

GridSearchCV in Scikit-learn

Slide 1: Title

GridSearchCV in Scikit-learn

Hyperparameter tuning for optimal model performance

Slide 2: Why Hyperparameter Tuning Matters

- Model performance depends on hyperparameters (e.g. C, kernel, max_depth)
- Manual tuning is inefficient and biased
- Goal: Systematically find the best hyperparameter combination

Analogy: Like tuning engine parameters for maximum fuel efficiency

Slide 3: What is GridSearchCV?

- GridSearchCV = exhaustive search over specified hyperparameter values
- Uses cross-validation to evaluate combinations
- Returns the best parameters and the best model

Tool = `sklearn.model_selection.GridSearchCV`

Slide 4: Key Parameters

`GridSearchCV(estimator, param_grid, scoring=None, cv=None, verbose=0, n_jobs=None)`

- estimator: e.g. `SVC()`, `RandomForestClassifier()`
- param_grid: dict of parameter names and list of values
- scoring: metric to optimise (e.g. 'accuracy', 'f1')
- cv: number of cross-validation folds (e.g. 5)
- n_jobs: number of parallel jobs (e.g. -1 for all cores)

Slide 5: Example - Tuning an SVM

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
```

GridSearchCV in Scikit-learn

```
param_grid = {  
    'C': [0.1, 1, 10],  
    'kernel': ['linear', 'rbf']  
}  
  
grid = GridSearchCV(SVC(), param_grid, cv=5)  
grid.fit(X_train, y_train)  
  
print(grid.best_params_)  
print(grid.best_score_)
```

Slide 6: Visual Explanation

Visual: Grid matrix diagram

- Rows: different C values
- Columns: different kernel options
- Each cell = average cross-validation score

Slide 7: Accessing Results

```
grid.best_params_    # Best hyperparameter combination  
grid.best_score_     # CV score of best model  
grid.best_estimator_ # The model trained with best params
```

Bonus: `grid.cv_results_` gives full result matrix

Slide 8: Advantages and Limitations

Pros:

- Systematic and exhaustive
- Integrated CV for fair evaluation

Cons:

GridSearchCV in Scikit-learn

- Computationally expensive
- Doesn't handle continuous parameters well

Slide 9: Best Practices

- Start with a coarse grid, then refine
- Use `n_jobs=-1` to parallelise
- Combine with Pipeline to avoid data leakage

Slide 10: When to Use vs Alternatives

- Use GridSearchCV when:
 - Grid is small
 - Interpretability is important

Alternatives:

- RandomizedSearchCV (faster)
- Bayesian optimisation (e.g., Optuna, Hyperopt)

Slide 11: Summary

- GridSearchCV = brute-force but reliable search
- Performs K-fold CV for all combinations
- Returns best model + performance
- Powerful tool for improving model generalisation