Hyperparameter Tuning and Double Descent in Regularization for Machine Learning Models

Max Tost, Jéremy Salm, Gauthier Leurent
EPFL, Lausanne, Switzerland
max.tost@epfl.ch jeremy.salm@epfl.ch gauthier.leurent@epfl.ch

***Abstract*—This study investigates the impact of hyperparameter tuning on the regularization parameter in a machine learning model, focusing on patterns that improve model generalization. Using k-fold cross-validation and custom functions in 'run.py', we explored the double descent phenomenon in test errors, highlighting the importance of regularization and extensive data cleaning. Our results emphasize pre-processing steps, including feature normalization and handling irrelevant data, which are essential for reproducible and accurate outcomes.**

I. Introduction

Hyperparameter tuning plays a crucial role in machine learning model performance, where the regularization parameter particularly influences model generalization. This project examined how varying regularization values affected the test error, especially under different optimization times. We observed a double descent pattern in test error across iterations, suggesting that prolonged optimization can reduce overfitting. Additionally, we identified key preprocessing steps, such as data normalization and feature selection, as critical factors in achieving accurate results. Our methodology leverages custom functions in 'run.py', ensuring the approach is replicable and adaptable.

II. Methods

Our approach involved a structured workflow comprising data loading, preprocessing, cross-validation, and hyperparameter tuning, with each step detailed to support reproducibility.

A. Data Loading and Preprocessing

Data loading and preparation were handled by 'run.py', which included custom functions tailored to the project's requirements. Specifically:

- `load_csv_data()` loads data from CSV files, converting features X and binary labels y (from -1, 1 to 0, 1).
- `clean_and_standardize()` preprocesses features by normalizing them, imputing missing values with the column mean, and adding a bias term to each data point. Normalization ensured that features were scaled, improving gradient-based optimization.
- During preprocessing, irrelevant features like identifiers and time stamps were removed to prevent spurious correlations. Initial tests showed that including identifiers led to disproportionately high weights for these features, likely due to non-informative data patterns that interfered with learning.

This cleaning process is essential for reproducibility, as it ensures that the model learns from meaningful patterns in the data rather than from labels, et cetera.

B. Cross-Validation and Hyperparameter Tuning

To systematically assess the effect of regularization, we used k-fold cross-validation, implemented in `cross_validation_reg_log()` to evaluate a range of regularization parameters λ. This setup is particularly effective for generalization testing and model stability.

Key functions for this process include:

- `build_k_indices(y, k_fold, seed)` generates indices for k-fold cross-validation, where $k = 5$ was selected to balance training and validation set sizes.
- For each λ value, `cross_validation()` iterates over k-folds, training on $k - 1$ folds and validating on the remaining fold. This function integrates `reg_logistic_regression()` which optimizes model weights for regularized logistic regression.
- The cross-validation process calculates average training and test losses across folds, with the optimal λ identified based on the lowest test error.

C. Observed Double Descent Phenomenon

Double descent, a recurring pattern in modern machine learning, was observed in our tests as follows: at low iteration counts, test error exhibited a minimum at specific λ values, suggesting optimal performance. However, as iterations increased, test error for lower regularization values stabilized, becoming largely independent of λ. This indicates that extended optimization can mitigate overfitting, underscoring the importance of balancing regularization strength and optimization time.

III. Results

The results section presents key findings, including the influence of regularization and iteration counts on test and

training errors, with visual representations to illustrate observed behaviors.

A. *Effect of Regularization on Test and Training Error*

The test and training errors varied notably across different regularization values. Figure 3 presents initial tuning results with 300 iterations and a learning rate of 0.1, alongside refined tuning results with 1000 iterations and step size 1. In the latter case, the extended tuning highlights the double descent pattern, where errors for step sizes below 10^{-3} become independent of λ.

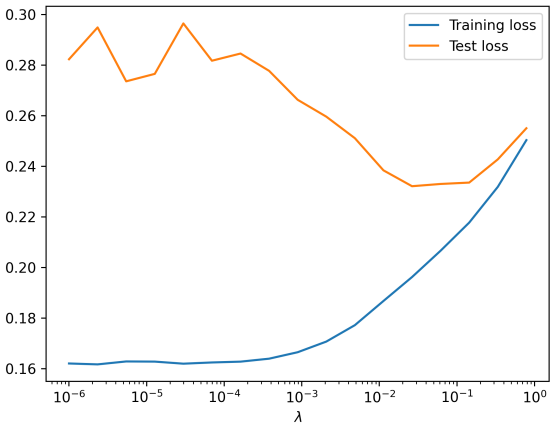

Figure 1. Initial tuning with 300 iterations, learning rate 0.1

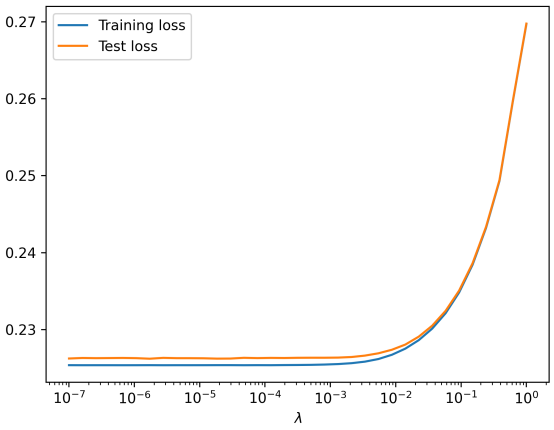

Figure 2. Extended tuning with 1000 iterations, step size 1

Figure 3. Training and test error behavior during hyperparameter tuning at different regularization and iteration levels.

B. *Impact of Data Preprocessing*

Data cleaning, including feature standardization and removal of non-informative features, significantly improved model accuracy. Without this preprocessing, the model often assigned high weights to irrelevant features, distorting the regularization effect and degrading test accuracy. This highlights the necessity of careful preprocessing for valid and interpretable results.

IV. Discussion

The double descent phenomenon observed in this project aligns with recent findings in machine learning, where complex models benefit from extended training to achieve generalization stability. In our case, the results confirm that iterative tuning of regularization can help reach a stable error plateau, balancing both underfitting and overfitting.

Our experiments demonstrate that feature selection and data normalization play critical roles in optimizing regularization's impact. When irrelevant data, like IDs, were included, the model tended to rely on them, falsely inflating test accuracy. This spurious correlation risk emphasizes the importance of thoughtful data cleaning before training.

V. Conclusion

Regularization remains a vital component of machine learning, significantly affecting generalization. Our exploration of double descent behavior confirms that balancing regularization with extended optimization can prevent overfitting. Effective data cleaning, feature normalization, and hyperparameter tuning are essential steps in obtaining replicable and accurate results.

Acknowledgments

We acknowledge the assistance of ChatGPT in optimizing language, refining the code structure, and improving clarity. All fundamental ideas, code implementations, and methodologies were independently developed by the authors.