Fitness Landscape Analysis of Automated Machine Learning Search Spaces
Supplementary Material

Cristiano G. Pimenta[1], Alex G. C. de Sá[1], Gabriela Ochoa[2], and Gisele L. Pappa[1]

[1] Computer Science Department, Universidade Federal de Minas Gerais, Brazil
{cgpimenta,alexgcsa,glpappa}@dcc.ufmg.br
[2] University of Stirling, United Kingdom
gabriela.ochoa@stir.ac.uk

1 Search Space

The search space is composed of 18 preprocessing algorithms and 23 classification algorithms. Table 1 shows the preprocessing algorithms and their respective hyperparameters. Tables 2 and 3 describe the classification methods.

1.1 Grammar

In our work, the pipelines are generated from a previously defined context-free grammar (CFG) [1]. Formally, a grammar G is represented by a four-tuple $<NT, T, PR, S>$, where NT represents a set of non-terminals, T a set of terminals, PR a set of production rules and S (a member of N) the start symbol. We will use a simplified version of the Backus Naur Form (BNF) to represent grammars. This means that each production rule has, for instance, the following form $<Start> ::= <A><B> \mid <C>\ d$. Symbols wrapped in "< >" represent non-terminals, whereas terminals (such as d) are not bounded by "< >". The special symbol "|" represents a choice. Additionally, the symbol "#" represents a comment in the grammar (i.e., it is ignored by the grammar parser). The choice of one among all elements connected by "|" is made using a uniform probability distribution.

Box 1.1 shows the top-level productions of the grammar. A classification pipeline always has one classification algorithm, optionally preceded by a preprocessing step. The preprocessing step can contain up to three algorithms, respecting the combinations imposed by the grammar. Boxes 1.2-1.4 show the production rules for each of the preprocessing groups, namely imputation, bounding and binarizer (grouped under the name of range manipulation) and dimensionality manipulation (feature selection and dimensionality reduction).

Since classification pipelines always contains exactly one classifier, they have not been grouped in the grammar. Boxes 1.5-1.11 show the production rules for each of the 23 classification algorithms of the grammar.

```
<Start> ::= <preprocessing> <classification> | <classification>

# Preprocessing
<preprocessing> ::= <imputation> | <bounding> | <binarizer> | <dimensionality> | <imputation> <bounding>
                  | <imputation> <binarizer> | <imputation> <dimensionality> | <bounding> <dimensionality>
                  | <imputation> <bounding> <dimensionality>

# Classification
<classification> ::= <gaussian_nb> | <bernoulli_nb> | <multinomial_nb> | <complement_nb> | <svc> | <nu_svc>
                   | <logistic_regression> | <perceptron> | <mlp> | <sgd> | <lda> | <qda> | <knn>
                   | <radius_neighbours> | <centroid> | <ridge> | <ridge_cv> | <decision_tree> | <extra_tree>
                   | <random_forest> | <extra_trees> | <ada_boost> | <gradient_boosting>
```

Box 1.1: Defined Grammar – Part 1: Top level rules.

Table 1: Preprocessing configuration space. #feats corresponds to the number of features of the dataset.

Algorithm	**Hyperparameter**	**Values**
SimpleImputer	strategy	{mean, median, most_frequent}
Normalizer	norm	{l1, l2, max}
MinMaxScaler	-	-
MaxAbsScaler	-	-
RobustScaler	with_scaling	{True, False}
	with_centering	{True, False}
StandardScaler	with_std	{True, False}
	with_mean	{True, False}
Binarizer	threshold	$[10^{-6}, 10^{3}]$
VarianceThreshold	-	-
SelectKBest	k	$[1, \#feats - 1]$
PCA	n_components	$[1, \#feats - 1]$
	whiten	{True, False}
	svd_solver	{auto, full, arpack, randomized}
	tol	$[0.0, 0.1]$
	iterated_power	$[2, 20]$
IncrementalPCA	n_components	$[1, \#feats - 1]$
	whiten	{True, False}
FastICA	n_components	$[1, \#feats - 1]$
	algorithm	{parallel, deflation}
	fun	{logcosh, exp, cube}
	max_iter	$[10, 10^{3}]$
	tol	$[10^{-10}, 10^{-1}]$
	whiten	{True, False}
GaussianRandomProjection	n_components	$[1, \#feats - 1]$
	eps	$[0.0, 1.0]$
SparseRandomProjection	n_components	$[1, \#feats - 1]$
	eps	$[0.0, 1.0]$
	density	$[10^{-5}, 1.0]$
	dense_output	{True, False}
FeatureAgglomeration	n_clusters	$[1, \#feats - 1]$
	affinity	{euclidean, l1, l2, manhattan, cosine
	linkage	{ward, complete, average}
	compute_full_tree	{True, False}
RBFSampler	n_components	$[1, \#feats - 1]$
	gamma	$[3.05 \times 10^{-5}, 8.0]$
Nystroem	n_components	$[1, \#feats - 1]$
	gamma	$[3.05 \times 10^{-5}, 8.0]$
	kernel	{linear, poly, rbf, sigmoid}
	degree	$[2, 10]$
	coef0	$[0.0, 10^{3}]$
TruncatedSVD	n_components	$[1, \#feats - 1]$
	algorithm	{arpack, randomized}
	n_iter	$[5, 10^{3}]$
	tol	$[10^{-10}, 10^{-1}]$

Table 2: Classification configuration space – Part 1. #feats corresponds to the number of features of the dataset.

Algorithm	**Hyperparameter**	**Values**
GaussianNB	-	-
BernoulliNB	binarize	$[0.0, 1.0]$
	alpha	$[0.0, 9.0]$
	fit_prior	{True, False}
MultinomialNB	alpha	$[0.0, 9.0]$
	fit_prior	{True, False}
ComplementNB	alpha	$[0.0, 9.0]$
	fit_prior	{True, False}
	norm	{True, False}
SVC	C	$[0.03125, 32768.0]$
	kernel	{linear, poly, rbf, sigmoid}
	degree	$[2, 10]$
	gamma	$[3.05 \times 10^{-5}, 8.0]$
	coef0	$[0.0, 10^3]$
	probability	{True, False}
	shrinking	{True,
	decision_function_shape	{ovo, ovr, None}
	tol	$[10^{-10}, 10^{-1}]$
	max_iter	$[10, 10^4]$
	class_weight	{balanced, None}
NuSVC	nu	$[10^{-10}, 1.0]$
	[SVC hyperparameters, except C]	
LogisticRegression	penalty	{l1, l2}
	tol	$[10^{-10}, 10^{-1}]$
	C	$[0.03125, 32768.0]$
	fit_intercept	{True, False}
	max_iter	$[10, 10^4]$
	warm_start	{True, False}
Perceptron	penalty	{l1, l2}
	tol	$[10^{-10}, 10^{-1}]$
	max_iter	$[10, 10^4]$
	warm_start	{True, False}
MLP	learning_rate	{constant, invscaling, adaptive}
	learning_rate_init	$[0.1, 1.0]$
	momentum	$[0.0, 1.0]$
	max_iter	$[10, 10^4]$
	activation	{identity, logistic, tanh, relu}
SGD	penalty	{l1, l2}
	tol	$[10^{-10}, 10^{-1}]$
	max_iter	$[10, 10^4]$
	loss	{hinge, log, modified_huber, squared_hinge, perceptron, squared_loss}
	warm_start	{True, False}
LinearDiscriminantAnalysis	n_components	$[1, \#feats - 1]$
	tol	$[10^{-10}, 10^{-1}]$
QuadraticDiscriminantAnalysis	reg_param	$[0.0, 1.0]$
	tol	$[10^{-10}, 10^{-1}]$

Table 3: Classification configuration space – Part 2.

Algorithm	**Hyperparameter**	**Values**
NearestNeighbors	n_neighbors	$[1, 30]$
	weights	{uniform, distance}
	algorithm	{auto, brute, kd_tree, ball_tree}
	leaf_size	$[5, 100]$
	p	$[1, 15]$
	metric	{euclidean, manhattan, chebyshev, minkowski}
RadiusNeighbors	radius	$[1.0, 30.0]$
	weights	{uniform, distance}
	algorithm	{auto, brute, kd_tree, ball_tree}
	leaf_size	$[5, 100]$
	p	$[1, 15]$
	metric	{euclidean, manhattan, chebyshev, minkowski}
NearestCentroid	shrink_threshold	$[0.0, 30.0]$
	metric	{euclidean, manhattan, chebyshev, minkowski}
Ridge	alpha	$[0.0, 1.0]$
	max_iter	$[10, 10^4]$
	copy_X	{True, False}
	solver	{auto, svd, cholesky, lsqr, sparse_cg, sag, saga}
	tol	$[10^{-10}, 10^{-1}]$
	normalize	{True, False}
	fit_intercept	{True, False}
RidgeCV	cv	$[2, 10]$
	normalize	{True, False}
	fit_intercept	{True, False}
DecisionTree ExtraTree	criterion	{gini, entropy}
	splitter	{best, random}
	max_depth	$[10, 100]$
	max_features	{sqrt, log2, auto}
	min_weight_fraction_leaf	$[0.0, 0.5]$
	max_leaf_nodes	$[2, 100]$
RandomForest ExtraTrees	criterion	{gini, entropy}
	bootstrap	{True, False}
	oob_score	{True, False}
	class_weight	{balanced, balanced_subtree, None}
	n_estimators	$[5, 50]$
	warm_start	{True, False}
	max_features	{sqrt, log2, auto}
	max_depth	$[10, 100]$
	min_weight_fraction_leaf	$[0.0, 0.5]$
	max_leaf_nodes	$[2, 100]$
AdaBoost	n_estimators	$[5, 50]$
	algorithm	{SAMME.R, SAMME}
	learning_rate	$[0.01, 2.0]$
GradientBoosting	loss	{deviance, exponential}
	tol	$[10^{-10}, 10^{-1}]$
	learning_rate	$[10^{-10}, 1.0]$
	presort	{True, False, auto}
	n_estimators	$[5, 50]$
	warm_start	{True, False}
	max_features	{sqrt, log2, auto}
	max_depth	$[10, 100]$
	min_weight_fraction_leaf	$[0.0, 0.5]$
	max_leaf_nodes	$[2, 100]$

```
# Imputation
<imputation> ::= <simpleImputer>

<simpleImputer> ::= SimpleImputer <strategy_imp>
<strategy_imp> ::= mean | median | most_frequent
```

Box 1.2: Defined Grammar – Part 2: Imputation algorithm.

```
# Bounding
<bounding> ::= <normalizer> | <minMaxScaler> | <maxAbsScaler> | <robust_scaler> | <standard_scaler>

<normalizer> ::= Normalizer <norm>
<norm> ::= l1 | l2 | max

<minMaxScaler> ::= MinMaxScaler

<maxAbsScaler> ::= MaxAbsScaler

<robust_scaler> ::= RobustScaler <with_scaling> <with_centering>
<with_scaling> ::= True | False
<with_centering> ::= True | False

<standard_scaler> ::= StandardScaler <with_std> <with_mean>
<with_std> ::= True | False
<with_mean> ::= True | False

# Binarizer
<binarizer> ::= <binarizer_alg>

<binarizer_alg> ::= Binarizer <threshold_bin>
<threshold_bin> ::= RANDFLOAT(0.000001,1000)
```

Box 1.3: Defined Grammar – Part 3: Range manipulation algorithms.

```
# Dimensionality
<dimensionality> ::= <varianceThreshold> | <selectKBest> | <pca> | <incremental_pca> | <fast_ica>
                     | <gaussian_projection> | <sparse_random_projection> | <feature_agglomeration>
                     | <rbf_sampler> | <nystroem> | <truncatedsvd>

<varianceThreshold> ::= VarianceThreshold

<selectKBest> ::= SelectKBest <features_dim>
<features_dim> ::= RANDATT(1,ATT-1)

<pca> ::= PCA <features_dim> <whiten> <svd_solver> <tol_pca> <iterated_power>
<whiten> ::= True | False
<svd_solver> ::= auto | full | arpack | randomized
<tol_pca> ::= RANDFLOAT(0.0,0.1)
<iterated_power> ::= RANDINT(2,20)

<incremental_pca> ::= IncrementalPCA <features_dim> <whiten>

<fast_ica> ::= FastICA <features_dim> <algorithm_fastica> <funct> <max_iter_fastica> <tol_dim> <whiten>
<algorithm_fastica> ::= parallel | deflation
<funct> ::= logcosh | exp | cube
<max_iter_fastica> ::= RANDINT(10,1000)
<tol_dim> ::= RANDFLOAT(0.0000000001,0.1)

<gaussian_projection> ::= GaussianRandomProjection <features_dim> <epsilon>
<epsilon> ::= RANDFLOAT(0.0,1.0)

<sparse_random_projection> ::= SparseRandomProjection <features_dim> <epsilon> <density>
<dense_output>
<density> ::= RANDFLOAT(0.00001,1.0)
<dense_output> ::= True | False

<feature_agglomeration> ::= FeatureAgglomeration <features_dim> <affinity> <compute_full_tree>
<affinity> ::= euclidean ward | euclidean complete | euclidean average | l1 complete | l1 average
               | l2 complete | l2 average | manhattan complete | manhattan average | cosine complete | cosine average
<compute_full_tree> ::= True | False

<rbf_sampler> ::= RBFSampler <features_dim> <gamma_kernelApprox>
<gamma_kernelApprox> ::= RANDFLOAT(0.000030518,8.0)

<nystroem> ::= Nystroem <features_dim> <kernel_dr> <gamma_kernelApprox> <degree_1> <coef0_dim>
<kernel_dr> ::= linear | poly | rbf | sigmoid
<degree_1> ::= RANDINT(2,10)
<coef0_dim> ::= RANDFLOAT(0.0,1000.0)

<truncatedsvd> ::= TruncatedSVD <features_dim> <n_iter> <tol_dim> <algorithm_tsvd>
<n_iter> ::= RANDINT(5,1000)
<algorithm_tsvd> ::= arpack | randomized
```

Box 1.4: Defined Grammar – Part 4: Dimensionality manipulation algorithms.

```
# Naïve Bayes
<gaussian_nb> ::= gaussianNB

<bernoulli_nb> ::= BernoulliNB <binarize> <alpha_nb> <fit_prior>
<binarize> ::= RANDFLOAT(0.0,1.0)
<alpha_nb> ::= RANDFLOAT(0.0,9.0)
<fit_prior> ::= True | False

<multinomial_nb> ::= MultinomialNB <alpha_nb> <fit_prior>

<complement_nb> ::= ComplementNB <alpha_nb> <fit_prior> <norm_cnb>
<norm_cnb> ::= True | False
```

Box 1.5: Defined Grammar – Part 5: Naïve Bayes classifiers.

```
# Lineal models
<svc> ::= SVC <C> <kernel> <degree_kernel> <gamma> <coef0> <probability> <shrinking>
          <decision_function_shape> <tol> <max_iter> <class_weight>
<C> ::= RANDFLOAT(0.03125,32768.0)
<kernel> ::= linear | poly | rbf | sigmoid
<degree_kernel> ::= RANDINT(2,10)
<gamma> ::= RANDFLOAT(0.000030518,8.0)
<coef0> ::= RANDFLOAT(0.0,1000.0)
<probability> ::= True | False
<shrinking> ::= True | False
<decision_function_shape> ::= ovo | ovr | None
<tol> ::= RANDFLOAT(0.0000000001,0.1)
<max_iter> ::= RANDINT(10,10000)
<class_weight> ::= balanced | None

<nu_svc> ::= NuSVC <nu> <kernel> <degree_kernel> <gamma> <coef0> <probability> <shrinking>
             <decision_function_shape> <tol> <max_iter> <class_weight>
<nu> ::= RANDFLOAT(0.0000000001, 1.0)

<logistic_regression> ::= LogisticRegression <penalty> <tol> <C> <fit_intercept> <max_iter> <warm_start>
<penalty> ::= l1 | l2
<fit_intercept> ::= True | False
<warm_start> ::= True | False

<perceptron> ::= Perceptron <penalty> <tol> <max_iter> <warm_start>

<sgd> ::= SGDClassifier <penalty> <tol> <max_iter> <loss> <warm_start>
<loss> ::= hinge | log | modified_huber | squared_hinge | perceptron | squared_loss

<ridge> ::= RidgeClassifier <alpha> <max_iter> <copy_X> <solver_ridge> <tol> <normalize> <fit_intercept>
<alpha> ::= RANDFLOAT(0.0,1.0)
<copy_X> ::= True | False
<solver_ridge> ::= auto | svd | cholesky | lsqr | sparse_cg | sag | saga
<normalize> ::= True | False

<ridge_cv> ::= RidgeClassifierCV <cv> <normalize> <fit_intercept>
<cv> ::= RANDINT(2,10)
```

Box 1.6: Defined Grammar – Part 6: Linear model classifiers.

```
# Neural networks
<mlp> ::= MLPClassifier <learning_rate> <learning_rate_init> <momentum> <max_iter> <activation>
<learning_rate> ::= constant | invscaling | adaptive
<learning_rate_init> ::= RANDFLOAT(0.1,1.0)
<momentum> ::= RANDFLOAT(0.0,1.0)
<max_iter>  ::= RANDINT(10,10000)
<activation> ::= identity | logistic | tanh | relu
```

Box 1.7: Defined Grammar – Part 7: Neural network algorithm.

```
# Nearest neighbors
<knn> ::= KNeighborsClassifier <k> <weights> <k_algorithm> <leaf_size> <p> <d_metric>
<k> ::= RANDINT(1,30)
<weights> ::= uniform | distance
<k_algorithm> ::= auto | brute | kd_tree | ball_tree
<leaf_size> ::= RANDINT(5,100)
<p> ::= RANDINT(1,15)
<d_metric> ::= euclidean | manhattan | chebyshev | minkowski

<radius_neighbours> ::= RadiusNeighborsClassifier <radius> <weights> <k_algorithm> <leaf_size> <p> <d_metric>
<radius> ::= RANDFLOAT(1.0,30.0)

<centroid> ::= NearestCentroid <shrinking_threshold> <d_metric>
<shrinking_threshold> ::= RANDFLOAT(0.0, 30.0)
```

Box 1.8: Defined Grammar – Part 8: Nearest neighbors algorithms.

```
# Discriminant analysis
<lda> ::= LinearDiscriminantAnalysis <features> <tol>
<features> ::= RANDATT(1,ATT-1)
<tol> ::= RANDFLOAT(0.0000000001,0.1)

<qda> ::= QuadraticDiscriminantAnalysis <reg_param> <tol>
<reg_param> ::= RANDFLOAT(0.0,1.0)
```

Box 1.9: Defined Grammar – Part 9: Discriminant analysis algorithms.

```
<decision_tree> ::= DecisionTreeClassifier <criterion> <splitter> <max_depth> <max_features>
                    <min_weight_fraction_leaf> <max_leaf_nodes>
<criterion> ::= gini | entropy
<splitter> ::= best | random
<max_depth> ::= RANDINT(10,100)
<max_features> ::= sqrt | log2 | auto
<min_weight_fraction_leaf> ::= RANDFLOAT(0.0,0.5)
<max_leaf_nodes> ::= RANDINT(2,100)

<extra_tree> ::= ExtraTreeClassifier <criterion> <splitter> <class_weight> <max_features> <max_depth>
                 <min_weight_fraction_leaf> <max_leaf_nodes>
<class_weight> ::= balanced | None
```

Box 1.10: Defined Grammar – Part 10: Tree algorithms.

```
# Ensembles
<random_forest> ::= RandomForestClassifier <criterion> <bootstrap_and_oob> <class_weight_Trees> <n_estimators>
                    <warm_start> <max_features> <max_depth> <min_weight_fraction_leaf> <max_leaf_nodes>
<criterion> ::= gini | entropy
<bootstrap_and_oob> ::= True True | True False | False False
<class_weight_Trees> ::= balanced | balanced_subsample | None
<n_estimators> ::= RANDINT(5,50)
<warm_start> ::= True | False
<max_features> ::= sqrt | log2 | auto
<max_depth> ::= RANDINT(10,100)
<min_weight_fraction_leaf> ::= RANDFLOAT(0.0,0.5)
<max_leaf_nodes> ::= RANDINT(2,100)

<extra_trees> ::= ExtraTreesClassifier <criterion> <bootstrap_and_oob> <class_weight_Trees> <n_estimators>
                  <warm_start> <max_features> <max_depth> <min_weight_fraction_leaf> <max_leaf_nodes>

<ada_boost> ::= AdaBoostClassifier <algorithm_ada> <n_estimators> <learning_rate_ada>
<algorithm_ada> ::= SAMME.R | SAMME
<learning_rate_ada> ::= RANDFLOAT(0.01,2.0)

<gradient_boosting> ::= GradientBoostingClassifier <loss_gradient> <tol> <learning_rate_gradient> <presort> <warm_start>
                        <n_estimators> <max_features> <max_depth> <min_weight_fraction_leaf> <max_leaf_nodes>
<loss_gradient> ::= deviance | exponential
<tol> ::= RANDFLOAT(0.0000000001,0.1)
<learning_rate_gradient> ::= RANDFLOAT(0.0000000001,1.0)
<presort> ::= True | False | auto
```

Box 1.11: Defined Grammar – Part 11: Ensemble algorithms.

2 Algorithms Partitioning

This section shows how preprocessing and classification algorithms were partitioned.

2.1 Preprocessing Algorithms

- Imputation algorithms (A_4)
  - SimpleImputer
- Data range manipulation algorithms (A_5)
  - Normalizer
  - MinMaxScaler
  - MaxAbsScaler
  - RobustScaler
  - StandardScaler
  - Binarizer
- Dimensionality manipulation algorithms (A_6)
  - VarianceThreshold
  - SelectKBest
  - PCA
  - IncrementalPCA
  - FastICA
  - GaussianRandomProjection
  - SparseRandomProjection
  - FeatureAgglomeration
  - RBFSampler
  - Nystroem
  - TruncatedSVD

2.2 Classification Algorithms

- Naïve Bayes (A_7)
  - GaussianNB
  - BernoulliNB
  - MultinomialNB
  - ComplementNB
- Linear models (A_8)
  - SVC
  - NuSVC
  - LogisticRegression
  - Perceptron
  - SGDClassifier
  - RidgeClassifier
  - RidgeClassifierCV
- Neural networks (A_9)
  - MLPClassifier
- Nearest neighbors (A_{10})
  - KNeighborsClassifier
  - RadiusNeighborsClassifier
  - NearestCentroidClassifier
- Discriminant analysis (A_{11})
  - LinearDiscriminantAnalysis
  - QuadraticDiscriminantAnalysis
- Trees (A_{12})
  - DecisionTreeClassifier
  - ExtraTreeClassifier
- Ensembles (A_{13})
  - RandomForestClassifier
  - ExtraTreesClassifier
  - AdaBoostClassifier
  - GradientBoostingClassifier

References

1. Sipser, M.: Introduction to the theory of computation. Cengage Learning, 3rd edn. (2012)