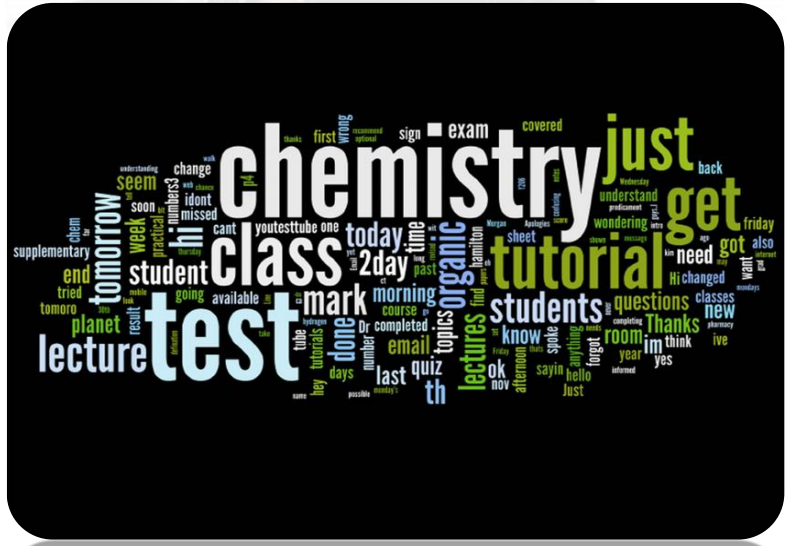
Optimized hyperparameters tuning of multi-class classification algorithms

By Seyed Mohammad Reza Ahmadi

The Bigger the business

the Bigger the problems

Classification in real-world problems



Which Algorithm?

Gradient Boosting (GB)	K-nearest neighbors(KNN)	Naïve Bayes	Random Forest	Decision Tree(DS)
Finance	Text mining	Text classification	Finance	Biomedical Engineering
Healthcare	Agriculture	Spam filtering	Manufacturing	Financial analysis
Credit scoring	Finance	Sentiment analysis	Social media analysis	Astronomy
Physics	Medical	Ranking pages	Healthcare & medicine	Manufacturing/Production
Facial analysis	Facial recognition	Social media analysis	E-commerce	Physics

Which Algorithm?

Gradient Boosting (GB)	K-nearest neighbors (KNN)	Naïve Bayes	Random Forest	Decision Tree (DS)	
Interpretability, Curbs overfitting easily	Simple to understand, fast and efficient	Efficient, not biased by outliers, works on non – linear problems, probabilistic approach.	Powerful and accurate, good performance on many problems, including non – linear.	Interpretability, no need for feature scaling, works on both linear / non – linear problems.	Pros
Sensitive to outliers	Need to manually choose the number of neighbours ‘k’.	Based in the assumption that the features have same statistical relevance.	No interpretability, overfitting can easily occur, need to choose the number of trees manually.	Poor results on very small datasets, overfitting can easily occur.	Cons

What Number?

Random Forest	**Gradient Boosting (GB)**	**Decision Tree (DS)**	**K-nearest neighbors (KNN)**	**Naïve Bayes**
N_estimators	N_estimators	Max_depth	K	Alpha
Max_features	Max_depth	Min_samples_leaf	Weight_options	
Max_depth	Learning_rate	Criterion		
Min_samples_leaf				
Min_samples_split				
Bootstrap				

NO

TIME

TO

CALCULATE

Hyperparameter

GridSearchCV

sklearn.model_selection.**GridSearchCV**(estimator, param_grid, scoring=None, n_jobs=None, refit=True, cv=None, return_train_score=False)

RandomizedSearchCV

sklearn.model_selection.**RandomizedSearchCV**(estimator, param_distributions, n_iter=10, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', random_state=None, error_score=nan, return_train_score=False)

GridSearch & RandomSearch

- GridSearchCV and RandomSearchCV are **libraries function that is a member of sklearn's model_selection package**

Grid Search starts with defining a search space **grid**. The grid consists of selected hyperparameter names and values, and **grid search** exhaustively searches the best combination of these given values.

In random search, we define **distributions** for each hyperparameter which can be defined *uniformly* or with a *sampling method*. The key difference from grid search is in random search, not all the values are tested and values tested are selected at random.

Grid Search vs Random Search

Note

- We don’t necessarily need to customize the range of numbers in Gridsearch and Randomsearch for every problem.

- With the following random numbers we would be able to get a reasonable result from our model.

Random Forest

	N_estimator	Min_samples_leaf	Max_depth	Max_features	Bootstrap
Gridsearch	[50, 100, 200, 300]	[1, 5, 10]	[2, 4, 6, 8, 10]	['auto', 'sqrt']	[True, False]
Randomizedsearch	list(range(100, 300, 10))	list(range(1, 20))	list(range(2, 50)	['auto', 'sqrt']	[True, False]

Max_depth	Max depth of the tree
N_estimator	The number of sequential trees to be modeled
Min_samples_leaf	Minimum number of data points in a leaf node
Bootstrap	Method of selecting samples for training each tree
Max_features	Number of features to consider at every split

Gradient Boosting(GB)

N_estimator	Max_depth	Learning_rate
[5,50,250,500]	[1,3,5,7,9]	[0.01,0.1,1,10,100]

Max_depth	Max depth of the tree
N_estimator	The number of sequential trees to be modeled
Learning_rate	This determines the impact of each tree on the final outcome

Decision Tree(DS)

Max_depth	Min_samples_leaf	Criterion
[2, 3, 5, 10, 20]	[5, 10, 20, 50, 100]	["gini", "entropy"]

Max_depth	Max depth of the tree
Min_samples_leaf	Minimum number of data points in a leaf node
Criterion	how the impurity of a split will be measured

K-nearest neighbors(KNN)

K

[1, 5, 8, 10, 20, 23, 30]

Weight_options

['uniform', 'distance']

Weight_options	uniform (all points in the neighborhood are weighted equally) distance (weights closer neighbors more heavily than further neighbors)
k	parameter that refers to the number of nearest neighbours to include in the majority of the voting process

Naïve Bayes

Alpha

[2, 3, 5, 10, 20]

alpha	As alpha increases, the likelihood probability moves towards uniform distribution (0.5). Most of the time, alpha = 1 is being used to remove the problem of zero probability.
	It accepts float value representing the additive smoothing parameter. The value of 0.0 represents no smoothing. The default value of this parameter is 1.0.

Optimized hyperparameters tuning of multi-class classification algorithms

By Seyed Mohammad Reza Ahmadi