On performance of Meta-learning Templates on Different Datasets

Pavel Kordík, Jan Černý

Czech Technical University in Prague
Department of Computer Systems
Faculty of Information Technology
Computational Intelligence Group
Overview

• Introduction to predictive modeling
  o Base models
  o Optimization of topology and parameters
  o Ensembles
• Meta-learning templates explained
• Benchmarking results
• Landmarking experiment
• Evaluation
• Conclusion
Meta-learning template – An Example

Ensembles:
- Bagging (Bootstrap Aggregating),
- Boosting,
- Stacking,
- and many others

Base algorithms:
- DT (Decision Tree),
- KNN (K-Nearest Neighbor),
- NN (Neural Network),
- SVM (Support Vector Machine),
- and many others

ClassifierBagging\{DecisionTree\(\text{splitmin}=5\),ClassifierBoosting\{2NearestNeighborClassifier,StackingClassifier(SVM)\{NeuralNetClassifier,DecisionTree\(\text{splitmin}=2\)\},5NearestNeighborClassifier,DecisionTree\(\text{splitmin}=10\)\}
Meta-learning template execution

Template execution:
- Distributing data
- Building base models
- Finalizing meta models

Meta-learning templates

Bootstrap sampling

DT (sm=5)

Sequential weighted sampling

5NN

DT (sm=10)

2NN

Stacking (SVM)
Meta-learning template execution

Template execution:
- Distributing data
- Building base models
- Finalizing meta models

Meta-learning template execution

Template execution:
- Distributing data
- Building base models
- Finalizing meta models
Meta-learning template

Template execution:

- Bagging
  - DT
  - Boosting
  - 5NN
  - DT
    - 2NN
    - Stacking (SVM)
      - NN
      - DT

Model finished
Using the model

Model recall:

- Input vectors propagation
- Base models recall
- Output blending and propagation

Input vector

Bagging

DT

Boosting

5NN

DT

2NN

Stacking (SVM)

NN

DT
Using the model

Model recall:

- Input vectors propagation
- Base models recall
- Output blending and propagation
Using the model

Model recall:

- Input vectors propagation
- Base models recall
- Output blending and propagation

Combined by the SVM meta-model
Using the model

**Model recall:**
- Input vectors propagation
- Base models recall
- Output blending and propagation

Combined by the weighted voting
Using the model

Model recall:

- Input vectors propagation
- Base models recall
- Output blending and propagation
Evolution of templates

Genetic programming:

Fitness of template – average generalization performance of models produced by the template
## Benchmarking results - templates

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Meta-learning templates evolved on benchmarking datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>StackingProbabilities{4x KNN(k=2, vote=true, measure=ManhattanDistance)}</td>
</tr>
<tr>
<td>Balance</td>
<td>Boosting{57x ClassifierModel{&lt;outputs&gt;x PolynomialModel(degree=4)}}</td>
</tr>
<tr>
<td>Breast</td>
<td>ClassifierModel{&lt;outputs&gt;x CascadeGenModel[7x CascadeGenModel[5x GaussianModel]]}</td>
</tr>
<tr>
<td>Diabetes</td>
<td>SVM(kernel=dot)</td>
</tr>
<tr>
<td>Ecoli</td>
<td>ClassifierArbitrating{2x ClassifierBagging{3x SVM(kernel=anova)}}</td>
</tr>
<tr>
<td>Heart</td>
<td>KNN(k=15, vote=false, measure=CosineSimilarity)</td>
</tr>
<tr>
<td>Texture1</td>
<td>ClassifierArbitrating{4x ClassifierModel{&lt;outputs&gt;x PolynomialModel(degree=2)}}</td>
</tr>
<tr>
<td>Texture2</td>
<td>CascadeGenProb{8x Boosting{2x ClassifierModel{&lt;outputs&gt;x ExpModel}}}</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>= DecisionTree(maxdepth=20, conf=0.25, alt=10)</td>
</tr>
<tr>
<td>Spirals</td>
<td>CascadeGenProb{8x ClassifierArbitrating{4x KNN(k=3, vote=false, measure=MixedEuclideanDistance)}}</td>
</tr>
<tr>
<td>Pendigits</td>
<td>KNN(k=3, vote=false, measure=CosineSimilarity)</td>
</tr>
<tr>
<td>Vehicle</td>
<td>ClassifierArbitrating{6x ClassifierModel{&lt;outputs&gt;x DivideModel(mult=6.68)}7x PolynomialModel(degree=3)}}</td>
</tr>
<tr>
<td>Wine</td>
<td>CascadeGenProb{9x ClassifierModel{&lt;outputs&gt;x BoostingRTModel(tr=0.1)}8x GaussianModel}</td>
</tr>
<tr>
<td>Spambase</td>
<td>ClassifierModel{&lt;outputs&gt;x CascadeGenModel[9x SigmoidModel]}</td>
</tr>
<tr>
<td>Segment</td>
<td>Boosting{17x DecisionTree(maxdepth=24, conf=0.082, alt=0)}</td>
</tr>
<tr>
<td>Fourier</td>
<td>NeuralNetClassifier(net=-1x0, epsilon=0.00001, learn=0.3, momentum=0.2)</td>
</tr>
<tr>
<td>Spirals+3</td>
<td>KNN(k=3, vote=true, measure=EuclideanDistance)</td>
</tr>
<tr>
<td>Spread</td>
<td>CascadeGenProb{5x CascadeGenProb{3x KNN(k=9, vote=true, measure=CosineSimilarity)}}</td>
</tr>
<tr>
<td>Meta-learn. Templates</td>
<td>Glass</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>83.08</td>
</tr>
<tr>
<td>Baseline algorithms performance [%]</td>
<td></td>
</tr>
<tr>
<td>Bayes</td>
<td>46.38</td>
</tr>
<tr>
<td>DT</td>
<td>63.81</td>
</tr>
<tr>
<td>KNN</td>
<td>67.33</td>
</tr>
<tr>
<td>NeuNet</td>
<td>61.96</td>
</tr>
<tr>
<td>SVM</td>
<td>50.38</td>
</tr>
</tbody>
</table>
**Landmarking experiment**

Run evolved templates on all datasets

What is the accuracy of template evolved on dataset X and executed on dataset Y?

<table>
<thead>
<tr>
<th>Data / Cfg</th>
<th>Tbal</th>
<th>Tbre</th>
<th>Tdia</th>
<th>Tecm</th>
<th>Tfou</th>
<th>Tglm</th>
<th>Thea</th>
<th>Tion</th>
<th>Tseg</th>
<th>Tspin</th>
<th>Tspr</th>
<th>Ttxt1</th>
<th>Ttxt2</th>
<th>Tveh</th>
<th>Twine</th>
</tr>
</thead>
<tbody>
<tr>
<td>balance</td>
<td>0.99</td>
<td>0.88</td>
<td>0.91</td>
<td>0.87</td>
<td>0.79</td>
<td>0.97</td>
<td>0.90</td>
<td>0.89</td>
<td>0.42</td>
<td>0.80</td>
<td>0.90</td>
<td>0.91</td>
<td>0.98</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>breast</td>
<td>0.94</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td>0.88</td>
<td>0.96</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.72</td>
<td>0.75</td>
<td>0.78</td>
<td>0.77</td>
<td>0.71</td>
<td>0.66</td>
<td>0.76</td>
<td>0.73</td>
<td>0.70</td>
<td>0.73</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>0.76</td>
<td>0.75</td>
</tr>
<tr>
<td>ecoli</td>
<td>0.41</td>
<td>0.87</td>
<td>0.12</td>
<td>0.85</td>
<td>0.82</td>
<td>0.77</td>
<td>0.81</td>
<td>0.86</td>
<td>0.82</td>
<td>0.84</td>
<td>0.84</td>
<td>0.12</td>
<td>0.84</td>
<td>0.86</td>
<td>0.80</td>
</tr>
<tr>
<td>fourier</td>
<td>0.80</td>
<td>0.82</td>
<td>0.81</td>
<td>0.80</td>
<td>0.82</td>
<td>0.81</td>
<td>0.77</td>
<td>0.78</td>
<td>0.74</td>
<td>0.81</td>
<td>0.73</td>
<td>0.81</td>
<td>0.84</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>glass</td>
<td>0.66</td>
<td>0.69</td>
<td>0.63</td>
<td>0.51</td>
<td>0.70</td>
<td>0.76</td>
<td>0.50</td>
<td>0.62</td>
<td>0.66</td>
<td>0.69</td>
<td>0.64</td>
<td>0.64</td>
<td>0.66</td>
<td>0.67</td>
<td>0.48</td>
</tr>
<tr>
<td>heart</td>
<td>0.54</td>
<td>0.81</td>
<td>0.51</td>
<td>0.80</td>
<td>0.78</td>
<td>0.80</td>
<td>0.82</td>
<td>0.76</td>
<td>0.74</td>
<td>0.78</td>
<td>0.84</td>
<td>0.53</td>
<td>0.82</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.47</td>
<td>0.84</td>
<td>0.51</td>
<td>0.85</td>
<td>0.88</td>
<td>0.86</td>
<td>0.90</td>
<td>0.91</td>
<td>0.92</td>
<td>0.87</td>
<td>0.86</td>
<td>0.52</td>
<td>0.92</td>
<td>0.89</td>
<td>0.91</td>
</tr>
<tr>
<td>segment</td>
<td>0.27</td>
<td>0.95</td>
<td>0.15</td>
<td>0.88</td>
<td>0.97</td>
<td>0.96</td>
<td>0.80</td>
<td>0.95</td>
<td>0.96</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
<td>0.97</td>
<td>0.95</td>
<td>0.80</td>
</tr>
<tr>
<td>spirals</td>
<td>0.49</td>
<td>0.46</td>
<td>0.45</td>
<td>0.47</td>
<td>0.78</td>
<td>0.49</td>
<td>0.48</td>
<td>0.49</td>
<td>0.42</td>
<td>0.85</td>
<td>0.45</td>
<td>0.47</td>
<td>0.46</td>
<td>0.47</td>
<td>0.48</td>
</tr>
<tr>
<td>spread</td>
<td>0.24</td>
<td>0.25</td>
<td>0.10</td>
<td>0.07</td>
<td>0.63</td>
<td>0.06</td>
<td>0.03</td>
<td>0.06</td>
<td>0.03</td>
<td>0.57</td>
<td>0.83</td>
<td>0.11</td>
<td>0.86</td>
<td>0.06</td>
<td>0.03</td>
</tr>
<tr>
<td>texture1</td>
<td>0.99</td>
<td>0.93</td>
<td>0.99</td>
<td>0.95</td>
<td>0.97</td>
<td>0.95</td>
<td>0.88</td>
<td>0.99</td>
<td>0.90</td>
<td>0.95</td>
<td>0.97</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.88</td>
</tr>
<tr>
<td>texture2</td>
<td>0.47</td>
<td>0.45</td>
<td>0.40</td>
<td>0.22</td>
<td>0.43</td>
<td>0.40</td>
<td>0.22</td>
<td>0.34</td>
<td>0.43</td>
<td>0.63</td>
<td>0.40</td>
<td>0.66</td>
<td>0.44</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>vehicle</td>
<td>0.76</td>
<td>0.72</td>
<td>0.75</td>
<td>0.66</td>
<td>0.71</td>
<td>0.73</td>
<td>0.43</td>
<td>0.73</td>
<td>0.73</td>
<td>0.76</td>
<td>0.74</td>
<td>0.78</td>
<td>0.74</td>
<td>0.74</td>
<td>0.45</td>
</tr>
<tr>
<td>wine</td>
<td>0.95</td>
<td>0.96</td>
<td>0.98</td>
<td>0.99</td>
<td>0.95</td>
<td>0.96</td>
<td>0.92</td>
<td>0.98</td>
<td>0.92</td>
<td>0.97</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>
Landmarking experiment

Resulting performances on individual datasets

balance  breast  diabetes  ecoli
fourier  glass  heart  ionosphere
segment  spirals  spread  texture1

texture2  vehicle  wine

Tfourier  Tecoli  Tdiab  Tgla
Thear  Tbrea  Tbal  Twine
Tions  Tseg  Tspirals  Tspread
Ttext1  Ttext2
Concluding remarks

• Hierarchical templates are specially useful for problems with a complex decision boundary.
• They are best in average, even without being optimized to concrete datasets.
MODGEN provides you with the meta-learning templates for predictive modeling. Upload your data and we compute the performance of several well-known algorithms, evolve the best combination of these, and produce the report.

Coming soon ...