OptiML and Fusions
Automating Machine Learning

Charles Parker
VP ML Algorithms, BigML, Inc
Model Choices

<table>
<thead>
<tr>
<th>Early Stage</th>
<th>Mid Stage</th>
<th>Late Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapid Prototyping</td>
<td>Proven Application</td>
<td>Critical Performance</td>
</tr>
</tbody>
</table>

- Single Tree Models
- Random Decision Forests
- Deepnets

- Logistic Regression
- Decision Trees
- Boosted Trees

Increasing Data Size/Complexity

Decreasing Interpretability / Better Representation / Longer Training

TOO HARD
The success of a Deepnet is dependent on getting the right network structure for the dataset.

But, there are too many parameters:
- Nodes, layers, activation function, learning rate, etc…

And setting them takes significant expert knowledge.

Solution:
- Metalearning (a good initial guess)
- Network search (try a bunch)
Key Insight: We can solve any parameter selection problem in a similar way.

- Each resource has several parameters that impact quality
  - Number of trees, missing splits, nodes, weight
- Rather than trial and error, we can use ML to find ideal parameters
- Why not make the model type, Decision Tree, Boosted Tree, etc, a parameter as well?
- Similar to Deepnet network search, but finds the optimum machine learning algorithm and parameters for your data automatically
OptiML Demo
Fusions Inside

*Key Insight*: ML algorithms each have unique strengths and weaknesses

- Fuse any set of models into a new “fusion”
  - Must have the same objective type
  - Inputs and feature space can differ
- Weights can be added
  - Give more importance to individual models
- Fusions can be fused as well
  - Especially useful for fusing OptiML models
Performance thru Diversity

Dataset

Optimized Deepnet

Optimized Ensemble

Optimized Logistic Regression

Better?
Fusion Demo #1
**Fusions: Under the Hood**

### Classification

<table>
<thead>
<tr>
<th>Model</th>
<th>Prediction</th>
<th>Probability</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble</td>
<td>TRUE</td>
<td>%56</td>
<td>1</td>
</tr>
<tr>
<td>Deepnet</td>
<td>FALSE</td>
<td>%67</td>
<td>1</td>
</tr>
<tr>
<td>Model</td>
<td>TRUE</td>
<td>%78</td>
<td>2</td>
</tr>
</tbody>
</table>

Fusion:

\[
P(\text{TRUE}) = \frac{[56+(100-67)+2*78]}{4}
\]

### Regression

<table>
<thead>
<tr>
<th>Model</th>
<th>Prediction</th>
<th>Error</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble</td>
<td>156.78</td>
<td>12.56</td>
<td>1</td>
</tr>
<tr>
<td>Deepnet</td>
<td>139.55</td>
<td>9.88</td>
<td>1</td>
</tr>
<tr>
<td>Model</td>
<td>172.10</td>
<td>23.76</td>
<td>2</td>
</tr>
</tbody>
</table>

Fusion:

160.13  17.49
Fusions: Like any BigML Model

- Fully accessible thru API and WhizzML
- Bindings have support for local predictions

```whizzml
;; WhizzML - create a fusion
(define my-fusion (create-fusion "models" my-best-models))
```

```python
# Python - create a fusion
fusion = api.create_fusion(["model/5af06df94e17277501000010",
                            "logisticregression/5af06df84e17277502000019",
                            "deepnet/5af06df84e17277502000016",
                            "ensemble/5af06df74e1727750100000d"])
```
**Decision Boundary Smoothness**

**Single Tree:**
- Outcome changes abruptly near decision boundary
- And not at all parallel to the boundary
- This can be “surprising”

**Single Tree + Deepnet:**
- Keep the interpretability of the tree
- But with a more nuanced decision boundary
Feature Stability

Feature Importance: Different subsets of features may have similar modeling performance

Fusing models gives better resilience against missing values as well as ensuring that all relevant features are utilized.
Data significance over time:

- Some data may change significance in different times
- Short-term user behavior versus long-term
- Weights can set to account for significance of time

1 Day

1 Week

1 Month
Consider a 3-class objective

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
<th>Maybe</th>
</tr>
</thead>
</table>

- Really only care about “yes” versus “not yes”
- A single model may struggle to separate the two negative classes
Feature Space Optimization

Model Skills: Some ML algorithms “generally” do better on some feature types:

- RDF for sparse text vectors
- LR/Deepnets for numeric features
- Trees for categorical features
Fusions Demo #2
Your Turn!

• *Note*: best to work in groups to limit computation time
• Configure an OptiML of the Diabetes 80%
  • Limit number of model candidates to 10
  • Disable Deepnets from the search
  • Optimize for identifying diabetes

• While the OptiML is running:
  • Build a Fusion from any set of Diabetes 80% models
  • Evaluate the fusion with the 20%
  • How does it compare to previous models?

• Returning to the OptiML
  • Evaluate the top performing model with the 20%
  • How does it perform?