AutoML challenge winners

Victor Kocheganov

March 21, 2016

The most frequent prize-winners

djajetic (Damir Jajetic (Croatia))

4 out of 8 prizes: Final2 (2nd), AutoML3 (only one), Final3 (2nd), AutoML4 (2nd)

aad_freiburg Frank Hutter and collaborators (Freiburg, Germany)

7 out of 8 prizes: Final0 (3rd), AutoML1 (1st), Final1 (1st), AutoML2 (2nd), Final2 (3rd), Final3 (1st), AutoML4 (1st)

The most frequent prize-winners

djajetic (Damir Jajetic (Croatia))

4 out of 8 prizes: Final2 (2nd), AutoML3 (only one), Final3 (2nd), AutoML4 (2nd)

aad_freiburg Frank Hutter and collaborators (Freiburg, Germany)

7 out of 8 prizes: Final0 (3rd), AutoML1 (1st), Final1 (1st), AutoML2 (2nd), Final2 (3rd), Final3 (1st), AutoML4 (1st)

ideal.intel.analytics

4 out of 4 code-free prizes: Final0 (1st), Final1 (2nd), Final2 (2nd), Final3 (3rd)

The most frequent prize-winners

djajetic (Damir Jajetic (Croatia))

4 out of 8 prizes: Final2 (2nd), AutoML3 (only one), Final3 (2nd), AutoML4 (2nd)

aad_freiburg Frank Hutter and collaborators (Freiburg, Germany)

7 out of 8 prizes: Final0 (3rd), AutoML1 (1st), Final1 (1st), AutoML2 (2nd), Final2 (3rd), Final3 (1st), AutoML4 (1st)

ideal.intel.analytics

4 out of 4 code-free prizes: Final0 (1st), Final1 (2nd), Final2 (2nd), Final3 (3rd)

djajetic approach

Simple cross validation through models

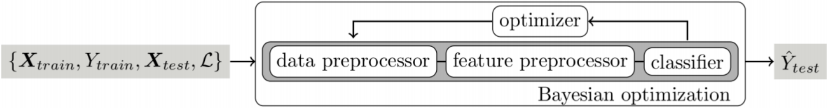
linear_model.LogisticRegression
naive_bayes.GaussianNB
f16.RandomForestClassifier
ensemble.GradientBoostingClassifier
f16.ExtraTreesClassifier
ensemble.AdaBoostClassifier
ensemble.AdaBoostClassifier (base_estimator=f16.RF)
neighbors.KNeighborsClassifier

djajetic approach

```
models = [
        {"model": 'linear_model.LogisticRegression(random_state=1)',
        "blend_group":   "LC",
            "getter":   "Lestimators = model_last.get_params()['penalty']",
            "updater":   "tries_left = 0",
             "setter":   "return in updater will end process",
             "generator":  "penalty='l2', dual=False, C=1.0 @@ " \
                            "penalty='l1', dual=False, C=1.0 @@ " \
                            "penalty='l2', dual=False, C=2.0 @@ " \
                            "penalty='l2', dual=False, C=0.5 @@ " \
                            "penalty='l2', dual=True, C=1.0 @@ " \
                            "penalty='l2', dual=True, C=0.5 @@ " \
                            "penalty='l2', dual=False, C=4.0 @@ " \
                            "penalty='l2', dual=False, C=8.0 @@ "
                            "penalty='l2', dual=False, C=1.0 @@ " \
                            "penalty='l1', dual=False, C=1.0 @@ " \
                            "penalty='l2', dual=False, C=2.0 @@ " \
                            "penalty='l2', dual=False, C=0.5 @@ " \
                            "penalty='l2', dual=True, C=1.0 @@ " \
                            "penalty='l2', dual=True, C=0.5 @@ " \
                            "penalty='l2', dual=False, C=4.0 @@ " \
                            "penalty='l2', dual=False, C=8.0 @@ "
        },
        {"model": 'naive_bayes.GaussianNB()',
        "blend_group":   "NB",
            "getter":   "Lestimators = model_last.get_params()",
            "updater":   "tries_left = 0",
             "setter":   "",
             "generator":  ""
        },
```

aad_freiburg approach

State-of-the-art before this guy (Auto-WEKA):



Data preprocessing

Algorithms that change data values

1. Rescaling of inputs
2. Imputation of missing values
3. Balancing of the target classes

Feature preprocessing

Algorithms that don't change data, but reduce number of features

1. PCA
2. Linear SVM (non zero model coefficients)
3. Feature agglomeration
4. Random forest
5. Extremly randomized forest
6. ...

Feature preprocessing

Well-established classification algorithms

1. General linear models
2. SVM
3. Discriminant analysis
4. Nearest neighbors
5. Naive Bayes
6. Decision trees
7. Ensemble methods

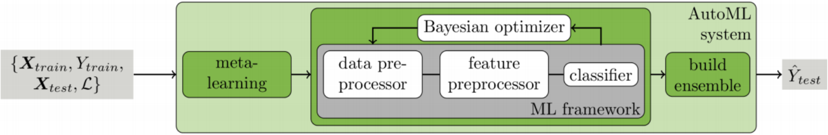
Auto-sklearn features result list

name	#λ	cat (cond)	cont (cond)
AdaBoost (AB)	3	-	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	3	1 (-)	2 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	2	-	2 (-)
linear SVM	5	3 (-)	2 (-)
kernel SVM	8	3 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
ridge regression (RR)	2	-	2 (-)
SGD	9	3 (-)	6 (3)

name	#λ	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (-)
feature agglomeration	3	2 (-)	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	5	3 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

aad_freiburg approach

This guy added two more stages:



Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
    - Simple features (N samples, N features, N classes etc)

Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
    - Simple features (N samples, N features, N classes etc)
    - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)

Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
   - Simple features (N samples, N features, N classes etc)
   - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)
   - Statistical metafeatures (Departure from normality, univariate skewness and kurosis etc)

Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
    - Simple features (N samples, N features, N classes etc)
    - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)
    - Statistical metafeatures (Departure from normality, univariate skewness and kurosis etc)
2. Using Bayesian optimizer for each dataset choose ML framework with strongest performance

Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
   - Simple features (N samples, N features, N classes etc)
   - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)
   - Statistical metafeatures (Departure from normality, univariate skewness and kurosis etc)
2. Using Bayesian optimizer for each dataset choose ML framework with strongest performance
3. For a new dataset D find k nearest from the given 140 ones (L_1-metric on metafeatures dataset).

Meta-Learning

This is used to warm-start Bayesian optimizer

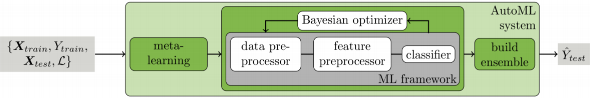
1. Given 140 datasets, compute it's metafeatures (total 38)
    - Simple features (N samples, N features, N classes etc)
    - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)
    - Statistical metafeatures (Departure from normality, univariate skewness and kurosis etc)
2. Using Bayesian optimizer for each dataset choose ML framework with strongest performance
3. For a new dataset D find k nearest from the given 140 ones (L_1-metric on metafeatures dataset).
4. Warm-start Bayesian optimizer with k found parameters

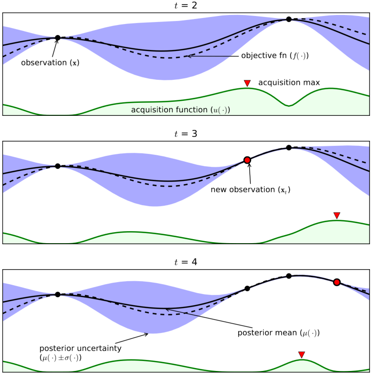
Meta-Learning

This is used to warm-start Bayesian optimizer

1. Given 140 datasets, compute it's metafeatures (total 38)
    - Simple features (N samples, N features, N classes etc)
    - Information-theoretic features (entropy of features, Entropy of classes, feature noisiness etc)
    - Statistical metafeatures (Departure from normality, univariate skewness and kurosis etc)
2. Using Bayesian optimizer for each dataset choose ML framework with strongest performance
3. For a new dataset D find k nearest from the given 140 ones (L_1-metric on metafeatures dataset).
4. Warm-start Bayesian optimizer with k found parameters

Bayesian optimization reminder





Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a validation set

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a validation set
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a validation set
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used
4. Return the ensemble from the nested set of ensembles that has maximum performance on the validation set

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a validation set
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used
4. Return the ensemble from the nested set of ensembles that has maximum performance on the validation set

Ensemble building

After Bayesian optimization one gets set of ML frameworks:

$$\mathrm{M} = \{M_i\}$$

Algorithm:

1. Start with the empty ensemble
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the error metric on a validation set
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used
4. Return the ensemble from the nested set of ensembles that has maximum performance on the validation set

Ensemble building

With these improvements

1. Selection with replacement: adds model only if it increases performance

Ensemble building

With these improvements

1. Selection with replacement: adds model only if it increases performance
2. Sorted Ensemble Initialization: start with non empty ensemble

Ensemble building

With these improvements

1. Selection with replacement: adds model only if it increases performance
2. Sorted Ensemble Initialization: start with non empty ensemble
3. Bagged Ensemble Selection: repeat all the above several (20) times for bagged models

Ensemble building

With these improvements

1. Selection with replacement: adds model only if it increases performance
2. Sorted Ensemble Initialization: start with non empty ensemble
3. Bagged Ensemble Selection: repeat all the above several (20) times for bagged models

Ensemble building

With these improvements

1. Selection with replacement: adds model only if it increases performance
2. Sorted Ensemble Initialization: start with non empty ensemble
3. Bagged Ensemble Selection: repeat all the above several (20) times for bagged models

Auto-sklearn results