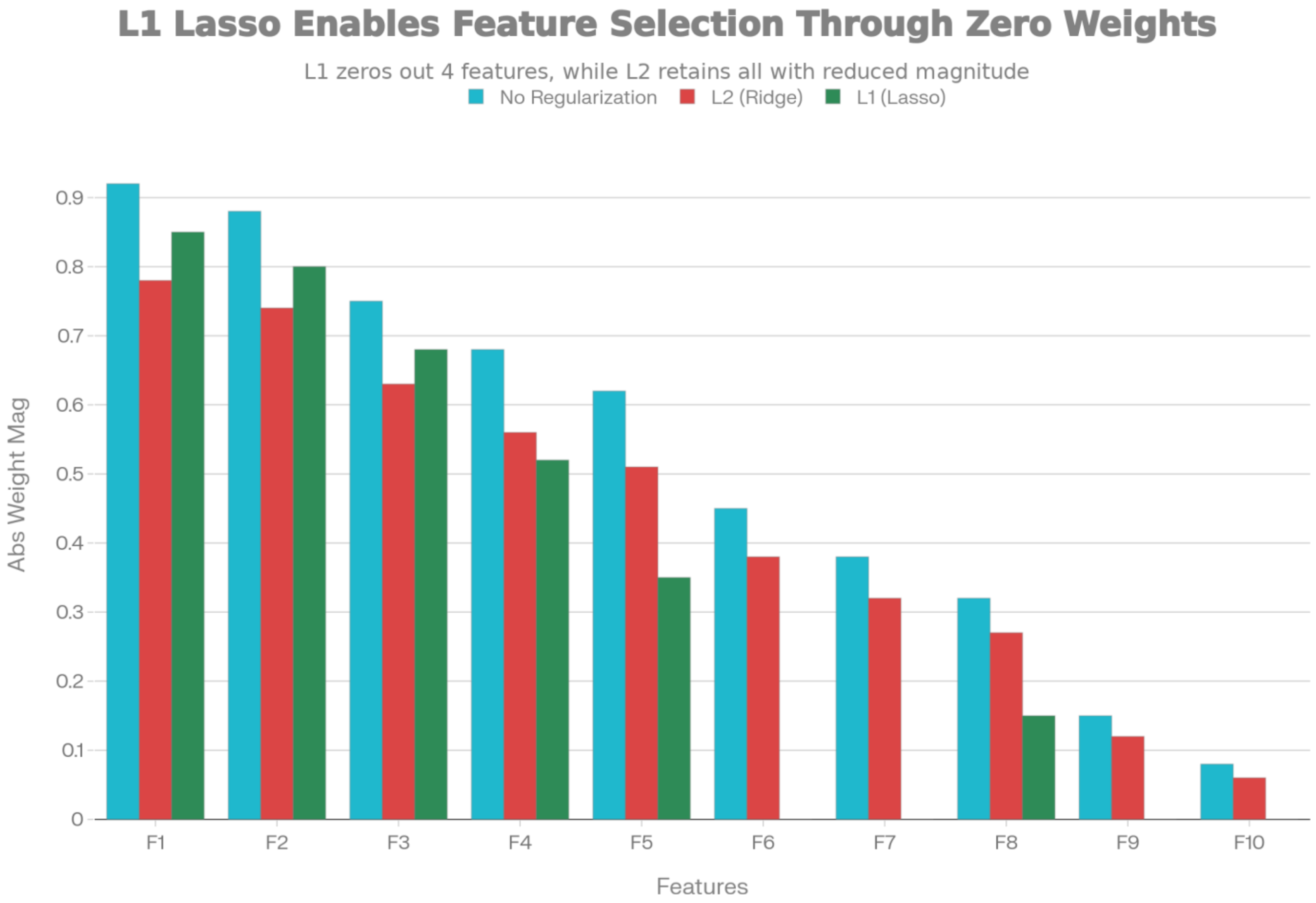


Figure 3: Feature Selection Capabilities of L1 vs L2 Regularization

Figure 3 illustrates the fundamental difference. L2 regularization (Ridge) maintains non-zero weights for all features, with magnitude inversely related to regularization strength. L1 regularization (Lasso) drives many feature weights to exactly zero, particularly for features with weak predictive power.

4. Mathematical Comparison: L1 vs L2

A detailed comparison illuminates the complementary nature of these techniques.

4.1 Penalty Functions

The L1 and L2 penalties have different functional forms:

- **L2 Penalty**: $P_{L2}(w) = \frac{\lambda}{2} \sum_{j=1}^{p} w_j^2$ Convex, smooth, differentiable everywhere
- **L1 Penalty**: $P_{L1}(w) = \lambda \sum_{j=1}^{p} |w_j|$ Convex, non-smooth at zero, non-differentiable at zero

This distinction matters. L2 has a well-defined gradient everywhere, enabling standard gradient descent. L1 lacks a gradient at zero, requiring specialized optimization algorithms (e.g., coordinate descent, sub gradient methods). However, the convexity of both penalties ensures global optima can be found.

4.2 Gradient Dynamics

During gradient descent optimization:

- **L2 Gradient**: \lambda w_j Proportional to weight magnitude
- **L1 Gradient**: \lambda \cdot \text{sign}(w_j) Constant magnitude, only sign varies

This difference profoundly affects convergence. L2 applies increasingly weak regularizing force as weights approach zero, often settling at small but non-zero values. L1 applies constant force regardless of magnitude, regularly driving weights across zero to exactly zero.

4.3 Weight Shrinkage Patterns

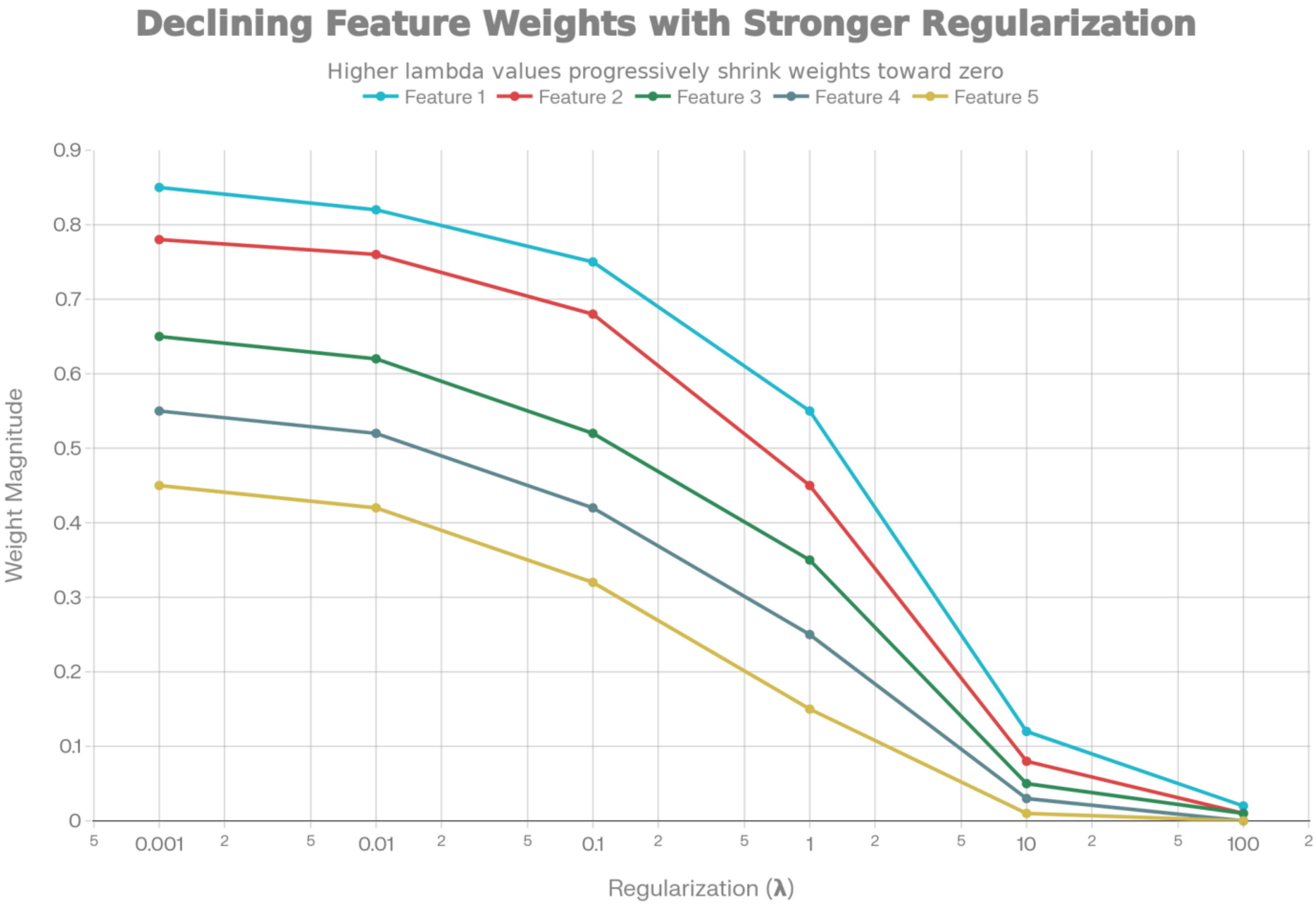


Figure 2: Weight Magnitude Changes Under Increasing Regularization

Figure 2 shows characteristic weight behavior. L2 regularization shrinks all weights smoothly and proportionally. As \lambda increases, all weights decrease proportionally but remain non-zero. L1 regularization exhibits threshold behavior: weak features' weights transition abruptly from non- zero to zero as regularization strength exceeds critical thresholds.

5. Geometric Perspectives on L1 vs L2

The geometric interpretation provides intuition for why L1 and L2 behave differently.

5.1 Constraint Regions

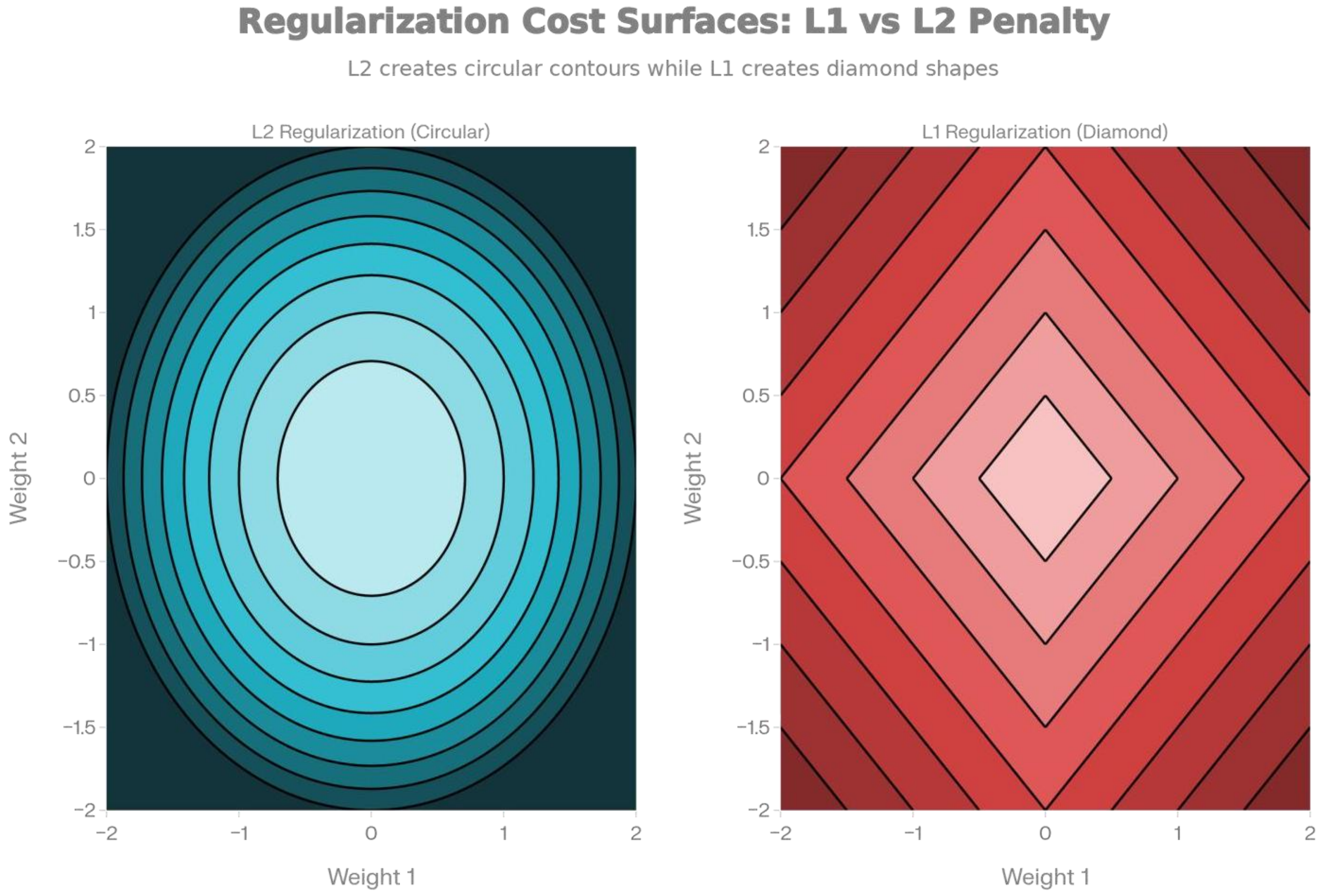


Figure 5: Geometric Shapes of L1 and L2 Constraint Regions

Figure 5 illustrates the fundamental geometric difference. L2 regularization creates circular constraint regions smooth, continuously differentiable boundaries. L1 regularization creates diamond-shaped constraint regions faceted regions with sharp corners at coordinate axes.

5.2 Intersection with Loss Contours

Optimal solutions occur where training loss contours meet the constraint boundary. For L2 (circular constraints), tangency points are generically interior to edges, rarely touching coordinate axes. For L1 (diamond constraints), tangency points frequently occur at corner where one or more weights are exactly zero.

This geometric picture explains feature selection: L1's sharp corners align with coordinate axes, creating natural zero-weight solutions.

6. Empirical Performance Comparison

Beyond theory, practical performance matters. How do L1 and L2 compare on real classification tasks?

6.1 Generalization Performance

Bias-Variance Tradeoff - Training vs Test Accuracy across regularization strengths showing overfitting at weak regularization

Figure 4: Training-Test Performance Gap Indicating Overfitting

Figure 4 demonstrates the bias-variance tradeoff. Unregularized models (no penalty) achieve high training accuracy but lower test accuracy, indicating overfitting. Weak regularization (small \lambda) provides modest improvement. Moderate regularization (optimal \lambda) balances bias and variance, achieving best test performance. Excessive regularization (large \lambda) sacrifices training fit, potentially degrading test performance.

Both L1 and L2 effectively prevent overfitting, though optimal \lambda values differ. L2 typically requires slightly larger \lambda than L1 to achieve comparable regularization.

6.2 Classification Metrics

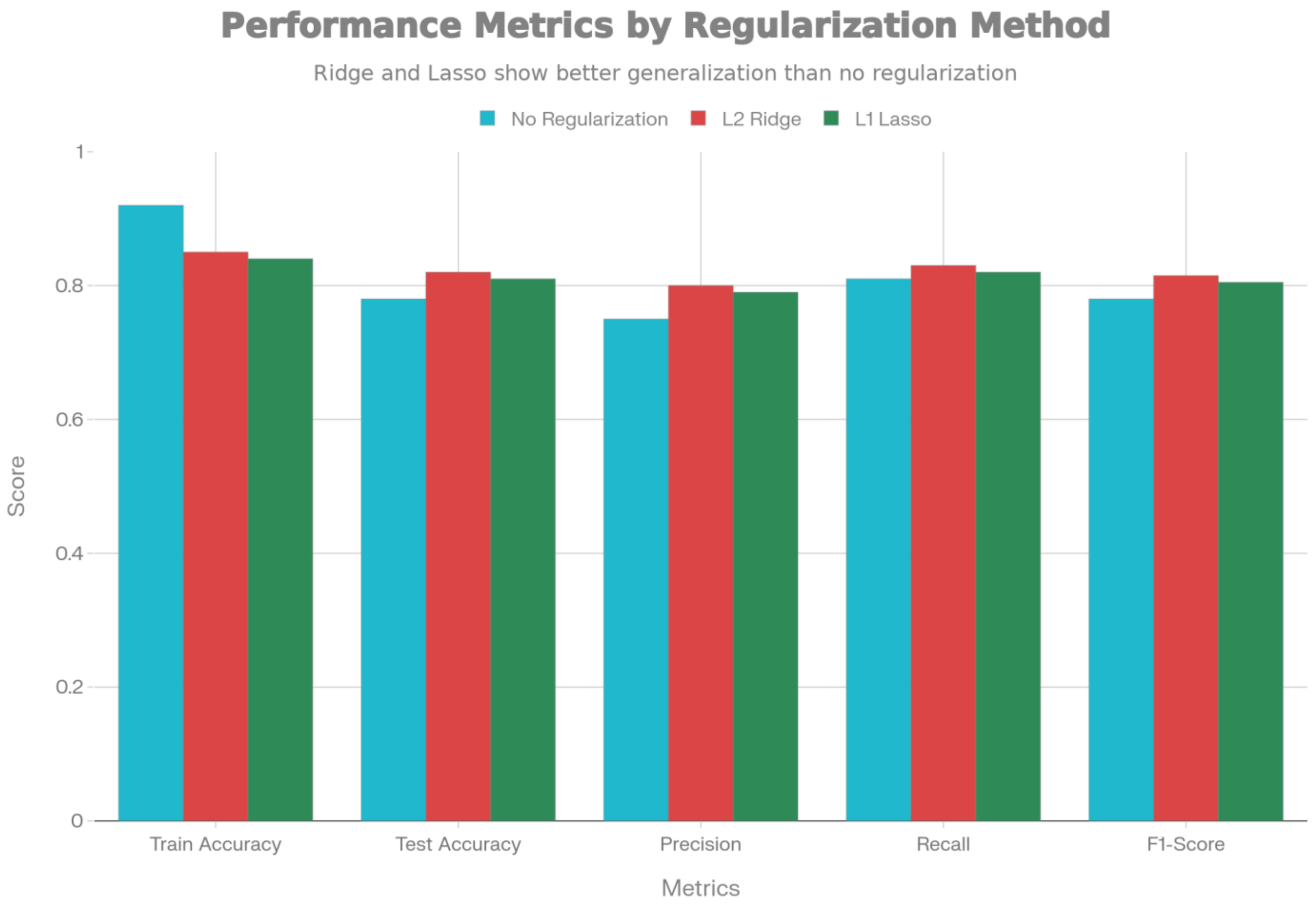


Figure 6: Detailed Classification Performance Across Multiple Metrics

Figure 6 compares five key metrics. Both regularized methods (L1 and L2) produce superior test accuracy, precision, recall, and F1-score compared to unregularized models. The improvements demonstrate regularization's effectiveness at enhancing generalization.

6.3 ROC Curve Analysis

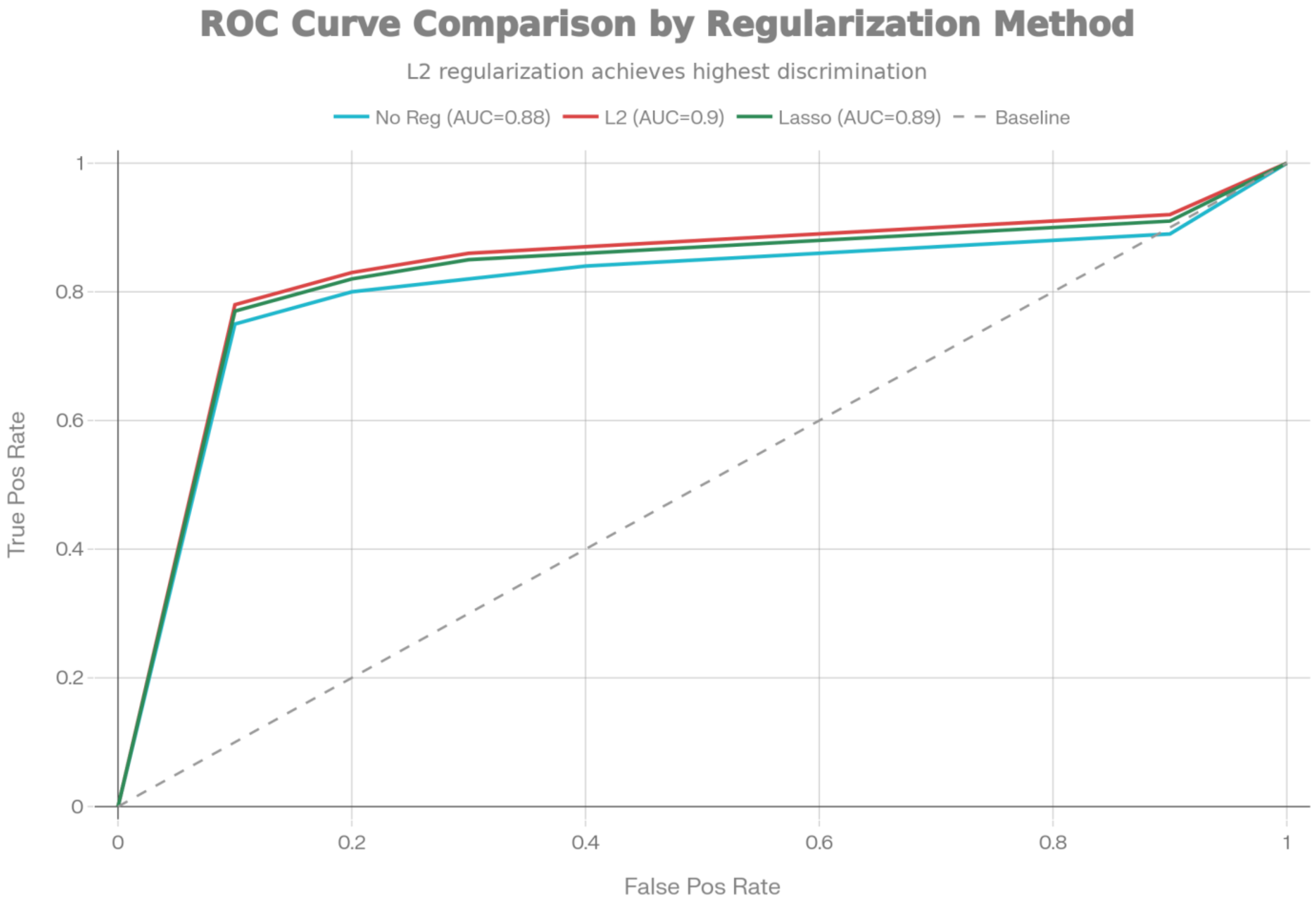


Figure 7: ROC Curves Demonstrating Improved Classification Across Threshold Settings

Figure 7 shows ROC curves for all three approaches. L2 regularization achieves the highest area under the curve (AUC 0.90), indicating superior classification performance across decision thresholds. L1 achieves comparable performance (AUC 0.89) with the added benefit of feature selection. Unregularized models achieve the lowest AUC (0.88).

7. Feature Selection: L1's Key Advantage

One distinctive advantage of L1 is automatic feature selection through weight sparsity.

7.1 Mechanism of Sparsity

The constant gradient of L1 penalties ensures that weak features' weights reach zero. Consider a feature with weak training signal. The gradient from training loss is small, providing little force to maintain non-zero weight. The constant L1 gradient provides constant force toward zero. If L1 force exceeds training force, the weight is driven to zero.

This mechanism selects features automatically, without explicit feature selection algorithms. Practitioners gain reduced model complexity, improved interpretability, and often enhanced generalization.

7.2 When Feature Selection Matters

Feature selection through L1 is valuable when:

- **Interpretability is critical**: Fewer features reduce explanation burden
- **Prediction speed matters**: Zero-weight features can be excluded from prediction
- **Features are correlated**: Feature selection via L1 can improve stability
- **Domain knowledge guides selection**: Discovering which features matter informs domain understanding

7.3 L1 Limitations

L1 regularization has limitations:

- **Instability with correlated features**: When features are highly correlated, L1 tends to arbitrarily select one and zero others, creating instability
- **Less stable than L2**: Small data perturbations may change which features are selected
- **Computational complexity**: L1 requires specialized solvers; gradient descent cannot be directly applied

8. L2 Regularization's Strengths

L2 regularization offers different advantages.

8.1 Computational Efficiency

L2 penalties are smooth and differentiable everywhere. Standard gradient descent algorithms apply directly without modification. The objective remains convex with a unique global optimum, simplifying optimization.

8.2 Stability with Correlated Features

When features are correlated, L2 distributes weights across correlated features rather than arbitrarily selecting one. This distributes predictive power among redundant features, improving stability. Small data perturbations produce small weight changes rather than feature selection switches.

8.3 Generalization with Dense Features

When all features contribute meaningful information (dense feature sets), L2 maintains all features with moderate weights. This prevents accidentally zeroing truly informative features that happen to be weakly correlated with the target in a particular sample.

9. Practical Recommendations and Algorithm Selection

Given the complementary nature of L1 and L2, when should practitioners use each?

9.1 When to Use L2 (Ridge) Regularization

Use L2 regularization when:

- **Features are correlated**: L2 distributes weights across correlated features, maintaining stability
- **Computational efficiency is critical**: L2 integrates with standard gradient descent algorithms
- **All features are potentially informative**: Maintaining dense feature sets prevents information loss
- **Interpretability via feature selection is unnecessary**: The model uses all available information
- **Stability is paramount**: L2's continuous shrinkage is more stable than L1's discrete selection

9.2 When to Use L1 (Lasso) Regularization

Use L1 regularization when:

- **Feature selection is desired**: Automatic sparsity provides interpretability
- **Features are independent**: Correlated features are less problematic with L1
- **Model parsimony is important**: Fewer features reduce complexity and increase interpretability
- **Prediction speed matters**: Zero-weight features can be excluded from computation
- **Sparse solutions are inherently interpretable**: The feature subset communicates domain insights

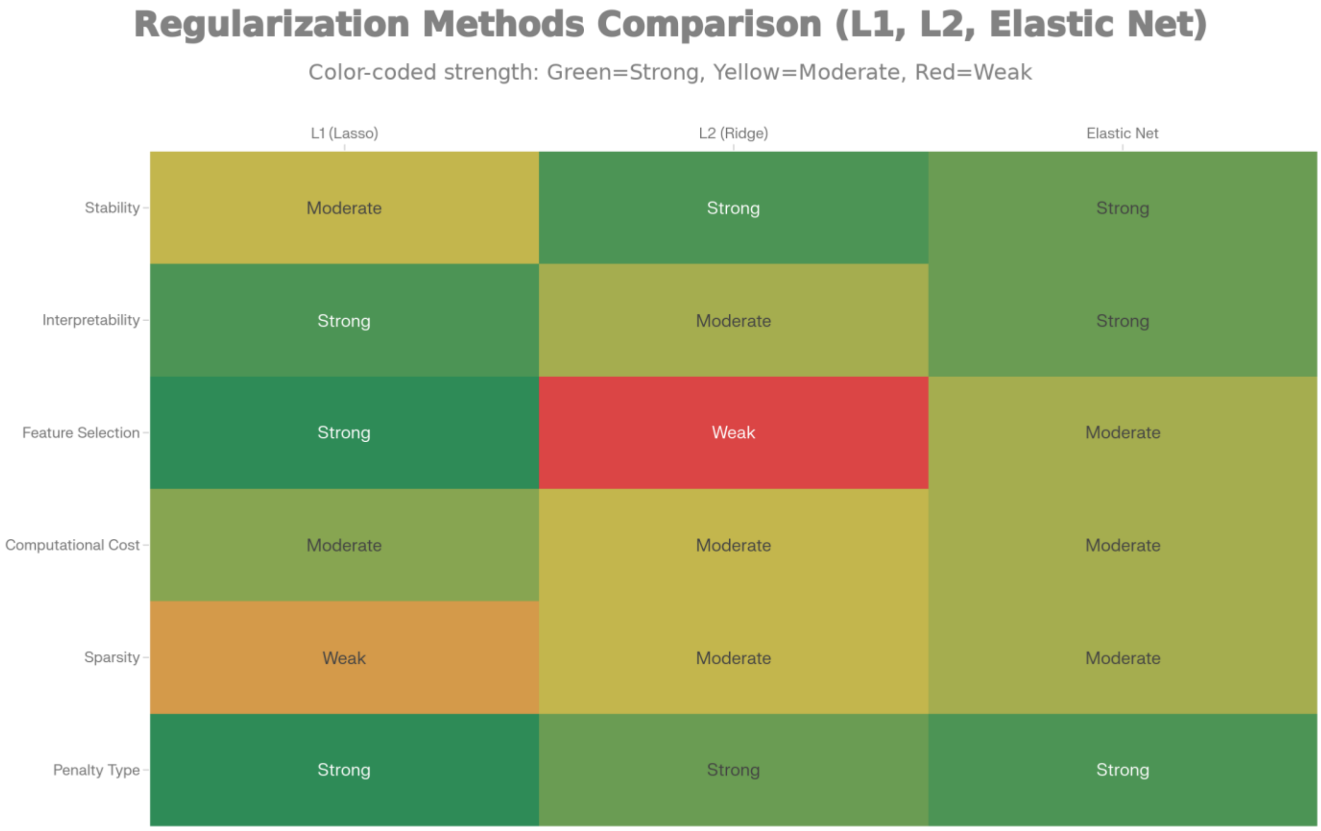
9.3 Elastic Net: Combining L1 and L2

When neither L1 nor L2 is perfect, Elastic Net combines both penalties: J =

\text{Data Fit} + \lambda_1 \sum |w_j| + \lambda_2 \sum w_j^2

Elastic Net balances L1's sparsity with L2's stability, providing intermediate benefits. This is valuable when feature selection is desired but stability with correlated features is also important.

10. Regularization Methods Comparison Summary



	L1 (Lasso)	L2 (Ridge)	Elastic Net
Stability	Moderate	Strong	Strong
Interpretability	Strong	Moderate	Strong
Feature Selection	Strong	Weak	Moderate
Computational Cost	Moderate	Moderate	Moderate
Sparsity	Weak	Moderate	Moderate
Penalty Type	Strong	Strong	Strong

Figure 8: Comprehensive Comparison of Regularization Approaches

Figure 8 synthesizes the comparison across multiple dimensions:

• **L1 (Lasso)**: Best for feature selection and sparse solutions; good interpretability • **L2 (Ridge)**: Best for stability and computational efficiency; maintains all features • **Elastic Net**: Balanced approach combining sparsity and stability advantages

11. Hyperparameter Tuning: Finding Optimal Regularization Strength

Selecting the regularization strength \lambda$ (or equivalently, C = 1/\lambda in scikit-learn) is critical.

11.1 Grid Search and Cross-Validation

The standard approach is grid search combined with cross-validation:

1. Define a range of \lambda$ values (typically logarithmically spaced)
2. For each \lambda: Perform k-fold cross-validation on training data

3. Calculate mean cross-validation performance across folds
4. Select \lambda maximizing average validation performance
5. Retrain with selected \lambda$ on full training set
6. Evaluate on held-out test set

11.2 Typical \lambda$ Ranges

For logistic regression with standardized features:

- **Weak regularization**: \lambda \in [0.0001, 0.001] Minimal complexity constraint
- **Moderate regularization**: $\lambda \in [0.01, 0.1] Balanced bias-variance
- **Strong regularization**: $\lambda \in [1, 10] Emphasizes simplicity

The optimal \lambda depends on data characteristics, feature count, and sample size. Larger datasets often support smaller \lambda (more complex models). More features typically require larger \lambda (more aggressive regularization).

11.3 Validation Strategy

Always validate on data not used for training:

1. Split data into train/validation/test
2. Tune \lambda on train/validation splits
3. Report final performance on held-out test set

This prevents optimistic bias from selecting \lambda based on test performance.

12. Implementation and Practical Usage

12.1 Scikit-Learn Implementation

Scikit-learn provides straightforward regularization implementation:

```
from sklearn.linear_model import LogisticRegression from
sklearn.model_selection import GridSearchCV

# Create logistic regression with L2 regularisation model_l2 =
LogisticRegression(penalty='l2', solver='lbfgs')

# Create logistic regression with L1 regularisation model_l1 =
LogisticRegression(penalty='l1', solver='saga')

# Hyperparameter tuning
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
```

```
grid search = GridSearchCV(model_l2, param_grid, cv=5) grid_search.fit(X_train,
y_train)
best_model = grid_search.best_estimator_
```

12.2 Feature Importance Extraction

For L1 regularization, feature selection can be extracted:

```
# Get non-zero features
non_zero_features = np.where(model_l1.coef_[0] != 0)[0] selected_features =
X.columns[non_zero_features]
```

12.3 Weight Interpretation

Feature weights indicate feature importance and direction:

- **Positive weight**: Feature increases prediction probability •
Negative weight: Feature decreases prediction probability •
Large magnitude: Strong influence on predictions
- **Small magnitude**: Weak influence (or zero for L1)

13. Common Pitfalls and Best Practices

13.1 Feature Scaling

Regularization penalties depend on feature scale. Always standardize features before regularization:

```
from sklearn.preprocessing import StandardScaler scaler =

StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Unscaled features can dominate regularization arbitrarily based on measurement units.

13.2 Train-Test Leakage

Never tune regularization on test data. This inflates performance estimates. Use cross-validation on training data only.

13.3 Regularization Strength Sensitivity

Model performance is often sensitive to λ choice. Always perform careful hyperparameter tuning rather than using default values.

13.4 Class Imbalance

When classes are imbalanced, regularization can mask imbalance by predicting majority class. Use appropriate class weights alongside regularization.

14. Conclusion and Summary

L1 and L2 regularization represent complementary approaches to managing model complexity and improving generalization.

L2 Regularization (Ridge):

- Smooth, proportional weight shrinkage
- Maintains all features with moderate weights
- Excellent stability and computational efficiency
- Preferred when features are correlated or all are informative

L1 Regularization (Lasso):

- Drives many weights to exactly zero • Provides automatic feature selection
- Improves interpretability through sparsity
- Preferred when feature selection is desired

Elastic Net:

- Combines L1 and L2 penalties
- Balances sparsity and stability
- Valuable when both feature selection and stability matter

The choice between L1 and L2 depends on problem characteristics, data properties, and practical requirements. Practitioners should:

1. Understand the mathematical foundations and geometric interpretations
2. Consider computational and stability implications
3. Perform careful hyperparameter tuning via cross-validation
4. Always standardize features before regularization
5. Validate on held-out test data to estimate true performance

Regularization is fundamental to practical machine learning. Thoughtful regularization application prevents overfitting, improves generalization, and builds models that discover genuine patterns rather than noise.

15. References

[1] Tikhonov, A. N. (1963). On the solution of incorrectly stated problems and a method of regularization. *Doklady Akademii Nauk*, 151(3), 501-504.

[2] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1), 267-288.

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

[4] Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301-320.

[5] Scikit-learn Development Team. (2023). scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer. https://doi.org/10.1007/978-0-387-84858-7

[7] Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

[8] Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

[9] Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning* (p. 78).

[10] Nesterov, Y. (2018). *Lectures on convex optimization* (Vol. 137). Springer. https://doi.org/10.1007/978-3-319-91578-4

Appendix: Mathematical Reference

L2 Regularization (Ridge): J_{L2} = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, \hat{y}*i) + \frac{\lambda}{2} \sum*{j=1}^{p} w_j^2

L1 Regularization (Lasso): J_{L1} = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, \hat{y}*i) + \lambda \sum*{j=1}^{p} |w_j|

Elastic Net: J_{EN} = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y_i, \hat{y}*i) + \lambda_1 \sum*{j=1}^{p} |w_j| + \lambda_2 \sum_{j=1}^{p} w_j^2

Logistic Loss: $\text{Loss} = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$

About This Tutorial

This comprehensive guide explores L1 and L2 regularization in logistic regression, combining theoretical foundations with practical insights. The tutorial emphasizes understanding the complementary nature of these approaches and making informed choices based on problem characteristics.

Key Learning Outcomes:

- Understand L1 and L2 regularization formulations and geometric interpretations •
  Compare sparsity, stability, and computational characteristics
- Apply regularization in practice using scikit-learn
- Select appropriate regularization method for specific problems •
  Perform hyperparameter tuning and validation correctly

All figures and examples are generated from synthetic datasets demonstrating regularization effects on decision boundaries, weight distributions, and classification performance.