Feature Selection with Monte-Carlo Tree Search

Robert Pinsler
20.01.2015
Agenda

1. Feature Selection
2. Feature Selection as a Markov Decision Process
3. Feature UCT Selection
4. Experimental Validation
5. Summary and Outlook
Motivation

- less to store and collect
- faster to process

```
Less data
```

```
Reduced
generalization error
```

- less noise (less irrelevant features)
- simpler hypothesis spaces (less redundant features)

```
Better understanding
```

- easier to understand
- easier to visualize
Supervised Approaches

- **Filter**
  - independently rank features with score function, select top n
  - no correlations or redundancy

- **Wrapper**
  - explore superset of feature, measure generalization error of all subsets
  - whole combinatorial optimization problem

- **Embedded**
  - combine feature selection and learning
  - no correlations or redundancy

- exploration vs. exploitation dilemma
FS as a Markov Decision Process

\[ \mathcal{M} = (S, A, P, R) \]

- \( \mathcal{F} \) set of features plus \textit{stopping} feature \( f_s \)
- \( S = 2^\mathcal{F} \) final states: all states \( F \subseteq \mathcal{F} \) containing \( f_s \)
- \( A = \{ \text{add } f, f \in \mathcal{F} \} \) action space
- \( P : S \times \mathcal{F} \times S \mapsto \mathbb{R}^+ \) transition function
- \( V : S \mapsto [0, 1] \) reward function (also denoted as \( R \))
- \( \pi : S \mapsto A \) policy

Goal: find \textit{optimal} policy

\[ \pi^* = \arg \min_{\pi} \text{Err} \left( A(F_{\pi}) \right) \]

- \( A(F \setminus \{ f_s \}) \) learned hypothesis based on \( F \)
- \( \text{Err}(A(F)) \) generalization error of learned hypothesis
Finding an Optimal Policy

\[ \pi^* = \arg\min_{\pi} \text{Err} \left( \mathcal{A}(F_{\pi}) \right) \]

Following Bellman’s optimality principle

\[ V^*(F) = \begin{cases} 
\text{Err}(\mathcal{A}(F)) & \text{if } F \text{ is final} \\
\min_{f \in \mathcal{F}\setminus F} V^*(F \cup \{f\}) & \text{otherwise}
\end{cases} \]

\[ \pi^*(F) = \arg\min_{f \in \mathcal{F}\setminus F} V^*(F \cup \{f\}) \]

optimal, but *intractable* (state space exponential in #features)

Why not cast problem into 1-player game and use MCTS with UCT?
Feature Selection as a 1-Player Game

- Formalize FS as Markov Decision Process
- MDP can be solved with Reinforcement Learning
- Cast problem as 1-player game
- Use MCTS with UCT!
Restrict number of arms

**UCB1-tuned** instead of UCB1

<table>
<thead>
<tr>
<th>limit exploration term by including empirical variance of rewards</th>
</tr>
</thead>
</table>

\[
a^* = \arg \max_{a \in A} \left\{ \hat{\mu}_{F,a} + \sqrt{\frac{c_e \ln(T_F)}{t_{F,a}}} \min \left( \frac{1}{4}, \hat{\sigma}^2_{F,a} + \sqrt{\frac{2 \ln(T_F)}{t_{F,a}}} \right) \right\}
\]

**Continuous heuristic**
set \( c_e \) to very small value

**Discrete heuristic**
consider only \( \lfloor T_F^b \rfloor \) children \((b < 1)\)

→ **progressive widening**
Rapid Action Value Estimation (RAVE)

**AMAF heuristic**
incorporate additional knowledge gained within search

\[
g\text{-RAVE}_f = \text{average}\{V(F_t), f \in F_t\} \\
\ell\text{-RAVE}_{F,f} = \text{average}\{V(F_t), F \sim F_t, f \in F_t\}
\]

associate *RAVE score* to each size of feature set:

\[
g\text{-RAVE}_{f_s(d)} = \text{average}\{V(F_t), |F_t| = d + 1\}
\]
Selection of New Nodes

**Discrete heuristic**
select top-ranked feature after RAVE whenever integer part of $T_F^b$ is incremented

**Continuous heuristic**
replace *UCB1-tuned* formula by

$$(1-\alpha) \cdot \hat{\mu}_{F,f} + \alpha \left( (1 - \beta) \cdot \ell\text{-RAVE}_{F,f} + \beta \cdot g\text{-RAVE}_f \right)$$

$$+ \sqrt{\frac{c_e \ln (T_F)}{t_{F,f}}} \min \left( \frac{1}{4}, \frac{\sigma^2_{F,f}}{t_{F,f}} + \sqrt{\frac{2 \ln (T_F)}{t_{F,f}}} \right)$$

\[\alpha = \frac{c}{c + t_{F,f}}\] impact of \(\ell\text{-RAVE}\)

\[\beta = \frac{i}{c_l + t_l}\] impact of \(g\text{-RAVE}\)

\(t_l\) no. of iterations involved in \(\ell\text{-RAVE}\) computation

\(t_{F,f}\) no. of times feature \(f\) has been selected in \(F\)

\(c, c_l\) parameter
Instant Reward Function

**k-nearest neighbor (k-NN)**

\[ s_F(z) = \left| \{ z' \in \mathcal{N}_{F,k}(x), \ y' > 0 \} \right| \]

- \( d_F \): Euclidean distance based on features in \( F \)
- \( \mathcal{L} \): training set
- \( \mathcal{V} \): aggressive subsample of \( \mathcal{L} \)
- \( z = (x, y) \): labeled example in \( \mathcal{V} \)
- \( \mathcal{N}_{F,k}(x) \): set of k-NN of \( x \) in \( \mathcal{L} \) after \( d_F \)
- \( s_F(z) \): number of positive examples among \( \mathcal{N}_{F,k}(x) \)

**Area under the ROC curve (AUC)** *

aka Mann Whitney Wilcoxon sum of ranks test

\[
V(F) = \frac{\left| \{(z, z') \in \mathcal{V}^2, s_F(x) < s_F(x'), y < y'\} \right|}{\left| \{(z, z') \in \mathcal{V}^2, y < y'\} \right|}
\]

* Note that 0 really is the minimum as we do not simply predict a class which we could change. Instead we want to find a feature set with minimum generalization error.
**Feature UCT Selection (FUSE)**

**FUSE**

**Input:** number of iterations $T$ and many-armed behavior MA

**Output:** search tree $T$ and g-RAVE score

Initialize $T \leftarrow \emptyset$, $\forall f, \text{g-RAVE}(f) = 0$

for $t = 1$ to $T$ do

Iterate($T$, g-RAVE, $\emptyset$)

end for

**Iterate**

**Input:** search tree $T$, score g-RAVE, subset $F$

**Output:** reward $V$

if $F$ final then

$V \leftarrow V(F \setminus \{f_s\})$ ; Update g-RAVE

else

if $t(F) \neq 0$ then

if $\text{MA} = \text{progressive widening}$ then

$f^* \leftarrow \arg\max_{f \in \text{AllowedFeatures}(F)} \text{UCB1-tuned}(F, f)$

else

$f^* \leftarrow \arg\max_{f \in F \setminus F} \text{tradeoff UCB/RAVE}(F, f)$

end if

$V \leftarrow \text{iterate}(T, \text{g-RAVE}, F \cup \{f^*\})$

else

$V \leftarrow \text{iterate}_{\text{random}}(T, \text{g-RAVE}, F)$

end if

Update $T_F, t_f, \hat{\mu}_{F,f}, \sigma^2_{F,f}$ and $l$-RAVE$_F$.

end if

**Iterate_random**

**Input:** search tree $T$, score g-RAVE, subset $F$

**Output:** reward $V$

while $\text{rand()} < q^{|F|}$ do

$f^* \leftarrow$ uniformly selected feature in $F \setminus (F \cup \{f_s\})$

$F \leftarrow F \cup \{f^*\}$

end while

$V \leftarrow V(F)$ ; Update g-RAVE
### Output
- Search tree (most visited path)

### Algorithm
- **FUSE**
  - RAVE score guides FUSE exploration
- **FUSE\textsuperscript{R}**
  - FUSE helps build RAVE score, indicating feature relevance

### FS approach
- **Wrapper**
Experimental Validation

<table>
<thead>
<tr>
<th>Data set</th>
<th>Samples</th>
<th>Features</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>2,600</td>
<td>500</td>
<td>XOR-like</td>
</tr>
<tr>
<td>Arcene</td>
<td>200</td>
<td>10,000*</td>
<td>disjunction of overlapping sub concepts</td>
</tr>
<tr>
<td>Colon</td>
<td>62</td>
<td>2,000</td>
<td>„easy“</td>
</tr>
</tbody>
</table>

* only top 2000 are considered for FUSE and CFS, ranked after their ANOVA score

Baseline approaches

- Correlation-based Feature Selection (CFS)
- RandomForest-based Gini score (Gini-RF) *
- Lasso
- RAND^R – average RAVE score built from random 20-feature subsets

- 200,000 iterations
- Gaussian SVM as end learner (5-fold CV optimized hyper-parameters)

* with 1,000 trees
Results

FUSE algorithms “best of both worlds”
- detect feature interdependencies (like Gini-RF, better with few features)
- filter out redundant features (like CFS, better with many features)
Results (contd.)

- all equal on colon
- **FUSE vs. FUSE\textsuperscript{R}:** FUSE does not control depth of search tree efficiently → FUSE\textsuperscript{R} better
- **discrete vs. continuous:** same performance with optimal parameters → discrete more robust due to less parameters

**Performance on Madelon dataset**
- FUSE\textsuperscript{R} converges more slowly than FUSE but improves after 10,000 iterations
- FUSE\textsuperscript{R} is faster by an order of magnitude than RAND\textsuperscript{R}
- runtime 45 minutes (Arcene: 5min, Colon: 4min) *

* on Intel Core 2×2.6GHz CPU with 2GB memory, only considering FS on the training set
Summary and Outlook

Contributions

- formalized FS task as a Reinforcement Learning problem
- proposed efficient approximation for optimal policy
- used UCT to define FUSE algorithm
- according to benchmark state of the art, but costly

Future directions

- extend to multi-class problems
- extend to mixed (continuous and discrete) search spaces
- combine FUSE with other end learners
- reconsider instant reward
- extend to feature construction
Critical Evaluation

- original approach for FS
- promising validation results

However…
- many degrees of freedom
  - interdependencies not fully understood
  - problem is simply shifted
- inherits problems from k-NN when working with
  - high dimensionality
  - skewed class distributions
- extensions probably further increase computational costs
- RF, Lasso as wrappers is fair for comparison, but unlike (usually) used in practice
Thank you!
Questions?

See next slide for sources
Sources

- Guyon, Isabelle; Elisseeff, André: *An Introduction to Feature Extraction*. In: Guyon, Isabelle et. al. (editors): Feature Extraction. 2006.